Agenzia nazionale per le nuove tecnologie, l'energia e lo sviluppo economico sostenibile

MINISTERO DELLA TRANSIZIONE ECOLOGICA

Ricerca di Sistema elettrico

Sviluppo del modello aggregato per la stima dell'affidabilità di reti ibride AC/DC in MT e BT in ambiente simulativo

R. Ciavarella, M. Valenti, G. Adinolfi, A. Merola, A. Ricca

Report RdS/PTR(2021)/065

SVILUPPO DEL MODELLO AGGREGATO PER LA STIMA DELL'AFFIDABILITÀ DI RETI IBRIDE AC/DC IN MT E BT IN AMBIENTE SIMULATIVO

R. Ciavarella (ENEA), M. Valenti (ENEA), G. Adinolfi (ENEA), A. Merola (ENEA), A. Ricca (ENEA)

Dicembre 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero della Transizione Ecologica - ENEA
Piano Triennale di Realizzazione 2019-2021 - III annualità (2021)
Obiettivo: *Sistema Elettrico*
Progetto: 2.7 Modelli e strumenti per incrementare l'efficienza energetica nel ciclo di produzione, trasporto, distribuzione dell'elettricità
Work package: Analisi delle problematiche di gestione per l'integrazione nelle attuali reti in AC di nuove reti in DC in MT/BT (Media Tensione/Bassa Tensione)
Linea di attività: *LA1.16 Sviluppo del modello aggregato per la stima dell'affidabilità di reti ibride AC/DC in MT e BT in ambiente simulativo*
Responsabile del Progetto: Maria Valenti, ENEA

Indice

Sommario

Le attività della LA1.16 sono state focalizzate sulla progettazione e l'implementazione del modello affidabilistico aggregato e sulla progettazione e programmazione di ORAtool, uno strumento software programmato ad hoc per il calcolo dell'affidabilità e l'ottimizzazione delle risorse energetiche dei sistemi di potenza in corrente alternata, continua o ibride (AC/DC). La progettazione di ORAtool, e la successiva programmazione, sono state condotte nell'ottica di realizzare un prodotto intuitivo e open source. In tal senso, il software è stato codificato utilizzando il linguaggio di programmazione Python e la navigazione è stata realizzata in accordo ad una logica di tipo wizard, ovvero basata su una modalità di navigazione guidata ottenuta per scomposizione in step elementari di ciascuna funzione. L'interfaccia di tipo wizard di ORAtool guida l'utente nella scelta delle diverse funzionalità del software, con la possibilità di poterle applicare sia su reti benchmark già integrate nel software (architetture e reti implementate nell'ambito della LA1.5), sia su reti definite dall'utente nei due ambienti di simulazione esterni Neplan e PowerFactory DigSilent.
I risultati delle elaborazioni del software sono espressi in duplice modalità, grafica e reportistica, così da permetterne una visualizzazione immediata ma anche un salvataggio in formato pdf.
I principali risultati della presente LA consistono in:

- ORAtool strumento software open source per il calcolo dell'affidabilità e l'ottimizzazione delle risorse energetiche dei sistemi di potenza in corrente alternata, continua o ibride (AC/DC). Il software è reso disponibile agli utenti finali attraverso il sito della Ricerca di Sistema Elettrico ENEA dedicata al progetto 2.7 (https://www.enea.it/it/Ricerca_sviluppo/lenergia/ricerca-di-sistema-elettrico/accordo-di-programma-MiSE-ENEA-2019-2021/sistema-elettrico/modelli-e-strumenti-per-incrementare-efficienza-energetica-nel-ciclo-di-produzione-trasporto-e-distribuzione-di-elettricita) – RdS/PTR(2021)/070
- Manuale d'uso del software – RdS/PTR(2021)/068
- Rapporto tecnico Report RdS/PTR(2021)/065 costituito da report descrittivo e allegato integrativo contenente i principali codici Python programmati per lo sviluppo del software.

1 TOOL PER L'ANALISI DELL'AFFIDABILITÀ DI RETI IBRIDE AC/DC

Il software ORATool mette a sistema modelli, funzionalità e le logiche di controllo sviluppate nelle diverse LA del progetto 2.7 opportunamente integrate in un unico software. In accordo con le specifiche del progetto, il tool consente all'utente di:

- selezionare una **logica di controllo singola** o effettuare una scelta multipla (**logiche di controllo multipla**), applicandola ad un modello di rete reso disponibile dal software (per dettagli sulle reti benchmark si veda il rapporto tecnico RdS/PTR(2020)/002) o caricato dall'utente (software esterni per il caricamento delle reti ammessi: Powerfactory DigSilent o Neplan).
- scegliere uno **scenario energetico** precaricato nel software (scenari 2020, 2030, 2040 – base-centralizzato o decentrato – per dettagli si veda il report RdS/PTR(2020)/001) o impostare un proprio scenario operativo;
- stimare l'**affidabilità di ogni componente della rete e quella complessiva a livello di fornitura elettrica** (per dettagli sui modelli affidabilistici si veda il rapporto tecnico RdS/PTR(2020)/007));
- effettuare un **dimensionamento delle protezioni**, fornendo per ciascuna protezione: costo, sovratensione e sovracorrente nel caso duplice di dispositivo di tipo elettromeccanico e di tipo elettronico;
- valutare **indici economici, energetici e affidabilistici**.

Uno schema concettuale di impostazioni e funzionalità rese disponibili dal tool è di seguito riportato in Figura 1.

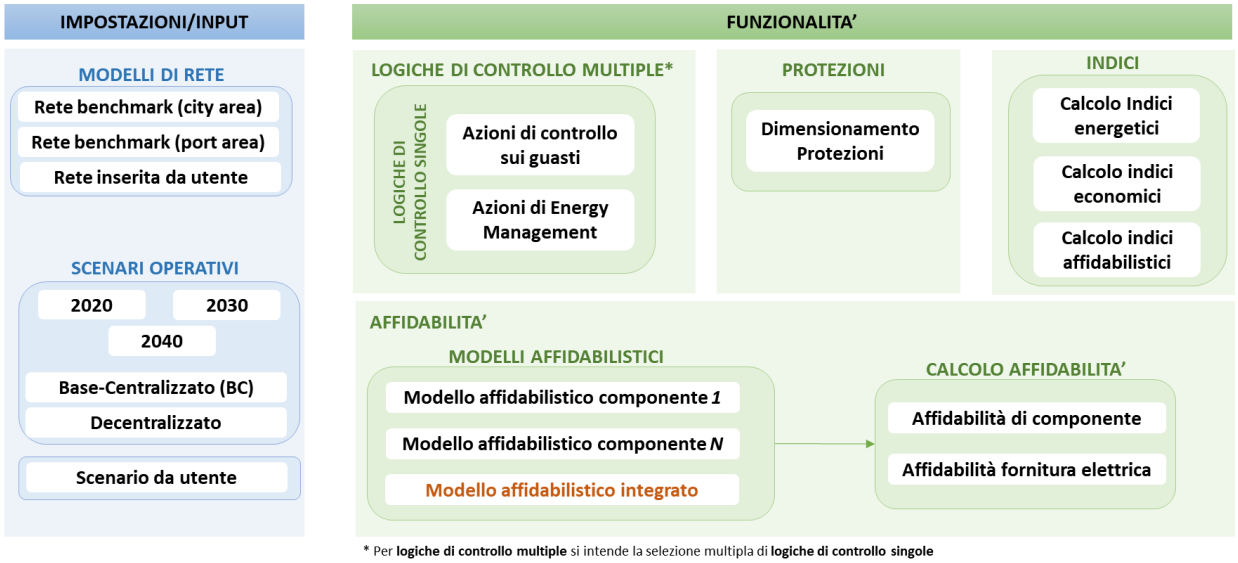


Figura 1: Schema concettuale di input e funzionalità dell'ORAtool

L'insieme delle scelte operate definisce la tipologia di configurazione. In accodo con quanto stabilito in sede progettuale e riportato nel capitolato di progetto, il tool consente di operare su:

- ***configurazioni base*** ovvero reti a cui non è applicata alcuna logica di controllo. In tal caso, l'utente potrà selezionare le diverse impostazioni (modelli e scenari) e studiarne l'affidabilità sia a livello di fornitura che di componente. Il software, in particolare, è stato sviluppato in modo tale che l'analisi di affidabilità fornisca entrambe le tipologie di risultati (componente e fornitura) in maniera automatica ovvero senza la necessità di selezionare particolari opzioni;
- ***configurazioni avanzate su logica singola*** ovvero reti a cui viene applicata una sola logica di controllo specifica tra quelle evidenziate nello schema di principio riportato in Figura 1. Sostanzialmente le configurazioni avanzate sono concettualmente equiparabili a quelle base con l'aggiunta di una logica di controllo. A valle dell'applicazione della logica, sarà possibile sia misurarne gli "impatti" attraverso specifici indici energetici ed economici che studiarne l'affidabilità, valutandone i relativi indici affidabilistici;
- ***configurazioni avanzate su logiche multiple*** configurazioni analoghe a quelle su logica singola ma in cui l'utente ha selezionate più logiche singole. Come nel caso precedente, gli impatti delle logiche saranno valutati attraverso specifici indici energetici ed economici, e sarà possibile effettuare lo studio dell'affidabilità della configurazione, anche valutandone i relativi indici affidabilistici.

Nel prosieguo del report saranno descritte le modalità di definizione del modello aggregato di stima dell'affidabilità e i diversi step logici seguiti per l'implementazione del software.

1.1 ARCHITETTURA DEL SOFTWARE

Il primo passo nella progettazione di ORAtool è consistito nella definizione dell'architettura e dell'alberatura dell'applicativo. In tal senso, il sistema è stato scomposto in sottosistemi (schema di principio in Figura 1) ed è stata stabilita l'interazione tra sottosistemi da realizzare attraverso le interfacce. A partire dalla struttura così ottenuta si è proceduto ad individuare le "proprietà visibili" e quelle "non visibili" del tool. In particolare, si è stabilito che rientrano tra le proprietà non visibili tutti gli algoritmi relativi a:

- realizzazione delle funzionalità;
- interfacciamento con i software esterni;
- gestione dei flussi di dati in ingresso e uscita.

Rientrano tra le proprietà visibili tutte le interfacce di comunicazione con l'utente del software (es. input/output), come sinteticamente rappresentato dallo schema di Figura 2.

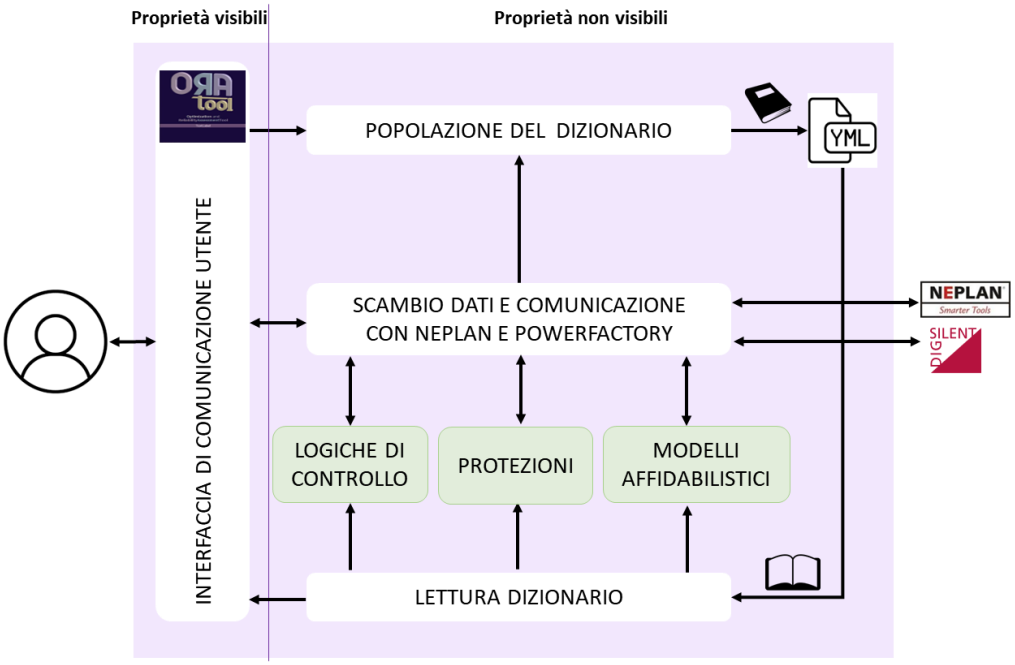


Figura 2: Proprietà visibili e non visibili ORAtool

La **gestione dei flussi di dati da e verso l'esterno** è stata implementata mediante la creazione di un apposito dizionario che è popolato da dati embedded (es. reti modello), dati provenienti dall'utente (input) e dati provenienti dai software esterni (Neplan e DigSilent). In generale, un **dizionario** è una mappa costituita da un insieme di chiavi (key) e un insieme di valori (value) che associa ad ogni chiave un valore: dict (key, value). Ad ogni chiave è associato un unico valore e non possono esserci due coppie con lo stesso valore della chiave (le coppie sono uniche). Nel caso specifico, il dizionario è stato costruito mediante chiavi annidate ed è stato salvato in un file YAML contenente tutti gli elementi della rete, così da poter realizzare un agevole scambio di informazioni tra le diverse classi del tool e permettere una facile interazione tra le diverse funzionalità. In tal senso, ad ogni elemento è stata associata una chiave principale del dizionario caratterizzata da 7 sottochiavi: category, conn, ems, parameters, protections, reliability, results, di cui 4 relative alle caratteristiche peculiari dell'elemento (category, conn, parameters, results) e 3 alla funzionalità (ems, protections, reliability). Di seguito, si riporta un estratto di dizionario relativo ad un elemento "carico" (Pros_DC-Load).

```
Pros_DC-Load:
  category: DC-Load
  conn:
    Pros_BB_DC2: true
    h: Pros_BB_DC2
  ems:
    profile:
      load:
      - 0.34
    type: load
    unmet_cost: 7.0
  parameters:
    I: 28.333
    P: 17.0
    R: 21.176
    control: 0
```

```
    profile:
      constant: false
      curve:
      - 0.02
     name: DC-Load1
  protections:
    In: 28.333
    Pn: 16.999799999999997
    Vn: 0.6
    category: DC-Load
    results:
      comparison:
        el:
          cost: 101.65332543405258
          overcurrent: 28.333
          overvoltage: 0.8220000000000001
        em:
          cost: 36.28937013360243
          overcurrent: 283.33
          overvoltage: 1.38
      cost: 101.65332543405258
      delay_I: 0.5
      delay_Vmax: 0.3
      delay_Vmin: 0.1
      soglia_I: 32.58295
      soglia_Vmax: 0.63
      soglia_Vmin: 0.51
      type: Interruttore elettronico
  reliability:
    Pi_E: 1.0
    Pi_Q: 5.5
    T0: 30.0
    alfa: 438000.0
    beta: 1.0
    results:
      MTBF_anni: 0
      MTBF_ore: 0
      R: 0
      lambda: 0
  results:
    I: 14.16666666666667
    Iangle: 0.005729577951308233
    LimitViolated: false
    P: 8.5
    Q: 0.0
    S: 8.5
    U: 0.6
    cosPhi: 1.0
```


Per lo scambio esterno dei dati relativo alla **comunicazione con il DigSilent PowerFactory e il Neplan** sono state, quindi, definite due specifiche **classi**[1] di comunicazione con i software. Parimenti, sono state sviluppate attraverso l'utilizzo di classi tutte le funzionalità del tool, sia in relazione alla categoria di proprietà visibili che a quella di proprietà non visibili (Figura 2). L'approccio per classi ha comportato due grandi vantaggi:

- è stato possibile implementare un software caratterizzato da una spinta modularità e, pertanto, facilmente estendibile per sviluppi futuri;

[1] Le classi sono raggruppamenti di dati e funzionalità che rappresentano un nuovo tipo facilmente replicabile all'interno del software.

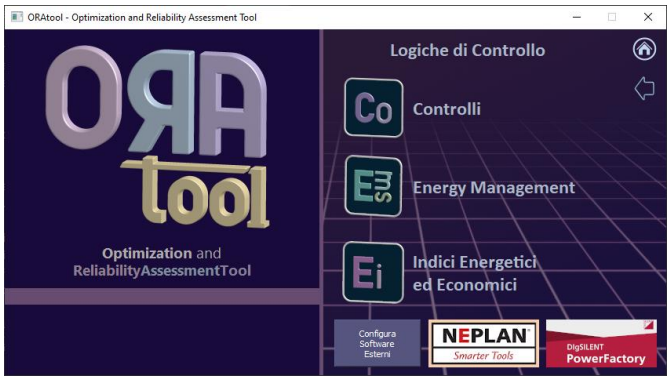
- i software DigSilent PowerFactory e Neplan prevedono la possibilità di interagire con Python mediante specifiche classi e di effettuare calcoli e valutazioni anche senza aprire la loro interfaccia grafica, sfruttando la modalità engine mode per DigSilent e i Webservices per Neplan.

1.2 FUNZIONALITA'

1.2.1 Logiche di controllo

Le Logiche di Controllo implementate nel tool consentono all'utente di scegliere tra le funzioni:

- Controlli;
- Energy Management.



Mediante la **<u>funzione "*Controlli*"</u>** è possibile analizzare gli stati del sistema, definiti come la combinazione di una configurazione di rete, uno specifico scenario energetico, selezionato tra quelli disponibili nel tool o definiti dall'utente stesso, e un evento di failure. Per ciascuno stato del sistema, è possibile analizzare il comportamento della rete in presenza del guasto, ovvero verificare il miglioramento dell'affidabilità e della continuità di servizio per effetto dell'azione del controllo attraverso i seguenti parametri/indicatori (per i dettagli sui controlli e sugli indicatori si veda il Report RdS/PTR2021/062):

1. indice di autonomia;
2. indice di flessibilità;
3. indice di modulazione;
4. percentuale di generazione da fonte rinnovabile rispetto alla generazione complessiva della rete/microrete/porzione di rete funzionante in isola (RES);
5. flessibilità del carico (FLEX);
6. rapporto tra la potenza fornita dai sistemi di accumulo e la potenza totale dei sistemi di generazione della rete/microrete/porzione di rete funzionante in isola (BESS);
7. durata delle interruzioni (Ti);
8. energia non fornita (Ens);
9. rapporto tra il numero di generatori che possono ridurre contemporaneamente la propria produzione di energia ed il numero complessivo di generatori, mediato sulla base delle taglie dei generatori presenti in una microrete (Ng);
10. riserva di energia dei sistemi di accumulo installati rispetto al consumo giornaliero della rete/microrete/porzione di rete funzionante in isola (ST);
11. indice di grid forming (GF)
12. rapporto di inerzia (RI).

Gli eventi di failure integrati nel programma sono:

- Fuori servizio del bus di alimentazione (Fuori servizio di una linea, di un trasformatore o di un convertitore);
- Guasto (cortocircuito o interruzione) in una linea di collegamento;
- Improvvisa perdita di generazione o di carico con conseguente squilibrio delle potenze nell'impianto;
- Creazione di isole involontarie;

- Malfunzionamento di un dispositivo di protezione;
- Malfunzionamento di un generatore (non necessariamente dell'unico non di alimentazione);
- Malfunzionamento del sistema di controllo di una risorsa flessibile.
- Malfunzionamento del sistema di distacco dei carichi.

In accordo con quanto descritto nel rapporto tecnico RdS/PTR(2020)/003, la funzionalità tiene conto delle seguenti contingenze:

- eventi di failure che portano al funzionamento in isola di parte della rete ibrida;
- eventi di failure che richiedono il supporto in regime statico della rete DC alla rete AC;
- eventi di failure che richiedono il supporto in regime dinamico della rete DC alla rete AC.

Data la già richiamata modularità di ORAtool, gli eventi di failure possono essere estesi aggiungendo delle nuove classi al software.

La logica di controllo proposta è stata applicata ai due benchmark disponibili nel tool: City Area e Port Area. Per i dettagli operativi di utilizzo del controllo, si veda il manuale ORAtool (RdS/PTR(2021)/068).

Un esempio dell'interfaccia della funzionalità "controlli" è di seguito riportata in Figura 3.

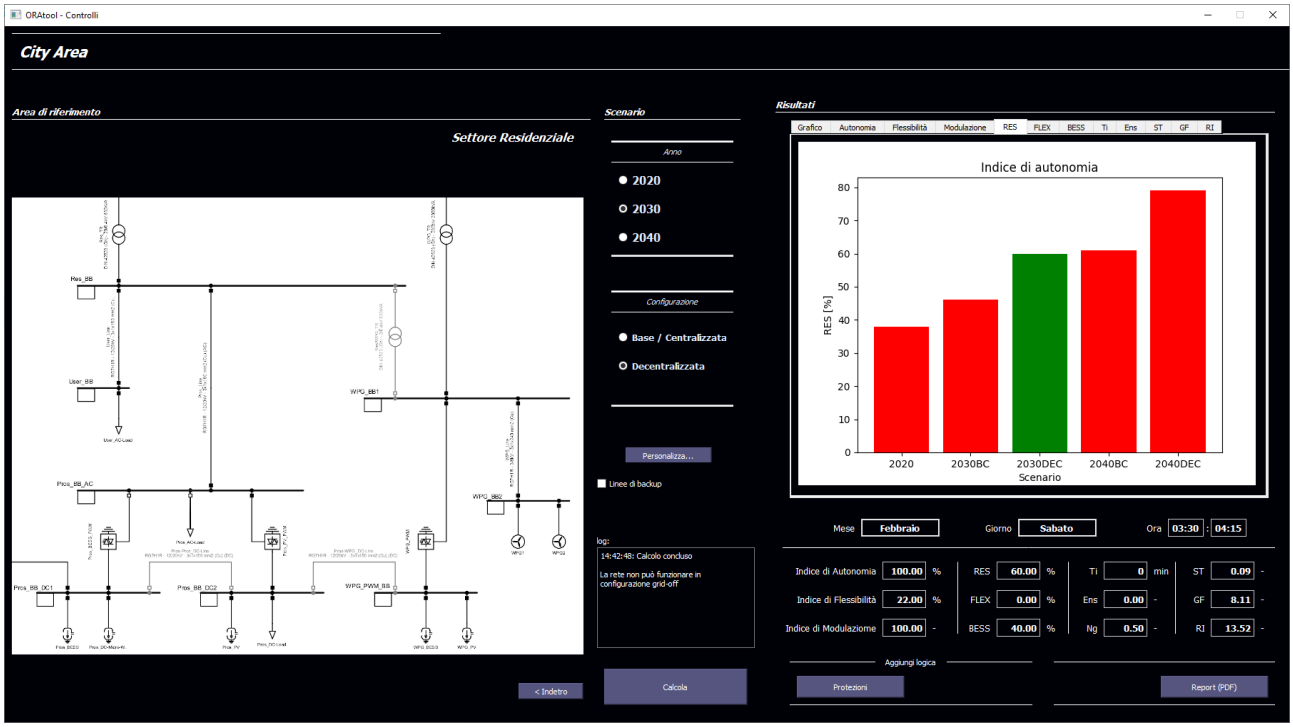


Figura 3: Interfaccia della funzionalità "controlli"

La funzione **Energy Management** propone un EMS (Energy Management System) per risolvere problemi di dispacciamento per un sistema elettrico su scala microrete caratterizzato da diversi componenti di generazione elettrica rinnovabile, accumulo elettrochimico, carico, convertitori, linee di distribuzione e trasformatori con l'obiettivo di minimizzare il costo operativo per il sistema tenendo conto dei prezzi di acquisto e vendita dell'energia verso la rete e del costo opportunità dell'utilizzo degli accumuli elettrici.

Inoltre, la logica comprende metodi di gestione predittivi ed un approccio simulativo per la valutazione delle incertezze e parametri di affidabilità per reti miste AC/DC. L'utente può definire scenari di guasto in modo arbitrario e valutare i seguenti indici:

1. Energia non fornita da ciascun utente (LPENS);
2. Energia non fornita per il sistema (ENS);
3. Costo di interruzione per ciascun utente (LPEIC)
4. Costo totale per le interruzioni (EIC)

Un esempio dell'interfaccia della funzionalità "controlli" è di seguito riportata in Figura 4.

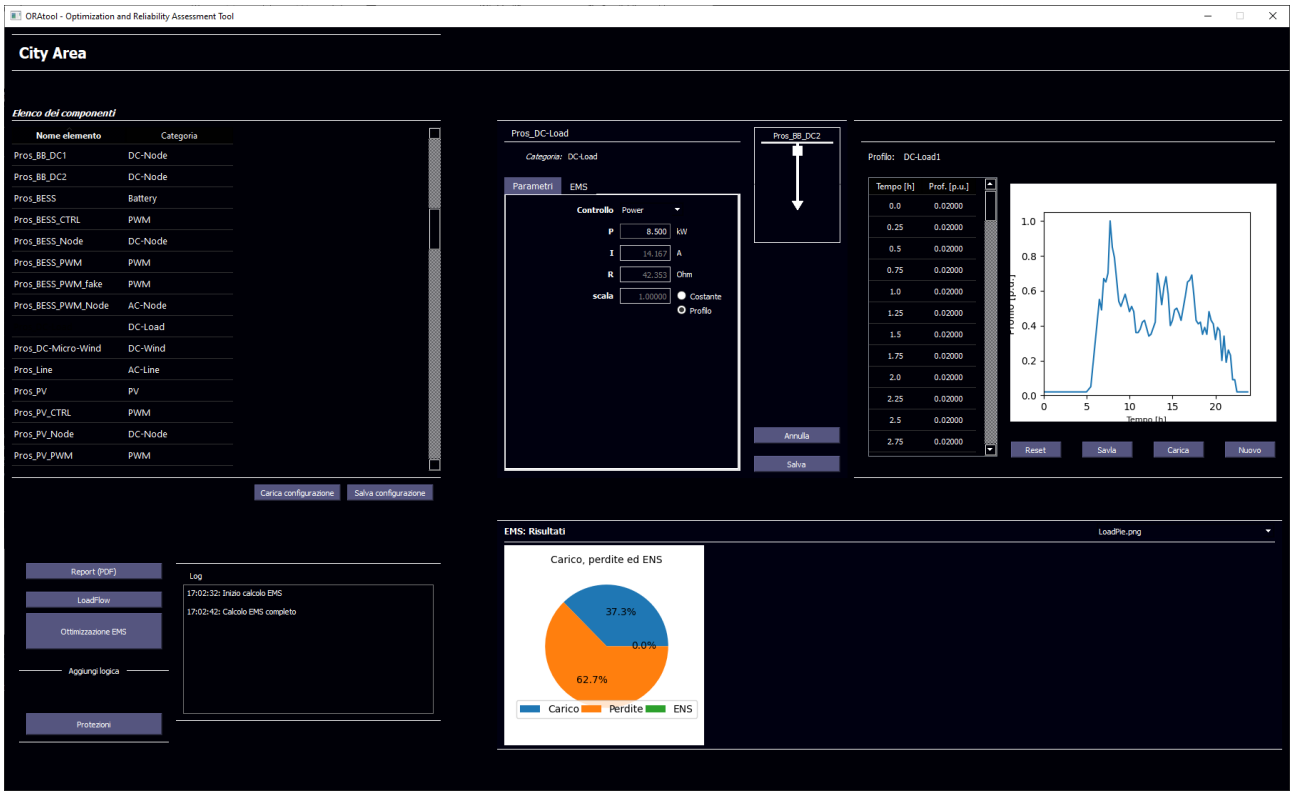
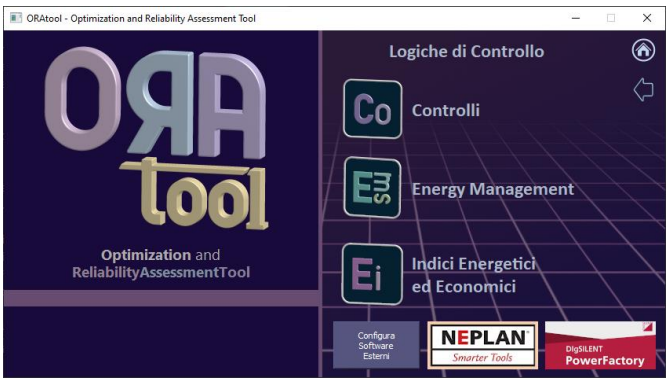


Figura 4: Interfaccia della funzionalità "Energy Management"

1.2.2 Indici energetici ed economici

La funzione "*Indici energetici ed economici*" permette, a partire da uno specifico scenario di guasto, di valutare i seguenti indici:

- energia non fornita da ciascun utente (LPENS);
- energia non fornita per il sistema (ENS);
- costo di interruzione per ciascun utente (LPEIC);
- costo totale per le interruzioni (EIC).



La funzione per il calcolo degli "Indici energetici ed economici" è riportata nella sezione del tool relativa alle Logiche di controllo, poiché il calcolo di tali indici viene eseguito dal software prima e dopo l'applicazione di una logica al fine di valutarne l'efficacia. Pertanto, la funzionalità è concettualmente riconducibile alla sezione Logiche di Controllo del software.
Un esempio dell'interfaccia della funzionalità "controlli" è di seguito riportata in Figura 5.

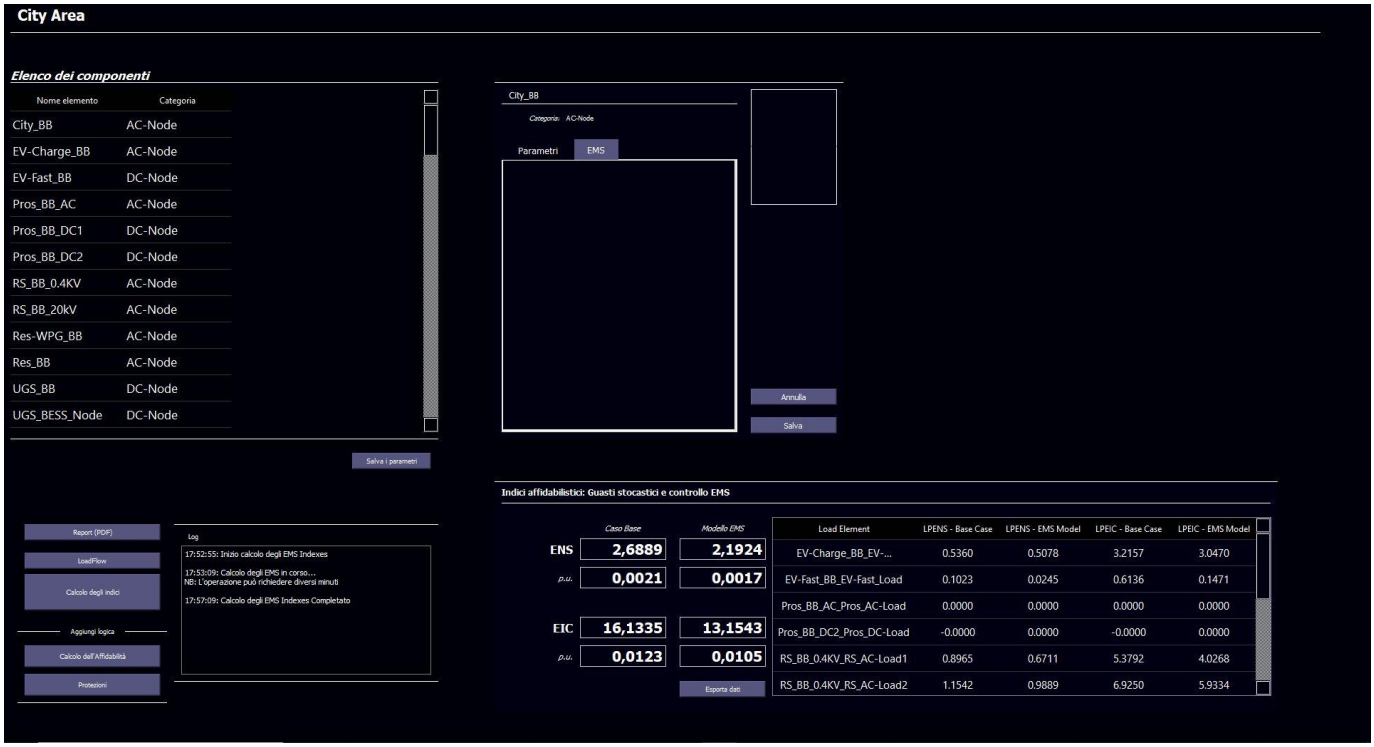
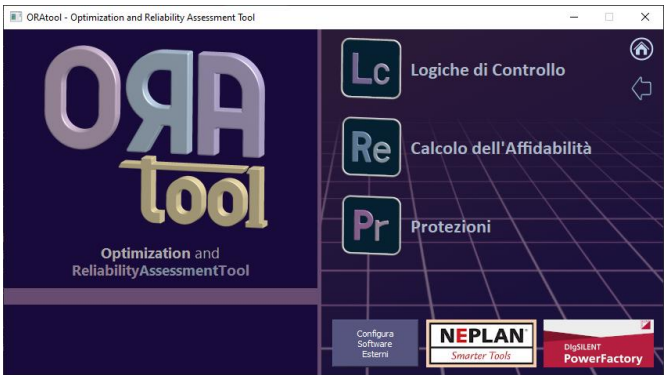


Figura 5: Interfaccia della funzionalità "Indici energetici ed economici"

1.2.3 Calcolo dell'affidabilità

La funzione "*Calcolo dell'affidabilità*" consente di valutare le prestazioni affidabilistiche di una rete benchmark precaricata, o di una qualsiasi rete caricata esternamente dall'utente.



Il tool, in base alle risorse presenti nella rete, valuta sia l'affidabilità per ciascun componente a partire dai relativi modelli affidabilistici sia l'affidabilità della fornitura elettrica. L'utente, in particolare, ha la possibilità di analizzare per ciascun componente le seguenti informazioni:

- Il tasso di guasto λ;
- L'affidabilità R(t);
- Il tempo medio fra i guasti (MTBF-Mean Time Between Failures) in ore (MTBF_ore) e in anni (MTBF_anni);
- Affidabilità della fornitura di ciascun carico presente nella rete.

Riguardo l'ultimo punto, il calcolo dell'affidabilità della fornitura per ciascun carico si basa sull'utilizzo del modulo Python NetworkX che consente di studiare reti complesse mediante la creazione di grafi. A partire dall'analisi dei path minimi di collegamento tra generazione e carico presenti in una rete, il tool genera una serie di grafi per descrivere i collegamenti generazione-carico. A ciascuno di questi grafi viene poi associato

un Reliability Block Diagram (RBD), grazie al modulo fiabilipy, valutando in questo modo l'affidabilità della fornitura per ciascun carico presente in rete. Un esempio di grafo è di seguito riportato in Figura 6:

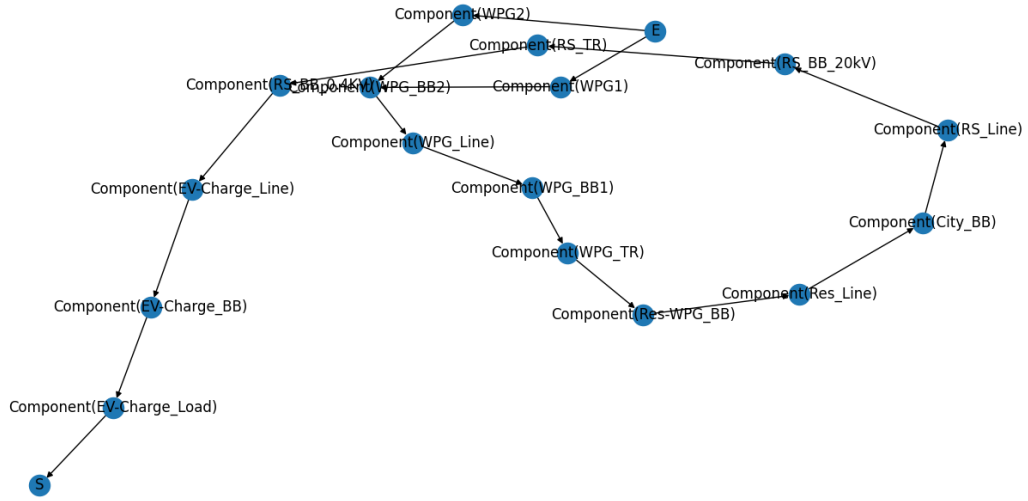


Figura 6: Esempio di grafo ottenuto nel caso del benchmark City Area

Il grafo rappresentato in figura è stato ottenuto nel caso del benchmark "City Area" disponibile nel tool. Tale grafo rappresenta i path di collegamento tra i generatori "WPG1" e "WPG2" verso il carico "Charge_Load". I nodi del grafo rappresentano tutti i componenti di rete (linee, trasformatori, convertitori, etc.) presenti lungo il percorso di collegamento tra i generatori ed il carico. A ciascuno di questi componenti sono associate le informazioni affidabilistiche ottenute grazie ai modelli affidabilistici descritti. Un esempio dei risultati ottenuti selezionando la funzionalità protezioni è di seguito riportato in Figura 7.

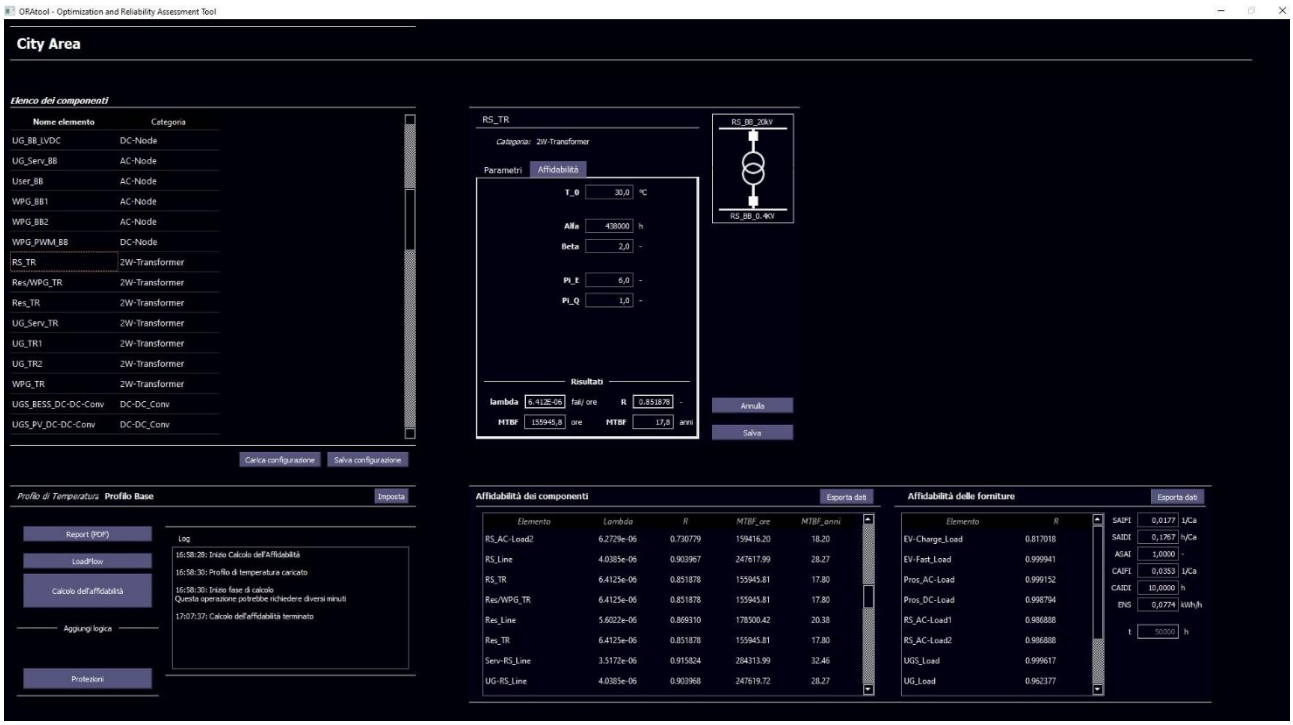


Figura 7: Esempio di risultati della funzionalità "Calcolo dell'affidabilità"

1.2.4 Protezioni

La funzione "*Protezioni*" consente di dimensionare le protezioni applicabili alla rete, valutandone le prestazioni tecnico-economiche nei due casi di utilizzo di un dispositivo di tipo elettromeccanico e di uno tipo elettronico.

Le protezioni dei vari componenti DC vengono dimensionate in funzione della tensione nominale, della corrente e del tipo di componente che si sta analizzando. Inoltre, per ciascuna protezione viene calcolato il costo mettendo in evidenza il costo minimo, massimo, ed il costo della soluzione proposta per l'intera rete. Un esempio dei risultati ottenuti selezionando la funzionalità protezioni è di seguito riportato in Figura 8.

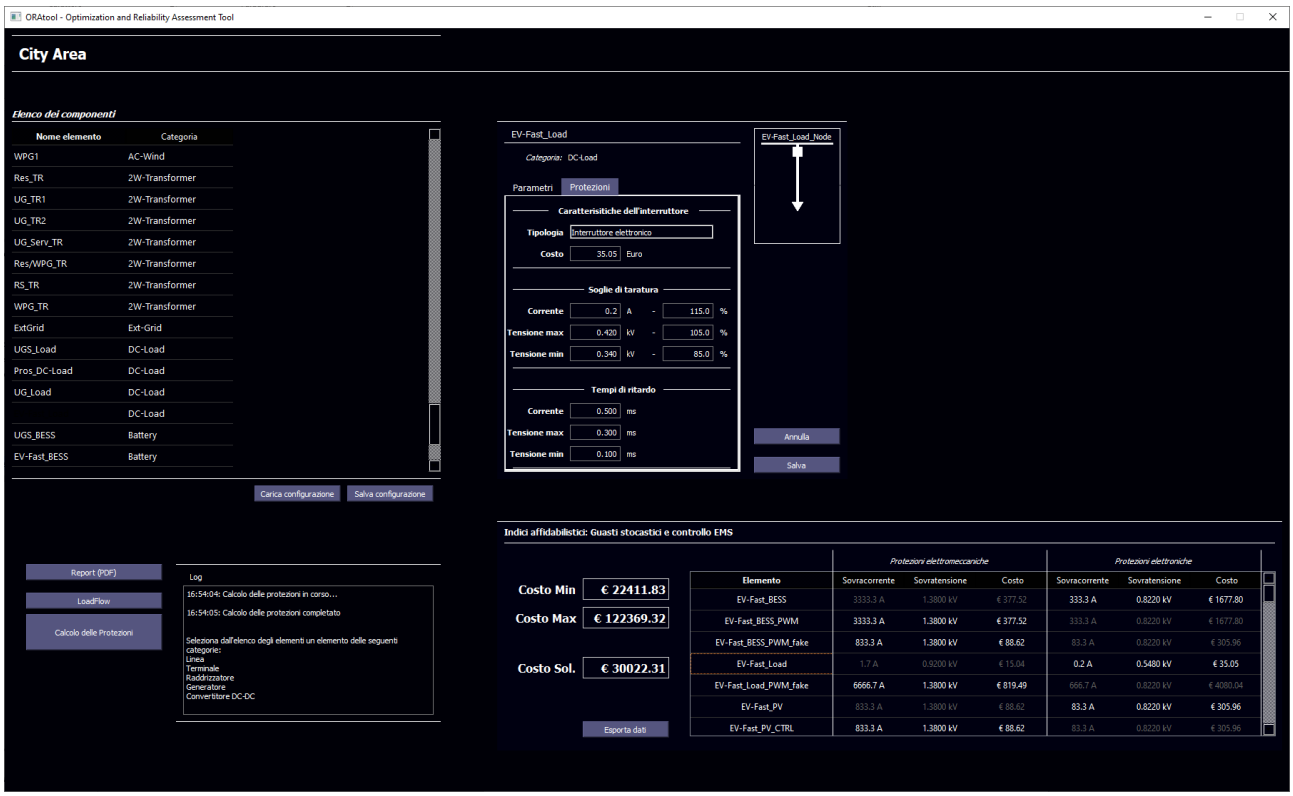


Figura 8: Esempio di risultati della funzionalità "Protezioni"

2 FLOW CHART LOGICI DELL'ORATool

In questa sezione sono riportati i flow chart logici di funzionamento del software. Tali schemi hanno il solo fine di illustrare i passi logici che l'utente deve seguire nell'utilizzo del tool. Per la guida dettagliata all'utilizzo, si rimanda al manuale d'uso (RdS/PTR(2021)/068). Come evidente dalle Figure 9-13, il primo passo da compiere, indipendentemente dalle successive scelte, è comune a tutte le azioni ed è relativo alla selezione di un modello di rete o al caricamento di una rete esterna. La scelta della rete, in particolare, richiede l'inserimento di tutti i parametri necessari alla completa definizione delle risorse della rete stessa.

Caratterizzata completamente la rete e le sue risorse, l'utilizzo del software prevede step connessi alla specifica funzionalità.

2.1 LOGICHE DI CONTROLLO

2.1.1 Controlli

La Figura 9 mostra il percorso logico relativo alla selezione della funzionalità "Logiche di Controllo" e, successivamente, alla funzione "Controlli". In tal caso, scelto il modello di rete da analizzare e l'opzione logiche di controllo, l'utente dovrà selezionare anche il settore/area di rete sulla quale applicare il controllo (se previsto per la specifica rete). Successivamente, l'utente dovrà inserire i seguenti input obbligati:

- scenario energetico (scelto tra quelli già disponibili o personalizzato);
- configurazione ("Base/ Centralizzata": diffusione centralizzata degli impianti di generazione rinnovabile e di accumulo; "Decentralizzata": diffusione in prossimità dei punti di utilizzo di impianti di generazione da rinnovabile e di accumulo).

Definiti i dati di input, l'utente potrà avviare la sessione di simulazione per il calcolo dei relativi indici e la produzione di un report in formato pdf.

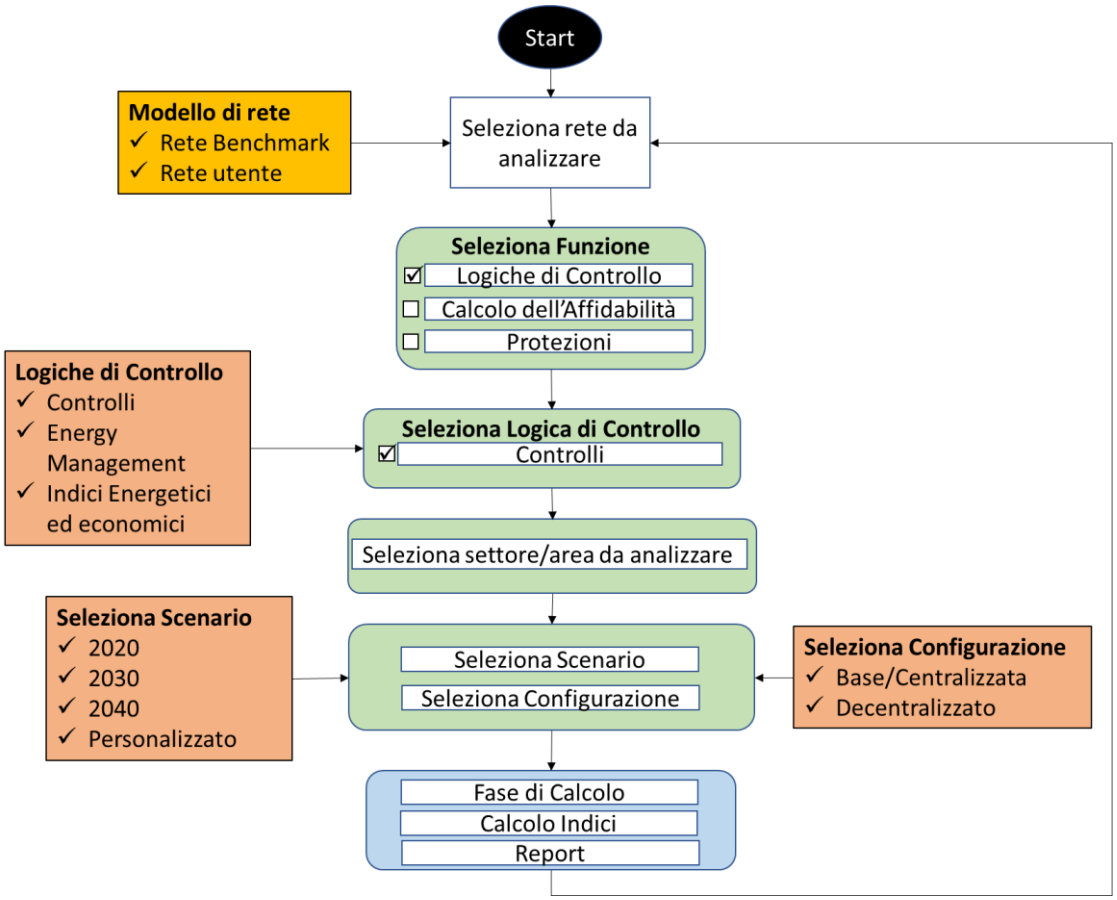


Figura 9: Azione di Controllo su guasti

2.1.2 Energy management

La Figura 10 mostra il caso in cui l'utente scelga di applicare la logica di controllo "Energy Management" alla rete selezionata.
L'utente viene guidato nell'inserimento, componente per componente, di tutti i parametri necessari. Dopodiché è possibile eseguire la fase di calcolo che consiste in un'ottimizzazione economica che minimizzi il costo operativo per il sistema in considerazione tenendo conto dei prezzi di acquisto e vendita dell'energia verso la rete e del costo opportunità dell'utilizzo degli accumuli elettrici. Inoltre, i risultati, come in tutti i casi, vengono resi disponibili sia nella schermata del software che in un report in formato pdf.

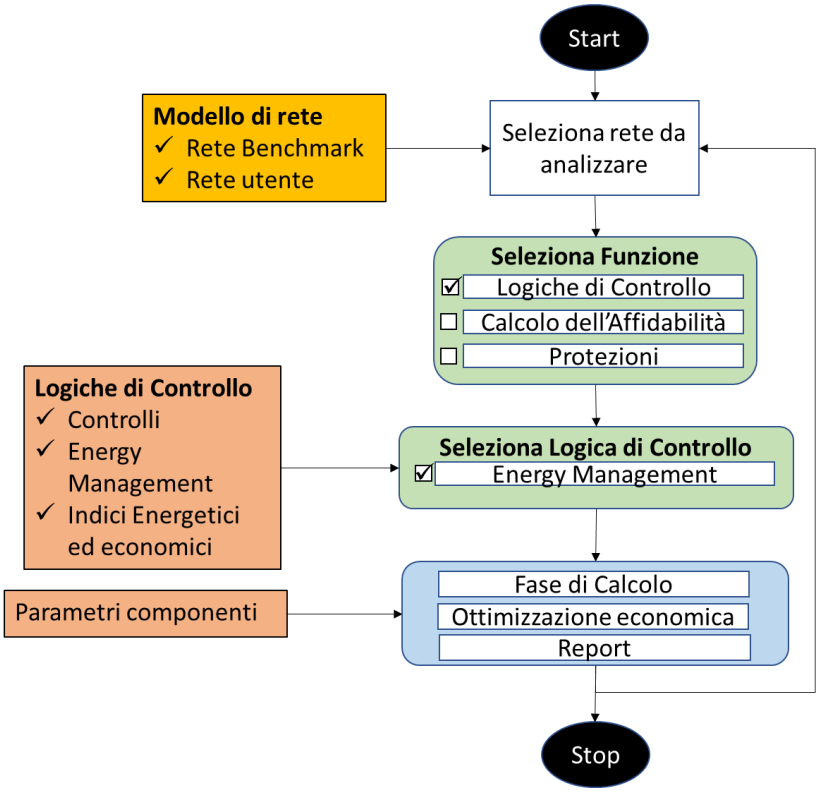


Figura 10: Energy Management

2.2 INDICI ENERGETICI ED ECONOMICI

In Figura 11 è riportato lo schema logico generale del tool nel caso in cui l'utente scelga di eseguire la logica di controllo "Indici energetici ed economici".
In questo caso, una volta scelta la rete da analizzare, è necessario che l'utente definisca lo scenario di guasto da considerare. La definizione dello scenario di guasto avviene tramite una procedura guidata che indica all'utente le informazioni da inserire nel tool.
Dopo aver definito lo scenario di guasto, è possibile avviare la fase di simulazione per il calcolo degli indici: energia non fornita da ciascun utente (LPENS), energia non fornita per il sistema (ENS), costo di interruzione per ciascun utente (LPEIC), costo totale per le interruzioni (EIC).
Gli indici vengono calcolati sia nel caso base, ovvero senza l'azione del controllo "Energy management" che considerando l'azione del controllo.
I risultati, come in tutti i casi, vengono resi disponibili sia nella schermata del software che in un report reso disponibile in formato pdf.

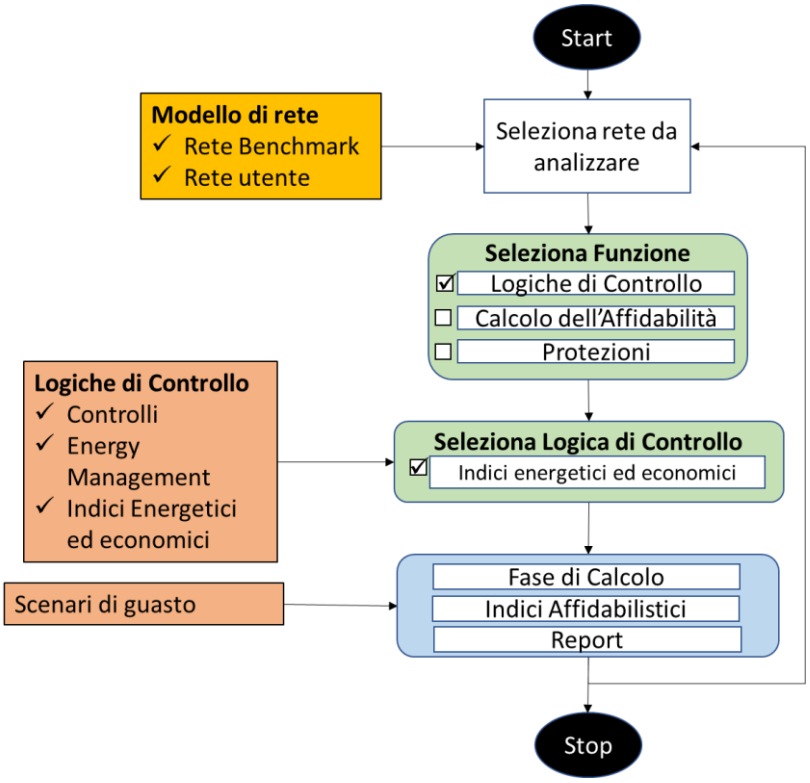


Figura 11: Indici energetici ed economici

2.3 CALCOLO DELL'AFFIDABILITÀ

In Figura 12 è mostrata la sequenza logica delle azioni che l'utente deve compiere nel caso in cui venga selezionata la funzione "Calcolo dell'Affidabilità". Per poter procedere con il calcolo è necessario che l'utente imposti un profilo di temperatura, scelto tra profili di default o appositamente definito, e inserire tutti i parametri affidabilistici. Per quanto riguarda le condizioni operative dei componenti, queste vengono lette in automatico dal modello di rete selezionato.
Una volta definiti tutti gli input necessari, l'utente può eseguire la fase di calcolo che consiste nel determinare sia l'affidabilità per componente che l'affidabilità della fornitura per ciascun carico presente nella rete. Infine, vengono calcolati gli indici affidabilistici di rete (SAIFI, SAIDI, ENS, etc.)
I risultati, come in tutti i casi, vengono resi disponibili sia nella schermata del software che in un report reso disponibile in formato pdf.

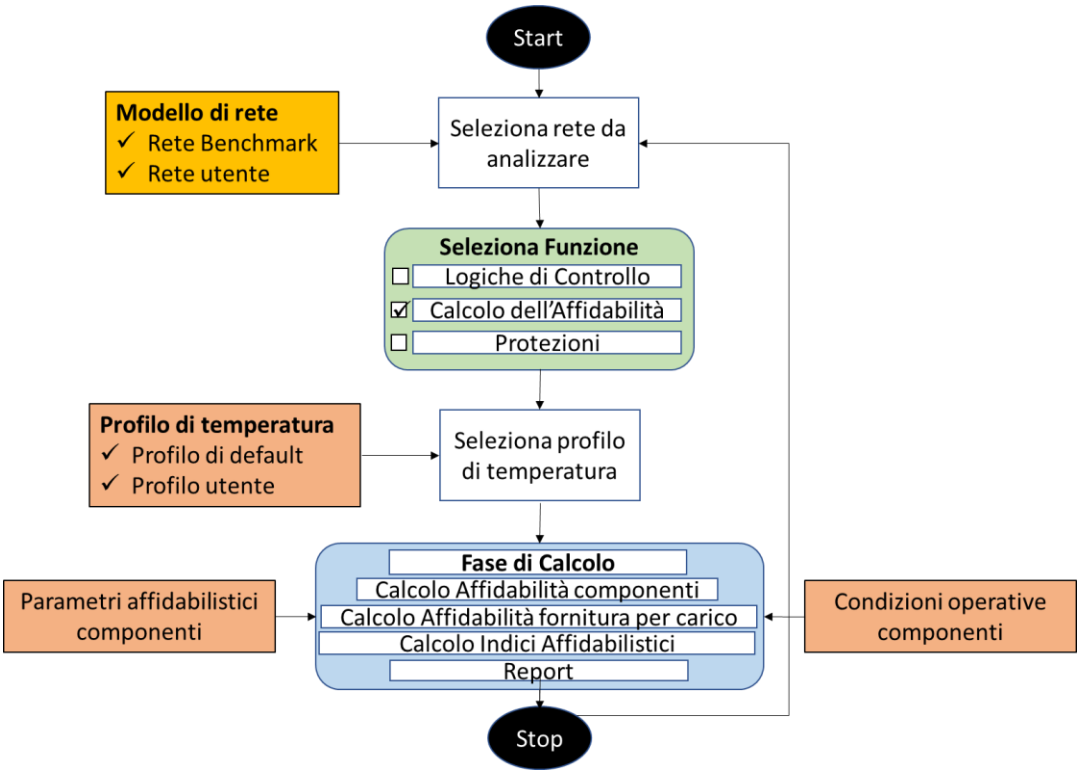


Figura 12: Calcolo dell'Affidabilità

2.4 PROTEZIONI

La Figura 13 mostra la sequenza logica delle azioni che l'utente deve compiere nel caso in cui si desideri determinare le caratteristiche delle protezioni da adottare per i vari componenti presenti in rete.

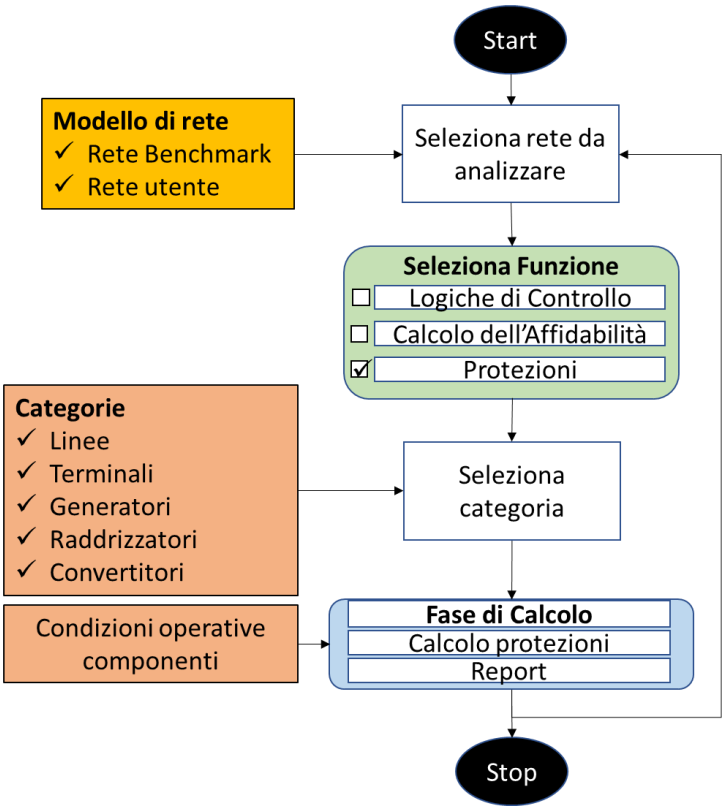


Figura 13: Calcolo caratteristiche protezioni

Per il calcolo delle protezioni, i componenti di rete sono stati suddivisi in 5 categorie (Linee, Terminali, generatori, Raddrizzatori, Convertitori). L'utente ha la possibilità di scegliere una o più categorie da analizzare. Altro input essenziale è rappresentato dalle condizioni operative dei vari componenti (corrente, tensione, etc.). Queste informazioni vengono ottenute in automatico dal tool grazie alle funzioni appositamente sviluppate per la comunicazione con ambienti di modellazione esterni quali il Neplan e il PowerFactory.
Dopo aver acquisito i necessari input, è possibile avviare la fase di calcolo che, per ciascuna protezione fornirà le seguenti informazioni:

- ✓ Tipo di protezione (elettromeccanica/elettronica)
- ✓ Sovracorrente
- ✓ Sovratensione
- ✓ Costo protezione

Infine, vengono fornite le seguenti informazioni generali sulla soluzione proposta a livello dell'intera rete:

- ✓ Costo minimo
- ✓ Costo massimo
- ✓ Costo soluzione proposta

Infine, i risultati, come nei precedenti casi, vengono resi disponibili sia nella schermata del software che in un report in formato pdf.

3 SVILUPPO DEL SOFTWARE: PROGETTO E ALGORITMI PYTHON

Il presente paragrafo descrive tutte le classi, scritte in Python, del tool prodotto (ORAtool). Data la complessità della struttura del progetto, il documento è stato organizzato in sezioni in base alle funzioni sviluppate e riportate nel seguente albero di navigazione.

Albero di navigazione

4 STRUTTURA DEL PROGETTO

```
|  main.py
|
+---Functionalities
|  +---Controls
|  |  |  port.py
|  |  |  residential.py
|  |  |  roadservices.py
|  |  |  underground.py
|  |
|  +---EMS
|  |  |  BaseCase_IndicesCalculation.py
|  |  |  EMS.py
|  |  |  ems_dict_correct.py
|  |  |  EMS_module.py
|  |  |  EMS_stoch_sim.py
|  |  |  utils.py
|  |
|  +---PDF
|  |  |     pdf_creator.py
|  |
|  \---Protections
|     |  protection.py
|           |
+---Softwares
|  +---Neplan
|  |  |  neplan.py
|  |  |  webservices.py
|  |
|  \---PowerFactory
|        pf.py
|        pf_class.py
|
+---UI
|  +---Main
|  |  |  MainUI.py
|  |  |  UI.py
|  |  |
|  |  +---Controls
|  |  |  |  controls.py
|  |  |  |  controlsUI.py
|  |  |
|  |  +---Elements
|  |  |  +---ACLine
|  |  |  |  |  acline.py
|  |  |  |  |  aclineUI.py
|  |  |  |
|  |  |  +---ACLoads
|  |  |  |  |  acload.py
|  |  |  |  |  acloadUI.py
|  |  |  |
|  |  |  +---ACnode
|  |  |  |  |  acnode.py
|  |  |  |  |  acnodeUI.py
|  |  |  |
|  |  |  +---ACwind
```

```
|   |   |   |   |   acwind.py
|   |   |   |   |   acwindUI.py
|   |   |   |
|   |   |   +---Battery
|   |   |   |   |   battery.py
|   |   |   |   |   batteryUI.py
|   |   |   |
|   |   |   +---DCDC_converter
|   |   |   |   |   dcdc_conv.py
|   |   |   |   |   dcdc_convUI.py
|   |   |   |
|   |   |   +---DCLine
|   |   |   |   |   dcline.py
|   |   |   |   |   dclineUI.py
|   |   |   |
|   |   |   +---DCLoads
|   |   |   |   |   dcload.py
|   |   |   |   |   dcloadUI.py
|   |   |   |
|   |   |   +---DCnode
|   |   |   |   |   dcnode.py
|   |   |   |   |   dcnodeUI.py
|   |   |   |
|   |   |   +---DCwind
|   |   |   |   |   dcwind.py
|   |   |   |   |   dcwindUI.py
|   |   |   |
|   |   |   +---ExtGrid
|   |   |   |   |   extgrid.py
|   |   |   |   |   extgridUI.py
|   |   |   |
|   |   |   +---PV
|   |   |   |   |   pv.py
|   |   |   |   |   pvUI.py
|   |   |   |
|   |   |   +---PWM
|   |   |   |   |   pwm.py
|   |   |   |   |   pwmUI.py
|   |   |   |
|   |   |   \---Transformers_TwoWings
|   |   |       |   transformer_TwoWings.py
|   |   |       |   transformer_TwoWingsUI.py
|   |   |
|   |   +---EMS
|   |   |   |   ems_results.py
|   |   |   |   ems_resultsUI.py
|   |   |
|   |   +---EMS_stoch
|   |   |   |   ems_stoch_results.py
|   |   |   |   ems_stoch_resultsUI.py
|   |   |
|   |   +---Protections
|   |   |       protections.py
|   |   |       protectionsUI.py
|   |   |
|   |   +---Reliability
|   |   |   |   reliability_results.py
```

```
|  |  |  |  reliability_resultsUI.py
|  |
|  +---SplashScreen
|  |  |  splash.py
|  |  |  splashScreenUI.py
|  |  |
|  |  +---widgets
|  |  |  |  functionalities.py
|  |  |  |  functionalitiesUI.py
|  |  |  |  logics.py
|  |  |  |  logicsUI.py
|  |  |  |  open.py
|  |  |  |  openUI.py
|  |  |  |  scenarios.py
|  |  |  |  scenariosUI.py
|  |  |  |  specificlogics.py
|  |  |  |  specificlogicsUI.py
|  |
|  \---_general
|     +---Popup
|     |  +---Configurations
|     |  |  |  configurations.py
|     |  |  |  configurations_ui.py
|     |  |  |
|     |  +---EMS_Scenarios
|     |  |  |  popup_EMS_scen.py
|     |  |  |  popup_EMS_scen_ui.py
|     |  |  |
|     |  +---Protections
|     |  |  |     popup_prot.py
|     |  |  |     popup_prot_ui.py
|     |  |  |
|     |  \---Reliability
|     |     |  popup_rel.py
|     |     |  popup_rel_ui.py
|     |
|     +---Profile
|     |  |  profile.py
|     |  |  profileUI.py
|     |
|     +---RbdProfile
|     |     rbdprofile.py
|     |     rbdprofileUI.py
|     |
|     \---YearProfile
|        |  yearprofile.py
|        |  yearprofileUI.py
|
\---__shared__
   |  variables.py
   |
   +---plugins
   |  \---fiabilipy
   |        component.py
   |        markov.py
   |        system.py
   |        test_markov.py
```

| test_system.py
| voter.py
| __init__.py

4.1 Main.py

```
import os

import yaml

from PyQt5 import QtWidgets, QtCore, QtGui

from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
import matplotlib.pyplot as plt

from UI.SplashScreen.splash import Splash
from Functionalities.EMS.EMS import EMS

from Functionalities.EMS.ems_dict_correct import dict_correct as ems_dict_correct

from UI.Main.UI import UI
from UI.Main.Controls.controls import Controls

from UI.Main.Elements.ACLoads.acload import ACLoad
from UI.Main.Elements.DCLoads.dcload import DCLoad
from UI.Main.Elements.Transformers_TwoWings.transformer_TwoWings import Tr2W
from UI.Main.Elements.PWM.pwm import PWM
from UI.Main.Elements.DCDC_converter.dcdc_conv import DCDC_conv
from UI.Main.Elements.ACwind.acwind import ACWind
from UI.Main.Elements.DCwind.dcwind import DCWind
from UI.Main.Elements.ACLine.acline import ACLine
from UI.Main.Elements.DCLine.dcline import DCLine
from UI.Main.Elements.ACnode.acnode import ACnode
from UI.Main.Elements.DCnode.dcnode import DCnode
from UI.Main.Elements.PV.pv import PV
from UI.Main.Elements.ExtGrid.extgrid import ExtGrid
from UI.Main.Elements.Battery.battery import Battery

from UI._general.YearProfile.yearprofile import YearProfile

from UI._general.Popup.Reliability.popup_rel import PopupRel
from UI._general.Popup.EMS_Scenarios.popup_EMS_scen import PopupEMSscen

from UI._general.Profile.profile import Profile
from UI.Main.EMS.ems_results import EmsResults
from UI.Main.EMS_stoch.ems_stoch_results import EmsReliabilityResults
from UI.Main.Reliability.reliability_results import ReliabilityResults

from __shared__ import variables as v

from functools import partial

import datetime as dt
import copy


class Main:
    def __init__(self):
        self.application = QtWidgets.QApplication([])

        # inizializzazione variabili
        self.ID = ""
        self.element = None
        self.to_set_params = []

        self.node_types = ['AC-Node', 'DC-Node']
        self.link_types = ['AC-Line', 'DC-Line', 'DC-DC_Conv', 'PWM', '2W-Transformer']

        self.ems_dict = dict()
        self.project = None
        self.software = None
        v.soft_check = False
        self.UI = None
        self.current_profile = None
        self.ems_img_dir = None
        self.ems_res_wg = None
        self.rel_img_dir = None
        self.rel_res_wg = None
        self.tp_WG = None
        self.prot_res_wg = None
        self.current_widget = None

        self.canvas = FigureCanvas(plt.Figure(figsize=(15, 6)))
        self.ax = self.canvas.figure.subplots()
        self.line, = self.ax.plot([0], [0])

        self.start()

    # Azzeramento variabili globali
    def clear_vars(self):
        self.application.closeAllWindows()  # Chiudo tutte le applicazioni aperte
        self.application.quit()             # Termino l'esecuzione delle applicazioni

        v.functionality = None

        v.elements = dict()
        v.ext_grid = None

        v.temperature = dict()
        v.temperature['name'] = None
        v.temperature['profile'] = None

        v.neplan_connections = dict()
        v.neplan_extgrid = ''

        v.T_vector = []

        v.features = dict()
        v.ilf = False
        v.plf = False
```

```
        v.ems = False
        v.ems2 = False
        v.reliability = False
        v.protections = False
        v.executed = []

        v.next_ml = False

    #
    def start(self):
        self.ems_dict = dict()

        # Attivazione della Splash screen
        if not v.next_ml:                       # Verifica se sono selezionate logiche multiple
            select = self.splash_screen()
        elif v.features['pf']['exists']:        # Verifica se PowerFactory è disponibile
            select = 'pf'
        elif v.features['neplan']['exists']:    # Verifica se Neplan è disponibile
            select = 'neplan'
        else:
            select = None

        try:
            if v.functionality == 'Controls':
                self.UI = Controls()
                self.UI.setWindowFlags(QtCore.Qt.WindowCloseButtonHint | QtCore.Qt.WindowMinimizeButtonHint)
                self.UI.showMaximized()
                self.application.exec()

                # una volta che l'applicazione è chiusa...
                if not v.next_ml:
                    self.clear_vars()
                self.start()

            elif v.functionality is not None:
                v.next_ml = False
                self.project = v.features[select]['project_name']

                # Instanziamento della classe Software
                self.init_software(self.project, select)

                # Inizializzazione dell'interfaccia principale
                self.UI = UI()
                self.UI.ui.statusbar.hide()
                self.UI.ui.gridname_LBL.setText(v.features['name'])

                # Se si è selezionato il Calcolo dell'affidabilità, mostrare la possibilità di settare la temperatura
                self.UI.ui.temp_WGT.setVisible(v.functionality == 'Reliability')

                self.UI.ui.plf_BTN.setVisible(False)
                self.UI.ui.ems_BTN.setVisible(False)
                self.UI.ui.ems_BTN_2.setVisible(False)
                self.UI.ui.reliability_BTN.setVisible(False)

                self.UI.setWindowFlags(QtCore.Qt.WindowCloseButtonHint | QtCore.Qt.WindowMinimizeButtonHint)
                self.UI.showMaximized()
                v.model_error = False

                self.items_table_fill()

                # Importazione dei dati di rete precedentemente salvati
                self.get_data()

                # Definizione dell'azione dei pulsanti e degli elementi
                self.UI.ui.tableWidget.currentCellChanged.connect(self.element_selected)
                self.UI.ui.ilf_BTN.clicked.connect(self.results)
                self.UI.ui.plf_BTN.clicked.connect(self.prof_results)
                self.UI.ui.saveParams_BTN.clicked.connect(partial(self.store_attributes))
                self.UI.ui.temp_prof_BTN.clicked.connect(self.show_temp_profile)
                self.UI.ui.temp_prof_canc_BTN.clicked.connect(self.cancel_T_profile)

                self.UI.ui.lm_protections_BTN.clicked.connect(self.ml_protections)
                self.UI.ui.lm_reliability_BTN.clicked.connect(self.ml_reliability)
                self.UI.ui.pdf_BTN.clicked.connect(self.pdf_gen)

                self.func_check()

                self.UI.ui.ilf_BTN.setFocus()

                self.application.exec()

                # una volta che l'interfaccia principale è chiusa...
                self.clear_vars()
                self.start()
        except Exception as e:
            self.clear_vars()
            v.model_error = True
            self.start()

    # Definizione dell'azione del pulsante di calcolo per ogni funzionalità
    def func_check(self):
        try:    # Eliminare eventuali azioni sul pulsante
            self.UI.ui.calculate_BTN.clicked.disconnect()
        except: pass

        if v.functionality == 'Reliability':
            self.UI.ui.calculate_BTN.clicked.connect(self.rel_calc)
            self.UI.ui.calculate_BTN.setText("Calcolo dell'affidabilità")
        elif v.functionality == 'EMS':
            self.UI.ui.calculate_BTN.clicked.connect(self.ems_execute)
            self.UI.ui.calculate_BTN.setText("Ottimizzazione EMS")
        elif v.functionality == 'Energy Indexes':
            self.UI.ui.calculate_BTN.clicked.connect(self.ems_stoich)
            self.UI.ui.calculate_BTN.setText("Calcolo degli indici")
        elif v.functionality == 'Protections':
            self.UI.ui.calculate_BTN.clicked.connect(self.protections_calc)
            self.UI.ui.calculate_BTN.setText("Calcolo delle Protezioni")
        self.UI.ui.multiplelogics_WGT.setVisible(False)     # Nasconde le azioni delle logiche multiple

    # Definizione dei pulsanti di azione delle logiche multiple
    def ml_check(self):
        prot_view = 'prot' not in v.executed and v.functionality != 'Protections' and (v.features['pf']['exists'] or
                                                                                       v.features['neplan']['exists'])
        rel_view = 'rel' not in v.executed and v.functionality != 'Reliability' and v.features['pf']['exists']
        self.UI.ui.lm_protections_BTN.setVisible(prot_view)
```

```
        self.UI.ui.lm_reliability_BTN.setVisible(rel_view)
        self.UI.ui.multiplelogics_WGT.setVisible(rel_view or prot_view)

    def ml_reliability(self):
        v.functionality = 'Reliability'
        self.func_check()
        self.UI.ui.temp_WGT.setVisible(v.functionality == 'Reliability')

    def ml_protections(self):
        v.functionality = 'Protections'
        self.func_check()
        # self.ml_check()

    # Imposta il software da utilizzare
    def init_software(self, project, select):
        try:
            del self.software
        except: pass
        if select == 'neplan':
            from Softwares.Neplan.neplan import Neplan
            self.software = Neplan(project)
        else:
            from Softwares.PowerFactory.pf_class import PowerFactory
            self.software = PowerFactory(project)

        # Inizializzazione del sotto-dizionario "EMS" e "Protections", e delle connessioni degli elementi
        try:
            links = ['AC-Line', 'DC-Line', 'PWM', 'DC-DC_Conv', '2W-Transformer']
            elems = ['AC-Load', 'DC-Load', 'AC-Wind', 'DC-Wind', 'PV']
            base_dict = yaml.safe_load(open(os.getcwd() + '/__shared__/attributes_template.yml'))
            for element in v.elements:
                v.elements[element]['ems'] = copy.deepcopy(base_dict[v.elements[element]['category']]['ems'])
                try:
                    v.elements[element]['protections'] = \
                        copy.deepcopy(base_dict[v.elements[element]['category']]['protections'])
                except Exception:
                    pass
                if v.elements[element]['category'] in links:
                    self.ems_connections(element)
                elif v.elements[element]['category'] in elems:
                    self.ems_profile(element)
        except:
            print('errore generato2')

        pass

    # Definizione delle connessioni degli elementi
    def ems_connections(self, element):
        v.elements[element]['ems']['in'] = v.elements[element]['conn']['h']
        conn_list = list(v.elements[element]['conn'].keys())
        conn_list.remove('h')
        conn_list.remove(v.elements[element]['ems']['in'])
        v.elements[element]['ems']['out'] = conn_list[0]

    # Definizione del profilo di carico o di generazione:
    # crea il vettore del profilo in EMS->profile
    def ems_profile(self, element):
        profile = []
        try:
            value = v.elements[element]['parameters']['profile']['curve']
        except:
            value = 1

        for i in range(0, 96):
            profile.append(value)
        for cat in v.elements[element]['ems']['profile']:
            v.elements[element]['ems']['profile'][cat] = profile

    # DA ELIMINARE
    def complete_dict(self):
        base_dict = yaml.safe_load(open(os.getcwd() + '/__shared__/attributes_template.yml'))
        for element in v.elements:
            v.elements[element]['ems'] = copy.deepcopy(base_dict[v.elements[element]['category']]['ems'])
            v.elements[element]['results'] = dict()

    # Recupero dati salvati del modello di rete
    def get_data(self):
        # se esiste una cartella con i dati, importarne i dati
        if os.path.exists(os.path.join(v.project_folder, v.software)):
            v.project_folder = os.path.join(v.project_folder, v.software)
            self.load_attributes()  # Caricamento degli attributi
            pass

        else:   # Se la cartella non esiste, ne creiamo una ed inseriamo i dati
            try:
                v.features['T_profile'] = None
                if not os.path.exists(v.project_folder):
                    os.mkdir(v.project_folder)
                v.project_folder = os.path.join(v.project_folder, v.software)
                os.mkdir(v.project_folder)
            except FileExistsError:
                pass
            self.store_attributes()

        # Inizialmente, i parametri di tutti gli elementi sono da impostare
        self.to_set_params = list(v.elements.keys())
        self.temperature_widget()   # Inizializzazione widget temperatura

    #
    def temperature_widget(self):
        if v.features['T_profile'] is not None:
            self.UI.ui.temp_prof_LBL.setText(v.features['T_profile'])
            self.UI.ui.temp_prof_BTN.setText('Vedi')
            self.UI.ui.temp_prof_canc_BTN.setVisible(True)
        else:
            self.UI.ui.temp_prof_LBL.setText('-- nessun profilo --')
            self.UI.ui.temp_prof_BTN.setText('Imposta')
            self.UI.ui.temp_prof_canc_BTN.setVisible(False)

    # Caricamento della temperatura
    def load_temperature(self):
        print('caricamento del profilo di Temperatura in corso... (questa operazione potrebbe richiedere molti secondi)')
        filename = os.path.join(v.project_folder, 'temperature.yml')
        v.temperature = yaml.safe_load(open(filename))
        print('caricamento della temperatura completato')
```

```
# Salvataggio della temperatura
def save_temperature(self):
    print('Salvataggio del profilo di Temperatura in corso... '
          '(questa operazione potrebbe richiedere molti secondi)')
    filename = os.path.join(v.project_folder, 'temperature.yml')
    with open(filename, 'w') as file:
        documents = yaml.dump(v.temperature, file)
    print('Salvataggio della temperatura completato\n')

    self.UI.ui.temp_prof_LBL.setText(v.features['T_profile'])

    filename = os.path.join(v.project_folder, 'features.yml')
    with open(filename, 'w') as file:
        documents = yaml.dump(v.features, file)

# Caricamento degli attributi elements, features e rbd
def load_attributes(self):
    v.elements = yaml.safe_load(open(os.path.join(v.project_folder, 'elements.yml')))
    try:
        v.features = yaml.safe_load(open(os.path.join(v.project_folder, 'features.yml')))
        v.temperature['name'] = v.features['T_profile']
    except FileNotFoundError:
        v.features['T_profile'] = None

    try:
        v.rbd = yaml.safe_load(open(os.path.join(v.project_folder, 'rbd.yml')))
    except FileNotFoundError:
        v.rbd = {}

# Salvataggio degli attributi elements e features
def store_attributes(self):
    filename = os.path.join(v.project_folder, 'elements.yml')
    with open(filename, 'w') as file:
        documents = yaml.dump(v.elements, file)
    file.close()

    filename = os.path.join(v.project_folder, 'features.yml')
    with open(filename, 'w') as file:
        documents = yaml.dump(v.features, file)
    file.close()

# creazione del dizionario "v.elements" dei risultati
def results(self):
    v.ilf = False
    v.plf = False
    v.ems = False
    v.ems2 = False
    v.reliability = False

    now = dt.datetime.now()
    self.UI.ui.log_TBR.append(now.strftime("%H:%M:%S") + ': Calcolo del load flow in corso...\n')
    hour = int((now.hour + now.minute/60)*4)/4

    # Invio dei parametri ai software esterni
    for element in self.to_set_params:
        try:
            self.software.set_params(element)
            log = "Caricamento dei parametri dell'elemento "
        except:
            log = "Errore nel caricamento dei parametri dell'elemento " + element
        self.log_write_wtime(log)
    self.to_set_params = []

    try:
        results_dict = copy.deepcopy(self.software.results(hour))     # creo il dizionario "v.elements"
        try:    # Lettura della potenza assorbita dalla rete esterna
            p_grid = v.elements[v.ext_grid]['results']['P']
            self.UI.ui.log_TBR.append('\n' + now.strftime("%H:%M:%S") + ": Potenza assorbita dala rete = %.3f kW"
                                      % p_grid)
        except:
            self.UI.ui.log_TBR.append('\n' + now.strftime("%H:%M:%S") + ': Potenza della Grid non disponibile')

        # Verifica degli elementi in errore
        elem_error = []
        limit_violated = False
        for elem in v.elements:
            if v.elements[elem]['results'] != {}:
                if v.elements[elem]['results']['LimitViolated']:
                    limit_violated = True
                    elem_error.append(elem)
        log = ''
        if limit_violated:
            for elem in elem_error:
                log = 'Limiti violati in ' + elem + '\n'
        else:
            log = 'Nessuna violazione dei limiti\n'
        self.UI.ui.log_TBR.append('\n' + now.strftime("%H:%M:%S") + ': ' + log)

        v.ilf = True
        v.executed.append('lf')
    except:
        self.UI.ui.log_TBR.append('\n' + now.strftime("%H:%M:%S") + ': Errore nel modello di rete. Verificare.\n')

# DA ELIMINARE
def prof_results(self):
    print('Profile Results')
    self.software.prof_results()
    print('Done')

# Imposta i valori di set per l'elemento selezionato, e calcola i risultati
def store(self):
    self.current_widget.store()
    prof = []
    if 'profile' in v.elements[self.element]['parameters'].keys():
        if self.current_widget.ui.sf_const_RB.isChecked():
            v.elements[self.element]['parameters']['profile']['curve'] = self.current_widget.ui.sf_DSB.value()
            v.elements[self.element]['parameters']['profile']['constant'] = True
            v.elements[self.element]['parameters']['profile']['name'] = 'constant'
            for i in range(0, 96):
                prof.append(self.current_widget.ui.sf_DSB.value())
        elif self.current_profile.profile[0] is not None:
            v.elements[self.element]['parameters']['profile']['curve'] = self.current_profile.profile
            v.elements[self.element]['parameters']['profile']['constant'] = False
            v.elements[self.element]['parameters']['profile']['name'] = self.current_profile.ui.profile_LBL.text()
```

```
            prof = self.current_profile.profile
        if 'profile' in v.elements[self.element]['ems'].keys():
            for elem_type in v.elements[self.element]['ems']['profile']:
                # if v.elements[self.element]['category'] == 'AC-Load':
                if v.elements[self.element]['category'] != 'Ext-Grid':
                    v.elements[self.element]['ems']['profile'][elem_type] = self.current_widget.calc_profile(prof)

        self.to_set_params.append(self.element)

    #
    def par_close(self):
        print('CLOSING...')
        self.UI.ui.stackedWidget.removeWidget(self.UI.ui.stackedWidget.currentWidget())
        self.UI.ui.profile_SW.removeWidget(self.UI.ui.profile_SW.currentWidget())

    # Popola la tabella degli elementi della rete
    def items_table_fill(self):
        self.UI.ui.tableWidget.clear()
        self.UI.ui.tableWidget.setHorizontalHeaderItem(0, QtWidgets.QTableWidgetItem('Nome elemento'))
        self.UI.ui.tableWidget.setHorizontalHeaderItem(1, QtWidgets.QTableWidgetItem('Categoria'))

        self.UI.ui.tableWidget.setRowCount(len(v.elements))
        i = 0
        for elem in v.elements:
            self.UI.ui.tableWidget.setItem(i, 0, QtWidgets.QTableWidgetItem(elem))
            self.UI.ui.tableWidget.setItem(i, 1, QtWidgets.QTableWidgetItem(v.elements[elem]['category']))
            i += 1
        self.UI.table_format()

    # Popola la finestra delle proprietà dell'elemento
    def element_selected(self):
        self.current_widget = None

        line = self.UI.ui.tableWidget.currentIndex().row()
        self.element = self.UI.ui.tableWidget.item(line, 0).text()
        cat = self.UI.ui.tableWidget.item(line, 1).text()

        if cat == 'AC-Load':
            self.current_widget = ACLoad(self.element)
        elif cat == 'DC-Load':
            self.current_widget = DCLoad(self.element)
        elif cat == '2W-Transformer':
            self.current_widget = Tr2W(self.element)
        elif cat == 'PWM':
            self.current_widget = PWM(self.element)
        elif cat == 'DC-DC_Conv':
            self.current_widget = DCDC_conv(self.element)
        elif cat == 'AC-Wind':
            self.current_widget = ACwind(self.element)
        elif cat == 'DC-Wind':
            self.current_widget = DCwind(self.element)
        elif cat == 'AC-Line':
            self.current_widget = ACLine(self.element)
        elif cat == 'DC-Line':
            self.current_widget = DCLine(self.element)
        elif cat == 'Battery':
            self.current_widget = Battery(self.element)
        elif cat == 'AC-Node':
            self.current_widget = ACnode(self.element)
        elif cat == 'DC-Node':
            self.current_widget = DCnode(self.element)
        elif cat == 'PV':
            self.current_widget = PV(self.element)
        elif cat == 'Ext-Grid':
            self.current_widget = ExtGrid(self.element)

        self.UI.ui.profile_SW.removeWidget(self.UI.ui.profile_SW.currentWidget())
        if 'profile' in v.elements[self.element]['parameters'].keys():
            self.current_widget.ui.sf_const_RB.setChecked(v.elements[self.element]['parameters']['profile']
                                                          ['constant'])
            self.current_widget.ui.sf_profile_RB.setChecked(not self.current_widget.ui.sf_const_RB.isChecked())
            self.init_profiles()
            self.current_widget.ui.sf_const_RB.clicked.connect(self.switch_profiles)
            self.current_widget.ui.sf_profile_RB.clicked.connect(self.switch_profiles)

        try:
            for i in range(1, 5):
                self.current_widget.ui.tabWidget.setTabVisible(i, False)

            self.current_widget.ui.tabWidget.setTabVisible(1, v.ilf)
            self.current_widget.ui.tabWidget.setTabVisible(2, v.functionality == 'EMS' or
                                                           v.functionality == 'Energy Indexes')
            self.current_widget.ui.tabWidget.setTabVisible(3, v.functionality == 'Reliability')
            self.current_widget.ui.tabWidget.setTabVisible(4, v.protections and
                                                           v.elements[self.element]['protections'] != {})

            if v.functionality == 'Reliability':
                self.current_widget.ui.tabWidget.setCurrentIndex(3)
            elif v.functionality == 'EMS':
                self.current_widget.ui.tabWidget.setCurrentIndex(2)
            elif v.functionality == 'Energy Indexes':
                self.current_widget.ui.tabWidget.setCurrentIndex(2)
            elif v.protections and v.elements[self.element]['protections']['results'] != {}:
                self.current_widget.ui.tabWidget.setCurrentIndex(4)
            elif v.ilf:
                self.current_widget.ui.tabWidget.setCurrentIndex(1)
        except:
            try:
                if v.ilf:
                    self.current_widget.ui.tabWidget.setCurrentIndex(1)
            except: pass

        self.UI.ui.stackedWidget.removeWidget(self.UI.ui.stackedWidget.currentWidget())
        if self.current_widget:
            try:
                self.current_widget.ui.control_CB.currentIndexChanged.connect(partial(self.current_widget.control_mode,
                                                                                       self.current_widget.ui))
            except: pass

            self.UI.ui.stackedWidget.addWidget(self.current_widget.ui.widget)
            self.UI.ui.stackedWidget.setCurrentIndex(0)
            self.current_widget.ui.store_BTN.clicked.connect(self.store)  # inizializzazione del pulsante SET
            self.current_widget.ui.cancel_BTN.clicked.connect(self.par_close)

    #
```

```
def tab_widget_changed(self):
    self.UI.ui.profile_SW.removeWidget(self.UI.ui.profile_SW.currentWidget())

# Inizializzazione dei profili
def init_profiles(self):
    self.current_widget.ui.sf_DSB.setEnabled(self.current_widget.ui.sf_const_RB.isChecked())
    self.UI.ui.profile_SW.removeWidget(self.UI.ui.profile_SW.currentWidget())
    if self.current_widget.ui.sf_const_RB.isChecked():
        self.current_widget.ui.sf_DSB.setValue(v.elements[self.element]['parameters']['profile']['curve'])
        self.current_widget.ui.sf_DSB.setStyleSheet("color: rgb(255, 255, 255);")
    else:
        self.current_widget.ui.sf_DSB.setStyleSheet("color: rgb(127, 127, 127);")
        self.current_profile = Profile(v.elements[self.element]['parameters']['profile']['name'],
                                       v.elements[self.element]['parameters']['profile']['curve'], 'p.u.')
        self.UI.ui.profile_SW.addWidget(self.current_profile)
        self.UI.ui.profile_SW.setCurrentIndex(0)

#
def switch_profiles(self):
    self.UI.ui.profile_SW.removeWidget(self.UI.ui.profile_SW.currentWidget())
    self.current_widget.ui.sf_DSB.setEnabled(self.current_widget.ui.sf_const_RB.isChecked())
    if self.current_widget.ui.sf_const_RB.isChecked():
        self.current_widget.ui.sf_DSB.setStyleSheet("color: rgb(255, 255, 255);")
        if type(v.elements[self.element]['parameters']['profile']['curve']) is float:
            self.current_widget.ui.sf_DSB.setValue(v.elements[self.element]['parameters']['profile']['curve'])
        else:
            self.current_widget.ui.sf_DSB.setValue(1)
    else:
        self.current_widget.ui.sf_DSB.setStyleSheet("color: rgb(127, 127, 127);")
        if type(v.elements[self.element]['parameters']['profile']['curve']) is list:
            self.current_profile = Profile(v.elements[self.element]['parameters']['profile']['name'],
                                           v.elements[self.element]['parameters']['profile']['curve'], 'p.u.')
        else:
            prof = []
            for i in range(0, 96):
                prof.append(v.elements[self.element]['parameters']['profile']['curve'])
            self.current_profile = Profile('From File', prof, 'p.u.')
        self.UI.ui.profile_SW.addWidget(self.current_profile)
        self.UI.ui.profile_SW.setCurrentIndex(0)

# Richiamo della splash screeen e acquisizione del nome del progetto da attivare
def splash_screen(self):
    my_splash = Splash()
    my_splash.show()

    if v.model_error:
        msg = QtWidgets.QMessageBox(QtWidgets.QMessageBox.Information, 'Errore modello di rete',
                                    'Il modello di rete risulta difforme dai parametri necessari.\n'
                                    'Si prega di consultare il manuale utente.')
        msg.exec_()

    self.application.exec()
    self.application.closeAllWindows()
    self.application.quit()
    return v.software

#
def show_temp_profile(self):
    if v.temperature['name']:
        if not v.temperature['profile']:
            if os.path.exists(os.path.join(v.project_folder, 'temperature.yml')):
                self.load_temperature()
    else:
        v.temperature['profile'] = None

    self.UI.ui.stackedWidget.removeWidget(self.UI.ui.stackedWidget.currentWidget())
    self.UI.ui.profile_SW.removeWidget(self.UI.ui.profile_SW.currentWidget())
    self.tp_WG = YearProfile(v.temperature['name'], v.temperature['profile'])
    self.UI.ui.profile_SW.addWidget(self.tp_WG)
    self.tp_WG.ui.confirm_BTN.clicked.connect(self.confirm_T_profile)
    self.tp_WG.ui.exit_BTN.clicked.connect(self.exit_T_profile)
    # self.tp_WG.ui.confirm_BTN.clicked.connect(self.save_temperature)
    pass

#
def confirm_T_profile(self):
    self.UI.ui.log_TBR.append('Salvataggio della temperatura in corso...')  # Non viene mostrato immediatamente

    v.T_vector = []
    v.temperature['name'] = self.tp_WG.name
    v.temperature['profile'] = self.tp_WG.profile
    self.set_T_profile()
    self.create_T_vector()
    self.save_temperature()
    self.temperature_widget()
    self.UI.ui.log_TBR.append('Salvataggio della temperatura completato')

#
def exit_T_profile(self):
    self.UI.ui.profile_SW.removeWidget(self.UI.ui.profile_SW.currentWidget())

#
def create_T_vector(self):
    prof_error = False
    for m in v.temperature['profile']:
        for d in v.temperature['profile'][m]['prof']:
            d_av = v.temperature['profile'][m]['prof'][d]['av']
            p_day = False
            for h in v.temperature['profile'][m]['prof'][d]['prof']:
                v.T_vector.append(v.temperature['profile'][m]['prof'][d]['prof'][h])
                if d_av != v.temperature['profile'][m]['prof'][d]['prof'][h]:
                    p_day = True
            if not p_day and not prof_error:
                prof_error = True

#
def cancel_T_profile(self):
    v.temperature = dict()
    v.temperature['name'] = None
    v.temperature['profile'] = None
    try:
        os.remove(v.project_folder + '/temperature.yml')
    except: pass
    self.set_T_profile()
    self.temperature_widget()
```

```
#
def set_T_profile(self):
    v.features['T_profile'] = v.temperature['name']
    filename = os.path.join(v.project_folder, 'features.yml')
    with open(filename, 'w') as file:
        documents = yaml.dump(v.features, file)

    if v.temperature['name']:
        self.UI.ui.temp_prof_LBL.setText(v.temperature['name'])
    self.UI.ui.profile_SW.removeWidget(self.UI.ui.profile_SW.currentWidget())

def ems_dict_compile(self):
    ems_dict_correct()

    self.ems_dict['parameters'] = dict()
    self.ems_dict['parameters']['time_res'] = 0.25
    self.ems_dict['parameters']['n_steps'] = 96
    self.ems_dict['nodes'] = dict()
    self.ems_dict['links'] = dict()

    for element in v.elements.keys():
        if v.elements[element]['category'] in self.node_types:
            self.ems_dict['nodes'][element] = dict()
            self.ems_dict['nodes'][element]['techs'] = dict()
            connections = list(v.elements[element]['conn'].keys())
            connections.remove('h')
            for conn in connections:
                # print(conn)
                if conn in v.elements.keys():
                    if v.elements[conn]['category'] not in self.link_types:
                        self.ems_dict['nodes'][element]['techs'][conn] = v.elements[conn]['ems']

        elif v.elements[element]['category'] in self.link_types:
            self.ems_dict['links'][element] = v.elements[element]['ems']

    filename = os.path.join(v.project_folder, 'ems.yml')
    with open(filename, 'w') as file:
        documents = yaml.dump(self.ems_dict, file)
    file.close()

    ems = EMS(self.ems_dict)
    return ems

#
def ems_execute(self):
    try:
        self.UI.ui.log_TBR.clear()
        self.log_write_wtime('Inizio calcolo EMS', 1)
        ems = self.ems_dict_compile()
        ems_path = os.path.join(v.project_folder, 'EMS', 'results')
        ems.execute_EMS(generate_plots=True, output_folder=ems_path)
        self.log_write_wtime('Calcolo EMS completo')
        v.ems = True
        v.executed.append('ems')
        self.ml_check()
        self.ems_show_results()
    except Exception as e:
        msg = QtWidgets.QMessageBox(QtWidgets.QMessageBox.Information,
                                    'Errore inatteso', 'Si è creato un errore nella procedura\n\n\n' + str(e))
        msg.exec_()
        self.UI.ui.log_TBR.clear()

#
def ems_show_results(self):
    self.UI.ui.results_WG.removeWidget(self.UI.ui.results_WG.currentWidget())
    self.ems_res_wg = EmsResults()
    self.UI.ui.results_WG.addWidget(self.ems_res_wg.ui.widget)

    self.ems_res_wg.ui.comboBox.clear()
    self.ems_img_dir = os.path.join(v.project_folder, 'EMS', 'results')
    if os.path.exists(self.ems_img_dir):
        for file in os.listdir(self.ems_img_dir):
            if file.endswith('.png'):
                self.ems_res_wg.ui.comboBox.addItem(file)

    file_path = os.path.join(self.ems_img_dir, self.ems_res_wg.ui.comboBox.currentText())

    image = QtGui.QPixmap(file_path)

    self.ems_res_wg.ui.label.setPixmap(image)
    self.ems_res_wg.ui.comboBox.currentTextChanged.connect(self.ems_res_update)

#
def ems_res_update(self):
    file_path = os.path.join(self.ems_img_dir, self.ems_res_wg.ui.comboBox.currentText())
    image = QtGui.QPixmap(file_path)
    self.ems_res_wg.ui.label.setPixmap(image)

#
def ems_stoich(self):
    try:
        ems_fault_scen = dict()
        popup = PopupEMSscen(self.project)

        self.UI.ui.log_TBR.clear()
        self.log_write_wtime('Inizio calcolo degli EMS Indexes', 1)

        if popup.exec_():
            ems_fault_scen = popup.get_value()
            popup.close()

        if popup.is_confirmed:
            self.log_write_wtime('Calcolo degli EMS in corso...')
            self.log_write("NB: L'operazione può richiedere diversi minuti", 1)
            ems = self.ems_dict_compile()

            reliability_indices_ENEA, reliability_indices, rel_ind_raw = \
                ems.execute_reliability_analysis(generate_plots=False, fault_scenarios=ems_fault_scen)
            self.log_write_wtime('Calcolo degli EMS Indexes Completato', 2)

            filename = os.path.join(v.project_folder, 'ems_fault_results_ENEA.yml')
            with open(filename, 'w') as file:
                documents = yaml.dump(reliability_indices_ENEA, file)
            file.close()
```

```
            filename = os.path.join(v.project_folder, 'ems_fault_results.yml')
            with open(filename, 'w') as file:
                documents = yaml.dump(reliability_indices, file)
            file.close()
            v.ems2 = True
            v.executed.append('ei')
            self.ml_check()
            self.ems_stoch_show_results()
        except Exception as e:
            msg = QtWidgets.QMessageBox(QtWidgets.QMessageBox.Information,
                                        'Errore inatteso', 'Si è creato un errore nella procedura\n\n\n' + str(e))
            msg.exec_()
            self.UI.ui.log_TBR.clear()

    #
    def ems_stoch_show_results(self):
        self.UI.ui.results_WG.removeWidget(self.UI.ui.results_WG.currentWidget())
        self.rel_res_wg = EmsReliabilityResults()
        self.UI.ui.results_WG.addWidget(self.rel_res_wg.ui.widget)

    #
    def rel_calc(self):
        self.UI.ui.log_TBR.clear()
        self.log_write_wtime("Inizio Calcolo dell'Affidabilità", 1)
        v.ilf = False
        v.plf = False
        v.ems = False
        v.ems2 = False
        v.reliability = False
        rel_ok = False
        rel_time = 0
        rel_day = 1
        rel_month = 1
        months = ['Gen', 'Feb', 'Mar', 'Apr', 'Mag', 'Giu', 'Lug', 'Ago', 'Set', 'Ott', 'Nov', 'Dic']

        folder = os.path.join(v.project_folder, 'reliability')
        if not os.path.exists(folder):
            os.mkdir(folder)
        folder = os.path.join(folder, 'img')
        if not os.path.exists(folder):
            os.mkdir(folder)

        t_max = 438000
        for element in v.elements:
            try:
                t_max = min(v.elements[element]['reliability']['alfa'], t_max)
            except:
                pass

        popup = PopupRel(t_max)

        if popup.exec_():
            rel_ok, rel_time, rel_day, rel_month = popup.get_value()

        if rel_ok:
            filename = os.path.join(v.project_folder, 'temperature.yml')
            if v.temperature['profile'] is None and v.features['T_profile'] is not None:
                self.log_write_wtime('Caricamento del profilo di temperatura in corso...')
                v.temperature = yaml.safe_load(open(filename))

            if v.temperature['profile'] is not None:
                Ta = list(v.temperature['profile'][months[rel_month-1]]['prof'][rel_day]['prof'].values())
                self.log_write_wtime('Profilo di temperatura caricato', 1)

                self.log_write_wtime('Inizio fase di calcolo', 0)
                self.log_write('Questa operazione potrebbe richiedere diversi minuti', 1)

                try:
                    self.software.reliability(rel_time, Ta)
                    self.log_write_wtime("Calcolo dell'affidabilità terminato", 2)
                    filename = os.path.join(v.project_folder, 'rbd.yml')
                    with open(filename, 'w') as file:
                        documents = yaml.dump(v.rbd, file)
                    file.close()

                    filename = os.path.join(v.project_folder, 'elements.yml')
                    with open(filename, 'w') as file:
                        documents = yaml.dump(v.elements, file)
                    file.close()

                    v.reliability = True
                    v.executed.append('rel')
                    self.ml_check()
                    self.reliability_show_results(rel_time)
                except:
                    self.log_write_wtime('Errore di calcolo')
                    msg = QtWidgets.QMessageBox(QtWidgets.QMessageBox.Information, 'Calcolo non riuscito',
                                                'Si è verificato un errore durante la procedura di calcolo')
                    msg.exec_()
            else:
                msg = QtWidgets.QMessageBox(QtWidgets.QMessageBox.Information,
                                            'Errore Temperatura', 'Nessun profilo di temperatura caricato')
                msg.exec_()

    #
    def reliability_show_results(self, rel_time):
        self.UI.ui.results_WG.removeWidget(self.UI.ui.results_WG.currentWidget())
        self.rel_res_wg = ReliabilityResults()
        self.UI.ui.results_WG.addWidget(self.rel_res_wg.ui.widget)

        self.rel_res_wg.ui.load_time_DSB.setValue(rel_time)
        self.rel_res_wg.ui.comp_rel_TW.clear()
        self.rel_res_wg.ui.loads_rel_TW.clear()
        self.rel_img_dir = os.path.join(v.project_folder, 'reliability', 'img')

        # Formattazione della tabella dei risultati
        self.rel_res_wg.ui.comp_rel_TW.setHorizontalHeaderLabels(['Elemento', 'Lambda', 'R', 'MTBF_ore', 'MTBF_anni'])
        stylesheet_comp = "QHeaderView::section{color:rgb(196,196,196); " \
                          "Background-color:rgb(1,1,1); border - radius: 14 px; font-style: italic}"
        self.rel_res_wg.ui.comp_rel_TW.horizontalHeader().setStyleSheet(stylesheet_comp)
        sizes = [150, 100, 100, 100, 100]
        for c in range(0, 5):
            self.rel_res_wg.ui.comp_rel_TW.setColumnWidth(c, sizes[c])
```

```
    # Formattazione della tabella dell'affidabilità delle forniture
    self.rel_res_wg.ui.loads_res_WGT.setVisible(False)
    self.rel_res_wg.ui.loads_rel_TW.setHorizontalHeaderLabels(['Elemento', 'R(t)'])
    stylesheet_loads = "QHeaderView::section{color:rgb(196,196,196); " \
                       "Background-color:rgb(1,1,1); border - radius: 14 px; font-style: italic}"
    self.rel_res_wg.ui.loads_rel_TW.horizontalHeader().setStyleSheet(stylesheet_loads)
    sizes = [250, 100]
    for c in range(0, 1):
        self.rel_res_wg.ui.loads_rel_TW.setColumnWidth(c, sizes[c])
    self.rel_res_wg.ui.loads_rel_TW.setColumnWidth(0, 180)
    self.rel_res_wg.ui.loads_rel_TW.setColumnWidth(1, 100)
    self.rel_res_wg.ui.loads_grafo_BTN.clicked.connect(self.rel_grafo_show)

    # Inserimento dei valori nella tabella dei risultati
    for element in v.elements:
        if v.elements[element]['category'] not in ['AC-Node', 'DC-Node', 'Extc-Grid']:
            i = self.rel_res_wg.ui.comp_rel_TW.rowCount()
            self.rel_res_wg.ui.comp_rel_TW.setRowCount(i+1)
            self.rel_res_wg.ui.comp_rel_TW.setItem(i, 0, QtWidgets.QTableWidgetItem(element))
            c = 1
            for index in ['lambda', 'R', 'MTBF_ore', 'MTBF_anni']:

                # formattazione del numero in funzione del suo valore
                d = v.elements[element]['reliability']['results'][index]
                if d == 0:
                    d_str = '0'
                elif d < 0.01:
                    d_str = '%.4e' % d
                elif d < 1:
                    d_str = '%.6f' % d
                else:
                    d_str = '%.2f' % d
                self.rel_res_wg.ui.comp_rel_TW.setItem(i, c, QtWidgets.QTableWidgetItem(d_str))
                self.rel_res_wg.ui.comp_rel_TW.item(i, c).setTextAlignment(QtCore.Qt.AlignCenter)
                c += 1

            if v.elements[element]['category'] in ['AC-Load', 'DC-Load']:
                i = self.rel_res_wg.ui.loads_rel_TW.rowCount()
                self.rel_res_wg.ui.loads_rel_TW.setRowCount(i+1)
                self.rel_res_wg.ui.loads_rel_TW.setItem(i, 0, QtWidgets.QTableWidgetItem(element))
                if v.elements[element]['reliability']['results']['load_rel'] < 0.004:
                    rel = '%.4e' % v.elements[element]['reliability']['results']['load_rel']
                else:
                    rel = '%.6f' % v.elements[element]['reliability']['results']['load_rel']
                self.rel_res_wg.ui.loads_rel_TW.setItem(i, 1, QtWidgets.QTableWidgetItem(rel))

    self.rel_res_wg.ui.loads_rel_TW.currentCellChanged.connect(self.reliability_res_update)
    self.rel_res_wg.ui.comp_rel_TW.currentCellChanged.connect(self.reliability_comp_selected)

#
def reliability_res_update(self):
    if self.rel_res_wg.ui.loads_rel_TW.currentIndex().row() >= 0:
        self.rel_res_wg.ui.loads_res_WGT.setVisible(True)
        element = \
            self.rel_res_wg.ui.loads_rel_TW.item(self.rel_res_wg.ui.loads_rel_TW.currentIndex().row(), 0).text()
        self.rel_res_wg.ui.loads_name_LBL.setText(element)
        self.rel_res_wg.ui.loads_rel_LE.setText(
            self.rel_res_wg.ui.loads_rel_TW.item(self.rel_res_wg.ui.loads_rel_TW.currentIndex().row(), 1).text())
        self.reliability_load_selected()

#
def reliability_load_selected(self):
    element = self.rel_res_wg.ui.loads_rel_TW.item(self.rel_res_wg.ui.loads_rel_TW.currentIndex().row(), 0).text()
    for row in range(self.UI.ui.tableWidget.rowCount()):
        if element == self.UI.ui.tableWidget.item(row, 0).text():
            self.UI.ui.tableWidget.setCurrentCell(row, 0)
            break

    for row in range(self.rel_res_wg.ui.comp_rel_TW.rowCount()):
        if element == self.rel_res_wg.ui.comp_rel_TW.item(row, 0).text():
            self.rel_res_wg.ui.comp_rel_TW.setCurrentCell(row, 0)
            break

#
def reliability_comp_selected(self):
    element = self.rel_res_wg.ui.comp_rel_TW.item(self.rel_res_wg.ui.comp_rel_TW.currentIndex().row(), 0).text()
    for row in range(self.UI.ui.tableWidget.rowCount()):
        if element == self.UI.ui.tableWidget.item(row, 0).text():
            self.UI.ui.tableWidget.setCurrentCell(row, 0)
            break

    self.rel_res_wg.ui.loads_res_WGT.setVisible(False)
    for row in range(self.rel_res_wg.ui.loads_rel_TW.rowCount()):
        self.rel_res_wg.ui.loads_rel_TW.setCurrentCell(-1, 0)
        if element == self.rel_res_wg.ui.loads_rel_TW.item(row, 0).text():
            self.rel_res_wg.ui.loads_rel_TW.setCurrentCell(row, 0)
            self.reliability_res_update()
            break

#
def rel_grafo_show(self):
    from PIL import Image
    image = Image.open(v.project_folder + '/reliability/img/'+ self.rel_res_wg.ui.loads_name_LBL.text() + '.png')
    image.show()

#
def protections_calc(self):
    items = ['Linea', 'Terminale', 'Raddrizzatore', 'Generatore', 'Convertitore DC-DC']
    from UI._general.Popup.Protections.popup_prot import PopupProt
    from Functionalities.Protections.protection import Protection
    popup = PopupProt()

    if popup.exec_():
        pass
    choice = popup.selection

    if popup.is_confirmed:
        prot_dict = dict()
        for elem in v.elements:
            v.elements[elem]['protections'] = dict()
            if v.elements[elem]['category'] == 'Battery' and choice[2]:
                elem_class = Battery(elem)
            elif v.elements[elem]['category'] == 'DC-DC_Conv' and choice[4]:
                elem_class = DCDC_conv(elem)
            elif v.elements[elem]['category'] == 'DC-Line' and choice[0]:
```

```
                    elem_class = DCLine(elem)
                elif v.elements[elem]['category'] == 'DC-Load' and choice[1]:
                    elem_class = DCLoad(elem)
                elif v.elements[elem]['category'] == 'DC-Wind' and choice[2]:
                    elem_class = DCwind(elem)
                elif v.elements[elem]['category'] == 'PV' and choice[2]:
                    elem_class = PV(elem)
                elif v.elements[elem]['category'] == 'PWM' and choice[3]:
                    elem_class = PWM(elem)
                else:
                    elem_class = None

                if elem_class is not None:
                    elem_class.calculate()
                    elem_class.store()
                    prot_dict[elem] = copy.deepcopy(v.elements[elem]['protections'])
                    prot_dict[elem]['category'] = v.elements[elem]['category']

            self.UI.ui.log_TBR.clear()
            self.log_write_wtime('Calcolo delle protezioni in corso...', 1)

            filename = os.path.join(v.project_folder, 'protections.yml')
            with open(filename, 'w') as file:
                documents = yaml.dump(prot_dict, file)

            prot = Protection(prot_dict, choice)
            v.protections = True
            v.executed.append('prot')
            self.ml_check()

            self.log_write_wtime('Calcolo delle protezioni completato', 2)
            self.log_write("Seleziona dall'elenco degli elementi un elemento delle seguenti categorie:")
            for i in range(0, len(choice)):
                if choice[i]:
                    self.log_write(items[i])
            pass
            self.protections_show_results()

    #
    def protections_show_results(self):
        from UI.Main.Protections.protections import Protections
        self.UI.ui.results_WG.removeWidget(self.UI.ui.results_WG.currentWidget())
        self.prot_res_wg = Protections()
        self.UI.ui.results_WG.addWidget(self.prot_res_wg.ui.widget)
        self.prot_res_wg.ui.indices_TW.currentCellChanged.connect(self.protection_comp_selected)

    #
    def protection_comp_selected(self):
        element = self.prot_res_wg.ui.indices_TW.item(self.prot_res_wg.ui.indices_TW.currentIndex().row(), 0).text()
        for row in range(self.UI.ui.tableWidget.rowCount()):
            if element == self.UI.ui.tableWidget.item(row, 0).text():
                self.UI.ui.tableWidget.setCurrentCell(row, 0)
                break

    #
    def pdf_gen(self):
        from Functionalities.PDF.pdf_creator import PDF
        pdf = PDF()
        pdf.save()

    #
    def log_write_wtime(self, text="", lines=0):
        now = dt.datetime.now()
        ln = ''
        for i in range(0, lines):
            ln = ln + '\n'
        self.UI.ui.log_TBR.append(now.strftime("%H:%M:%S") + ': ' + text + ln)
        self.UI.ui.log_TBR.repaint()

    #
    def log_write(self, text='', lines=0):
        ln = ''
        for i in range(0, lines):
            ln = ln + '\n'
        self.UI.ui.log_TBR.append(text + ln)
        self.UI.ui.log_TBR.repaint()


myMain = Main()
```

4.2 Functionalities

4.2.1 Controls

4.2.1.1 Port.py

```
import os

import numpy as np
import matplotlib.pyplot as plt

path_img = os.getcwd() + '/_images/Controls/'
path_data = os.getcwd() + '/_functionalities/Controls/'


class Port:
    def calc(self, custom, scenario, lock_ev=False, backup=True, p_backup=5000):
        # custom indica se si è scelto uno scenario Standard (custom = False) o Personalizzato (custom = True)
        # se si è scelto uno scenario standard, la variabile 'scenario' indica il nome dello scenario;
        # se si è scelto uno scenario Personalizzato, la variabile 'scenario' indica il vettore dei parametri;

        results = []
        indexes = []
        Funz_rete = ''

        # LETTURA DATI

        # PRODUZIONE - EOLICO
        WF = np.genfromtxt(path_data + 'WF.txt', usecols=(0,1,2,3,4,5,6,7,8,9,10,11) )
        WF = np.transpose(WF)
        # PRODUZIONE - FOTOVOLTAICO
        PV = np.genfromtxt(path_data + 'PV.txt', usecols=(0,1,2,3,4,5,6,7,8,9,10,11) )
        PV = np.transpose(PV)
        # CONSUMI - PORT AREA
        porto = np.genfromtxt(path_data + 'porto.txt', usecols=(0) )
        porto = np.transpose(porto)
        # ELECTRIC VEHICLE
        # percentuale utenti EV collegati alla rete
        EV_per = np.genfromtxt(path_data + 'EV.txt', usecols=0)
        # consumi per ricarica veloce
        EV_fast = np.genfromtxt(path_data + 'EV_fast.txt', usecols=0)
        #consumi per ricarica lenta
        EV_dumb = np.genfromtxt(path_data + 'EV_dumb.txt', usecols=0)

        # # DATI SCENARI
        # if not custom:
        #     preset_scenarios = ['scen_2020', 'scen_2030BC', 'scen_2030DEC', 'scen_2040BC', 'scen_2040DEC']
        #     i = preset_scenarios.index(scenario)
        #     scen = np.genfromtxt(path + 'scenari_porto.txt', usecols=[i])
        # else:
        #     scen = scenario

        preset_scenarios = ['scen_2020', 'scen_2030BC', 'scen_2030DEC', 'scen_2040BC', 'scen_2040DEC']

        # -- @AR: Nuova aggiunta confronto scenari---
        if custom:
            preset_scenarios.append('custom')
            i_sel = len(preset_scenarios)-1
        else:
            i_sel = preset_scenarios.index(scenario)

        # Estrazione del mese
        m = np.random.randint(1, 13)
        # m = 6   # TODO: da eliminare

        # estrazione istante di guasto
        ist = np.random.randint(1, 93)
        # ist = 76    # TODO: Da eliminare

        # IDENTIFICAZIONE DEL RANGE DI GUASTO
        # durata media di interruzione per guasto in MT (60 minuti) e guasto in BT (30 minuti)
        g_MT = 1  # considerando 60 minuti di guasto in valori decimali
        g_BT = 0.75  # considerando 30 minuti di guasto in valori decimali
        # ********************************************************************************************
        # decido togliendo/inserendo "#" se il guasto e' in MT o in BT
        # ********************************************************************************************
        # guasto=g_MT
        guasto = g_BT

        ist_iniziale = ist * 0.25  # in termini temporali
        ist_finale = ist_iniziale + guasto
        time_g = np.arange(ist_iniziale, (ist_finale + 0.25), 0.25)  # mi serve successivamente per plottarlo
        fine = int(ist_finale / 0.25)

        soc_cold=np.random.randint(10,91)                    # estrazione dello storage dell'impianto PV
        # soc_cold = 30   # TODO: Da eliminare

        if lock_ev:
            n_col = 10
            C_ev = 40
            Pev_fast = 5
            Pev_slow = 5

            n_ev=round((EV_per[ist-1]/100)*n_col)

            # numero EV fast ed slow che verranno sovrascritti
            n_EV_fast=0
            n_EV_slow=0
            ricarica=np.zeros(int(n_ev), dtype=int)
            for i in range(n_ev):
                ricarica[i]=np.random.randint(1,3)
                if ricarica[i]==1:
                    n_EV_fast=n_EV_fast+1
                elif ricarica[i]==2:
                    n_EV_slow=n_EV_slow+1
                if n_EV_fast>(n_col/2):
                    n_EV_slow=n_EV_slow+(n_EV_fast-n_col/2)
                    n_EV_fast=n_col/2
                elif n_EV_slow>n_col/2:
                    n_EV_fast=n_EV_fast+(n_EV_slow-n_col/2)
                    n_EV_slow=n_col/2
```

```
                if (n_EV_fast+n_EV_slow)==n_ev: break
            # n_EV_fast, n_EV_slow, ricarica = 4, 4, [1, 1, 2, 2, 1, 2, 1, 2]  # TODO: Da eliminare

            # creo vettore i cui elementi rappresentano lo stato di carica dei veicoli fast + vettore con i rispettivi tempi di
carica e tempi di scarica
            t_fast_c=np.zeros(int(n_EV_fast), dtype=float)  # CARICA EV FAST
            t_fast_s=np.zeros(int(n_EV_fast), dtype=float)  # SCARICA EV FAST
            SOC_fast=np.zeros(int(n_EV_fast), dtype=int)
            soc=np.zeros(int(n_EV_fast), dtype=int)
            j=0

            # soc = [28, 56, 19, 10]  # TODO: Da eliminare
            if n_EV_fast>0:
                for i in range(int(n_EV_fast)):
                    soc[i]=np.random.randint(10,91)   # TODO: Da ripristinare
                    SOC_fast[j]=soc[i]
                    t_fast_c[j]=((90-soc[i])/100)*C_ev/Pev_fast         # carica = assorbimento
                    t_fast_s[j]=(np.abs(soc[i]-10)/100)*C_ev/Pev_fast   # scarica = erogazione in V2G
                    j=j+1
            else:
                SOC_fast=0
                t_fast_c=0
                t_fast_s=0

            # creo vettore i cui elementi rappresentano lo stato di carica dei veicoli slow + vettore con i rispettivi tempi di
ricarica
            t_slow=np.zeros(int(n_EV_slow), dtype=float)  # CARICA EV FAST
            SOC_slow=np.zeros(int(n_EV_slow), dtype=int)
            socs=np.zeros(int(n_EV_slow), dtype=int)
            k=0
            # socs = [22, 57, 73, 63]  # TODO: Da eliminare
            if n_EV_slow>0:
                for i in range(int(n_EV_slow)):
                    socs[i] = np.random.randint(10, 91)   # TODO: Da ripristinare
                    SOC_slow[k]=socs[i]
                    t_slow[k]=((90-socs[i])/100)*C_ev/Pev_slow          # carica = assorbimento
                    k=k+1
            else:
                SOC_slow=0
                t_slow=0

        # Linea di BackUp ---------------------------------------------
        linea_backup = int(backup)
        if not backup:
            p_backup = 0
        Plb = p_backup
        # -------------------------------------------------------------

        #
        for study_case in preset_scenarios:
            if study_case == 'custom':
                scen = scenario
            else:
                i = preset_scenarios.index(study_case)
                scen = np.genfromtxt(path_data + 'scenari_porto.txt', usecols=[i])

            # il valore va impostato in relazione alla rete che si sta analizzando
            an=25000

            # DATI PRINCIPALI
            C=scen[0]         # capacita' kWh PORTO AREA
            Pc=scen[1]        # potenza nominale PORT AREA picco
            CC=scen[2]        # Carico controllabile [kW]
            Ppv=scen[3]       # potenza nominale fotovoltaico
            Pwf=scen[4]       # potenza nominale eolico
            P_cold=scen[5]    # potenza cold ironing
            C_cold=scen[6]    # capacità cold ironing
            # n_col=scen[7]     # Numero di veicoli
            # nev_fast=scen[8]  # Numero EV fast
            # nev_slow=scen[9]  # Numero EV slow
            # Pev_fast=scen[10] # Potenza dei veicoli elettrici a ricarica veloce (10 kW)
            # Pev_slow=scen[11] # Potenza dei veicoli elettrici a ricarica veloce (3 kW)
            C_ev=40           # Capacità EV [kWh]

            WF_pu = WF[m-1][:] # produzione eolica nel mese estratto [p.u.]
            PV_pu = PV[m-1][:] # produzione fotovoltaica nel mese estratto [p.u.]

            # PROFILI IN kW
            PV_scen=Ppv*PV_pu     # Produzione fotovoltaica in kW
            WF_scen=Pwf*WF_pu     # Produzione eolica in kW
            PORTO = Pc*porto      # Consumi

            time = np.arange(0.,24.,0.25)

            if not lock_ev:
                n_col = scen[7]  # Numero di veicoli
                Pev_fast = scen[10]  # Potenza dei veicoli elettrici a ricarica veloce (10 kW)
                Pev_slow = scen[11]  # Potenza dei veicoli elettrici a ricarica veloce (3 kW)


                # n. di veicoli collegati nell'istante di guasto
                # prendo la percentuale di veicoli connessi alla rete nell'istante di guasto

                n_ev=round((EV_per[ist-1]/100)*n_col)

                # numero EV fast ed slow che verranno sovrascritti
                n_EV_fast=0
                n_EV_slow=0
                ricarica=np.zeros(int(n_ev), dtype=int)
                for i in range(n_ev):
                    ricarica[i]=np.random.randint(1,3)
                    if ricarica[i]==1:
                        n_EV_fast=n_EV_fast+1
                    elif ricarica[i]==2:
                        n_EV_slow=n_EV_slow+1
                    if n_EV_fast>(n_col/2):
                        n_EV_slow=n_EV_slow+(n_EV_fast-n_col/2)
                        n_EV_fast=n_col/2
                    elif n_EV_slow>n_col/2:
                        n_EV_fast=n_EV_fast+(n_EV_slow-n_col/2)
                        n_EV_slow=n_col/2
                    if (n_EV_fast+n_EV_slow)==n_ev: break
```

```
                # creo vettore i cui elementi rappresentano lo stato di carica dei veicoli fast + vettore con i rispettivi tempi di
carica e tempi di scarica
                t_fast_c=np.zeros(int(n_EV_fast), dtype=float)  # CARICA EV FAST
                t_fast_s=np.zeros(int(n_EV_fast), dtype=float)  # SCARICA EV FAST
                SOC_fast=np.zeros(int(n_EV_fast), dtype=int)
                soc=np.zeros(int(n_EV_fast), dtype=int)
                j=0

                if n_EV_fast>0:
                    for i in range(int(n_EV_fast)):
                        soc[i]=np.random.randint(10,91)
                        SOC_fast[j]=soc[i]
                        t_fast_c[j]=((90-soc[i])/100)*C_ev/Pev_fast         # carica = assorbimento
                        t_fast_s[j]=(np.abs(soc[i]-10)/100)*C_ev/Pev_fast   # scarica = erogazione in V2G
                        j=j+1
                else:
                    SOC_fast=0
                    t_fast_c=0
                    t_fast_s=0

                # creo vettore i cui elementi rappresentano lo stato di carica dei veicoli slow + vettore con i rispettivi tempi di
ricarica
                t_slow=np.zeros(int(n_EV_slow), dtype=float)  # CARICA EV FAST
                SOC_slow=np.zeros(int(n_EV_slow), dtype=int)
                socs=np.zeros(int(n_EV_slow), dtype=int)
                k=0
                if n_EV_slow>0:
                    for i in range(int(n_EV_slow)):
                        socs[i]=np.random.randint(10,91)
                        SOC_slow[k]=socs[i]
                        t_slow[k]=((90-socs[i])/100)*C_ev/Pev_slow          # carica = assorbimento
                        k=k+1
                else:
                    SOC_slow=0
                    t_slow=0

            # COLD IRONING: SOC + tempo di carica (assorbimento) + tempo scarica (erogazione)
            # soc_cold=np.random.randint(10,91)                  # estrazione dello storage dell'impianto PV
            t_carica=((90-soc_cold)/100)*C_cold/P_cold             # tempo in fase di carica
            t_scarica=(abs(soc_cold-10)/100)*C_cold/P_cold         # tempo in fase di scarica

            self.output = (study_case, m, ist, soc_cold)
            # SI RICAVANO I VALORI D'INTERESSE NELL'INTERVALLO DI GUASTO PER EV e COLD IRONING
            # EV FAST in fase di carica
            P_fast_c=np.zeros((n_EV_fast,len(time_g)), dtype=float)
            q=0
            for i in range(int(n_EV_fast)):
                for j in range(len(time_g)):
                    if t_fast_c[q]>guasto or t_fast_c[q]==guasto:
                        P_fast_c[i][j]=Pev_fast
                    elif t_fast_c[q]<guasto:
                        for y in range((round(len(time_g)*t_fast_c[q]))):
                            P_fast_c[i][y]=Pev_fast
                        for k in range((round(len(time_g)*t_fast_c[q])),len(time_g)):
                            P_fast_c[i][k]=0
                q=q+1

            # si sommano tutte le righe per avere un unico profilo di assorbimento dedicato alla ricarica veloce: VETTORE DI CARICA
            p_fast_g1=np.zeros((len(t_fast_c)), dtype=float)
            if len(t_fast_c)>1:
                p_fast_g1=np.sum(P_fast_c, axis=0)
            elif len(t_fast_c)==1:
                p_fast_g1=P_fast_c
                if len(t_fast_c)==0:
                    p_fast_g1=np.zeros(len(time_g))

            # EV FAST in fase di scarica
            P_fast_s=np.zeros((n_EV_fast,len(time_g)), dtype=float)
            q=0
            for i in range(int(n_EV_fast)):
                for j in range(len(time_g)):
                    if t_fast_s[q]>guasto or t_fast_s[q]==guasto:
                        P_fast_s[i][j]=Pev_fast
                    elif t_fast_s[q]<guasto:
                        for y in range((round(len(time_g)*t_fast_s[q]))):
                            P_fast_s[i][y]=Pev_fast
                        for k in range((round(len(time_g)*t_fast_s[q])),len(time_g)):
                            P_fast_s[i][k]=0
                q=q+1

            # si sommano tutte le righe per avere un unico profilo di assorbimento dedicato alla ricarica veloce: VETTORE DI SCARICA
            p_fast_g2=np.zeros((len(t_fast_s)), dtype=float)
            if len(t_fast_s)>1:
                p_fast_g2=np.sum(P_fast_s, axis=0)
            elif len(t_fast_s)==1:
                p_fast_g2=P_fast_s
                if len(t_fast_s)==0:
                    p_fast_g2=np.zeros(len(time_g))

            # EV SLOW
            P_slow=np.zeros((n_EV_slow,len(time_g)), dtype=float)
            l=0
            for i in range(int(n_EV_slow)):
                for j in range(len(time_g)):
                    if t_slow[l]>guasto or t_slow[l]==guasto:
                        P_slow[i][j]=Pev_slow
                    elif t_slow[l]<guasto:
                        for y in range((round(len(time_g)*t_slow[l]))):
                            P_slow[i][y]=Pev_slow
                        for k in range((round(len(time_g)*t_slow[l])),len(time_g)):
                            P_slow[i][k]=0
                l=l+1

            # si sommano tutte le righe per avere un unico profilo di assorbimento dedicato alla ricarica lenta
            p_slow_g=np.zeros((len(t_slow)), dtype=float)
            if len(t_slow)>1:
                p_slow_g=np.sum(P_slow, axis=0)
            elif len(t_slow)==1:
                p_slow_g=P_slow
                if len(t_slow)==0:
                    p_slow_g=np.zeros(len(time_g))

            # COLD IRONING
            # CARICA
            cold_c=np.zeros((len(time_g)), dtype=float)
```

```
            for i in range(int(len(time_g))):
                if t_carica>guasto or t_carica==guasto:
                        cold_c[i]=P_cold
                elif t_carica<guasto:
                        for y in range((round(len(time_g)*t_carica))):
                            cold_c[y]=P_cold
                        for k in range((round(len(time_g)*t_carica)),len(time_g)):
                            cold_c[k]=0

            # SCARICA
            cold_s=np.zeros((len(time_g)), dtype=float)

            for i in range(int(len(time_g))):
                if t_scarica>guasto or t_scarica==guasto:
                        cold_s[i]=P_cold
                elif t_scarica<guasto:
                        for y in range((round(len(time_g)*t_scarica))):
                            cold_s[y]=P_cold
                        for k in range((round(len(time_g)*t_scarica)),len(time_g)):
                            cold_s[k]=0

            # INTERVALLO DI GUASTO - PRODUZIONE
            # FOTOVOLTAICO
            PV_g=np.zeros(len(time_g), dtype=float)
            i=0
            for j in range(ist,fine+1):
                PV_g[i]=round(PV_scen[j-1],2)
                i=i+1
            # EOLICO
            WF_g=np.zeros(len(time_g), dtype=float)
            i=0
            for j in range(ist,fine+1):
                WF_g[i]=round(WF_scen[j-1],2)
                i=i+1

            # INTERVALLO DI GUASTO - CONSUMI
            load1 = np.zeros(len(time_g), dtype=float)
            i = 0
            for j in range(ist, fine + 1):
                load1[i] = round(PORTO[j - 1], 2)
                i = i + 1

            # INTERVALLO DI GUASTO - CONSUMI E LINEA DI BACKUP
            load = np.zeros(len(time_g), dtype=float)
            i = 0
            for j in range(ist, fine + 1):
                load[i] = max(1000, round((load1[i] - Plb), 2))
                if load[i] < 0:
                    load[i] = 0
                i = i + 1

            # CARICO CONTROLLABILE
            CC_g=CC*np.ones(len(time_g))
            if n_EV_slow==0:
                p_slow_g=np.zeros(len(time_g))

            CC_tot=CC_g+p_slow_g       # il carico totale che posso staccare: EV lenti + Carico Controllabile

            # LOGICA RETE
            gen=PV_g+WF_g

            delta=gen-load

            P_fast_g=np.zeros(len(time_g))
            COLD=np.zeros(len(time_g))
            nuovo_delta=np.zeros(len(time_g))
            load_g=np.zeros(len(time_g))
            DR=np.zeros(len(time_g))

            if (n_EV_fast)!=0:
                Pev_fast=(np.sum(p_fast_g1, axis=0))/len(p_fast_g1)
            elif n_EV_fast==0:
                Pev_fast=0

            # LOGICA CARICA
            for i in range(len(time_g)):
                if delta[i]>P_cold and delta[i]>Pev_fast and np.abs(sum(delta)/len(time_g))>P_cold and
np.abs(sum(delta)/len(time_g))>Pev_fast:
                    COLD[i]=cold_c[i]
                    if delta[i]-COLD[i]<p_fast_g1[i] and n_EV_fast!=0:
                        P_fast_g[i]=delta[i]-COLD[i]
                    if delta[i]-COLD[i]>p_fast_g1[i] and n_EV_fast!=0:
                        P_fast_g[i]=p_fast_g1[i]
                elif delta[i]>0 and delta[i]<P_cold:
                    if t_carica>=guasto:
                        COLD[i]=delta[i]            # lo storage si carica con un assorbimento pari alla meta' della differenza
                    if p_fast_g1[i]<delta[i]:
                        P_fast_g[i]=p_fast_g1       # gli EV fast si caricano con un assorbimento pari alla meta' della differenza

            # LOGICA DI SCARICA
            if n_EV_fast==0:
                p_fast_g2=np.zeros(len(time_g))

            if t_scarica>guasto:
                for i in range(len(time_g)):
                    if delta[i]<0 and np.abs(delta[i])>P_cold:
                        load_g[i]=load[i]-CC_tot[i]
                        nuovo_delta[i]=gen[i]-load_g[i]
                        COLD[i]=cold_s[i]
                        if np.abs(nuovo_delta[i]-COLD[i])<p_fast_g2[i] and n_EV_fast!=0:
                            P_fast_g[i]=np.abs(nuovo_delta[i])-COLD[i]
                        elif np.abs(nuovo_delta[i]-COLD[i])>p_fast_g2[i] and n_EV_fast!=0:
                            P_fast_g[i]=p_fast_g2[i]
                            DR[i]=np.abs(nuovo_delta[i])-COLD[i]-P_fast_g[i]
                        elif delta[i]<0 and np.abs(delta[i])<P_cold:
                            load_g[i]=load[i]-CC_tot[i]
                            nuovo_delta[i]=gen[i]-load_g[i]
                            COLD[i]=np.abs(nuovo_delta[i])
                            P_fast_g[i]=0

            if t_scarica<=guasto:
                for i in range(len(time_g)):
                    if delta[i]<0:
                        load_g[i]=load[i]-CC_tot[i]
```

```
                        nuovo_delta[i]=gen[i]-load_g[i]
                        for y in range (round(len(time_g)*t_scarica)):
                            if np.abs(nuovo_delta[y])>P_cold:
                                COLD[y]=cold_s[y]
                                if np.abs(nuovo_delta[y]-COLD[y])<p_fast_g2[y] and n_EV_fast!=0:
                                    P_fast_g[y]=np.abs(nuovo_delta[y])-COLD[y]
                                elif np.abs(nuovo_delta[y]-COLD[y])>=p_fast_g2[y] and n_EV_fast!=0:
                                    P_fast_g[y]=p_fast_g2[y]
                                    DR[y]=np.abs(nuovo_delta[y])-COLD[y]-P_fast_g[y]
                        for k in range((round(len(time_g)*t_scarica)),(len(time_g))):
                            COLD[k]=0
                            if np.abs(nuovo_delta[k])<p_fast_g2[k] and n_EV_fast!=0:
                                P_fast_g[k]=np.abs(nuovo_delta[k])
                            elif np.abs(nuovo_delta[k])>p_fast_g2[k] and n_EV_fast!=0:
                                P_fast_g[k]=p_fast_g2[k]
                                DR[k]=np.abs(nuovo_delta[k])-P_fast_g[k]
                        for l in range(round(len(time_g)*t_scarica)):
                            if np.abs(nuovo_delta[l])<=P_cold:
                                COLD[l]=np.abs(nuovo_delta[l])
                                P_fast_g[l]=0
                        for l1 in range(round(len(time_g)*t_scarica),len(time_g)):
                            COLD[l1]=0
                            if (np.abs(nuovo_delta[l1]))<p_fast_g2[l1] and n_EV_fast!=0:
                                P_fast_g[l1]=np.abs(nuovo_delta[l1])

            info=np.column_stack((m,ist,fine))
            output=np.column_stack((time_g,load,CC_tot,load_g,PV_g,WF_g,p_slow_g,P_fast_g,COLD,DR))
            # print(output)

            # CALCOLO DEGLI INDICATORI CARATTERIZZANTI LO SCENARIO
            media=np.mean(output, axis=0)
            # print(media)

            # Indicatore "RES" - valuta la percentuale di generazione da fonte rinnovabile rispetto alla generazione complessiva
della rete [%].
            RES=round(((media[4]+media[5])/(media[1]))*100)

            # Indicatore FLEX - valuta la flessibilità del carico definita come l'energia che può essere spostata nell'intervallo
temporale di riferimento
            # agendo sui soli carichi flessibili in rapporto all'energia totale richiesta dai carichi nello stesso intervallo [%].
            FLEX=round(np.abs((media[2]/media[1]))*100)

            # Indicatore BESS - valuta il rapporto tra la potenza fornita dai sistemi di accumulo e la totale potenza dei sistemi di
generazione nella microrete [%].
            BESS=round(((media[7]+media[8])/media[1])*100)

            # print(RES)
            # print(FLEX)
            # print(BESS)

            # CALCOLO DEGLI INDICATORI DI AFFIDABILITA'
            isola=np.zeros(((len(time_g))-1), dtype=int)
            output_g=output[1:,0:10]
            dim_output_g=np.shape(output_g)
            produzione_storage=np.zeros((dim_output_g[0]), dtype=float)

            intervallo=int(guasto/0.25)
            for i in range(intervallo):
                produzione_storage[i]=output_g[i][4]+output_g[i][5]+output_g[i][7]+output_g[i][8]
                if produzione_storage[i]==output_g[i][3]:
                    isola[i]=1
                else:
                    isola[i]=0

            # indice di autonomia della microrete "i1" - misura la capacità di funzionamento in isola se richiesto dalla rete
principale [%].
            i1=round(((np.sum(isola, axis=0)*0.25)/guasto)*100)

            # indice di flessibilità "i2" - valuta la variazione di potenza attiva disponibile in aumento o riduzione nel punto di
connessione [%].
            i2=round(((media[2]+media[4]+media[5]+media[7]+media[8])/an)*100)

            # indice di capacità di modulazione del profilo di potenza in un tempo convenzionale "i3" -
            # misura la possibilità di variare con continuità per un tempo stabilito il profilo di potenza in un nodo [%].
            i3=round(((media[2]+media[7]+media[8])/(media[2]+media[7]+media[8]+media[9]))*100)

            # print(i1)
            # print(i2)
            # print(i3)

            # ********************************************************************** NUOVI INDICATORI
**************************************************************************************************

            # Durata delle interruzioni "TI" - differenza tra la durata media delle interruzioni della microrete (assunta
convenzionalmente pari a 45 min in BT e 60 min in MT)
            # e l'intervallo di tempo (Δt) espresso in minuti in cui, azionando le dovute logiche di controllo, si riescono ad
alimentare almeno le utenze critiche della microrete [min].

            TI = (guasto*60)-((np.sum(isola, axis=0)*0.25)*60)
            print(TI)

            # Energia non fornita "Ens" - rapporto tra l'energia che i carichi della microrete ricevono durante l'evento di failure e
l'energia che gli stessi carichi
            # avrebbero richiesto in assenza di guasto secondo il previsto profilo giornaliero di consumo [adim.].

            Ens = round(1 -
((np.sum(PV_g)+np.sum(WF_g)+np.sum(P_fast_g)+np.sum(COLD))*0.25)/((np.sum(load)+(np.sum(p_slow_g))*0.25)), 4)

            print(Ens)

            # Generazione flessible "Ng" - indicatore che tiene conto della presenza della generazione flessibile sul totale della
generazione della microrete operante in isola [adim.].

            Kpv=50  # [%] Valore stabilito dall'untente - Percentuale di generezione fotovoltaica controllabile rispetto totale
installata
            Kwf=50  # [%] Valore stabilito dall'untente - Percentuale di generezione eolica controllabile rispetto alla totale
installata

            Ng= round((((Kpv/100)*Ppv+(Kwf/100)*Pwf)/(Ppv+Pwf)), 4)
            print(Ng)

            # Riserva dello storage "ST" - indicatore che valuta la riserva di energia dei sistemi di accumulo installati nella
microrete rispetto al consumo giornaliero della microrete.

            Eday= np.sum(PORTO)*0.25     # energia richiesta dai carichi della microrete in isola in una giornata [kWh]
```

```
            ST = round(((0.8*C_cold)/(1.05*1.11*Eday)), 4)

            print(ST)

            # Capacità di grid forming "GF" - rapporto tra la potenza dei convertitori funzionanti in grid-forming e la potenza
totale necessaria [adim.]

            GF = round((Ppv+Pwf+P_cold)/((np.max(PORTO)*0.1)), 4)
            print(GF)

            # Rapporto di inerzia "IR" - rapporto tra l'inerzia dei convertitori esistenti e quella necessaria riproporzionata sulla
potenza effettivamente installata.

            H = 5    # [s] inerzia del sistema - Valore definito dall'utente
            IR = round(((H*(Ppv+Pwf+P_cold))/(3*(np.max(PORTO)*0.1))), 4)

            print(IR)

            #
**********************************************************************************************************************************************
***********************************************

            # ANDAMENTO DEI VALORI NELL'INTERVALLO DI GUASTO
            if custom or scenario == study_case:
                # ASSE TEMPORALE - CONVERSIONE DECIMALE -> SESSAGESIMALE
                tempo=time_g
                ore=np.zeros(len(tempo), dtype=int)
                minuti=np.zeros(len(tempo), dtype=int)
                assex=np.zeros(len(tempo), dtype=list)

                for i in range(len(tempo)):
                    ore[i]=int(tempo[i])
                    minuti[i]=int((tempo[i]-ore[i])*60) & 63
                    assex[i]="%d:%d" % (ore[i], minuti[i])

                fig, ax = plt.subplots(figsize=(6.2, 5))
                ax.plot(assex, load, label="Consumi")
                ax.plot(assex, PV_g, label="Fotovoltaico")
                ax.plot(assex, WF_g, label="Eolico")
                ax.plot(assex, CC_tot, label="carico controllabile")
                ax.plot(assex, COLD, label="cold ironing")
                ax.plot(assex, p_slow_g, label="ricarica lenta")
                ax.plot(assex, P_fast_g, label="ricarica veloce")
                ax.plot(assex, DR, label="Demand Response")
                # fig.legend(loc="upper right")
                ax.set_xlabel('Time [hh:ss]')
                ax.set_ylabel('Potenze [kW]')
                ax.set_title('PORT AREA: ANDAMENTI NEL RANGE DI GUASTO', y=1.15)
                # plt.grid()
                plt.legend(bbox_to_anchor=(0, 1.02, 1, 0.2), loc="lower left", mode="expand", borderaxespad=0,
                           ncol=4, fontsize=8)
                plt.tight_layout()
                fig.savefig(path_img + 'PORT.png')
                plt.cla()

                # FUNZIONAMENTO DELLA RETE
                # Imposta valore per linea_backup:
                # 1 = Linea esistente
                # 0 = Linea assente
                linea_backup = int(backup)

                # Legenda
                # Pfer = Potenza prodotta da impianti di generazione da FER
                # Pload = Potenza richiesta dai carichi
                # Pstorage = Potenza disponibile da storage
                # PERC_DR = Percentuale di Demand Response disponibile in  zona - VALORE RICAVATO DALLO SCENARIO SCELTO
                # Plb = Massima potenza che può circolare sul cavo di collegamento o sul trasformatore a monte. DATO FORNITO
DALL'UTENTE!

                # Imposta valori:
                Pfer = round(np.sum(gen, axis=0))
                Pload = round(np.sum(load, axis=0))
                Pstorage = round(np.sum(COLD, axis=0))
                PERC_DR = CC / Pc
                Plb = p_backup  # VALORE IMPOSTABILE DALL'UTENTE CHE VUOLE AVVIARE LA SIMULAZIONE [kW]

                # il fattore 0.8 presente nello script è un coefficiente di sicurezza mediante il quale si tiene conto del fatto che
l'energia prodotta dagli impianti di generazione da FER
                # deve compensare anche l'energia reattiva richiesta in zona.

                if linea_backup == 1:
                    if Pfer - Pload > 0:
                        Funz_rete = 'Funzionamento della rete in configurazione grid-off e la linea di backup viene caricata dal
surplus di potenza prodotta da impianti di generazione da FER'
                    elif Pfer - Pload < 0:
                        if Plb >= (Pload - Pfer) / 0.8:
                            Funz_rete = 'Funzionamento della rete in configurazione grid-off grazie alla potenza fornita dalla linea
di backup'
                        elif Plb < ((Pload - Pfer) / 0.8):
                            if Plb >= ((Pload - Pfer - Pstorage) / 0.8):
                                Funz_rete = 'Funzionamento della rete in configurazione grid-off grazie alla potenza fornita dalla
linea di backup e allo storage presente in zona'
                            elif Plb <= ((Pload - Pfer - Pstorage) / 0.8):
                                Pdr = (
                                        PERC_DR) * Pload  # si calcola la potenza distaccabile dalla rete grazie a interventi di
Demand Response
                                Pload_dr = Pload - Pdr  # potenza richiesta dai carichi decurtata da quella distaccata a seguito di
azioni di DR
                                if Pfer + Pstorage - Pload_dr > 0:
                                    Funz_rete = 'Funzionamento della rete in configurazione grid-off sfruttando tutte le risorse
flessibili presenti in zona'
                                elif Plb + Pfer + Pstorage - Pload_dr >= 0:
                                    Funz_rete = 'Funzionamento della rete in configurazione grid-off sfruttando tutte le risorse
flessibili presenti in zona e la potenza fornita dalla linea di backup'
                                elif Plb + Pfer + Pstorage - Pload_dr < 0:
                                    Funz_rete = 'La rete non può funzionare in configurazione grid-off'
                elif linea_backup == 0:
                    if Pfer - Pload >= 0:
                        Funz_rete = 'Funzionamento della rete in configurazione grid-off'
                    elif Pfer - Pload < 0:
                        if Pstorage >= (Pload / 0.8) - Pfer:
                            Funz_rete = 'Funzionamento della rete in configurazione grid-off grazie allo storage'
                        elif Pstorage < (Pload / 0.8) - Pfer:
                            Pdr = (
```

```
                                    PERC_DR) * Pload  # si calcola la potenza distaccabile dalla rete grazie a interventi di Demand
Response
                            Pload_dr = Pload - Pdr  # potenza richiesta dai carichi decurtata da quella distaccata a seguito di
azioni di DR
                            if Pfer + Pstorage - (Pload_dr / 0.8) >= 0:
                                Funz_rete = 'Funzionamento della rete in configurazione grid-off sfruttando tutte le risorse
flessibili presenti in zona'
                            elif Pfer + Pstorage - (Pload_dr / 0.8) < 0:
                                Funz_rete = 'La rete non può funzionare in configurazione grid-off'

                print(Funz_rete)

            indexes.append([i1, i2, i3])
            results.append([TI, Ens, Ng, ST, GF, IR])

        return indexes, [m, 0, ist_iniziale, ist_finale], results, i_sel, Funz_rete

        # return [i1, i2, i3], [m, 0, ist_iniziale, ist_finale], [TI, Ens, Ng, ST, GF, IR]
```

4.2.1.2 residential.py

```
import os

import numpy as np
import matplotlib.pyplot as plt

path_img = os.getcwd() + '/_images/Controls/'
path_data = os.getcwd() + '/_functionalities/Controls/'


class Residential:
    def calc(self, custom, scenario, lock_ev=False, backup=True, p_backup=100):
        results = []
        indexes = []
        Funz_rete = ''

        # custom indica se si è scelto uno scenario Standard (custom = False) o Personalizzato (custom = True)
        # se si è scelto uno scenario standard, la variabile 'scenario' indica il nome dello scenario;
        # se si è scelto uno scenario Personalizzato, la variabile 'scenario' indica il vettore dei parametri;

        # LETTURA DATI

        # PRODUZIONE - EOLICO
        WF = np.genfromtxt(path_data + 'WF.txt', usecols=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11))
        WF = np.transpose(WF)
        # PRODUZIONE - FOTOVOLTAICO
        PV = np.genfromtxt(path_data + 'PV.txt', usecols=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11))
        PV = np.transpose(PV)
        # CONSUMI - GIORNO FERIALE
        MTbt_fer = np.genfromtxt(path_data + 'MTbt_fer.txt', usecols=(0, 1, 2, 3))
        MTbt_fer = np.transpose(MTbt_fer)
        # CONSUMI - SABATO
        MTbt_sab = np.genfromtxt(path_data + 'MTbt_sab.txt', usecols=(0, 1, 2, 3))
        MTbt_sab = np.transpose(MTbt_sab)
        # CONSUMI - GIORNO FESTIVO
        MTbt_fest = np.genfromtxt(path_data + 'MTbt_fest.txt', usecols=(0, 1, 2, 3))
        MTbt_fest = np.transpose(MTbt_fest)

        # ELECTRIC VEHICLE
        # percentuale utenti EV collegati alla rete
        EVperc = np.genfromtxt(path_data + 'EV.txt', usecols=0)
        # consumi per ricarica veloce
        EVfast = np.genfromtxt(path_data + 'EV_fast.txt', usecols=0)
        # consumi per ricarica lenta
        EVdumb = np.genfromtxt(path_data + 'EV_dumb.txt', usecols=0)

        preset_scenarios = ['scen_2020', 'scen_2030BC', 'scen_2030DEC', 'scen_2040BC', 'scen_2040DEC']

        # -- @AR: Nuova aggiunta confronto scenari---
        if custom:
            preset_scenarios.append('custom')
            i_sel = len(preset_scenarios)-1
        else:
            i_sel = preset_scenarios.index(scenario)

        # AR: eliminato dopo aggiunta confronto scenari ------------
        # # DATI SCENARI
        # if not custom:
        #     i = preset_scenarios.index(scenario)
        #     scen = np.genfromtxt(path + 'scenari.txt', usecols=[i])
        # else:
        #     scen = scenario
        # ------------------------------------------------------------

        # Estrazione del mese -------------------------------------
        m = np.random.randint(1, 13)
        # m = 1   # TODO: Da eliminare

        WF_pu = WF[m - 1][:]  # produzione eolica nel mese estratto [p.u.]
        PV_pu = PV[m - 1][:]  # produzione fotovoltaica nel mese estratto [p.u.]

        u = np.shape(MTbt_fer)
        MTBT = np.zeros((u[0], u[1]), dtype=float)

        # estrazione del giorno
        g = np.random.randint(1, 8)
        # g = 4   # TODO Da eliminare

        if g >= 1 and g <= 5:
            MTBT = MTbt_fer
        elif g == 6:
            MTBT = MTbt_sab
        else:
            MTBT = MTbt_fest

        M = [[12, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]]

        t = np.shape(M)

        for i in range(t[0]):
            for j in range(t[1]):
                if m == M[i][j]:
                    s = i

        MTBT_pu = MTBT[s][:]  # consumi nell'area residenziale [p.u.]

        time = np.arange(0., 24., 0.25)

        # estrazione istante di guasto
        ist = np.random.randint(1, 93)
        # ist = 50    # TODO: Da Eliminare

        # IDENTIFICAZIONE DEL RANGE DI GUASTO
        # durata media di interruzione per guasto in MT (60 minuti) e guasto in BT (30 minuti)
        g_MT = 1  # considerando 60 minuti di guasto in valori decimali
        g_BT = 0.75  # considerando 30 minuti di guasto in valori decimali
        # ********************************************************************************
        # decido togliendo/inserendo "#" se il guasto e' in MT o in BT
        # ********************************************************************************
        # guasto=g_MT
        guasto = g_BT

        ist_iniziale = ist * 0.25  # in termini temporali
```

```
ist_finale = ist_iniziale + guasto
time_g = np.arange(ist_iniziale, (ist_finale + 0.25), 0.25)  # mi serve successivamente per plottarlo
fine = int(ist_finale / 0.25)

# STORAGE
# storage totale: SOC + tempo di carica (assorbimento) + tempo scarica (erogazione)
soc = np.random.randint(10, 91)
# soc = 20    # TODO: Da eliminare
# ------------------------------------------------------------

# Linea di BackUp ----------------------------------------------
linea_backup = int(backup)
if not backup:
    p_backup = 0
Plb = p_backup
# ------------------------------------------------------------

#
for study_case in preset_scenarios:
    if study_case == 'custom':
        scen = scenario
    else:
        i = preset_scenarios.index(study_case)
        scen = np.genfromtxt(path_data + 'scenari.txt', usecols=[i])

    # POTENZA APPARENTE DEL NODO CUI E' CONNESSA LA RETE [kVA]
    # il valore va impostato in relazione alla rete che si sta analizzando
    an=400

    # DATI PRINCIPALI
    Ccr = scen[0]  # capacita' kWh CABINA RESIDENZIALE
    Pcr = scen[1]  # potenza nominale CABINA RESIDENZIALE
    Ppv = scen[2]  # potenza nominale PV
    Pwf = scen[3]  # potenza nominale WF
    P_ess = scen[4]  # potenza storage (valore ipotizzato)
    C_ess = scen[5]  # capacita' storage (valore ipotizzato)
    # DATI SECONDARI
    n_veicoli = scen[6]  # Numnero di veicoli
    n_PHEV = scen[7]  # Numero di veicoli elettrici Plug-in Hybrid rispetto al totale (20%)
    n_EV1 = scen[8]  # Numero di veicoli elettrici a ricarica veloce rispetto al totale (40%)
    n_EV2 = scen[9]  # Numero di veicoli elettrici a ricarica lenta rispetto al totale (40%)
    P_PHEV = scen[10]  # Potenza di ricarica per i veicoli PHEV (3 kW)
    P_EV1 = scen[11]  # Potenza di ricarica per i veicoli EV1_veloce (10 kW)
    P_EV2 = scen[12]  # Potenza di ricarica per i veicoli EV2_lenta (3 kW)
    PERC_DR = scen[13]  # PERCENTUALE/100 di utenti che partecipano alla Demand Response. DATO DA CORREGGERE!

    # PROFILI IN kW
    MTBT_scen = Pcr * MTBT_pu  # assorbimento cabina residenziale in kW
    PV_scen = Ppv * PV_pu  # Produzione fotovoltaica in kW
    WF_scen = Pwf * WF_pu  # Produzione eolica in kW


    # STORAGE
    # storage totale: SOC + tempo di carica (assorbimento) + tempo scarica (erogazione)
    # soc = np.random.randint(10, 91)
    t_carica = round(((90 - soc) / 100) * C_ess / P_ess, 2);  # tempo di carica
    t_scarica = round((np.abs(soc - 10) / 100) * C_ess / P_ess, 2);  # tempo di scarica

    # LOGICA DI CARICA
    storage_c = np.zeros((len(time_g)), dtype=float)
    for i in range(len(time_g)):
        if t_carica > guasto or t_carica == guasto:
            storage_c[i] = P_ess
        elif t_carica < guasto:
            for y in range((round(len(time_g) * t_carica))):
                storage_c[y] = P_ess
            for k in range((round(len(time_g) * t_carica)), (len(time_g))):
                storage_c[k] = 0

                # LOGICA DI SCARICA
    storage_s = np.zeros((len(time_g)), dtype=float)
    for i in range(len(time_g)):
        if t_scarica > guasto or t_scarica == guasto:
            storage_s[i] = P_ess
        elif t_scarica < guasto:
            for y in range((round(len(time_g) * t_scarica))):
                storage_s[y] = P_ess
            for k in range((round(len(time_g) * t_scarica)), (len(time_g))):
                storage_s[k] = 0

    # INTERVALLO DI GUASTO - CONSUMI
    MTBT_g1 = np.zeros(len(time_g), dtype=float)
    i = 0
    for j in range(ist, fine + 1):
        MTBT_g1[i] = round(MTBT_scen[j - 1], 2)
        i = i + 1
    # INTERVALLO DI GUASTO - CONSUMI E LINEA DI BACKUP
    MTBT_g = np.zeros(len(time_g), dtype=float)
    i = 0
    for j in range(ist, fine + 1):
        MTBT_g[i] = max(0.000001, round((MTBT_g1[i] - Plb), 2))
        if MTBT_g[i] < 0:
            MTBT_g[i] = 0
        i = i + 1

    # INTERVALLO DI GUASTO - PRODUZIONE
    # FOTOVOLTAICO
    PV_g = np.zeros(len(time_g), dtype=float)
    i = 0
    for j in range(ist, fine + 1):
        PV_g[i] = round(PV_scen[j - 1], 2)
        i = i + 1
    # EOLICO
    WF_g = np.zeros(len(time_g), dtype=float)
    i = 0
    for j in range(ist, fine + 1):
        WF_g[i] = round(WF_scen[j - 1], 2)
        i = i + 1

    # STORAGE

    GEN = PV_g + WF_g
    delta = GEN - MTBT_g
    nuovo_delta = np.zeros(len(time_g))
    DR = np.zeros(len(time_g))
```

```
            storage = np.zeros(len(time_g))

            # LOGICA CARICA
            for i in range(len(time_g)):
                if delta[i] > P_ess and (sum(delta) / len(time_g)) > P_ess:
                    storage[i] = storage_c[i]
                    nuovo_delta[i] = delta[i] - storage[i]
                    if nuovo_delta[i] > 0 or nuovo_delta[i] < 0:
                        DR[i] = nuovo_delta[i]
                elif delta[i] < P_ess:
                    storage[i] = delta[i]

            # LOGICA DI SCARICA
            for i in range(len(time_g)):
                if delta[i] < 0 and np.abs(sum(delta) / len(time_g)) > P_ess:
                    storage[i] = storage_s[i]
                    nuovo_delta[i] = delta[i] + storage[i]
                    if nuovo_delta[i] > 0 or nuovo_delta[i] < 0:
                        DR[i] = nuovo_delta[i]
                elif delta[i] < 0 and np.abs(sum(delta) / len(time_g)) < P_ess:
                    for y in range(len(time_g)):
                        storage[y] = np.abs(delta[y])
                    for k in range((round(len(time_g) * t_scarica)), (len(time_g))):
                        storage[k] = 0
                        DR[k] = delta[k]

            print(GEN)

            info = np.column_stack((m, g, ist, fine, soc))
            output = np.column_stack((time_g, MTBT_g, PV_g, WF_g, storage, DR))
            print(info)
            print(output)

            # CALCOLO DEGLI INDICATORI CARATTERIZZANTI LO SCENARIO
            media = np.mean(output, axis=0)

            # Indicatore "RES" - valuta la percentuale di generazione da fonte rinnovabile rispetto alla generazione complessiva
della rete [%].
            RES = round(((media[2] + media[3]) / (media[1])) * 100)

            # Indicatore FLEX - valuta la flessibilità del carico definita come l'energia che può essere spostata nell'intervallo
temporale di riferimento
            # agendo sui soli carichi flessibili in rapporto all'energia totale richiesta dai carichi nello stesso intervallo [%].
            FLEX = round(np.abs(media[5] / media[1]) * 100)

            # Indicatore BESS - valuta il rapporto tra la potenza fornita dai sistemi di accumulo e la totale potenza dei sistemi di
generazione nella microrete [%].
            BESS = round((media[4] / media[1]) * 100)

            # CALCOLO DEGLI INDICATORI DI AFFIDABILITA'
            isola = np.zeros(((len(time_g)) - 1), dtype=int)
            output_g = output[1:, 0:6]
            dim_output_g = np.shape(output_g)
            produzione_storage = np.zeros((dim_output_g[0]), dtype=float)

            intervallo = int(guasto / 0.25)
            for i in range(intervallo):
                produzione_storage[i] = output_g[i][2] + output_g[i][3] + output_g[i][4]
                if produzione_storage[i] == output_g[i][1]:
                    isola[i] = 1
                else:
                    isola[i] = 0

            # indice di autonomia della microrete "i1" - misura la capacità di funzionamento in isola se richiesto dalla rete
principale [%].
            i1 = round(((np.sum(isola, axis=0) * 0.25) / guasto) * 100)

            # indice di flessibilità "i2" - valuta la variazione di potenza attiva disponibile in aumento o riduzione nel punto di
connessione [%].
            i2 = round(((media[2] + media[3] + media[4] + (np.abs(media[5]))) / an) * 100)

            # indice di capacità di modulazione del profilo di potenza in un tempo convenzionale "i3" -
            # misura la possibilità di variare con continuità per un tempo stabilito il profilo di potenza in un nodo [%].
            i3 = round(media[4] / (media[4] + (np.abs(media[5]))) * 100)

            print(i1)
            print(i2)
            print(i3)

            # ********************************************************************************* NUOVI INDICATORI
*****************************************************************************************

            # Durata delle interruzioni "TI" - differenza tra la durata media delle interruzioni della microrete (assunta
convenzionalmente pari a 45 min in BT e 60 min in MT)
            # e l'intervallo di tempo (Δt) espresso in minuti in cui, azionando le dovute logiche di controllo, si riescono ad
alimentare almeno le utenze critiche della microrete [min].

            TI = (guasto*60)-((np.sum(isola, axis=0)*0.25)*60)
            print(TI)

            # Energia non fornita "Ens" - rapporto tra l'energia che i carichi della microrete ricevono durante l'evento di failure e
l'energia che gli stessi carichi
            # avrebbero richiesto in assenza di guasto secondo il previsto profilo giornaliero di consumo [adim.].

            Ens = round(1 - (((np.sum(PV_g) + np.sum(WF_g) + np.sum(storage)) * 0.25) / (np.sum(MTBT_g) * 0.25)), 2)

            print(Ens)

            # Generazione flessible "Ng" - indicatore che tiene conto della presenza della generazione flessibile sul totale della
generazione della microrete operante in isola [adim.].

            Kpv=50  # [%] Valore stabilito dall'untente - Percentuale di generezione fotovoltaica controllabile rispetto totale
installata
            Kwf=50  # [%] Valore stabilito dall'untente - Percentuale di generezione eolica controllabile rispetto alla totale
installata

            Ng= round((((Kpv/100)*Ppv+(Kwf/100)*Pwf)/(Ppv+Pwf)),2)
            print(Ng)

            # Riserva dello storage "ST" - indicatore che valuta la riserva di energia dei sistemi di accumulo installati nella
microrete rispetto al consumo giornaliero della microrete.

            Eday= np.sum(MTBT_scen)*0.25                            # energia richiesta dai carichi della microrete in isola in una
giornata [kWh]
            ST = round(((0.8*C_ess)/(1.05*1.11*Eday)),2)
```

```
            print(ST)

            # Capacità di grid forming "GF" - rapporto tra la potenza dei convertitori funzionanti in grid-forming e la potenza
totale necessaria [adim.]

            GF = round((Ppv+Pwf+P_ess)/((np.max(MTBT_scen)*0.1667)),2)
            print(GF)

            # Rapporto di inerzia "IR" - rapporto tra l'inerzia dei convertitori esistenti e quella necessaria riproporzionata sulla
potenza effettivamente installata.

            H = 5    # [s] inerzia del sistema - Valore definito dall'utente
            IR = round(((H*(Ppv+Pwf+P_ess))/(3*(np.max(MTBT_scen)*0.1667))), 2)

            print(IR)

            #
**********************************************************************************************************************************************************************
**************************************************

            # ANDAMENTO DEI VALORI NELL'INTERVALLO DI GUASTO
            if custom or scenario == study_case:
                # ASSE TEMPORALE - CONVERSIONE DECIMALE -> SESSAGESIMALE
                tempo = time_g
                ore = np.zeros(len(tempo), dtype=int)
                minuti = np.zeros(len(tempo), dtype=int)
                assex = np.zeros(len(tempo), dtype=list)

                for i in range(len(tempo)):
                    ore[i] = int(tempo[i])
                    minuti[i] = int((tempo[i] - ore[i]) * 60) & 63
                    assex[i] = "%d:%d" % (ore[i], minuti[i])

                # plt.plot(assex, MTBT_g, label="Consumi")
                # plt.plot(assex, PV_g, label="Fotovoltaico")
                # plt.plot(assex, WF_g, label="Eolico")
                # plt.plot(assex, storage, label="Storage")
                # plt.plot(assex, DR, label="Demand Response")
                # plt.legend(loc="upper right")
                # plt.xlabel('Time [hh:ss]')
                # plt.ylabel('Potenze [kW]')
                # plt.title('RESIDENTIAL AREA: ANDAMENTI NEL RANGE DI GUASTO')
                # plt.grid()
                # plt.savefig(path + '/RES.png')
                # plt.cla()
                # plt.close()

                fig, ax = plt.subplots(figsize=(6.2, 5))
                ax.plot(assex, MTBT_g, label='Consumi')
                ax.plot(assex, PV_g, label="Fotovoltaico")
                ax.plot(assex, WF_g, label="Eolico")
                ax.plot(assex, storage, label="Storage")
                ax.plot(assex, DR, label="Demand Response")
                ax.set_title('RESIDENTIAL AREA: ANDAMENTI NEL RANGE DI GUASTO', y=1.15)
                # plt.title('RESIDENTIAL AREA: ANDAMENTI NEL RANGE DI GUASTO')
                # fig.legend(loc='upper right')
                plt.legend(bbox_to_anchor=(0, 1.02, 1, 0.2), loc="lower left", mode="expand", borderaxespad=0,
                           ncol=4, fontsize=8)
                # plt.legend(bbox_to_anchor=(1.05, 0), loc="lower left", fontsize=8)
                ax.set_xlabel('Time [hh:mm]')
                ax.set_ylabel('Potenze [kW]')
                plt.tight_layout()
                # fig.grid()
                fig.savefig(path_img + 'RES.png')
                plt.cla()

                # FUNZIONAMENTO DELLA RETE
                # Legenda
                # Pfer = Potenza prodotta da impianti di generazione da FER
                # Pload = Potenza richiesta dai carichi
                # Pstorage = Potenza disponibile da storage
                # PERC_DR = Percentuale di Demand Response disponibile in  zona - VALORE RICAVATO DALLO SCENARIO SCELTO
                # Plb = Massima potenza che può circolare sul cavo di collegamento o sul trasformatore a monte. DATO FORNITO
DALL'UTENTE!

                # Imposta valori:
                Pfer = round(np.sum(GEN, axis=0))
                Pload = round(np.sum(MTBT_g, axis=0))
                Pstorage = round(np.sum(storage, axis=0))
                Plb = p_backup  # VALORE IMPOSTABILE DALL'UTENTE CHE VUOLE AVVIARE LA SIMULAZIONE [kW]

                # il fattore 0.8 presente nello script è un coefficiente di sicurezza mediante il quale si tiene conto del fatto che
l'energia prodotta dagli impianti di generazione da FER
                # deve compensare anche l'energia reattiva richiesta in zona.

                if linea_backup == 1:
                    if Pfer - Pload > 0:
                        Funz_rete = 'Funzionamento della rete in configurazione grid-off e la linea di backup viene caricata dal
surplus di potenza prodotta da impianti di generazione da FER'
                    elif Pfer - Pload < 0:
                        if Plb >= (Pload - Pfer) / 0.8:
                            Funz_rete = 'Funzionamento della rete in configurazione grid-off grazie alla potenza fornita dalla linea
di backup'
                        elif Plb < ((Pload - Pfer) / 0.8):
                            if Plb >= ((Pload - Pfer - Pstorage) / 0.8):
                                Funz_rete = 'Funzionamento della rete in configurazione grid-off grazie alla potenza fornita dalla
linea di backup e allo storage presente in zona'
                            elif Plb <= ((Pload - Pfer - Pstorage) / 0.8):
                                Pdr = (PERC_DR) * Pload  # si calcola la potenza distaccabile dalla rete grazie a interventi di
Demand Response
                                Pload_dr = Pload - Pdr  # potenza richiesta dai carichi decurtata da quella distaccata a seguito di
azioni di DR
                                if Pfer + Pstorage - Pload_dr > 0:
                                    Funz_rete = 'Funzionamento della rete in configurazione grid-off sfruttando tutte le risorse
flessibili presenti in zona'
                                elif Plb + Pfer + Pstorage - Pload_dr >= 0:
                                    Funz_rete = 'Funzionamento della rete in configurazione grid-off sfruttando tutte le risorse
flessibili presenti in zona e la potenza fornita dalla linea di backup'
                                elif Plb + Pfer + Pstorage - Pload_dr < 0:
                                    Funz_rete = 'La rete non può funzionare in configurazione grid-off'
                elif linea_backup == 0:
                    if Pfer - Pload >= 0:
                        Funz_rete = 'Funzionamento della rete in configurazione grid-off'
                    elif Pfer - Pload < 0:
```

```
                        if Pstorage >= (Pload / 0.8) - Pfer:
                            Funz_rete = 'Funzionamento della rete in configurazione grid-off grazie allo storage'
                        elif Pstorage < (Pload / 0.8) - Pfer:
                            Pdr = (
                                    PERC_DR) * Pload  # si calcola la potenza distaccabile dalla rete grazie a interventi di Demand
Response
                            Pload_dr = Pload - Pdr  # potenza richiesta dai carichi decurtata da quella distaccata a seguito di
azioni di DR
                            if Pfer + Pstorage - (Pload_dr / 0.8) >= 0:
                                Funz_rete = 'Funzionamento della rete in configurazione grid-off sfruttando tutte le risorse
flessibili presenti in zona'
                            elif Pfer + Pstorage - (Pload_dr / 0.8) < 0:
                                Funz_rete = 'La rete non può funzionare in configurazione grid-off'

                print(Funz_rete)

            indexes.append([i1, i2, i3])
            results.append([TI, Ens, Ng, ST, GF, IR])

        return indexes, [m, g, ist_iniziale, ist_finale], results, i_sel, Funz_rete
```

4.2.1.3 roadservices.py

```
import os
import numpy as np
import matplotlib.pyplot as plt

path_img = os.getcwd() + '/_images/Controls/'
path_data = os.getcwd() + '/_functionalities/Controls/'


class RoadServices:
    def calc(self, custom, scenario, lock_ev=True, backup=True, p_backup=10):
        # custom indica se si è scelto uno scenario Standard (custom = False) o Personalizzato (custom = True)
        # se si è scelto uno scenario standard, la variabile 'scenario' indica il nome dello scenario;
        # se si è scelto uno scenario Personalizzato, la variabile 'scenario' indica il vettore dei parametri;

        results = []
        indexes = []
        Funz_rete = ''

        # LETTURA DATI

        # PRODUZIONE - FOTOVOLTAICO
        PV = np.genfromtxt(path_data + 'PV.txt', usecols=(0,1,2,3,4,5,6,7,8,9,10,11) )
        PV = np.transpose(PV)
        # CONSUMI
        consumi= np.genfromtxt(path_data + 'stazione_ric2020.txt', usecols=(0,1,2,3,4) )
        consumi = np.transpose(consumi)

        # ELECTRIC VEHICLE
        # percentuale utenti EV collegati alla rete
        EV_per = np.genfromtxt(path_data + 'EV.txt', usecols=0)
        # consumi per ricarica veloce
        EV_fast_pu = np.genfromtxt(path_data + 'EV_fast.txt', usecols=0)
        #consumi per ricarica lenta
        EV_dumb_pu = np.genfromtxt(path_data + 'EV_dumb.txt', usecols=0)

        # ILLUMINAZIONE NOTTURNA
        size=np.shape(consumi)
        ill_not=consumi[0:(size[0]-1),0:size[1]]

        # SERVIZI STRADALI
        servizi=consumi[4][:]

        # GUARDIOLA CUSTODE
        custode=np.ones(len(servizi))

        # VIDEOSORVEGLIANZA
        video=np.ones(len(servizi))

        # # DATI SCENARI
        # if not custom:
        #     preset_scenarios = ['scen_2020', 'scen_2030BC', 'scen_2030DEC', 'scen_2040BC', 'scen_2040DEC']
        #     i = preset_scenarios.index(scenario)
        #     scen = np.genfromtxt(path + 'scenari1.txt', usecols=[i])
        # else:
        #     scen = scenario

        preset_scenarios = ['scen_2020', 'scen_2030BC', 'scen_2030DEC', 'scen_2040BC', 'scen_2040DEC']

        # -- @AR: Nuova aggiunta confronto scenari---
        if custom:
            preset_scenarios.append('custom')
            i_sel = len(preset_scenarios)-1
        else:
            i_sel = preset_scenarios.index(scenario)

        # Estrazione del mese
        m = np.random.randint(1, 13)
        # m = 6   # TODO: Da eliminare

        M = [[12, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]]
        t = np.shape(M)

        for i in range(t[0]):
            for j in range(t[1]):
                if m == M[i][j]:
                    s = i

        # estrazione istante di guasto
        ist = np.random.randint(1, 93)
        ist = 24    # TODO: Da eliminare

        soc_pv = np.random.randint(10, 91)  # estrazione dello storage dell'impianto PV
        # soc_pv = 30  # TODO Da eliminare

        # # ordine dati vettore:
        # # PICCO CONSUMO; ENERGIA CONSUMO; QUOTA PV; POT. STORAGE; CAP. STORAGE; N. EV
        # scen_2020 = scenari[0][:]
        # scen_2030BC = scenari[1][:]
        # scen_2030DEC = scenari[2][:]
        # scen_2040BC = scenari[3][:]
        # scen_2040DEC = scenari[4][:]

        # DATI SECONDARI
        Pev_fast = 10
        Pev_slow = 3
        C_ev = 40

        # Linea di BackUp ---------------------------------------------
        linea_backup = int(backup)
        if not backup:
            p_backup = 0
        Plb = p_backup
        # -------------------------------------------------------------

        if lock_ev:
            n_col = 10
            n_EV_fast = 0
            n_EV_slow = 0
            n_ev = EV_per[ist] / 100 * n_col
            n_ev = round(n_ev)  # e' necessario un numero intero di veicoli

            ricarica = np.zeros(int(n_ev), dtype=int)
```

```
            for i in range(n_ev):
                ricarica[i] = np.random.randint(1, 3)
                if ricarica[i] == 1:
                    n_EV_fast = n_EV_fast + 1
                elif ricarica[i] == 2:
                    n_EV_slow = n_EV_slow + 1
                if n_EV_fast > (n_col / 2):
                    n_EV_slow = n_EV_slow + (n_EV_fast - n_col/2)
                    n_EV_fast = n_col/2
                elif n_EV_slow > n_col/2:
                    n_EV_fast = n_EV_fast+(n_EV_slow-n_col/2)
                    n_EV_slow = n_col/2
                if (n_EV_fast+n_EV_slow) == n_ev:
                    break

            # n_EV_fast, n_EV_slow, ricarica = 4, 4, [1, 1, 2, 2, 1, 2, 1, 2]  # TODO: Da eliminare

            # creo vettore i cui elementi rappresentano lo stato di carica dei veicoli fast + vettore con i rispettivi tempi di
carica e tempi di scarica
            t_fast_c = np.zeros(int(n_EV_fast), dtype=float)  # CARICA EV FAST
            t_fast_s = np.zeros(int(n_EV_fast), dtype=float)  # SCARICA EV FAST
            SOC_fast = np.zeros(int(n_EV_fast), dtype=int)
            soc = np.zeros(int(n_EV_fast), dtype=int)
            j = 0

            # soc = [28, 56, 19, 10]     # TODO: Da eliminare
            for i in range(int(n_EV_fast)):
                soc[i] = np.random.randint(10, 91)    # TODO: Da ripristinare
                SOC_fast[j] = soc[i]
                t_fast_c[j] = ((90-soc[i])/100) * C_ev/Pev_fast         # carica = assorbimento
                t_fast_s[j] = (np.abs(soc[i]-10)/100) * C_ev/Pev_fast   # scarica = erogazione in V2G
                j=j+1

            print(SOC_fast)

            # creo vettore i cui elementi rappresentano lo stato di carica dei veicoli slow + vettore con i rispettivi tempi di
ricarica
            t_slow = np.zeros(int(n_EV_slow), dtype=float)  # CARICA EV FAST
            SOC_slow = np.zeros(int(n_EV_slow), dtype=int)
            socs = np.zeros(int(n_EV_slow), dtype=int)
            k = 0
            # socs = [22, 57, 73, 63]     # TODO: Da eliminare
            for i in range(int(n_EV_slow)):
                socs[i] = np.random.randint(10, 91)   # TODO: Da ripristinare
                SOC_slow[k] = socs[i]
                t_slow[k] = ((90-socs[i])/100)*C_ev/Pev_slow          # carica = assorbimento
                k = k+1

        for study_case in preset_scenarios:
            if study_case == 'custom':
                scen = scenario
            else:
                i = preset_scenarios.index(study_case)
                scen = np.genfromtxt(path_data + 'scenari1.txt', usecols=[i])

            # POTENZA APPARENTE DEL NODO CUI E' CONNESSA LA RETE [kVA]
            # il valore va impostato in relazione alla rete che si sta analizzando
            an=400

            # #*****************************************************************************
            # # DA QUI SCELGO QUALE SCENARIO SIMULARE TOGLIENDO/INSERENDO IL COMMENTO "#"
            # #*****************************************************************************
            # #scen=scen_2020
            # #scen=scen_2030BC
            # #scen=scen_2030DEC
            # #scen=scen_2040BC
            # scen=scen_2040DEC

            # DATI PRINCIPALI
            Pc=scen[0]          # kW picco consumo
            Cc=scen[1]          # kWh consumo
            Ppv=scen[2]         # potenza nominale PV
            Ppv_ess=scen[3]     # potenza storage
            C_pv=scen[4]        # capacita' storage
            # n_col=scen[5]       # Numnero di veicoli (SPOSTATO)

            ILL=ill_not[s][:]  # assorbimento illuminazione notturna in kW


            PV_pu = PV[m-1][:] # produzione fotovoltaica nel mese estratto [p.u.]
            PV_scen=Ppv*PV_pu     # Produzione fotovoltaica in kW

            time = np.arange(0.,24.,0.25)

            # n. di veicoli collegati nell'istante di guasto

            if not lock_ev:
                n_col = scen[5]  # Numero di veicoli
                # prendo la percentuale di veicoli connessi alla rete nell'istante di guasto
                n_ev = EV_per[ist] / 100 * n_col
                n_ev = round(n_ev)  # e' necessario un numero intero di veicoli

                print(n_ev)
                # numero EV fast ed slow che verranno sovrascritti
                n_EV_fast=0
                n_EV_slow=0
                ricarica=np.zeros(int(n_ev), dtype=int)
                for i in range(n_ev):
                    ricarica[i]=np.random.randint(1,3)
                    if ricarica[i]==1:
                        n_EV_fast=n_EV_fast+1
                    elif ricarica[i]==2:
                        n_EV_slow=n_EV_slow+1
                    if n_EV_fast>(n_col/2):
                        n_EV_slow=n_EV_slow+(n_EV_fast-n_col/2)
                        n_EV_fast=n_col/2
                    elif n_EV_slow>n_col/2:
                        n_EV_fast=n_EV_fast+(n_EV_slow-n_col/2)
                        n_EV_slow=n_col/2
                    if (n_EV_fast+n_EV_slow)==n_ev: break

                print(n_EV_fast)
                print(n_EV_slow)
```

```
            # creo vettore i cui elementi rappresentano lo stato di carica dei veicoli fast + vettore con i rispettivi tempi di
carica e tempi di scarica
            t_fast_c=np.zeros(int(n_EV_fast), dtype=float)  # CARICA EV FAST
            t_fast_s=np.zeros(int(n_EV_fast), dtype=float)  # SCARICA EV FAST
            SOC_fast=np.zeros(int(n_EV_fast), dtype=int)
            soc=np.zeros(int(n_EV_fast), dtype=int)
            j=0

            for i in range(int(n_EV_fast)):
                    soc[i]=np.random.randint(10,91)
                    SOC_fast[j]=soc[i]
                    t_fast_c[j]=((90-soc[i])/100)*C_ev/Pev_fast         # carica = assorbimento
                    t_fast_s[j]=(np.abs(soc[i]-10)/100)*C_ev/Pev_fast   # scarica = erogazione in V2G
                    j=j+1

            print(SOC_fast)

            # creo vettore i cui elementi rappresentano lo stato di carica dei veicoli slow + vettore con i rispettivi tempi di
ricarica
            t_slow=np.zeros(int(n_EV_slow), dtype=float)  # CARICA EV FAST
            SOC_slow=np.zeros(int(n_EV_slow), dtype=int)
            socs=np.zeros(int(n_EV_slow), dtype=int)
            k=0
            for i in range(int(n_EV_slow)):
                    socs[i]=np.random.randint(10,91)
                    SOC_slow[k]=socs[i]
                    t_slow[k]=((90-socs[i])/100)*C_ev/Pev_slow          # carica = assorbimento
                    k=k+1

            print(SOC_slow)

        # storage PV: SOC + tempo di carica (assorbimento) + tempo scarica (erogazione)
        t_carica=((90-soc_pv)/100)*C_pv/Ppv_ess                # tempo in fase di carica
        t_scarica=(abs(soc_pv-10)/100)*C_pv/Ppv_ess            # tempo in fase di scarica

        # IDENTIFICAZIONE DEL RANGE DI GUASTO
        # durata media di interruzione per guasto in MT (60 minuti) e guasto in BT (30 minuti)
        g_MT=1     # considerando 60 minuti di guasto in valori decimali
        g_BT=0.75  # considerando 30 minuti di guasto in valori decimali
        #*********************************************************************************************
        # decido togliendo/inserendo "#" se il guasto e' in MT o in BT
        #*********************************************************************************************
        #guasto=g_MT
        guasto=g_BT

        ist_iniziale=ist*0.25                                  # in termini temporali
        ist_finale=ist_iniziale+guasto
        time_g=np.arange(ist_iniziale,(ist_finale+0.25),0.25)  # mi serve successivamente per plottarlo
        fine=int(ist_finale/0.25)

        # SI RICAVANO I VALORI D'INTERESSE NELL'INTERVALLO DI GUASTO PER EV e STORAGE PV
        # EV FAST - fase di carica
        print(n_EV_fast, time_g, len(time_g))
        P_fast_c=np.zeros((n_EV_fast,len(time_g)), dtype=float)
        q=0
        for i in range(int(n_EV_fast)):
            for j in range(len(time_g)):
                if t_fast_c[q]>guasto or t_fast_c[q]==guasto:
                    P_fast_c[i][j]=Pev_fast
                elif t_fast_c[q]<guasto:
                    for y in range((round(len(time_g)*t_fast_c[q]))):
                        P_fast_c[i][y]=Pev_fast
                    for k in range((round(len(time_g)*t_fast_c[q])),len(time_g)):
                        P_fast_c[i][k]=0
            q=q+1

        # si sommano tutte le righe per avere un unico profilo di assorbimento dedicato alla ricarica veloce: VETTORE DI CARICA
        p_fast_g1=np.zeros((len(t_fast_c)), dtype=float)
        if len(t_fast_c)>1:
            p_fast_g1=np.sum(P_fast_c, axis=0)
        elif len(t_fast_c)<=1:
            p_fast_g1=P_fast_c
            if len(t_fast_c)==0:
                p_fast_g1=np.zeros(len(time_g))

        # EV FAST in fase di scarica
        P_fast_s=np.zeros((n_EV_fast,len(time_g)), dtype=float)
        q=0
        for i in range(int(n_EV_fast)):
            for j in range(len(time_g)):
                if t_fast_s[q]>guasto or t_fast_s[q]==guasto:
                    P_fast_s[i][j]=Pev_fast
                elif t_fast_s[q]<guasto:
                    for y in range((round(len(time_g)*t_fast_s[q]))):
                        P_fast_s[i][y]=Pev_fast
                    for k in range((round(len(time_g)*t_fast_s[q])),len(time_g)):
                        P_fast_s[i][k]=0
            q=q+1

        # si sommano tutte le righe per avere un unico profilo di assorbimento dedicato alla ricarica veloce: VETTORE DI SCARICA
        p_fast_g2=np.zeros((len(t_fast_s)), dtype=float)
        if len(t_fast_s)>1:
            p_fast_g2=np.sum(P_fast_s, axis=0)
        elif len(t_fast_s)<=1:
            p_fast_g2=P_fast_s
            if len(t_fast_s)==0:
                p_fast_g2=np.zeros(len(time_g))

        # EV SLOW
        P_slow=np.zeros((n_EV_slow,len(time_g)), dtype=float)
        l=0
        for i in range(int(n_EV_slow)):
            for j in range(len(time_g)):
                if t_slow[l]>guasto or t_slow[l]==guasto:
                    P_slow[i][j]=Pev_slow
                elif t_slow[l]<guasto:
                    for y in range((round(len(time_g)*t_slow[l]))):
                        P_slow[i][y]=Pev_slow
                    for k in range((round(len(time_g)*t_slow[l])),len(time_g)):
                        P_slow[i][k]=0
            l=l+1

        # si sommano tutte le righe per avere un unico profilo di assorbimento dedicato alla ricarica lenta
        p_slow_g=np.zeros((len(t_slow)), dtype=float)
        if len(t_slow)>1:
```

```
            p_slow_g=np.sum(P_slow, axis=0)
        elif len(t_slow)<=1:
            p_slow_g=P_slow
            if len(t_slow)==0:
                p_slow_g=np.zeros(len(time_g))

        # STORAGE PV
        # CARICA
        storage_c=np.zeros((len(time_g)), dtype=float)

        for i in range(int(len(time_g))):
            if t_carica>guasto or t_carica==guasto:
                    storage_c[i]=Ppv_ess
            elif t_carica<guasto:
                    for y in range((round(len(time_g)*t_carica))):
                        storage_c[y]=Ppv_ess
                    for k in range((round(len(time_g)*t_carica)),len(time_g)):
                        storage_c[k]=0

        # SCARICA
        storage_s=np.zeros((len(time_g)), dtype=float)

        for i in range(int(len(time_g))):
            if t_scarica>guasto or t_scarica==guasto:
                    storage_s[i]=Ppv_ess
            elif t_scarica<guasto:
                    for y in range((round(len(time_g)*t_scarica))):
                        storage_s[y]=Ppv_ess
                    for k in range((round(len(time_g)*t_scarica)),len(time_g)):
                        storage_s[k]=0

        # INTERVALLO DI GUASTO - PRODUZIONE
        # FOTOVOLTAICO
        PV_g=np.zeros(len(time_g), dtype=float)
        i=0
        for j in range(ist,fine+1):
            PV_g[i]=round(PV_scen[j-1],2)
            i=i+1

        # ILLUMINAZIONE NOTTURNA
        ILL_g=np.zeros(len(time_g), dtype=float)
        i=0
        for j in range(ist,fine+1):
            ILL_g[i]=round(ILL[j],2)
            i=i+1
        print(m)
        print(ist)
        print(ILL_g)

        # SERVIZI STRADALI
        servizi_g=np.zeros(len(time_g), dtype=float)
        i=0
        for j in range(ist,fine+1):
            servizi_g[i]=round(servizi[j],2)
            i=i+1

        # GUARDIOLA CUSTODE
        custode_g=np.zeros(len(time_g), dtype=float)
        i=0
        for j in range(ist,fine+1):
            custode_g[i]=round(custode[j],2)
            i=i+1

        # VIDEOSORVEGLIANZA
        video_g=np.zeros(len(time_g), dtype=float)
        i=0
        for j in range(ist,fine+1):
            video_g[i]=round(video[j],2)
            i=i+1

        # CARICO TOTALE = video+custode+servizi+illuminazione (+EV lenti)
        load1 = ILL_g + custode_g + video_g + servizi_g + p_slow_g

        # CARICO TOTALE E LINEA DI BACKUP
        print(min(load1))
        load = load1 - min(min(load1)-0.001, Plb)
        i = 0   # @AR-----------------------------------------------------------------------------------
        for j in range(ist, fine + 1):
            if load[i] < 0:
                load[i] = 0
            i = i + 1

        delta=PV_g-load
        P_fast_g=np.zeros(len(time_g))
        storage=np.zeros(len(time_g))
        nuovo_delta=np.zeros(len(time_g))

        Pev_fast=np.abs(np.sum(p_fast_g1, axis=0)/len(p_fast_g1))

        # LOGICA CARICA
        for i in range(len(time_g)):
            if delta[i]>Ppv_ess and delta[i]>Pev_fast and np.abs(np.sum(delta, axis=0))/(len(time_g))>Ppv_ess and
np.abs(np.sum(delta, axis=0))/(len(time_g))>Pev_fast:
                storage[i]=storage_c[i]
                if delta[i]-storage[i]<p_fast_g1[i]:
                    P_fast_g[i]=delta[i]-storage[i]
                if delta[i]-storage[i]>p_fast_g1[i]:
                    P_fast_g[i]=p_fast_g1[i]
            elif delta[i]>0 and delta[i]<Ppv_ess:
                if t_carica>=guasto:
                    storage[i]=delta[i]            # lo storage si carica con un assorbimento pari alla meta' della differenza
                if p_fast_g1[i]<delta[i]:
                    P_fast_g[i]=p_fast_g1[i]       # gli EV fast si caricano con un assorbimento pari alla meta' della differenza

        # LOGICA DI SCARICA
        if t_scarica>guasto:
            for i in range(len(time_g)):
                if delta[i]<0 and np.abs(delta[i])>Ppv_ess:
                    load[i]=load[i]--p_slow_g[i]-custode_g[i]      # stacco EV lenti + guardiola custode
                    delta[i]=PV_g[i]-load[i]
                    custode_g=np.zeros(len(time_g), dtype=float)
                    p_slow_g=np.zeros(len(time_g), dtype=float)
                    storage[i]=storage_s[i]
                    if np.abs(delta[i])-storage[i]<p_fast_g2[i] and n_EV_fast!=0:
                        P_fast_g[i]=np.abs(delta[i])-storage[i]
```

```
                    elif delta[i]<0 and np.abs(delta[i])<Ppv_ess:
                        load[i]=load[i]-p_slow_g[i]-custode_g[i]
                        delta[i]=PV_g[i]-load[i]
                        custode_g[i]=np.zeros(len(time_g), dtype=float)
                        p_slow_g[i]=np.zeros(len(time_g))
                        storage[i]=np.abs(delta[i])
                        P_fast_g[i]=0

            if t_scarica<=guasto:
                for i in range(len(time_g)):
                    if delta[i]<0:
                        load[i]=load[i]-custode_g[i]-p_slow_g[i]
                        delta[i]=PV_g[i]-load[i]
                        custode_g=np.zeros(len(time_g), dtype=float)
                        p_slow_g=np.zeros(len(time_g), dtype=float)
                        for y in range (round(len(time_g)*t_scarica)):
                            if np.abs(delta[y])>Ppv_ess:
                                storage[y]=storage_s[y]
                                if np.abs(delta[y])-storage[y]<p_fast_g2[y] and n_EV_fast!=0:
                                    P_fast_g[y]=np.abs(delta[y])-storage[y]
                        for k in range((round(len(time_g)*t_scarica)),(len(time_g))):
                            storage[k]=0
                            if np.abs(delta[k])<p_fast_g2[k]:
                                P_fast_g[k]=np.abs(delta[k])
                        for l in range(round(len(time_g)*t_scarica)):
                            if np.abs(delta[l])<=Ppv_ess:
                                storage[l]=np.abs(delta[l])
                                P_fast_g[l]=0
                        for l1 in range(round(len(time_g)*t_scarica),len(time_g)):
                            storage[l1]=0
                            if (np.abs(delta[l1]))<p_fast_g2[l1]:
                                P_fast_g[l1]=np.abs(delta[l1])

            EV_tot=P_fast_g+p_slow_g

            info=np.column_stack((m,ist,fine))
            output=np.column_stack((time_g,PV_g, storage, P_fast_g, p_slow_g, EV_tot, ILL_g, servizi_g, custode_g, video_g, load,
delta))
            print(output)

            # VALORI MEDI
            media=np.mean(output, axis=0)
            print(media)

            # CALCOLO DEGLI INDICATORI DI AFFIDABILITA'
            isola=np.zeros(((len(time_g))-1), dtype=int)
            output_g=output[1:,0:13]
            dim_output_g=np.shape(output_g)
            produzione_storage=np.zeros((dim_output_g[0]), dtype=float)

            intervallo=int(guasto/0.25)
            for i in range(intervallo):
                produzione_storage[i]=output_g[i][1]+output_g[i][2]+output_g[i][3]
                if produzione_storage[i]>=output_g[i][10]:
                    isola[i]=1
                else:
                    isola[i]=0

            # CALCOLO DEGLI INDICATORI CARATTERIZZANTI LO SCENARIO

            # Indicatore "RES" - valuta la percentuale di generazione da fonte rinnovabile rispetto alla generazione complessiva
della rete [%].
            if media[1]>=(media[2]+media[5]+media[6]+media[7]+media[8]+media[9]):
                RES=round(media[1]/(media[2]+media[5]+media[6]+media[7]+media[8]+media[9])*100)
            else:
                RES=round((media[1]/(media[10]))*100)

            # Indicatore FLEX - valuta la flessibilità del carico definita come l'energia che può essere spostata nell'intervallo
temporale di riferimento
            # agendo sui soli carichi flessibili in rapporto all'energia totale richiesta dai carichi nello stesso intervallo [%].

FLEX=round((((Pev_slow*n_EV_slow)+(custode[1]))/(media[6]+media[7]+(custode[1])+media[9]+(Pev_slow*n_EV_slow)+(Pev_fast*n_EV_fast)))*
100)

            # Indicatore BESS - valuta il rapporto tra la potenza fornita dai sistemi di accumulo e la totale potenza dei sistemi di
generazione nella microrete [%].
            BESS=np.zeros(1, dtype=int)
            if media[1]>=(media[2]+media[3]+media[10]):
                BESS=0
            elif media[11]<(media[2]+media[3]+media[10]):
                BESS=round(((media[2]+media[3])/media[10])*100)

            print(RES)
            print(FLEX)
            print(BESS)

            # CALCOLO DEGLI INDICATORI DI AFFIDABILITA'

            # indice di autonomia della microrete "i1" - misura la capacità di funzionamento in isola se richiesto dalla rete
principale [%].
            i1=round(((np.sum(isola, axis=0)*0.25)/guasto)*100)

            # indice di flessibilità "i2" - valuta la variazione di potenza attiva disponibile in aumento o riduzione nel punto di
connessione [%].
            i2=np.zeros(1, dtype=float)
            if media[4]==0 and n_EV_slow>0:
                if media[8]>0:
                    i2=round(((media[1]+media[2]+media[3]+media[8])/an)*100)
                elif media[8]==0:
                    i2=round(((media[1]+media[2]+media[3]+(Pev_slow*n_EV_slow)+custode[1])/an)*100)
            else:
                if media[8]>0:
                    i2=round(((media[1]+media[2]+media[3]+media[8])/an)*100)
                elif media[8]==0:
                    i2=round(((media[1]+media[2]+media[3]+custode[1])/an)*100)

            # indice di capacità di modulazione del profilo di potenza in un tempo convenzionale "i3" -
            # misura la possibilità di variare con continuità per un tempo stabilito il profilo di potenza in un nodo [%].
            i3=np.zeros(1, dtype=float)
            if media[11]>=0:
                i3=100
            else:
                if media[4]==0 and n_EV_slow>0:
                    if media[8]>0:
```

```
i3=round(((media[2]+media[3]+(Pev_slow*n_EV_slow))/(media[2]+media[3]+media[8]+(Pev_slow*n_EV_slow)+(np.abs(media[11]))))*100)
                    elif media[8]==0:

i3=round(((media[2]+media[3]+media[8]+(Pev_slow*n_EV_slow))/(media[2]+media[3]+media[8]+(Pev_slow*n_EV_slow)+(np.abs(media[11]))))*10
0)
                else:
                    if media[8]>0:

i3=round(((media[2]+media[3])/(media[2]+media[3]+media[4]+media[8]+(Pev_slow*n_EV_slow)+(np.abs(media[11]))))*100)
                    else:

i3=round(((media[2]+media[3]+media[8])/(media[2]+media[3]+media[4]+media[8]+(Pev_slow*n_EV_slow)+(np.abs(media[11]))))*100)

            print(i1)
            print(i2)
            print(i3)
            # time_g,PV_g, storage, P_fast_g, p_slow_g, EV_tot, ILL_g, servizi_g, custode_g, video_g, load, delta

            # ********************************************************************** NUOVI INDICATORI
**************************************************************************************************

            # Durata delle interruzioni "TI" - differenza tra la durata media delle interruzioni della microrete (assunta
convenzionalmente pari a 45 min in BT e 60 min in MT)
            # e l'intervallo di tempo (Δt) espresso in minuti in cui, azionando le dovute logiche di controllo, si riescono ad
alimentare almeno le utenze critiche della microrete [min].

            TI = (guasto*60)-((np.sum(isola, axis=0)*0.25)*60)
            print(TI)

            # Energia non fornita "Ens" - rapporto tra l'energia che i carichi della microrete ricevono durante l'evento di failure e
l'energia che gli stessi carichi
            # avrebbero richiesto in assenza di guasto secondo il previsto profilo giornaliero di consumo [adim.].

            Ens = round(((np.sum(PV_g)+np.sum(storage)+np.sum(P_fast_g))*0.25)/(np.sum(load)*0.25),2)

            print(Ens)

            # Generazione flessible "Ng" - indicatore che tiene conto della presenza della generazione flessibile sul totale della
generazione della microrete operante in isola [adim.].

            Kpv=50  # [%] Valore stabilito dall'untente - Percentuale di generezione fotovoltaica controllabile rispetto totale
installata

            Ng= round((((Kpv/100)*Ppv)/(Ppv)),2)
            print(Ng)

            # Riserva dello storage "ST" - indicatore che valuta la riserva di energia dei sistemi di accumulo installati nella
microrete rispetto al consumo giornaliero della microrete.

            Eday= np.sum(ILL)*0.25+np.sum(servizi)*0.25+np.sum(custode)*0.25+np.sum(video)*0.25             # energia richiesta dai
carichi della microrete in isola in una giornata [kWh]
            ST = round(((0.8*C_pv)/(1.05*1.11*Eday)),2)

            print(ST)

            # Capacità di grid forming "GF" - rapporto tra la potenza dei convertitori funzionanti in grid-forming e la potenza
totale necessaria [adim.]

            GF = round(((Ppv+Ppv_ess)/(((np.max(ILL))+(np.max(servizi))+(np.max(custode))+(np.max(video)))*0.1667)),2)
            print(GF)

            # Rapporto di inerzia "IR" - rapporto tra l'inerzia dei convertitori esistenti e quella necessaria riproporzionata sulla
potenza effettivamente installata.

            H = 5    # [s] inerzia del sistema - Valore definito dall'utente
            IR = round(((H*(Ppv+Ppv_ess))/(3*(((np.max(ILL))+(np.max(servizi))+(np.max(custode))+(np.max(video)))*0.1667))),2)

            print(IR)

            #
**************************************************************************************************************************************************************
**********************************************

            # ANDAMENTO DEI VALORI NELL'INTERVALLO DI GUASTO
            if custom or scenario == study_case:

                # ASSE TEMPORALE - CONVERSIONE DECIMALE -> SESSAGESIMALE
                tempo=time_g
                ore=np.zeros(len(tempo), dtype=int)
                minuti=np.zeros(len(tempo), dtype=int)
                assex=np.zeros(len(tempo), dtype=list)

                for i in range(len(tempo)):
                    ore[i]=int(tempo[i])
                    minuti[i]=int((tempo[i]-ore[i])*60) & 63
                    assex[i]="%d:%d" % (ore[i], minuti[i])

                fig, ax = plt.subplots(figsize=(6.2, 5))
                ax.plot(assex, PV_g, label="Fotovoltaico")
                ax.plot(assex, storage, label="Storage")
                ax.plot(assex, P_fast_g, label="Ricarica veloce")
                ax.plot(assex, p_slow_g, label="Ricarica lenta")
                ax.plot(assex, ILL_g, label="Illuminazione")
                ax.plot(assex, servizi_g, label="Servizi stradali")
                ax.plot(assex, custode_g, label="Guardiola custode")
                ax.plot(assex, video_g, label="videosorveglianza")
                ax.plot(assex, load, label="Carico")
                # fig.legend(loc="upper right")
                ax.set_xlabel('Time [hh:ss]')
                ax.set_ylabel('Potenze [kW]')
                ax.set_title('EVH AREA: ANDAMENTI NEL RANGE DI GUASTO', y=1.15)
                plt.legend(bbox_to_anchor=(0, 1.02, 1, 0.2), loc="lower left", mode="expand", borderaxespad=0,
                           ncol=4, fontsize=8)
                plt.tight_layout()

                fig.savefig(path_img + 'RS.png')
                plt.cla()

                # FUNZIONAMENTO DELLA RETE
                # Legenda
                # Pfer = Potenza prodotta da impianti di generazione da FER
                # Pload = Potenza richiesta dai carichi
                # Pstorage = Potenza disponibile da storage
```

```
                # Plb = Massima potenza che può circolare sul cavo di collegamento o sul trasformatore a monte. DATO FORNITO
DALL'UTENTE!

                # Imposta valori:
                Pfer = round(np.sum(PV_g, axis=0))
                Pload = round(np.sum(load, axis=0))
                Pstorage = round(np.sum(storage, axis=0)) + round(np.sum(P_fast_g, axis=0))
                Pdr = round(np.sum(p_slow_g, axis=0)) + round(np.sum(custode_g, axis=0))  # potenza distaccabile dalla rete grazie a
interventi di Demand Response [kW]
                Plb = p_backup  # VALORE IMPOSTABILE DALL'UTENTE CHE VUOLE AVVIARE LA SIMULAZIONE [kW]

                # il fattore 0.8 presente nello script è un coefficiente di sicurezza mediante il quale si tiene conto del fatto che
l'energia prodotta dagli impianti di generazione da FER
                # deve compensare anche l'energia reattiva richiesta in zona.

                if linea_backup == 1:
                    if Pfer - Pload > 0:
                        Funz_rete = 'Funzionamento della rete in configurazione grid-off e la linea di backup viene caricata dal
surplus di potenza prodotta da impianti di generazione da FER'
                    elif Pfer - Pload < 0:
                        if Plb >= (Pload - Pfer) / 0.8:
                            Funz_rete = 'Funzionamento della rete in configurazione grid-off grazie alla potenza fornita dalla linea
di backup'
                        elif Plb < ((Pload - Pfer) / 0.8):
                            if Plb >= ((Pload - Pfer - Pstorage) / 0.8):
                                Funz_rete = 'Funzionamento della rete in configurazione grid-off grazie alla potenza fornita dalla
linea di backup e allo storage presente in zona'
                            elif Plb <= ((Pload - Pfer - Pstorage) / 0.8):
                                Pload_dr = Pload - Pdr  # potenza richiesta dai carichi decurtata da quella distaccata a seguito di
azioni di DR
                                if Pfer + Pstorage - Pload_dr > 0:
                                    Funz_rete = 'Funzionamento della rete in configurazione grid-off sfruttando tutte le risorse
flessibili presenti in zona'
                                elif Plb + Pfer + Pstorage - Pload_dr >= 0:
                                    Funz_rete = 'Funzionamento della rete in configurazione grid-off sfruttando tutte le risorse
flessibili presenti in zona e la potenza fornita dalla linea di backup'
                                elif Plb + Pfer + Pstorage - Pload_dr < 0:
                                    Funz_rete = 'La rete non può funzionare in configurazione grid-off'
                elif linea_backup == 0:
                    if Pfer - Pload >= 0:
                        Funz_rete = 'Funzionamento della rete in configurazione grid-off'
                    elif Pfer - Pload < 0:
                        if Pstorage >= (Pload / 0.8) - Pfer:
                            Funz_rete = 'Funzionamento della rete in configurazioe grid-off grazie allo storage'
                        elif Pstorage < (Pload / 0.8) - Pfer:
                            Pload_dr = Pload - Pdr  # potenza richiesta dai carichi decurtata da quella distaccata a seguito di
azioni di DR
                            if Pfer + Pstorage - (Pload_dr / 0.8) >= 0:
                                Funz_rete = 'Funzionamento della rete in configurazione grid-off sfruttando tutte le risorse
flessibili presenti in zona'
                            elif Pfer + Pstorage - (Pload_dr / 0.8) < 0:
                                Funz_rete = 'La rete non può funzionare in configurazione grid-off'

                print(Funz_rete)

            indexes.append([i1, i2, i3])
            results.append([TI, Ens, Ng, ST, GF, IR])

        return indexes, [m, 0, ist_iniziale, ist_finale], results, i_sel, Funz_rete
```

4.2.1.4 underground.py

```
import os
import numpy as np
import matplotlib.pyplot as plt

path_img = os.getcwd() + '/_images/Controls/'
path_data = os.getcwd() + '/_functionalities/Controls/'


class Underground:
    def calc(self, custom, scenario, lock_ev=False, backup=True, p_backup=100):
        results = []
        indexes = []
        Funz_rete = ''

        # custom indica se si è scelto uno scenario Standard (custom = False) o Personalizzato (custom = True)
        # se si è scelto uno scenario standard, la variabile 'scenario' indica il nome dello scenario;
        # se si è scelto uno scenario Personalizzato, la variabile 'scenario' indica il vettore dei parametri;

        # LETTURA DATI

        # PRODUZIONE - FOTOVOLTAICO
        PV = np.genfromtxt(path_data + 'PV.txt', usecols=(0,1,2,3,4,5,6,7,8,9,10,11) )
        PV = np.transpose(PV)
        # CONSUMI - METRO
        metro = np.genfromtxt(path_data + 'metro.txt', usecols=(0,1,2,3,4,5,6,7,8,9,10,11) )
        metro = np.transpose(metro)


        # ELECTRIC VEHICLE
        # percentuale utenti EV collegati alla rete
        EVperc = np.genfromtxt(path_data + 'EV.txt', usecols=0)
        # consumi per ricarica veloce
        EVfast = np.genfromtxt(path_data + 'EV_fast.txt', usecols=0)
        #consumi per ricarica lenta
        EVdumb = np.genfromtxt(path_data + 'EV_dumb.txt', usecols=0)

        # # DATI SCENARI
        # if not custom:
        #     preset_scenarios = ['scen_2020', 'scen_2040']
        #     i = preset_scenarios.index(scenario)
        #     scen = np.genfromtxt(path + 'scenari_underground.txt', usecols=[i])
        # else:
        #     scen = scenario

        preset_scenarios = ['scen_2020', 'scen_2040']

        # -- @AR: Nuova aggiunta confronto scenari---
        if custom:
            preset_scenarios.append('custom')
            i_sel = len(preset_scenarios)-1
        else:
            i_sel = preset_scenarios.index(scenario)

        # Estrazione del mese
        m = np.random.randint(1,13)
        m = 6   # TODO: Da eliminare


        PV_pu = PV[m-1][:] # produzione fotovoltaica nel mese estratto [p.u.]

        time = np.arange(0.,24.,0.25)

        # estrazione istante di guasto
        ist = np.random.randint(1,93)
        # ist = 50    # TODO: Da eliminare


        # durata media di interruzione per guasto in MT (60 minuti) e guasto in BT (30 minuti)
        g_MT = 1  # considerando 60 minuti di guasto in valori decimali
        g_BT = 0.75  # considerando 30 minuti di guasto in valori decimali
        # ********************************************************************************
        # decido togliendo/inserendo "#" se il guasto e' in MT o in BT
        # ********************************************************************************
        # guasto=g_MT
        guasto = g_BT

        ist_iniziale = ist * 0.25  # in termini temporali
        ist_finale = ist_iniziale + guasto
        time_g = np.arange(ist_iniziale, (ist_finale + 0.25), 0.25)  # mi serve successivamente per plottarlo
        fine = int(ist_finale / 0.25)

        # STORAGE
        # storage totale: SOC + tempo di carica (assorbimento) + tempo scarica (erogazione)
        soc = np.random.randint(10, 91)
        # soc = 20    # TODO: Da eliminare

        # Linea di BackUp ------------------------------------------
        linea_backup = int(backup)
        if not backup:
            p_backup = 0
        Plb = p_backup
        # ----------------------------------------------------------

        #
        for study_case in preset_scenarios:
            if study_case == 'custom':
                scen = scenario
            else:
                i = preset_scenarios.index(study_case)
                scen = np.genfromtxt(path_data + 'scenari_underground.txt', usecols=[i])
            # ordine dati vettore:
            # Consumi metro - PV - kW STORAGE - kWh STORAGE
            # scen_2020 = scenari[0][:]
            # scen_2040 = scenari[1][:]

            # POTENZA APPARENTE DEL NODO CUI E' CONNESSA LA RETE [kVA]
            # il valore va impostato in relazione alla rete che si sta analizzando
            an=400


            #*************************************************************************
```

```
            # DA QUI SCELGO QUALE SCENARIO SIMULARE TOGLIENDO/INSERENDO IL COMMENTO "#"
            #*****************************************************************************
            # scen=scen_2020
            # #scen=scen_2040

            # DATI PRINCIPALI
            Cm=scen[0] # capacita' kWh METROPOLITANA
            Ppv=scen[1] # potenza nominale PV
            P_ess=scen[2] # potenza storage (valore ipotizzato)
            C_ess=scen[3] # capacita' storage (valore ipotizzato)

            # PROFILI IN kW
            METRO=metro[m-1][:] # assorbimento metropolitana in kW
            PV_scen=Ppv*PV_pu     # Produzione fotovoltaica in kW

            t_carica=round(((90-soc)/100)*C_ess/P_ess,2);           # tempo di carica
            t_scarica=round((np.abs(soc-10)/100)*C_ess/P_ess,2);    # tempo di scarica

            # LOGICA DI CARICA
            storage_c=np.zeros((len(time_g)), dtype=float)
            for i in range(len(time_g)):
                if t_carica>guasto or t_carica==guasto:
                    storage_c[i]=P_ess
                elif t_carica<guasto:
                    for y in range((round(len(time_g)*t_carica))):
                        storage_c[y]=P_ess
                    for k in range((round(len(time_g)*t_carica)),(len(time_g))):
                            storage_c[k]=0

            # LOGICA DI SCARICA
            storage_s=np.zeros((len(time_g)), dtype=float)
            for i in range(len(time_g)):
                if t_scarica>guasto or t_scarica==guasto:
                    storage_s[i]=P_ess
                elif t_scarica<guasto:
                    for y in range((round(len(time_g)*t_scarica))):
                        storage_s[y]=P_ess
                    for k in range((round(len(time_g)*t_scarica)),(len(time_g))):
                            storage_s[k]=0

            # INTERVALLO DI GUASTO - CONSUMI
            METRO_g1 = np.zeros(len(time_g), dtype=float)
            i = 0
            for j in range(ist, fine + 1):
                METRO_g1[i] = round(METRO[j - 1], 2)
                i = i + 1

            # INTERVALLO DI GUASTO - CONSUMI E LINEA DI BACKUP
            METRO_g = np.zeros(len(time_g), dtype=float)
            i = 0
            for j in range(ist, fine + 1):
                METRO_g[i] = max(0.000001, round((METRO_g1[i] - Plb), 2))
                if METRO_g[i] < 0:
                    METRO_g[i] = 0
                i = i + 1

            # INTERVALLO DI GUASTO - PRODUZIONE
            # FOTOVOLTAICO
            PV_g=np.zeros(len(time_g), dtype=float)
            i=0
            for j in range(ist,fine+1):
                PV_g[i]=round(PV_scen[j-1],2)
                i=i+1

            # STORAGE

            GEN=PV_g
            delta=GEN-METRO_g
            nuovo_delta=np.zeros(len(time_g))
            DR=np.zeros(len(time_g))
            storage=np.zeros(len(time_g))

            # LOGICA CARICA
            for i in range(len(time_g)):
                if delta[i]>P_ess and (sum(delta)/len(time_g))>P_ess:
                    storage[i]=storage_c[i]
                    nuovo_delta[i]=delta[i]-storage[i]
                    if nuovo_delta[i]>0 or nuovo_delta[i]<0:
                        DR[i]=nuovo_delta[i]
                elif delta[i]<P_ess:
                    storage[i]=delta[i]

            # LOGICA DI SCARICA
            for i in range(len(time_g)):
                if delta[i]<0 and np.abs(sum(delta)/len(time_g))>P_ess:
                    storage[i]=storage_s[i]
                    nuovo_delta[i]=delta[i]+storage[i]
                    if nuovo_delta[i]>0 or nuovo_delta[i]<0:
                        DR[i]=nuovo_delta[i]
                elif delta[i]<0 and np.abs(sum(delta)/len(time_g))<P_ess:
                    for y in range(len(time_g)):
                        storage[y]=np.abs(delta[y])
                    for k in range((round(len(time_g)*t_scarica)),(len(time_g))):
                        storage[k]=0
                        DR[k]=delta[k]

            info=np.column_stack((m,ist,fine,soc))
            output=np.column_stack((time_g,METRO_g,PV_g,storage,DR))
            print(info)
            print(output)

            # CALCOLO DEGLI INDICATORI CARATTERIZZANTI LO SCENARIO
            media=np.mean(output, axis=0)

            # Indicatore "RES" - valuta la percentuale di generazione da fonte rinnovabile rispetto alla generazione complessiva
della rete [%].
            RES=round(((media[2])/(media[1]))*100)

            # Indicatore FLEX - valuta la flessibilità del carico definita come l'energia che può essere spostata nell'intervallo
temporale di riferimento
            # agendo sui soli carichi flessibili in rapporto all'energia totale richiesta dai carichi nello stesso intervallo [%].
            FLEX=round(np.abs(media[4]/media[1])*100)

            # Indicatore BESS - valuta il rapporto tra la potenza fornita dai sistemi di accumulo e la totale potenza dei sistemi di
generazione nella microrete [%].
```

```
            BESS=round((media[3]/media[1])*100)

            # CALCOLO DEGLI INDICATORI DI AFFIDABILITA'
            isola=np.zeros(((len(time_g))-1), dtype=int)
            output_g=output[1:,0:6]
            dim_output_g=np.shape(output_g)
            produzione_storage=np.zeros((dim_output_g[0]), dtype=float)
            intervallo=int(guasto/0.25)
            for i in range(intervallo):
                produzione_storage[i]=output_g[i][2]+output_g[i][3]
                if produzione_storage[i]==output_g[i][1]:
                    isola[i]=1
                else:
                    isola[i]=0

            # indice di autonomia della microrete "i1" - misura la capacità di funzionamento in isola se richiesto dalla rete
principale [%].
            i1 = round(((np.sum(isola, axis=0)*0.25)/guasto)*100, 4)

            # indice di flessibilità "i2" - valuta la variazione di potenza attiva disponibile in aumento o riduzione nel punto di
connessione [%].
            i2=round(((media[2]+media[3]+(np.abs(media[4])))/an)*100, 4)

            # indice di capacità di modulazione del profilo di potenza in un tempo convenzionale "i3" -
            # misura la possibilità di variare con continuità per un tempo stabilito il profilo di potenza in un nodo [%].
            i3=round(media[3]/(media[3]+(np.abs(media[4])))*100, 4)

            print(media)
            print(i1)
            print(i2)
            print(i3)

            # ********************************************************************** NUOVI INDICATORI
**********************************************************************************************

            # Durata delle interruzioni "TI" - differenza tra la durata media delle interruzioni della microrete (assunta
convenzionalmente pari a 45 min in BT e 60 min in MT)
            # e l'intervallo di tempo (Δt) espresso in minuti in cui, azionando le dovute logiche di controllo, si riescono ad
alimentare almeno le utenze critiche della microrete [min].

            TI = (guasto*60)-((np.sum(isola, axis=0)*0.25)*60)
            print(TI)

            # Energia non fornita "Ens" - rapporto tra l'energia che i carichi della microrete ricevono durante l'evento di failure e
l'energia che gli stessi carichi
            # avrebbero richiesto in assenza di guasto secondo il previsto profilo giornaliero di consumo [adim.].

            Ens = round(1-((np.sum(PV_g)+np.sum(storage))*0.25)/(np.sum(METRO_g)*0.25), 4)

            print(Ens)

            # Generazione flessible "Ng" - indicatore che tiene conto della presenza della generazione flessibile sul totale della
generazione della microrete operante in isola [adim.].

            Kpv=50  # Valore stabilito dall'untente - Percentuale di generezione fotovoltaica controllabile rispetto totale
installata

            Ng= round((((Kpv/100)*Ppv)/(Ppv)), 4)
            print(Ng)

            # Riserva dello storage "ST" - indicatore che valuta la riserva di energia dei sistemi di accumulo installati nella
microrete rispetto al consumo giornaliero della microrete.

            Eday= np.sum(METRO)*0.25                              # energia richiesta dai carichi della microrete in isola in una
giornata [kWh]
            ST = round(((0.8*C_ess)/(1.05*1.11*Eday)), 4)

            print(ST)

            # Capacità di grid forming "GF" - rapporto tra la potenza dei convertitori funzionanti in grid-forming e la potenza
totale necessaria [adim.]

            GF = round((Ppv+P_ess)/((np.max(METRO)*0.1667)), 4)
            print(GF)

            # Rapporto di inerzia "IR" - rapporto tra l'inerzia dei convertitori esistenti e quella necessaria riproporzionata sulla
potenza effettivamente installata.

            H = 5    # [s] inerzia del sistema - Valore definito dall'utente
            IR = round(((H*(Ppv+P_ess))/(3*(np.max(METRO)*0.1667))), 4)

            print(IR)

            #
**********************************************************************************************************************************************************
**********************************************

            # ANDAMENTO DEI VALORI NELL'INTERVALLO DI GUASTO

            # ASSE TEMPORALE - CONVERSIONE DECIMALE -> SESSAGESIMALE
            if custom or scenario == study_case:
                tempo=time_g
                ore=np.zeros(len(tempo), dtype=int)
                minuti=np.zeros(len(tempo), dtype=int)
                assex=np.zeros(len(tempo), dtype=list)

                for i in range(len(tempo)):
                    ore[i]=int(tempo[i])
                    minuti[i]=int((tempo[i]-ore[i])*60) & 63
                    assex[i]="%d:%d" % (ore[i], minuti[i])

                fig, ax = plt.subplots(figsize=(6.2, 5))
                ax.plot(assex,METRO_g, label="Consumi")
                ax.plot(assex,PV_g,label="Fotovoltaico")
                ax.plot(assex,storage,label="Storage")
                ax.plot(assex,DR,label="Demand Response")
                ax.set_xlabel('Time [hh:ss]')
                ax.set_ylabel('Potenze [kW]')
                ax.set_title('UNDERGROUND AREA: ANDAMENTI NEL RANGE DI GUASTO', y=1.10)
                plt.legend(bbox_to_anchor=(0, 1.02, 1, 0.2), loc="lower left", mode="expand", borderaxespad=0,
                           ncol=4, fontsize=8)
                plt.tight_layout()
                fig.savefig(path_img + 'UG.png')
                plt.cla()
```

```
                # FUNZIONAMENTO DELLA RETE
                # Legenda
                # Pfer = Potenza prodotta da impianti di generazione da FER
                # Pload = Potenza richiesta dai carichi
                # Pstorage = Potenza disponibile da storage
                # PERC_DR = Percentuale di Demand Response disponibile in  zona - VALORE RICAVATO DALLO SCENARIO SCELTO
                # Plb = Massima potenza che può circolare sul cavo di collegamento o sul trasformatore a monte. DATO FORNITO
DALL'UTENTE!

                # Imposta valori:
                Pfer = round(np.sum(GEN, axis=0))
                Pload = round(np.sum(METRO_g, axis=0))
                Pstorage = round(np.sum(storage, axis=0))
                PERC_DR = 0
                Plb = p_backup  # VALORE IMPOSTABILE DALL'UTENTE CHE VUOLE AVVIARE LA SIMULAZIONE [kW]

                # il fattore 0.8 presente nello script è un coefficiente di sicurezza mediante il quale si tiene conto del fatto che
l'energia prodotta dagli impianti di generazione da FER
                # deve compensare anche l'energia reattiva richiesta in zona.

                if linea_backup == 1:
                    if Pfer - Pload > 0:
                        Funz_rete = 'Funzionamento della rete in configurazione grid-off e la linea di backup viene caricata dal
surplus di potenza prodotta da impianti di generazione da FER'
                    elif Pfer - Pload < 0:
                        if Plb >= (Pload - Pfer) / 0.8:
                            Funz_rete = 'Funzionamento della rete in configurazione grid-off grazie alla potenza fornita dalla linea
di backup'
                        elif Plb < ((Pload - Pfer) / 0.8):
                            if Plb >= ((Pload - Pfer - Pstorage) / 0.8):
                                Funz_rete = 'Funzionamento della rete in configurazione grid-off grazie alla potenza fornita dalla
linea di backup e allo storage presente in zona'
                            elif Plb <= ((Pload - Pfer - Pstorage) / 0.8):
                                Pdr = (
                                        PERC_DR) * Pload  # si calcola la potenza distaccabile dalla rete grazie a interventi di
Demand Response
                                Pload_dr = Pload - Pdr  # potenza richiesta dai carichi decurtata da quella distaccata a seguito di
azioni di DR
                                if Pfer + Pstorage - Pload_dr > 0:
                                    Funz_rete = 'Funzionamento della rete in configurazione grid-off sfruttando tutte le risorse
flessibili presenti in zona'
                                elif Plb + Pfer + Pstorage - Pload_dr >= 0:
                                    Funz_rete = 'Funzionamento della rete in configurazione grid-off sfruttando tutte le risorse
flessibili presenti in zona e la potenza fornita dalla linea di backup'
                                elif Plb + Pfer + Pstorage - Pload_dr < 0:
                                    Funz_rete = 'La rete non può funzionare in configurazione grid-off'
                elif linea_backup == 0:
                    if Pfer - Pload >= 0:
                        Funz_rete = 'Funzionamento della rete in configurazione grid-off'
                    elif Pfer - Pload < 0:
                        if Pstorage >= (Pload / 0.8) - Pfer:
                            Funz_rete = 'Funzionamento della rete in configurazione grid-off grazie allo storage'
                        elif Pstorage < (Pload / 0.8) - Pfer:
                            Pdr = (
                                    PERC_DR) * Pload  # si calcola la potenza distaccabile dalla rete grazie a interventi di Demand
Response
                            Pload_dr = Pload - Pdr  # potenza richiesta dai carichi decurtata da quella distaccata a seguito di
azioni di DR
                            if Pfer + Pstorage - (Pload_dr / 0.8) >= 0:
                                Funz_rete = 'Funzionamento della rete in configurazione grid-off sfruttando tutte le risorse
flessibili presenti in zona'
                            elif Pfer + Pstorage - (Pload_dr / 0.8) < 0:
                                Funz_rete = 'La rete non può funzionare in configurazione grid-off'

                print(Funz_rete)

            indexes.append([i1, i2, i3])
            results.append([TI, Ens, Ng, ST, GF, IR])

        return indexes, [m, 0, ist_iniziale, ist_finale], results, i_sel, Funz_rete
```

4.2.2 EMS

4.2.2.1 EMS.py

```
from .EMS_stoch_sim import *
from .BaseCase_IndicesCalculation import *


class EMS:
    """
    EMS class
    """

    def __init__(self, config, data_file_profiles="", solver_name="glpk"):
        """
        Initialization of the EMS

        Parameters
        ----------
        config: str or dict
            Configuration file storing the network layout, the components and the data of the power time series
        fault_scenarios: str or dict
            Scenario configuration for the stochastic simulations
        data_file_profiles: str
            Optional path to a csv file used to store time series to be used in the EMS simulations
        solver_name: str
            Solver name
        """

        # load the config
        if type(config) is str:
            file = open(config)
            self.config = yaml.load(file, Loader=yaml.Loader)
        elif type(config) is dict:
            self.config = config
        elif type(config) is not dict:
            raise Exception(f"Type of config ({type(config)}) is not recognized, it shall be a valid string or dictionary")

        self.data_file_profiles = data_file_profiles
        self.solver_name = solver_name

    #
    def execute_EMS(self, generate_plots=False, output_folder="./Img/"):
        """
        Function to execute the deterministic EMS only and output the results

        Parameters
        ----------
        generate_plots: bool
            When true, output plots are saved
        output_folder:
            When generate_plots is true, plots are saved in the folder specified by output_folder

        Outputs
        -------
        reliability_indices: dict
            Dictionary where the reliability outputs are stored
        model: Pyomo model
            Pyomo model of the EMS
        results: dict
            Results collection. The keys of this dictionary are the node names and the content is the dispatch of the resources by
node
        config: dict
            Configuration dictionary used in the EMS
        """
        return run_deterministic_EMS(self.config,
                                     data_file_profiles=self.data_file_profiles,
                                     solver_name=self.solver_name,
                                     generate_plots=generate_plots,
                                     ouput_folder=output_folder)

    #
    def execute_reliability_analysis(self,
                                     fault_scenarios="",
                                     generate_plots=False,
                                     output_folder_base="./Img/Img_bc/Img_%s/",
                                     output_folder_stoch="./Img/Img_s/Img_%s/"):
        """
        Function to execute the stochastic simulations on the deterministic simulation of the EMS
        This function executes the stochastic simulations on the deterministic EMS and compare them with the results
        of the base case configuration to show the benefits

        Parameters
        ----------
        generate_plots: bool
            When true, output plots are saved
        output_folder_base:
            When generate_plots is true, plots of the base case operation are saved in the folder specified by output_folder_base
        output_folder_stoch:
            When generate_plots is true, plots of the stochastic simulations are saved in the folder specified by output_folder_stoch

        Outputs
        -------
        reliability_indices: dict
            Dictionary where the reliability outputs are stored
            reliability_indices["Base case"] contains the reliability indices for the base case management
            reliability_indices["EMS model"] contains the reliability indices for the EMS

            Each of the two dictionary values, contain the reliability indices for every node and for the system.
            - reliability_indices[case][code_all_system] contains the reliability indices for the entire system
            - whereas reliability_indices[case][node_name] contains the reliability indices for the node node_name
            - the reliability indices that are considered are the following:
              - Energy Not Served (ENS): expected value of the energy not served among the considered scenarios
              - Energy Not Served cost (ENS_cost): expected value of the cost for the energy not served among the considered
scenarios
              - ENS per unit (ENS_pu): expected per unit ENS with respect to the demand
              - ENS cost per unit (ENS_cost_pu): expected per unit ENC cost with respect to the demand

        """

        # load the config
        if type(fault_scenarios) is str:
```

```
        file = open(fault_scenarios)
        self.fault_scenarios = yaml.load(file, Loader=yaml.Loader)
    elif type(fault_scenarios) is dict:
        self.fault_scenarios = fault_scenarios
    elif type(fault_scenarios) is not dict:
        raise Exception(
            f"Type of fault_scenarios ({type(fault_scenarios)}) is not recognized, it shall be a valid string or dictionary")

    # base case simulations
    print("Run base case simulations")
    reliability_indices_bc, reliability_indices_bc_raw, model_bc, results_bc, config_bc = \
        run_base_case_simulations(self.config,
            self.fault_scenarios,
            self.solver_name,
            data_file_profiles=self.data_file_profiles,
            generate_plots=generate_plots,
            output_folder=output_folder_base)

    # stochastic simulations
    print("Run EMS simulations")
    reliability_indices_s, reliability_indices_s_raw, model_s, results_s, config_s = \
        run_full_stoch_simulations(self.config,
            self.fault_scenarios,
            self.solver_name,
            data_file_profiles=self.data_file_profiles,
            generate_plots=generate_plots,
            output_folder=output_folder_stoch)

    # textual print of the results
    models = ["Base case", "EMS model"]
    reliability_indices = {"Base case":reliability_indices_bc,
                           "EMS model":reliability_indices_s}

    print("Results of the optimization (the lower the better):")
    text_code = "%18s|%14s|%14s"
    text_results_code = "%18s|%14.4f|%14.4f"
    print(text_code % ("Indicator", "Base case", "EMS model"))
    for k_v in ["ENS", "ENS_cost"]:  #, "ENS_pu", "ENS_cost_pu"
        print(text_results_code % (k_v,
                           reliability_indices["Base case"][code_all_system][k_v],
                           reliability_indices["EMS model"][code_all_system][k_v]
                           ))

    # list consumption blocks
    list_nodes = list(set(list(reliability_indices["Base case"].keys())) - set([code_all_system]))

    # calculation indices by specific
    # Energia totale non fornita per le interruzioni in un dato periodo di riferimento
    ENS = {
        "Base case": reliability_indices["Base case"][code_all_system]["ENS"],
        "EMS model": reliability_indices["EMS model"][code_all_system]["ENS"]
    }
    # energia non fornita per il nodo n per le interruzioni in un dato periodo di riferimento
    LPENS = {
        "Base case": {n: reliability_indices["Base case"][n]["ENS"] for n in list_nodes},
        "EMS model": {n: reliability_indices["EMS model"][n]["ENS"] for n in list_nodes}
    }
    # costo totale atteso per le interruzioni in un dato periodo di riferimento
    EIC = {
        "Base case": reliability_indices["Base case"][code_all_system]["ENS_cost"],
        "EMS model": reliability_indices["EMS model"][code_all_system]["ENS_cost"]
    }
    # costo medio di interruzione per il nodo n in un dato periodo di riferimento
    LPEIC = {
        "Base case": {n: reliability_indices["Base case"][n]["ENS_cost"] for n in list_nodes},
        "EMS model": {n: reliability_indices["EMS model"][n]["ENS_cost"] for n in list_nodes}
    }

    reliability_indices_ENEA = {
        "ENS": ENS,
        "LPENS": LPENS,
        "EIC": EIC,
        "LPEIC": LPEIC
    }

    reliability_indices_raw = {
        "Base case": reliability_indices_bc_raw,
        "EMS model": reliability_indices_s_raw
    }

    return reliability_indices_ENEA, reliability_indices, reliability_indices_raw
```

4.2.2.2 EMS_Stoch_sim.py

```
"""
This module provides the main functions for the stochastic evaluation of the reliability indices
for the deterministic EMS method
"""

from .EMS_module import *


def optimal_dispatch_stoch(prev_model, t_step, scenario, config,
                           data_file_profiles="",
                           solver_name="glpk"):

    # load the config
    if type(config) is str:
        file = open(config)
        config = yaml.load(file, Loader=yaml.Loader)
    elif type(config) is not dict:
        raise Exception(f"Type of config ({type(config)}) is not recognized, it shall be a valid string or dictionary")
    else:
        config = config

    # number of time steps and vector of time steps
    n_steps = int(get_parameter(config, "n_steps"))
    time_steps = range(0, n_steps)

    # nodes of the network
    nodes = get_nodes_list(config)
    n_nodes = len(nodes)

    # connection arcs
    links = get_links_list(config)

    # populate the config dictionary with the profiles of the resources if needed
    dataprofiles = pd.DataFrame()
    if len(data_file_profiles) > 0:
        dataprofiles = pd.read_csv(data_file_profiles, sep=';')

        for n in nodes:
            if get_node(config, n)["techs"] is not None:
                for t_n in get_node(config, n)["techs"]:
                    if "profile" in get_techs(config, n)[t_n].keys():
                        profile_values = get_techs(config, n)[t_n]["profile"]

                        for profile_name, profile_value in profile_values.items():
                            if type(profile_value) is str:
                                get_techs(config, n)[t_n]["profile"][profile_name] = dataprofiles[profile_value].to_list()

    # Update the data profiles to account for errors in the forecast
    for n in nodes:
        if get_node(config, n)["techs"] is not None:
            for t_n in get_node(config, n)["techs"]:
                if scenario["components"] is not None and t_n in scenario["components"].keys():
                        if "mod_profile" in scenario["components"][t_n]:
                            for p_name, p_value in scenario["components"][t_n]["mod_profile"].items():
                                if "name" in p_value.keys():
                                    if p_value["name"] not in dataprofiles:
                                        raise ValueError(f"History csv file does not contain the profile {p_value['name']}")
                                    get_techs(config, n)[t_n]["profile"][p_name][0:t_step] = \
                                        [x * p_value["multiplier"]
                                         for x in dataprofiles[p_value["name"]][0:t_step]]
                                else:
                                    get_techs(config, n)[t_n]["profile"][p_name][0:t_step] = [x * p_value["multiplier"]
                                         for x in p_value["profile"][0:t_step]]

    # creation of pyomo model
    model = pyo.ConcreteModel()

    # time ids
    model.T_id = pyo.Set(initialize=range(0, n_steps), ordered=True)

    # node ids
    model.N_id = pyo.Set(initialize=nodes)
    model.N_ren_id = pyo.Set(initialize=(id for id in model.N_id if has_tech_type(config, id, REN_T)))
    model.N_grid_id = pyo.Set(initialize=(id for id in model.N_id if has_tech_type(config, id, GRID_T)))
    model.N_batt_id = pyo.Set(initialize=(id for id in model.N_id if has_tech_type(config, id, BATT_T)))
    model.N_load_id = pyo.Set(initialize=(id for id in model.N_id if has_tech_type(config, id, LOAD_T)))

    # tech-node ids
    model.tech_RN_ids = pyo.Set(dimen=2, initialize=(
        (n, tech) for n in model.N_id if get_node(config, n)["techs"] is not None for tech, vt in get_node(config,
n)["techs"].items()
            if vt["type"] == REN_T))
    model.tech_BN_ids = pyo.Set(dimen=2, initialize=(
        (n, tech) for n in model.N_id if get_node(config, n)["techs"] is not None for tech, vt in get_node(config,
n)["techs"].items()
            if vt["type"] == BATT_T))
    model.tech_GN_ids = pyo.Set(dimen=2, initialize=(
        (n, tech) for n in model.N_id if get_node(config, n)["techs"] is not None for tech, vt in get_node(config,
n)["techs"].items()
            if vt["type"] == GRID_T))
    model.tech_LN_ids = pyo.Set(dimen=2, initialize=(
        (n, tech) for n in model.N_id if get_node(config, n)["techs"] is not None for tech, vt in get_node(config,
n)["techs"].items()
            if vt["type"] == LOAD_T))

    # links ids
    model.L_id = pyo.Set(initialize=links)

    # renewable production by timestep and node with renewable sources
    model.Pren = pyo.Var(model.T_id, model.tech_RN_ids, within=pyo.NonNegativeReals)
    # battery discharge by timestep and node with batteries
    model.PbattP = pyo.Var(model.T_id, model.tech_BN_ids, within=pyo.NonNegativeReals)
    # battery charge by timestep and node with batteries
    model.PbattN = pyo.Var(model.T_id, model.tech_BN_ids, within=pyo.NonNegativeReals)
    # Energy in the battery
    model.Ebatt = pyo.Var(model.T_id, model.tech_BN_ids, within=pyo.NonNegativeReals)
    # main grid sell by timestep and nodes of interconnection
    model.PgridP = pyo.Var(model.T_id, model.tech_GN_ids, within=pyo.NonNegativeReals)
    # main grid buy by timestep at the nodes of interconnection
    model.PgridN = pyo.Var(model.T_id, model.tech_GN_ids, within=pyo.NonNegativeReals)
    # main grid actual peak power at the nodes of interconnection exceeding the base value
    model.PgridPeakExcess = pyo.Var(model.tech_GN_ids, within=pyo.NonNegativeReals)
```

```
    # load curtailment by timestep and node with demand
    model.Plc = pyo.Var(model.T_id, model.tech_LN_ids, within=pyo.NonNegativeReals)

    # connections variable
    # Power flowing from node out to node in (when phisically flowing in that way, otherwise zero)
    model.PlinkP = pyo.Var(model.T_id, model.L_id, within=pyo.NonNegativeReals)
    # Power flowing from node in to node out (when phisically flowing in that way, otherwise zero)
    model.PlinkN = pyo.Var(model.T_id, model.L_id, within=pyo.NonNegativeReals)

    def ren_limit(model, t, n, r_t):
        return model.Pren[t, n, r_t] <= \
               get_techs(config, n)[r_t]["profile"]["ren"][t] * \
               get_techs(config, n)[r_t]["cap_pwr"] * get_cap_mult_stoch_comps(scenario, r_t, t)

    model.con_ren_limit = pyo.Constraint(model.T_id, model.tech_RN_ids,
        rule=ren_limit)

    def battery_balance(model, t, n, b_t):
        coef_Ebatt = 1
        if get_cap_mult_stoch_comps(scenario, b_t, t-1) >= 1e-3:
            coef_Ebatt = get_cap_mult_stoch_comps(scenario, b_t, t)/get_cap_mult_stoch_comps(scenario, b_t, t-1)
        return model.Ebatt[t, n, b_t] == (
                    get_tech(config, n, b_t)["cap_en"] * get_cap_mult_stoch_comps(scenario, b_t, t) *
                    get_tech(config, n, b_t)["init_SOC"]
                    if (t == model.T_id[1])
                    else model.Ebatt[pre(t, model.T_id), n, b_t] * coef_Ebatt
                ) \
                + model.PbattN[t, n, b_t] * get_tech(config, n, b_t)["eta_ch"] * get_parameter(config, "time_res") \
                - model.PbattP[t, n, b_t] / get_tech(config, n, b_t)["eta_dch"] * get_parameter(config, "time_res")

    model.con_batt_balance = pyo.Constraint(model.T_id, model.tech_BN_ids,
        rule=battery_balance)

    def final_battery_SOC(model, n, b_t):
        return model.Ebatt[model.T_id.last(), n, b_t] >= \
               get_tech(config, n, b_t)["cap_en"] * get_cap_mult_stoch_comps(scenario, b_t, model.T_id.last()) * \
               get_tech(config, n, b_t)["final_SOC"]

    model.con_final_batt_SOC = pyo.Constraint(model.tech_BN_ids,
        rule=final_battery_SOC)

    def battery_maxmin_SOC(model, t, n, b_t):
        return inequality(get_tech(config, n, b_t)["cap_en"] * get_cap_mult_stoch_comps(scenario, b_t, t) * get_tech(config, n,
b_t)["min_SOC"],
                    model.Ebatt[t, n, b_t],
                    get_tech(config, n, b_t)["cap_en"] * get_cap_mult_stoch_comps(scenario, b_t, t) * get_tech(config, n,
b_t)["max_SOC"])

    def battery_max_dch(model, t, n, b_t):
        return model.PbattP[t, n, b_t] <= \
               get_tech(config, n, b_t)["cap_en"] * get_cap_mult_stoch_comps(scenario, b_t, t) * get_tech(config, n, b_t)["max_dch"]

    def battery_max_ch(model, t, n, b_t):
        return model.PbattN[t, n, b_t] <= \
               get_tech(config, n, b_t)["cap_en"] * get_cap_mult_stoch_comps(scenario, b_t, t) * get_tech(config, n, b_t)["max_ch"]

    model.con_batt_SOC = pyo.Constraint(model.T_id, model.tech_BN_ids,
        rule=battery_maxmin_SOC)
    model.con_batt_max_dch = pyo.Constraint(model.T_id, model.tech_BN_ids,
        rule=battery_max_dch)
    model.con_batt_max_ch = pyo.Constraint(model.T_id, model.tech_BN_ids,
        rule=battery_max_ch)

    def grid_max_pwr(model, t, n, g_t):
        return model.PgridP[t, n, g_t] <= \
               get_tech(config, n, g_t)["cap_pwr"] * get_cap_mult_stoch_comps(scenario, g_t, t) * get_tech(config, n,
g_t)["max_supply"]

    def grid_min_pwr(model, t, n, g_t):
        return model.PgridN[t, n, g_t] <= \
               get_tech(config, n, g_t)["cap_pwr"] * get_cap_mult_stoch_comps(scenario, g_t, t) * get_tech(config, n,
g_t)["max_load"]

    model.con_grid_P = pyo.Constraint(model.T_id, model.tech_GN_ids,
        rule=grid_max_pwr)
    model.con_grid_N = pyo.Constraint(model.T_id, model.tech_GN_ids,
        rule=grid_min_pwr)

    def grid_peak_excess_pwr(model, t, n, g_t):
        return model.PgridPeakExcess[n, g_t] >= \
               model.PgridN[t, n, g_t] + model.PgridP[t, n, g_t] - get_tech(config, n, g_t)["base_peak"]

    model.con_grid_peak = pyo.Constraint(model.T_id, model.tech_GN_ids,
        rule=grid_peak_excess_pwr)

    def link_pwr_max_P(model, t, l):
        return model.PlinkP[t, l] <= \
               get_link(config, l)["cap_pwr"] * get_cap_mult_stoch_links(scenario, l, t) * get_link(config, l)["max_outin"]

    def link_pwr_max_N(model, t, l):
        return model.PlinkN[t, l] <= \
               get_link(config, l)["cap_pwr"] * get_cap_mult_stoch_links(scenario, l, t) * get_link(config, l)["max_inout"]

    model.con_link_P = pyo.Constraint(model.T_id, model.L_id,
        rule=link_pwr_max_P)
    model.con_link_N = pyo.Constraint(model.T_id, model.L_id,
        rule=link_pwr_max_N)

    def load_curt_max(model, t, n, lo_t):
        return model.Plc[t, n, lo_t] <= get_techs(config, n)[lo_t]["profile"]["load"][t]

    model.con_load_curt = pyo.Constraint(model.T_id, model.tech_LN_ids,
        rule=load_curt_max)

    def node_balance(model, t, n):
        if (get_techs(config, n) is None) and (
                    (get_links(config) is None) or all(
                        vl["in"] != n and vl["out"] != n for (l, vl) in get_links(config).items()
                    )
                ):
            return pyo.Constraint.Skip
        else:
            return (
                        0 if (get_techs(config, n) is None) else (
                            sum(model.Pren[t, n, tech]
```

```
                                for tech in get_techs_of_type(config, n, REN_T))
                            + sum(model.PbattP[t, n, tech] - model.PbattN[t, n, tech]
                                for tech in get_techs_of_type(config, n, BATT_T))
                            - sum(model.PgridP[t, n, tech] - model.PgridN[t, n, tech]
                                for tech in get_techs_of_type(config, n, GRID_T))
                            - sum(get_techs(config, n)[tech]["profile"]["load"][t] - model.Plc[t, n, tech]
                                for tech in get_techs_of_type(config, n, LOAD_T))
                            )
                    ) + (0 if (get_links(config) is None) else (
                            + sum(+ model.PlinkP[t, l] - model.PlinkN[t, l]/vl["etainout"]
                                for (l, vl) in get_links(config).items() if vl["in"] == n)
                            + sum(- model.PlinkP[t, l]/vl["etaoutin"] + model.PlinkN[t, l]
                                for (l, vl) in get_links(config).items() if vl["out"] == n)
                        )
                    ) == 0

    model.con_balance_node = pyo.Constraint(model.T_id, model.N_id, rule=node_balance)


    lc_cost = sum(model.Plc[t, n, lo_t]*get_techs(config, n)[lo_t]["unmet_cost"] * get_parameter(config, "time_res")
                  for t in model.T_id for (n, lo_t) in model.tech_LN_ids)
    grid_cost = sum(+ model.PgridN[t, n, g_t]*get_techs(config, n)[g_t]["profile"]["price_buy"][t] * get_parameter(config,
"time_res")
                    - model.PgridP[t, n, g_t]*get_techs(config, n)[g_t]["profile"]["price_sell"][t] * get_parameter(config,
"time_res")
                    for t in model.T_id for (n, g_t) in model.tech_GN_ids)
    peak_grid_cost = sum(
                    model.PgridPeakExcess[n, g_t]*get_techs(config, n)[g_t]["peak_cost"]
                    for (n, g_t) in model.tech_GN_ids)
    batt_cost = sum(+ model.PbattP[t, n, b_t]*get_techs(config, n)[b_t]["cost_dch"] * get_parameter(config, "time_res")
                    + model.PbattN[t, n, b_t]*get_techs(config, n)[b_t]["cost_ch"] * get_parameter(config, "time_res")
                    for t in model.T_id for (n, b_t) in model.tech_BN_ids)

    obj_rule = lc_cost + grid_cost + batt_cost + peak_grid_cost

    model.objective = pyo.Objective(expr=obj_rule, sense=pyo.minimize)

    # set previous dispatch variables for the dispatchable variables (the battery assets and grid)
    # till the current t_step, included
    for t in range(0, t_step+1):
        for (n, b_t) in model.tech_BN_ids:
            model.PbattP[t, n, b_t].fix(prev_model.PbattP[t, n, b_t].value)
            model.PbattN[t, n, b_t].fix(prev_model.PbattN[t, n, b_t].value)
        # for (n, g_t) in model.tech_GN_ids:
        #     model.PgridP[t, n, g_t].fix(prev_model.PgridP[t, n, g_t].value)
        #     model.PgridN[t, n, g_t].fix(prev_model.PgridN[t, n, g_t].value)

    # Should print all available solvers
    # opt = SolverFactory("gurobi", solver_io="python")
    opt = SolverFactory(solver_name)

    instance = model

    # Create a model instance and optimize
    results = opt.solve(instance)
    # model.display()

    return model, results, config


def run_single_stoch_simulation(s_name, s_value, det_model, config_orig,
                                data_file_profiles="datafile.csv", solver_name="glpk"):
    config = copy.deepcopy(config_orig)
    model_s = det_model
    results_s = None
    config_s = None
    for t in det_model.T_id:
        model_s, results_s, config_s = \
            optimal_dispatch_stoch(model_s, t, s_value, config,
                                   data_file_profiles=data_file_profiles,
                                   solver_name=solver_name)
        print(f"scenario: {s_name}; time: {t}/{len(det_model.T_id)}")
    return model_s, results_s, config_s


def run_full_stoch_simulations(config,
                               fault_scenarios,
                               solver_name="glpk",
                               data_file_profiles="",
                               generate_plots=False,
                               output_folder="./Img/Img_s/Img_%s/"):
    """
    This is the main function to run the complete stochastic simulations of the output of the deterministic EMS
    After solving the deterministic optimization, for every fault scenario, a rolling-horizon dispatch is performed.
    Every time step, the previous dispatch is set but with the real-time conditions in term of components availability.

    Parameters
    ----------
    config: str or dict
        Configuration file of the network
    fault_scenarios: str or dict
        Configuration file of the fault scenarios to simulate
    solver_name: str
        Solver name
    data_file_profiles (optional): str
        Path to a csv file containing additional time series that may be needed
    generate_plots: bool
        If true, it generates output plots
    output_folder:
        If generate_plots is true, folder where the output plots are saved

    Outputs
    -------
    reliability_indices: dict
        Dictionary where the reliability outputs are stored
    model_s: dict
        Dictionary of the Pyomo models by scenario
    results_s: dict
        Result dictionary by scenario
    config_s: dict
        Configuration dictionary by scenario
    """

    # initialize outputs
    model_s = {}
```

```
    results_s = {}
    config_s = {}
    out_results_s = {}
    reliability_indices_s = {}

    # execute deterministic optimization
    print("Run normal scenario")

    model_s[name_bc], out_results_s[name_bc], config_s[name_bc] = \
        optimal_dispatch(config, data_file_profiles=data_file_profiles, solver_name=solver_name)

    # obtain results of deterministic optimization
    out_results_s[name_bc], reliability_indices_s[name_bc] = get_results(
        model_s[name_bc], config_s[name_bc], generate_plots=True, folder=output_folder % name_bc)

    # load fault scenarios
    if type(fault_scenarios) is str:
        file = open(fault_scenarios)
        fault_scenarios = yaml.load(file, Loader=yaml.Loader)
    elif type(fault_scenarios) is dict:
        fault_scenarios = fault_scenarios
    elif type(fault_scenarios) is not dict:
        raise Exception(f"Type of fault_scenarios ({type(fault_scenarios)}) is not recognized, it shall be a valid string or
dictionary")

    # Initialize system reliability indices
    reliability_indices = initialize_reliability_dict(config_s[name_bc])

    # execute base model
    prob_bc = 1 - sum(s_value["prob"] for s_value in fault_scenarios["scenarios"].values())  # probability base scenario

    # calculation of reliability indices for the base case scenario
    update_reliability_dict(reliability_indices, reliability_indices_s[name_bc], prob_bc, config_s[name_bc])

    # execute stochastic scenarios
    for s_name, s_value in fault_scenarios["scenarios"].items():

        print(f"Run scenario {s_name}")

        # execute stochastic simulation
        model_s[s_name], results_s[s_name], config_s[s_name] = \
            run_single_stoch_simulation(s_name, s_value, model_s[name_bc], config_s[name_bc],
                                        data_file_profiles=data_file_profiles,
                                        solver_name=solver_name)

        # obtain results stochastic simulation
        out_results_s[s_name], reliability_indices_s[s_name] = get_results(
            model_s[s_name], config_s[s_name], generate_plots=generate_plots, folder=output_folder % s_name)

        # update reliability indices
        update_reliability_dict(reliability_indices, reliability_indices_s[s_name], s_value["prob"], config_s[name_bc])

    return reliability_indices, reliability_indices_s, model_s, results_s, config_s

if __name__ == "__main__":

    reliability_indices, reliability_indices_s, model_s, results_s, config_s = run_full_stoch_simulations("data.yml",
                                "fault_scenarios.yml",
                                solver_name="glpk",
                                data_file_profiles="datafile.csv",
                                generate_plots=True,
                                output_folder="./Img/Img_s/Img_%s/")
```

4.2.2.3 BaseCase_IndicesCalculation.py

```
"""
This module aims at providing the base case dispatch of a network based on a Load Following approach
Assets are used according to their marginal cost at every iteration without predictive dispatch,
which is instead performed in the EMS_module
The reliability indices are calculated accordingly.
"""

from .EMS_module import *


def build_meritorder_model(scenario, config, data_file_profiles=''):
    """
    Function to calculate and build the main model used for the merit-order simulation
    """

    # number of time steps and vector of time steps
    n_steps = int(get_parameter(config, "n_steps"))
    time_steps = range(0, n_steps)

    # nodes of the network
    nodes = get_nodes_list(config)
    n_nodes = len(nodes)

    # connection arcs
    links = get_links_list(config)

    # populate the config dictionary with the profiles of the resources if needed
    dataprofiles = pd.DataFrame()
    if len(data_file_profiles) > 0:
        dataprofiles = pd.read_csv(data_file_profiles, sep=';')

        for n in nodes:
            if get_node(config, n)["techs"] is not None:
                for t_n in get_node(config, n)["techs"]:
                    if "profile" in get_techs(config, n)[t_n].keys():
                        profile_values = get_techs(config, n)[t_n]["profile"]

                        for profile_name, profile_value in profile_values.items():
                            if type(profile_value) is str:
                                get_techs(config, n)[t_n]["profile"][profile_name] = dataprofiles[profile_value].to_list()

    # Update the data profiles to account for errors in the forecast
    for n in nodes:
        if get_node(config, n)["techs"] is not None:
            for t_n in get_node(config, n)["techs"]:
                if scenario["components"] is not None and t_n in scenario["components"].keys():
                        if "mod_profile" in scenario["components"][t_n]:
                            for p_name, p_value in scenario["components"][t_n]["mod_profile"].items():
                                if "name" in p_value.keys():
                                    if p_value["name"] not in dataprofiles:
                                        raise ValueError(f"History csv file does not contain the profile {p_value['name']}")
                                    get_techs(config, n)[t_n]["profile"][p_name][0:n_steps] = \
                                        [x * p_value["multiplier"]
                                         for x in dataprofiles[p_value["name"]][0:n_steps]]
                                else:
                                    get_techs(config, n)[t_n]["profile"][p_name][0:n_steps] = [x * p_value["multiplier"]
                                         for x in p_value["profile"][0:n_steps]]

    # creation of pyomo model
    model = pyo.ConcreteModel()

    # time ids
    model.T_id = pyo.Set(initialize=range(0, n_steps), ordered=True)

    # node ids
    model.N_id = pyo.Set(initialize=nodes)
    model.N_ren_id = pyo.Set(initialize=(id for id in model.N_id if has_tech_type(config, id, REN_T)))
    model.N_grid_id = pyo.Set(initialize=(id for id in model.N_id if has_tech_type(config, id, GRID_T)))
    model.N_batt_id = pyo.Set(initialize=(id for id in model.N_id if has_tech_type(config, id, BATT_T)))
    model.N_load_id = pyo.Set(initialize=(id for id in model.N_id if has_tech_type(config, id, LOAD_T)))

    # tech-node ids
    model.tech_RN_ids = pyo.Set(dimen=2, initialize=(
        (n, tech) for n in model.N_id if get_node(config, n)["techs"] is not None for tech, vt in get_node(config,
n)["techs"].items()
            if vt["type"] == REN_T))
    model.tech_BN_ids = pyo.Set(dimen=2, initialize=(
        (n, tech) for n in model.N_id if get_node(config, n)["techs"] is not None for tech, vt in get_node(config,
n)["techs"].items()
            if vt["type"] == BATT_T))
    model.tech_GN_ids = pyo.Set(dimen=2, initialize=(
        (n, tech) for n in model.N_id if get_node(config, n)["techs"] is not None for tech, vt in get_node(config,
n)["techs"].items()
            if vt["type"] == GRID_T))
    model.tech_LN_ids = pyo.Set(dimen=2, initialize=(
        (n, tech) for n in model.N_id if get_node(config, n)["techs"] is not None for tech, vt in get_node(config,
n)["techs"].items()
            if vt["type"] == LOAD_T))

    # links ids
    model.L_id = pyo.Set(initialize=links)

    # renewable production by timestep and node with renewable sources
    model.Pren = pyo.Var(model.T_id, model.tech_RN_ids, within=pyo.NonNegativeReals)
    # battery discharge by timestep and node with batteries
    model.PbattP = pyo.Var(model.T_id, model.tech_BN_ids, within=pyo.NonNegativeReals)
    # battery charge by timestep and node with batteries
    model.PbattN = pyo.Var(model.T_id, model.tech_BN_ids, within=pyo.NonNegativeReals)
    # Energy in the battery
    model.Ebatt = pyo.Var(model.T_id, model.tech_BN_ids, within=pyo.NonNegativeReals)
    # main grid sell by timestep and nodes of interconnection
    model.PgridP = pyo.Var(model.T_id, model.tech_GN_ids, within=pyo.NonNegativeReals)
    # main grid buy by timestep at the nodes of interconnection
    model.PgridN = pyo.Var(model.T_id, model.tech_GN_ids, within=pyo.NonNegativeReals)
    # main grid actual peak power at the nodes of interconnection exceeding the base value
    model.PgridPeakExcess = pyo.Var(model.tech_GN_ids, within=pyo.NonNegativeReals)
    # load curtailment by ti"mestep and node with demand
    model.Plc = pyo.Var(model.T_id, model.tech_LN_ids, within=pyo.NonNegativeReals)

    # connections variable
    # Power flowing from node out to node in (when phisically flowing in that way, otherwise zero)
    model.PlinkP = pyo.Var(model.T_id, model.L_id, within=pyo.NonNegativeReals)
```

```
    # Power flowing from node in to node out (when phisically flowing in that way, otherwise zero)
    model.PlinkN = pyo.Var(model.T_id, model.L_id, within=pyo.NonNegativeReals)

    def ren_limit(model, t, n, r_t):
        return model.Pren[t, n, r_t] <= \
               get_techs(config, n)[r_t]["profile"]["ren"][t] * \
               get_techs(config, n)[r_t]["cap_pwr"] * get_cap_mult_stoch_comps(scenario, r_t, t)

    model.con_ren_limit = pyo.Constraint(model.T_id, model.tech_RN_ids,
        rule=ren_limit)

    def battery_balance(model, t, n, b_t):
        coef_Ebatt = 1
        if get_cap_mult_stoch_comps(scenario, b_t, t-1) >= 1e-3:
            coef_Ebatt = get_cap_mult_stoch_comps(scenario, b_t, t)/get_cap_mult_stoch_comps(scenario, b_t, t-1)
        return model.Ebatt[t, n, b_t] == (
                    get_tech(config, n, b_t)["cap_en"] * get_cap_mult_stoch_comps(scenario, b_t, t) *
                    get_tech(config, n, b_t)["init_SOC"]
                    if (t == model.T_id[1])
                    else model.Ebatt[pre(t, model.T_id), n, b_t] * coef_Ebatt
                ) \
                + model.PbattN[t, n, b_t] * get_tech(config, n, b_t)["eta_ch"] * get_parameter(config, "time_res") \
                - model.PbattP[t, n, b_t] / get_tech(config, n, b_t)["eta_dch"] * get_parameter(config, "time_res")

    model.con_batt_balance = pyo.Constraint(model.T_id, model.tech_BN_ids,
        rule=battery_balance)

    def final_battery_SOC(model, n, b_t):
        return model.Ebatt[model.T_id.last(), n, b_t] >= \
               get_tech(config, n, b_t)["cap_en"] * get_cap_mult_stoch_comps(scenario, b_t, model.T_id.last()) * \
               get_tech(config, n, b_t)["final_SOC"]

    model.con_final_batt_SOC = pyo.Constraint(model.tech_BN_ids,
        rule=final_battery_SOC)

    def battery_maxmin_SOC(model, t, n, b_t):
        return inequality(get_tech(config, n, b_t)["cap_en"] * get_cap_mult_stoch_comps(scenario, b_t, t) * get_tech(config, n,
b_t)["min_SOC"],
                    model.Ebatt[t, n, b_t],
                    get_tech(config, n, b_t)["cap_en"] * get_cap_mult_stoch_comps(scenario, b_t, t) * get_tech(config, n,
b_t)["max_SOC"])

    def battery_max_dch(model, t, n, b_t):
        return model.PbattP[t, n, b_t] <= \
               get_tech(config, n, b_t)["cap_en"] * get_cap_mult_stoch_comps(scenario, b_t, t) * get_tech(config, n, b_t)["max_dch"]

    def battery_max_ch(model, t, n, b_t):
        return model.PbattN[t, n, b_t] <= \
               get_tech(config, n, b_t)["cap_en"] * get_cap_mult_stoch_comps(scenario, b_t, t) * get_tech(config, n, b_t)["max_ch"]

    model.con_batt_SOC = pyo.Constraint(model.T_id, model.tech_BN_ids,
        rule=battery_maxmin_SOC)
    model.con_batt_max_dch = pyo.Constraint(model.T_id, model.tech_BN_ids,
        rule=battery_max_dch)
    model.con_batt_max_ch = pyo.Constraint(model.T_id, model.tech_BN_ids,
        rule=battery_max_ch)

    def grid_max_pwr(model, t, n, g_t):
        return model.PgridP[t, n, g_t] <= \
               get_tech(config, n, g_t)["cap_pwr"] * get_cap_mult_stoch_comps(scenario, g_t, t) * get_tech(config, n,
g_t)["max_supply"]

    def grid_min_pwr(model, t, n, g_t):
        return model.PgridN[t, n, g_t] <= \
               get_tech(config, n, g_t)["cap_pwr"] * get_cap_mult_stoch_comps(scenario, g_t, t) * get_tech(config, n,
g_t)["max_load"]

    model.con_grid_P = pyo.Constraint(model.T_id, model.tech_GN_ids,
        rule=grid_max_pwr)
    model.con_grid_N = pyo.Constraint(model.T_id, model.tech_GN_ids,
        rule=grid_min_pwr)

    def grid_peak_excess_pwr(model, t, n, g_t):
        return model.PgridPeakExcess[n, g_t] >= \
               model.PgridN[t, n, g_t] + model.PgridP[t, n, g_t] - get_tech(config, n, g_t)["base_peak"]

    model.con_grid_peak = pyo.Constraint(model.T_id, model.tech_GN_ids,
        rule=grid_peak_excess_pwr)

    def link_pwr_max_P(model, t, l):
        return model.PlinkP[t, l] <= \
               get_link(config, l)["cap_pwr"] * get_cap_mult_stoch_links(scenario, l, t) * get_link(config, l)["max_outin"]

    def link_pwr_max_N(model, t, l):
        return model.PlinkN[t, l] <= \
               get_link(config, l)["cap_pwr"] * get_cap_mult_stoch_links(scenario, l, t) * get_link(config, l)["max_inout"]

    model.con_link_P = pyo.Constraint(model.T_id, model.L_id,
        rule=link_pwr_max_P)
    model.con_link_N = pyo.Constraint(model.T_id, model.L_id,
        rule=link_pwr_max_N)

    def load_curt_max(model, t, n, lo_t):
        return model.Plc[t, n, lo_t] <= get_techs(config, n)[lo_t]["profile"]["load"][t]

    model.con_load_curt = pyo.Constraint(model.T_id, model.tech_LN_ids,
        rule=load_curt_max)

    def node_balance(model, t, n):
        if (get_techs(config, n) is None) and (
                    (get_links(config) is None) or all(
                        vl["in"] != n and vl["out"] != n for (l, vl) in get_links(config).items()
                    )
                ):
            return pyo.Constraint.Skip
        else:
            return (
                        0 if (get_techs(config, n) is None) else (
                            sum(model.Pren[t, n, tech]
                                for tech in get_techs_of_type(config, n, REN_T))
                            + sum(model.PbattP[t, n, tech] - model.PbattN[t, n, tech]
                                  for tech in get_techs_of_type(config, n, BATT_T))
                            - sum(model.PgridP[t, n, tech] - model.PgridN[t, n, tech]
                                  for tech in get_techs_of_type(config, n, GRID_T))
                            - sum(get_techs(config, n)[tech]["profile"]["load"][t] - model.Plc[t, n, tech]
```

```
                                for tech in get_techs_of_type(config, n, LOAD_T))
                        )
                    ) + (0 if (get_links(config) is None) else (
                            + sum(+ model.PlinkP[t, l] - model.PlinkN[t, l]/vl["etainout"]
                                for (l, vl) in get_links(config).items() if vl["in"] == n)
                            + sum(- model.PlinkP[t, l]/vl["etaoutin"] + model.PlinkN[t, l]
                                for (l, vl) in get_links(config).items() if vl["out"] == n)
                        )
                    ) == 0

    model.con_balance_node = pyo.Constraint(model.T_id, model.N_id, rule=node_balance)

    return model, config


def add_no_discharge_when_selling(model, config, time_step):


    # Binary variable to avoid discharging batteries when selling to the grid
    if model.find_component('binGridSell') is None:  # add variable if not available
        model.binGridSell = pyo.Var(within=pyo.Binary)

    # delete existing constraint if available
    if model.find_component('con_batt_dch') is not None:
        model.del_component(model.con_batt_dch)
        model.del_component(model.con_grid_sell)

    #  Avoid discharging batteries when energy is sold to the grid

    # Set dispatch of battery to zero if energy is sold to the grid (binGridSell=1)
    def rule_no_discharge_bin(model):
        return sum(model.PbattP[time_step, n, b_t] for (n, b_t) in model.tech_BN_ids) <= \
               model.binGridSell * \
               sum(get_tech(config, n, b_t)["cap_en"] for (n, b_t) in model.tech_BN_ids)

    model.con_batt_dch = pyo.Constraint(rule=rule_no_discharge_bin)

    # Impose binGridSell to be 1 if energy is sold
    def rule_bin_if_sell(model):
        return sum(model.PgridP[time_step, n, g_t] for (n, g_t) in model.tech_GN_ids) <= \
               (1 - model.binGridSell) * \
               sum(get_tech(config, n, g_t)["cap_pwr"] for (n, g_t) in model.tech_GN_ids)

    model.con_grid_sell = pyo.Constraint(rule=rule_bin_if_sell)

    return model


def build_meritorder_objective(model, config, time_step, multiplier_lc=1.0):
    """
    Function to build the objective for the current time step in the merit-order technique
    """

    lc_cost = sum(model.Plc[time_step, n, lo_t]*get_techs(config, n)[lo_t]["unmet_cost"] * get_parameter(config, "time_res")
                  for (n, lo_t) in model.tech_LN_ids)
    grid_cost = sum(+ model.PgridN[time_step, n, g_t]*get_techs(config, n)[g_t]["profile"]["price_buy"][time_step] *
get_parameter(config, "time_res")
                    - model.PgridP[time_step, n, g_t]*get_techs(config, n)[g_t]["profile"]["price_sell"][time_step] *
get_parameter(config, "time_res")
                    for (n, g_t) in model.tech_GN_ids)
    peak_grid_cost = sum(
                     model.PgridPeakExcess[n, g_t]*get_techs(config, n)[g_t]["peak_cost"]
                     for (n, g_t) in model.tech_GN_ids)
    batt_cost = sum(+ model.PbattP[time_step, n, b_t]*get_techs(config, n)[b_t]["cost_dch"] * get_parameter(config, "time_res")
                    + model.PbattN[time_step, n, b_t]*get_techs(config, n)[b_t]["cost_ch"] * get_parameter(config, "time_res")
                    for (n, b_t) in model.tech_BN_ids)

    obj_rule = multiplier_lc*lc_cost + grid_cost + batt_cost + peak_grid_cost

    if model.find_component("objective"):
        model.del_component(model.objective)
    model.objective = pyo.Objective(expr=obj_rule, sense=pyo.minimize)

    return model

def load_config(config, data_file_profiles = ""):
    """
    Function to load the cleaned configuration files
    """

    # load the config
    if type(config) is str:
        file = open(config)
        config = yaml.load(file, Loader=yaml.Loader)
    elif type(config) is not dict:
        raise Exception(f"Type of config ({type(config)}) is not recognized, it shall be a valid string or dictionary")

    # number of time steps and vector of time steps
    n_steps = int(get_parameter(config, "n_steps"))
    time_steps = range(0, n_steps)

    # nodes of the network
    nodes = get_nodes_list(config)
    n_nodes = len(nodes)

    # populate the config dictionary with the profiles of the resources if needed
    if len(data_file_profiles) > 0:
        dataprofiles = pd.read_csv(data_file_profiles, sep=';')

        for n in nodes:
            if get_node(config, n)["techs"] is not None:
                for t_n in get_node(config, n)["techs"]:
                    if "profile" in get_techs(config, n)[t_n].keys():
                        profile_values = get_techs(config, n)[t_n]["profile"]

                        for profile_name, profile_value in profile_values.items():
                            if type(profile_value) is str:
                                get_techs(config, n)[t_n]["profile"][profile_name] = dataprofiles[profile_value].to_list()

    return config


def run_solveloop_meritorder_model(scenario, config_orig,
                                   data_file_profiles='', multiplier_lc=1.0, solver_name="glpk"):
    config = copy.deepcopy(config_orig)
```

```
    # build the raw optimization model
    model, config = build_meritorder_model(scenario, config, data_file_profiles=data_file_profiles)

    # define the solver
    opt = SolverFactory(solver_name)

    results = None

    # till the current t_step, included
    for t in model.T_id:

        # add constraint regarding no battery discharge when selling energy
        model = add_no_discharge_when_selling(model, config, t)

        # update the model objective function with the current iteration
        model = build_meritorder_objective(model, config, t, multiplier_lc)

        # solve the model
        results = opt.solve(model)

        print(f"time: {t}/{len(model.T_id)}")

        for (n, b_t) in model.tech_BN_ids:
            model.PbattP[t, n, b_t].fix()
            model.PbattN[t, n, b_t].fix()
            model.Ebatt[t, n, b_t].fix()
        for (n, g_t) in model.tech_GN_ids:
            model.PgridP[t, n, g_t].fix()
            model.PgridN[t, n, g_t].fix()
        for (n, r_t) in model.tech_RN_ids:
            model.Pren[t, n, r_t].fix()
        for (n, l_t) in model.tech_LN_ids:
            model.Plc[t, n, l_t].fix()
        for l in model.L_id:
            model.PlinkP[t, l].fix()
            model.PlinkN[t, l].fix()

    return model, results, config


def run_base_case_simulations(config,
                              fault_scenarios,
                              solver_name="glpk",
                              data_file_profiles="",
                              generate_plots=False,
                              output_folder="./Img/Img_bc/Img_%s/"):
    """
    This is the main function to run the complete stochastic simulations of the base case simulation.
    For every time step of the normal operation and every fault scenario, the system is operated to maximize the return for every
time step.

    Parameters
    ----------
    config: str or dict
        Configuration file of the network
    fault_scenarios: str or dict
        Configuration file of the fault scenarios to simulate
    solver_name: str
        Solver name
    data_file_profiles (optional): str
        Path to a csv file containing additional time series that may be needed
    generate_plots: bool
        If true, it generates output plots
    output_folder:
        If generate_plots is true, folder where the output plots are saved

    Outputs
    -------
    reliability_indices: dict
        Dictionary where the reliability outputs are stored
    model_s: dict
        Dictionary of the Pyomo models by scenario
    results_s: dict
        Result dictionary by scenario
    config_s: dict
        Configuration dictionary by scenario
    """

    # load and clean configuration files
    config = load_config(config, data_file_profiles=data_file_profiles)

    # load fault scenarios
    if type(fault_scenarios) is str:
        file = open(fault_scenarios)
        fault_scenarios = yaml.load(file, Loader=yaml.Loader)
    elif type(fault_scenarios) is dict:
        fault_scenarios = fault_scenarios
    elif type(fault_scenarios) is not dict:
        raise Exception(f"Type of fault_scenarios ({type(fault_scenarios)}) is not recognized, it shall be a valid string or
dictionary")

    # initialize outputs
    model_s = {}
    results_s = {}
    config_s = {}
    out_results_s = {}
    reliability_indices_s = {}

    # Initialize system reliability indices
    reliability_indices = initialize_reliability_dict(config)

    # execute base model
    prob_bc = 1 - sum(s_value["prob"] for s_value in fault_scenarios["scenarios"].values())  # probability base scenario

    # multiplier to avoid load curtailment
    multiplier_lc = int(get_parameter(config, "n_steps"))  # multiplier set equal to the number of time steps

    # execute base case scenario
    print("Run normal scenario")
    normal_scenario_dict = {"components": None, "links": None}
    model_s[name_bc], results_s[name_bc], config_s[name_bc] = \
        run_solveloop_meritorder_model(normal_scenario_dict, config, data_file_profiles=data_file_profiles,
multiplier_lc=multiplier_lc, solver_name=solver_name)
```

```
    # get results base case scenario
    out_results_s[name_bc], reliability_indices_s[name_bc] = get_results(
        model_s[name_bc], config_s[name_bc], generate_plots=True, folder=output_folder % name_bc)

    # calculation of reliability indices for the base case scenario
    update_reliability_dict(reliability_indices, reliability_indices_s[name_bc], prob_bc, config)

    for s_name, s_value in fault_scenarios["scenarios"].items():

        print(f"Run scenario {s_name}")

        model_s[s_name], results_s[s_name], config_s[s_name] = \
            run_solveloop_meritorder_model(s_value, config,
                                           data_file_profiles=data_file_profiles,
                                           multiplier_lc=multiplier_lc,
                                           solver_name=solver_name)

        out_results_s[s_name], reliability_indices_s[s_name] = get_results(
            model_s[s_name], config_s[s_name], generate_plots=generate_plots, folder=output_folder % s_name)

        # calculation of reliability indices
        update_reliability_dict(reliability_indices, reliability_indices_s[s_name], s_value["prob"], config)

    return reliability_indices, reliability_indices_s, model_s, results_s, config_s

if __name__ == "__main__":

    reliability_indices, reliability_indices_s, model_s, results_s, config_s = run_base_case_simulations("config.yml",
                               "fault_scenarios.yml",
                               solver_name="glpk",
                               data_file_profiles="datafile.csv",
                               generate_plots=True,
                               output_folder="./Img/Img_bc/Img_%s/")

    # # load and clean configuration files
    # config = load_config("data.yml", data_file_profiles="datafile.csv")
    #
    # file = open("fault_scenarios.yml")
    # fault_scenarios = yaml.load(file, Loader=yaml.Loader)
    #
    # # initialize outputs
    # model_s = {}
    # results_s = {}
    # config_s = {}
    # out_results_s = {}
    # reliability_indices_s = {}
    #
    # # Initialize system reliability indices
    # reliability_indices = initialize_reliability_dict(config)
    #
    # # execute base model
    # prob_bc = 1 - sum(s_value["prob"] for s_value in fault_scenarios["scenarios"].values())  # probability base scenario
    #
    # # multiplier to avoid load curtailment
    # multiplier_lc = int(get_parameter(config, "n_steps"))  # multiplier set equal to the number of time steps
    #
    # # execute base case scenario
    # model_s[name_bc], results_s[name_bc], config_s[name_bc] = \
    #     run_solveloop_meritorder_model(name_bc, config, multiplier_lc=multiplier_lc, solver_name="glpk")
    #
    # # get results base case scenario
    # out_results_s[name_bc], reliability_indices_s[name_bc] = get_results(
    #     model_s[name_bc], config_s[name_bc], generate_plots=True, folder=f"./Img/Img_bc/Img_{name_bc}/")
    #
    # # calculation of reliability indices for the base case scenario
    # update_reliability_dict(reliability_indices, reliability_indices_s[name_bc], prob_bc, config)
    #
    # for s_name, s_value in fault_scenarios["scenarios"].items():
    #     model_s[s_name], results_s[s_name], config_s[s_name] = \
    #         run_solveloop_meritorder_model(s_value, config, multiplier_lc=multiplier_lc,
    #                                        solver_name="glpk")
    #
    #     out_results_s[s_name], reliability_indices_s[s_name] = get_results(
    #         model_s[s_name], config_s[s_name], generate_plots=True, folder=f"./Img/Img_bc/Img_{s_name}/")
    #
    #     # calculation of reliability indices
    #     update_reliability_dict(reliability_indices, reliability_indices_s[s_name], s_value["prob"], config)
    #
    # reliability_indices_base_case = reliability_indices
```

4.2.2.4 EMS_module.py

```
"""
This module provides the main functions for the deterministic optimization of a network
"""

import yaml  # Used for reading config as dict
from pyomo.environ import *
from pyomo.opt import SolverFactory

import pandas as pd
import matplotlib.pyplot as plt
import os as os

import pyomo.core as pyo

from .utils import *


def optimal_dispatch(config, data_file_profiles="", solver_name="glpk"):
    """
    Function to execute the optimal dispatch
    Parameters
    ----------
    config : str or dict
        - File name related to a yaml file to import a dict
        - Dictionary of the config to import
    data_file_profiles : str
        File name related to a yaml file to import config; if null, the profile is searched in config
    Output
    ------
    model : Pyomo Concrete Model
        Pyomo concrete model storing the model and solution of the EMS
    opt_status_result : SolverResults object
        Status of the optimization result executed by the solver
    config : dict
        Dictionary of the configuration file used for the optimization
    """

    # load the config
    if type(config) is str:
        file = open(config)
        config = yaml.load(file, Loader=yaml.Loader)
    elif type(config) is not dict:
        raise Exception(f"Type of config ({type(config)}) is not recognized, it shall be a valid string or dictionary")

    # number of time steps and vector of time steps
    n_steps = int(get_parameter(config, "n_steps"))
    time_steps = range(0, n_steps)

    # nodes of the network
    nodes = get_nodes_list(config)
    n_nodes = len(nodes)

    # connection arcs
    links = get_links_list(config)

    # populate the config dictionary with the profiles of the resources if needed
    dataprofiles = pd.DataFrame()
    if len(data_file_profiles) > 0:
        dataprofiles = pd.read_csv(data_file_profiles, sep=';')

        for n in nodes:
            if get_node(config, n)["techs"] is not None:
                for t_n in get_node(config, n)["techs"]:
                    if "profile" in get_techs(config, n)[t_n].keys():
                        profile_values = get_techs(config, n)[t_n]["profile"]

                        for profile_name, profile_value in profile_values.items():
                            if type(profile_value) is str:
                                get_techs(config, n)[t_n]["profile"][profile_name] = dataprofiles[profile_value].to_list()

    # creation of pyomo model
    model = pyo.ConcreteModel()

    # time ids
    model.T_id = pyo.Set(initialize=range(0, n_steps), ordered=True)

    # node ids
    model.N_id = pyo.Set(initialize=nodes)
    model.N_ren_id = pyo.Set(initialize=(id for id in model.N_id if has_tech_type(config, id, REN_T)))
    model.N_grid_id = pyo.Set(initialize=(id for id in model.N_id if has_tech_type(config, id, GRID_T)))
    model.N_batt_id = pyo.Set(initialize=(id for id in model.N_id if has_tech_type(config, id, BATT_T)))
    model.N_load_id = pyo.Set(initialize=(id for id in model.N_id if has_tech_type(config, id, LOAD_T)))

    # tech-node ids
    model.tech_RN_ids = pyo.Set(dimen=2, initialize=(
        (n, tech) for n in model.N_id if get_node(config, n)["techs"] is not None for tech, vt in get_node(config,
n)["techs"].items()
            if vt["type"] == REN_T))
    model.tech_BN_ids = pyo.Set(dimen=2, initialize=(
        (n, tech) for n in model.N_id if get_node(config, n)["techs"] is not None for tech, vt in get_node(config,
n)["techs"].items()
            if vt["type"] == BATT_T))
    model.tech_GN_ids = pyo.Set(dimen=2, initialize=(
        (n, tech) for n in model.N_id if get_node(config, n)["techs"] is not None for tech, vt in get_node(config,
n)["techs"].items()
            if vt["type"] == GRID_T))
    model.tech_LN_ids = pyo.Set(dimen=2, initialize=(
        (n, tech) for n in model.N_id if get_node(config, n)["techs"] is not None for tech, vt in get_node(config,
n)["techs"].items()
            if vt["type"] == LOAD_T))

    # links ids
    model.L_id = pyo.Set(initialize=links)

    # renewable production by timestep and node with renewable sources
    model.Pren = pyo.Var(model.T_id, model.tech_RN_ids, within=pyo.NonNegativeReals)
    # battery discharge by timestep and node with batteries
    model.PbattP = pyo.Var(model.T_id, model.tech_BN_ids, within=pyo.NonNegativeReals)
    # battery charge by timestep and node with batteries
    model.PbattN = pyo.Var(model.T_id, model.tech_BN_ids, within=pyo.NonNegativeReals)
    # Energy in the battery
    model.Ebatt = pyo.Var(model.T_id, model.tech_BN_ids, within=pyo.NonNegativeReals)
```

```
# main grid sell by timestep and nodes of interconnection
model.PgridP = pyo.Var(model.T_id, model.tech_GN_ids, within=pyo.NonNegativeReals)
# main grid buy by timestep at the nodes of interconnection
model.PgridN = pyo.Var(model.T_id, model.tech_GN_ids, within=pyo.NonNegativeReals)
# main grid actual peak power at the nodes of interconnection exceeding the base value
model.PgridPeakExcess = pyo.Var(model.tech_GN_ids, within=pyo.NonNegativeReals)
# load curtailment by timestep and node with demand
model.Plc = pyo.Var(model.T_id, model.tech_LN_ids, within=pyo.NonNegativeReals)

# connections variable
# Power flowing from node out to node in (when phisically flowing in that way, otherwise zero)
model.PlinkP = pyo.Var(model.T_id, model.L_id, within=pyo.NonNegativeReals)
# Power flowing from node in to node out (when phisically flowing in that way, otherwise zero)
model.PlinkN = pyo.Var(model.T_id, model.L_id, within=pyo.NonNegativeReals)

def ren_limit(model, t, n, r_t):
    return model.Pren[t, n, r_t] <= \
           get_techs(config, n)[r_t]["profile"]["ren"][t] * get_techs(config, n)[r_t]["cap_pwr"]

model.con_ren_limit = pyo.Constraint(model.T_id, model.tech_RN_ids,
    rule=ren_limit)

def battery_balance(model, t, n, b_t):
    return model.Ebatt[t, n, b_t] == (
                get_tech(config, n, b_t)["cap_en"] * get_tech(config, n, b_t)["init_SOC"]
                if (t == model.T_id[1])
                else model.Ebatt[pre(t, model.T_id), n, b_t]
            ) \
            + model.PbattN[t, n, b_t] * get_tech(config, n, b_t)["eta_ch"] * get_parameter(config, "time_res") \
            - model.PbattP[t, n, b_t] / get_tech(config, n, b_t)["eta_dch"] * get_parameter(config, "time_res")

model.con_batt_balance = pyo.Constraint(model.T_id, model.tech_BN_ids,
    rule=battery_balance)

def final_battery_SOC(model, n, b_t):
    return model.Ebatt[model.T_id.last(), n, b_t] >= get_tech(config, n, b_t)["cap_en"] * get_tech(config, n, b_t)["final_SOC"]

model.con_final_batt_SOC = pyo.Constraint(model.tech_BN_ids,
    rule=final_battery_SOC)

def battery_maxmin_SOC(model, t, n, b_t):
    return inequality(get_tech(config, n, b_t)["cap_en"] * get_tech(config, n, b_t)["min_SOC"],
               model.Ebatt[t, n, b_t],
               get_tech(config, n, b_t)["cap_en"] * get_tech(config, n, b_t)["max_SOC"])

def battery_max_dch(model, t, n, b_t):
    return model.PbattP[t, n, b_t] <= \
           get_tech(config, n, b_t)["cap_en"] * get_tech(config, n, b_t)["max_dch"]

def battery_max_ch(model, t, n, g_t):
    return model.PbattN[t, n, g_t] <= \
           get_tech(config, n, g_t)["cap_en"] * get_tech(config, n, g_t)["max_ch"]

model.con_batt_SOC = pyo.Constraint(model.T_id, model.tech_BN_ids,
    rule=battery_maxmin_SOC)
model.con_batt_max_dch = pyo.Constraint(model.T_id, model.tech_BN_ids,
    rule=battery_max_dch)
model.con_batt_max_ch = pyo.Constraint(model.T_id, model.tech_BN_ids,
    rule=battery_max_ch)

def grid_max_pwr(model, t, n, g_t):
    return model.PgridP[t, n, g_t] <= \
           get_tech(config, n, g_t)["cap_pwr"] * get_tech(config, n, g_t)["max_supply"]

def grid_min_pwr(model, t, n, g_t):
    return model.PgridN[t, n, g_t] <= \
           get_tech(config, n, g_t)["cap_pwr"] * get_tech(config, n, g_t)["max_load"]

model.con_grid_P = pyo.Constraint(model.T_id, model.tech_GN_ids,
    rule=grid_max_pwr)
model.con_grid_N = pyo.Constraint(model.T_id, model.tech_GN_ids,
    rule=grid_min_pwr)

def grid_peak_excess_pwr(model, t, n, g_t):
    return model.PgridPeakExcess[n, g_t] >= \
           model.PgridN[t, n, g_t] + model.PgridP[t, n, g_t] - get_tech(config, n, g_t)["base_peak"]

model.con_grid_peak = pyo.Constraint(model.T_id, model.tech_GN_ids,
    rule=grid_peak_excess_pwr)

def link_pwr_max_P(model, t, l):
    return model.PlinkP[t, l] <= \
           get_link(config, l)["cap_pwr"] * get_link(config, l)["max_outin"]

def link_pwr_max_N(model, t, l):
    return model.PlinkN[t, l] <= \
           get_link(config, l)["cap_pwr"] * get_link(config, l)["max_inout"]

model.con_link_P = pyo.Constraint(model.T_id, model.L_id,
    rule=link_pwr_max_P)
model.con_link_N = pyo.Constraint(model.T_id, model.L_id,
    rule=link_pwr_max_N)

def load_curt_max(model, t, n, lo_t):
    return model.Plc[t, n, lo_t] <= get_techs(config, n)[lo_t]["profile"]["load"][t]

model.con_load_curt = pyo.Constraint(model.T_id, model.tech_LN_ids,
    rule=load_curt_max)

def node_balance(model, t, n):
    if (get_techs(config, n) is None) and (
                (get_links(config) is None) or all(
                    vl["in"] != n and vl["out"] != n for (l, vl) in get_links(config).items()
                )
            ):
        return pyo.Constraint.Skip
    else:
        return (
                    0 if (get_techs(config, n) is None) else (
                        sum(model.Pren[t, n, tech]
                            for tech in get_techs_of_type(config, n, REN_T))
                        + sum(model.PbattP[t, n, tech] - model.PbattN[t, n, tech]
                              for tech in get_techs_of_type(config, n, BATT_T))
                        + sum(model.PgridN[t, n, tech] - model.PgridP[t, n, tech]
                              for tech in get_techs_of_type(config, n, GRID_T))
```

```
                            - sum(get_techs(config, n)[tech]["profile"]["load"][t] - model.Plc[t, n, tech]
                                  for tech in get_techs_of_type(config, n, LOAD_T))
                            )
                        ) + (0 if (get_links(config) is None) else (
                                + sum(+ model.PlinkP[t, l] - model.PlinkN[t, l]/vl["etainout"]
                                      for (l, vl) in get_links(config).items() if vl["in"] == n)
                                + sum(- model.PlinkP[t, l]/vl["etaoutin"] + model.PlinkN[t, l]
                                      for (l, vl) in get_links(config).items() if vl["out"] == n)
                            )
                        ) == 0

    model.con_balance_node = pyo.Constraint(model.T_id, model.N_id, rule=node_balance)

    lc_cost = sum(model.Plc[t, n, lo_t]*get_techs(config, n)[lo_t]["unmet_cost"] * get_parameter(config, "time_res")
                  for t in model.T_id for (n, lo_t) in model.tech_LN_ids)
    grid_cost = sum(- model.PgridP[t, n, g_t]*get_techs(config, n)[g_t]["profile"]["price_sell"][t] * get_parameter(config,
"time_res")
                    + model.PgridN[t, n, g_t]*get_techs(config, n)[g_t]["profile"]["price_buy"][t] * get_parameter(config,
"time_res")
                    for t in model.T_id for (n, g_t) in model.tech_GN_ids)
    peak_grid_cost = sum(
                    model.PgridPeakExcess[n, g_t]*get_techs(config, n)[g_t]["peak_cost"]
                    for (n, g_t) in model.tech_GN_ids)
    batt_cost = sum(+ model.PbattP[t, n, b_t]*get_techs(config, n)[b_t]["cost_dch"] * get_parameter(config, "time_res")
                    + model.PbattN[t, n, b_t]*get_techs(config, n)[b_t]["cost_ch"] * get_parameter(config, "time_res")
                    for t in model.T_id for (n, b_t) in model.tech_BN_ids)

    obj_rule = lc_cost + grid_cost + batt_cost + peak_grid_cost

    model.objective = pyo.Objective(expr=obj_rule, sense=pyo.minimize)

    # Should print all available solvers
    # opt = SolverFactory("gurobi", solver_io="python")
    opt = SolverFactory(solver_name)

    # Create a model instance and optimize
    opt_status_result = opt.solve(model)
    # model.display()

    return model, opt_status_result, config


def get_results(model, config, generate_plots=True, folder="./Img/",
                text_axes_size=10, text_general_size=10, text_legend_size=10,
                column_spacing=0.2,
                figsize_dispatch=(6, 3.3), figsize_shares=(2.8, 2.8)):
    """Function to generate the plots of the model"""

    df_list = {}

    # number of time steps and vector of time steps
    n_steps = int(get_parameter(config, "n_steps"))
    time_steps = list(map(lambda x: x * get_parameter(config, "time_res"), range(0, n_steps)))

    # nodes of the network
    nodes = get_nodes_list(config)
    n_nodes = len(nodes)

    for n in nodes:
        data_node = pd.DataFrame()
        data_node[REN_T] = [0 if (get_techs(config, n) is None) else sum(
                    model.Pren[t, n, r_t].value for r_t in get_techs_of_type(config, n, REN_T)
                )
                for t in model.T_id]
        data_node[LOAD_T] = [0 if (get_techs(config, n) is None) else sum(
                    get_techs(config, n)[l_t]["profile"]["load"][t] for l_t in get_techs_of_type(config, n, LOAD_T)
                )
                for t in model.T_id]
        data_node[LOAD_T + "_lc"] = [0 if (get_techs(config, n) is None) else sum(
                    model.Plc[t, n, l_t].value for l_t in get_techs_of_type(config, n, LOAD_T)
                )
                for t in model.T_id]
        data_node[LOAD_T + "_lc_c"] = [0 if (get_techs(config, n) is None) else sum(
                    model.Plc[t, n, l_t].value * get_techs(config, n)[l_t]["unmet_cost"] * get_parameter(config, "time_res")
                    for l_t in get_techs_of_type(config, n, LOAD_T)
                )
                for t in model.T_id]
        data_node[BATT_T] = [0 if (get_techs(config, n) is None) else sum(
                    model.PbattP[t, n, b_t].value - model.PbattN[t, n, b_t].value for b_t in get_techs_of_type(config, n, BATT_T)
                )
                for t in model.T_id]
        data_node[BATT_T + "_E"] = [0 if (get_techs(config, n) is None) else sum(
                    model.Ebatt[t, n, b_t].value for b_t in get_techs_of_type(config, n, BATT_T)
                )
                for t in model.T_id]
        data_node[GRID_T] = [0 if (get_techs(config, n) is None) else sum(
                    model.PgridP[t, n, g_t].value - model.PgridN[t, n, g_t].value for g_t in get_techs_of_type(config, n, GRID_T)
                )
                for t in model.T_id]
        data_node[LINK_T] = [0 if (get_links(config) is None) else (
                    + sum(+ model.PlinkP[t, l].value - model.PlinkN[t, l].value / vl["etainout"]
                          for (l, vl) in get_links(config).items() if vl["in"] == n)
                    + sum(- model.PlinkP[t, l].value / vl["etaoutin"] + model.PlinkN[t, l].value
                          for (l, vl) in get_links(config).items() if vl["out"] == n)
                )
                for t in model.T_id]

        if get_techs(config, n) is not None:
            for (tech, val) in get_techs(config, n).items():
                if val["type"] == REN_T:
                    data_node[tech] = [model.Pren[t, n, tech].value for t in model.T_id]
                elif val["type"] == BATT_T:
                    data_node[tech + "_P"] = [model.PbattP[t, n, tech].value for t in model.T_id]
                    data_node[tech + "_N"] = [model.PbattN[t, n, tech].value for t in model.T_id]
                    data_node[tech + "_E"] = [model.Ebatt[t, n, tech].value for t in model.T_id]
                elif val["type"] == GRID_T:
                    data_node[tech + "_P"] = [model.PgridP[t, n, tech].value for t in model.T_id]
                    data_node[tech + "_N"] = [model.PgridN[t, n, tech].value for t in model.T_id]
                elif val["type"] == LOAD_T:
                    data_node[tech] = [get_techs(config, n)[tech]["profile"]["load"][t] for t in model.T_id]
                    data_node[tech + "_lc"] = [model.Plc[t, n, tech].value for t in model.T_id]

        if get_links(config) is not None:
            for (l, val) in get_links(config).items():
                if val["in"] == n:
```

```
                data_node[l] = [model.PlinkP[t, l].value - model.PlinkN[t, l].value / val["etainout"] for t in model.T_id]
            elif val["out"] == n:
                data_node[l] = [- model.PlinkP[t, l].value / val["etaoutin"] + model.PlinkN[t, l].value for t in model.T_id]

        df_list[n] = data_node


    # reliability indices

    # PREVIOUS IMPLEMENTATION: indices by nodes
    # ENS = {n: float(df_list[n][LOAD_T + "_lc"].values.sum()) for n in nodes}  # Energy Not Served by node
    # ENS_cost = {n: float(df_list[n][LOAD_T + "_lc_c"].values.sum()) for n in nodes}  # Cost of the Energy Not Served by node
    # load_node = {n: float(df_list[n][LOAD_T].values.sum()) for n in nodes}  # Total demand by node
    #
    # ENS_pu = {n: float(ENS[n]/load_node[n] if load_node[n] > 0.0 else 0.0) for n in nodes}  # specific energy not served by node
    # ENS_cost_pu = {n: float(ENS_cost[n]/load_node[n] if load_node[n] > 0.0 else 0.0) for n in nodes}  # specific cost of the energy
not served by node
    #
    # reliability_indices = {n: {"ENS": ENS[n], "ENS_cost": ENS_cost[n], "ENS_pu": ENS_pu[n], "ENS_cost_pu": ENS_cost_pu[n],
"tot_load": load_node[n]} for n in nodes}

    # New implementation: indices by load ids

    ENS = {f"{n}_{l_t}": sum(list(model.Plc[:, n, l_t].value)) * get_parameter(config, "time_res") for (n, l_t) in model.tech_LN_ids}
# Energy Not Served by node&user
    ENS_cost = {f"{n}_{l_t}": sum(list(model.Plc[:, n, l_t].value)) * get_techs(config, n)[l_t]["unmet_cost"] * get_parameter(config,
"time_res") for (n, l_t) in model.tech_LN_ids}  # Cost of the Energy Not Served by node&user
    load_nodeuser = {f"{n}_{l_t}": sum(get_techs(config, n)[l_t]["profile"]["load"]) * get_parameter(config, "time_res") for (n, l_t)
in model.tech_LN_ids}  # Total demand by node&user

    ENS_pu = {n: float(ENS[n]/load_nodeuser[n] if load_nodeuser[n] > 0.0 else 0.0) for n in ENS.keys()}  # specific energy not served
by node
    ENS_cost_pu = {n: float(ENS_cost[n]/load_nodeuser[n] if load_nodeuser[n] > 0.0 else 0.0) for n in ENS.keys()}  # specific cost of
the energy not served by node

    reliability_indices = {n: {"ENS": ENS[n], "ENS_cost": ENS_cost[n], "ENS_pu": ENS_pu[n], "ENS_cost_pu": ENS_cost_pu[n],
"tot_load": load_nodeuser[n]} for n in ENS.keys()}


    reliability_indices[code_all_system] = {
        "ENS": float(sum(ENS.values())),
        "ENS_cost": float(sum(ENS_cost.values())),
        "ENS_pu": float(sum(ENS.values())/sum(load_nodeuser.values()) if sum(load_nodeuser.values()) > 0.0 else 0.0),
        "ENS_cost_pu": float(sum(ENS_cost.values())/sum(load_nodeuser.values()) if sum(load_nodeuser.values()) > 0.0 else 0.0),
        "tot_load": float(sum(load_nodeuser.values())),
        "objective": float(value(model.objective)),
    }

    # list of axis plots

    if generate_plots:

        os.makedirs(folder, exist_ok=True)

        plt.rc('font', size=text_general_size)  # controls default text sizes
        plt.rc('axes', titlesize=text_axes_size)  # fontsize of the axes title
        plt.rc('axes', labelsize=text_axes_size)  # fontsize of the x and y labels
        plt.rc('xtick', labelsize=text_axes_size)  # fontsize of the tick labels
        plt.rc('ytick', labelsize=text_axes_size)  # fontsize of the tick labels
        plt.rc('legend', fontsize=text_legend_size)  # legend fontsize
        plt.rc('figure', titlesize=text_general_size)  # fontsize of the figure title

        figs = []
        axs = []

        for n in nodes:
            fig, ax = plt.subplots(figsize=figsize_dispatch)
            ax.set_title("Power balance - %s" % n)
            axs.append(ax)
            figs.append(fig)

        # plot conventions: positive value inflow the node, negative value outflow
        for (n, n_i) in zip(nodes, range(n_nodes)):
            if has_tech_type(config, n, REN_T):
                axs[n_i].plot(time_steps, df_list[n][REN_T], label="Renewables")
            if has_tech_type(config, n, BATT_T):
                axs[n_i].plot(time_steps, df_list[n][BATT_T], label="Battery")
            if has_tech_type(config, n, GRID_T):
                axs[n_i].plot(time_steps, -df_list[n][GRID_T], label="Grid")
            if has_tech_type(config, n, LOAD_T):
                axs[n_i].plot(time_steps, -df_list[n][LOAD_T], label="Load")
                axs[n_i].plot(time_steps, df_list[n][LOAD_T + "_lc"], label="Load curtailment")

            if get_links(config) is not None:
                for (l, val) in get_links(config).items():
                    if val["in"] == n or val["out"] == n:
                        axs[n_i].plot(time_steps, df_list[n][l], label=l)

            lines, labels = axs[n_i].get_legend_handles_labels()

            if has_tech_type(config, n, BATT_T):
                ax2 = axs[n_i].twinx()
                ax2.plot(time_steps, df_list[n][BATT_T + "_E"], '--', label="Batt. energy")
                ax2.set_ylabel("Energy [kWh]")
                lines2, labels2 = ax2.get_legend_handles_labels()
                lines = lines + lines2
                labels = labels + labels2

            axs[n_i].legend(lines, labels, loc='upper center',ncol=2, bbox_to_anchor=(0.5, -0.22))
            axs[n_i].set_xlabel("Time [h]")
            axs[n_i].set_ylabel("Power [kW]")

        for (fig, ax) in zip(figs, axs):
            fig.tight_layout()
            fig.savefig(folder + '/%s.png' % ax.title.get_text(), dpi=fig.dpi)

        # calculate the total demand and production including losses
        tot_demand = 0  # total demand
        losses = 0  # total losses
        load_curt = 0  # total load curtailment
        ren_production = 0  # total renewable production
        battery_flow_dch = 0  # total battery discharge (bus side)
        battery_flow_ch = 0  # total battery charge (bus side)
        grid_buy = 0  # total energy bought from the grid
```

```
        grid_sell = 0  # total energy sold to the grid
        delta_storage = 0  # difference between initial and final energy storage

        for (n, n_i) in zip(nodes, range(n_nodes)):
            for (tech_name, tech_val) in get_techs(config, n).items():
                if is_tech_type(tech_val, REN_T):
                    ren_production += sum(df_list[n][tech_name]) * get_parameter(config, "time_res")
                if is_tech_type(tech_val, BATT_T):
                    battery_flow_ch += sum(df_list[n][tech_name + "_N"]) * get_parameter(config, "time_res")
                    battery_flow_dch += sum(df_list[n][tech_name + "_P"]) * get_parameter(config, "time_res")
                    delta_storage += df_list[n][tech_name + "_E"].iloc[-1] - get_tech(config, n, tech_name)["cap_en"] *
get_tech(config, n, tech_name)["init_SOC"]
                    losses += \
                        + sum(df_list[n][tech_name + "_N"]) * get_parameter(config, "time_res") * (1 - get_tech(config, n,
tech_name)["eta_ch"]) + \
                        + sum(df_list[n][tech_name + "_P"]) * get_parameter(config, "time_res") * (1/get_tech(config, n,
tech_name)["eta_dch"] - 1)
                if is_tech_type(tech_val, GRID_T):
                    grid_buy += sum(df_list[n][tech_name + "_N"]) * get_parameter(config, "time_res")
                    grid_sell += sum(df_list[n][tech_name + "_P"]) * get_parameter(config, "time_res")
                if is_tech_type(tech_val, LOAD_T):
                    curr_curt = sum(df_list[n][tech_name + "_lc"]) * get_parameter(config, "time_res")
                    tot_demand += sum(df_list[n][tech_name]) * get_parameter(config, "time_res")
                    load_curt += curr_curt

        # calculate losses due to the links
        for (l, val) in get_links(config).items():
            losses += sum(abs(df_list[val["in"]][l] + df_list[val["out"]][l])) * get_parameter(config, "time_res")

        tot_demand_and_losses = tot_demand + losses

        # print the energy balance, this number shall be zero
        # print(tot_demand_and_losses - ren_production + delta_storage - grid_buy + grid_sell - load_curt)

        # fix numerical issues
        if load_curt > -1e-6:
            load_curt = max(load_curt, 0.0)

        load_pie = [tot_demand-load_curt, losses, load_curt]
        #labels_pie_load = ["Demand", "Losses", "ENS"]
        labels_pie_load = ["Carico", "Perdite", "ENS"]

        fig_load, ax_load = plt.subplots(figsize=figsize_shares)
        #ax_load.set_title("Demand, losses and ENS")
        ax_load.set_title("Carico, perdite ed ENS")
        ax_load.pie(load_pie, autopct='%1.1f%%')
        ax_load.legend(labels_pie_load, loc="lower center", ncol=len(labels_pie_load),
                       columnspacing=column_spacing)
        fig_load.savefig(folder + '/LoadPie.png', dpi=fig.dpi)

        ren_self_cons = ren_production - grid_sell - delta_storage
        self_cons_pie = [grid_buy, ren_self_cons, load_curt]
        #labels_pie_self_cons = ["Grid buy", "PV prod + Batt", "ENS"]
        labels_pie_self_cons = ["Acquisto rete", "Prod. PV + Batt", "ENS"]

        fig_self, ax_self = plt.subplots(figsize=figsize_shares)
        #ax_self.set_title("Self consumption")
        ax_self.set_title("Autoconsumo")
        ax_self.pie(self_cons_pie, autopct='%1.1f%%')
        ax_self.legend(labels_pie_self_cons, loc="lower center", ncol=2,  # ncol=len(labels_pie_self_cons),
                       columnspacing=column_spacing)
        fig_self.savefig(folder + '/SelfConsPie.png', dpi=fig.dpi)

    plt.close('all')

    return df_list, reliability_indices


def run_deterministic_EMS(config,
                          solver_name="glpk",
                          data_file_profiles="",
                          generate_plots=True,
                          ouput_folder="./Img/"):
    """
    This is the main function to run the deterministic EMS

    Parameters
    ----------
    config: str or dict
        Configuration file of the network
    solver_name: str
        Solver name
    data_file_profiles (optional): str
        Path to a csv file containing additional time series that may be needed
    generate_plots: bool
        If true, it generates output plots
    output_folder:
        If generate_plots is true, folder where the output plots are saved

    Outputs
    -------
    reliability_indices: dict
        Dictionary where the reliability outputs are stored
    model: Pyomo model
        Pyomo model of the EMS
    results: dict
        Results collection
    config: dict
        Configuration dictionary used in the EMS
    """
    # execute the EMS
    model, opt_status_result, config = optimal_dispatch(config,  data_file_profiles=data_file_profiles,
                                    solver_name=solver_name)

    # obtain results
    outresults, reliability_indices = get_results(model, config, generate_plots=generate_plots, folder=ouput_folder)

    return reliability_indices, model, outresults, config


if __name__ == "__main__":
    # model, opt_status_result, config = optimal_dispatch("ems-data-15_10_2021.yml",  # data_file_profiles="datafile.csv",
    #                                 solver_name="glpk")
    model, opt_status_result, config = optimal_dispatch("data.yml",  data_file_profiles="datafile.csv",
                                    solver_name="glpk")
```

```
with open('config.yml', 'w') as outfile:
    yaml.dump(config, outfile, default_flow_style=False)

outresults, reliability_indices = get_results(model, config, generate_plots=True)
```

4.2.2.5 ems_dict_correct.py

```
from __shared__ import variables as v
from UI.Main.Elements.ACLine.acline import ACLine
from UI.Main.Elements.ACLoads.acload import ACLoad
from UI.Main.Elements.ACwind.acwind import  ACwind
from UI.Main.Elements.Battery.battery import Battery
from UI.Main.Elements.DCDC_converter.dcdc_conv import DCDC_conv
from UI.Main.Elements.DCLine.dcline import DCLine
from UI.Main.Elements.DCLoads.dcload import DCLoad
from UI.Main.Elements.DCwind.dcwind import DCwind
from UI.Main.Elements.PV.pv import PV
from UI.Main.Elements.PWM.pwm import PWM
from UI.Main.Elements.Transformers_TwoWings.transformer_TwoWings import Tr2W


def dict_correct():
    for elem in v.elements:
        if v.elements[elem]['category'] == 'AC-Line':
            elem_class = ACLine(elem)
        elif v.elements[elem]['category'] == 'AC-Load':
            elem_class = ACLoad(elem)
        elif v.elements[elem]['category'] == 'AC-Wind':
            elem_class = ACwind(elem)
        elif v.elements[elem]['category'] == 'Battery':
            elem_class = Battery(elem)
        elif v.elements[elem]['category'] == 'DC-DC_Conv':
            elem_class = DCDC_conv(elem)
        elif v.elements[elem]['category'] == 'DC-Line':
            elem_class = DCLine(elem)
        elif v.elements[elem]['category'] == 'DC-Load':
            elem_class = DCLoad(elem)
        elif v.elements[elem]['category'] == 'DC-Wind':
            elem_class = DCwind(elem)
        elif v.elements[elem]['category'] == 'PV':
            elem_class = PV(elem)
        elif v.elements[elem]['category'] == 'PWM':
            elem_class = PWM(elem)
        elif v.elements[elem]['category'] == '2W-Transformer':
            elem_class = Tr2W(elem)
        else:
            elem_class = None

        if elem_class:
            elem_class.calculate()
            elem_class.store()
```

4.2.2.6 utils.py

```
# utils file
REN_T = "renewable"
GRID_T = "grid"
LOAD_T = "load"
BATT_T = "battery"
LINK_T = "link"

# Reliability indices codes
REL_INDICES = ["ENS", "ENS_cost", "ENS_pu", "ENS_cost_pu", "tot_load"]
# name to identify the reliability of the entire system
code_all_system = "ALL_SYSTEM"
name_bc = "base_model"  # name base scenario


def get_parameters(data):
    return data["parameters"]


def get_parameter(data, param_name):
    return get_parameters(data)[param_name]


def get_nodes(data):
    return data["nodes"]


def get_node(data, node_name):
    return get_nodes(data)[node_name]


def get_techs(data, node_name):
    return get_node(data, node_name)["techs"]


def get_techs_of_type(data, node_name, type_t):
    return [t for t, vt in get_techs(data, node_name).items() if vt["type"] == type_t]


def get_tech(data, node_name, tech_name):
    return get_techs(data, node_name)[tech_name]


def is_tech_type(tech, type_t):
    return tech["type"] == type_t


def is_tech_type_by_name(data, node_name, tech_name, type_t):
    return is_tech_type(get_tech(data, node_name, tech_name), type_t)


def get_nodes_list(data):
    return list(get_nodes(data).keys())


def has_tech_type(data, node_name, type_t):
    tech_list = get_techs(data, node_name)
    return tech_list and any([vt["type"] == type_t for nt, vt in tech_list.items()])


def get_links(data):
    return data["links"]


def get_link(data, name_link):
    return get_links(data)[name_link]


def get_links_list(data):
    return list(get_links(data).keys())


def pre(t, time_set):
    if t == time_set[1]:
        return time_set.last()
    else:
        return time_set.prev(t)


def get_cap_mult_stoch_comps(scenario, tech, t):
    multiplier = 1
    if type(scenario) == dict and "components" in scenario.keys() and \
            scenario["components"] is not None and \
            tech in scenario["components"].keys():
        if "mod_capacity" in scenario["components"][tech].keys():
            if (t >= scenario["components"][tech]["mod_capacity"]["start"]) & (t <=
scenario["components"][tech]["mod_capacity"]["end"]):
                multiplier = scenario["components"][tech]["mod_capacity"]["multiplier"]
    return multiplier


def get_cap_mult_stoch_links(scenario, tech, t):
    multiplier = 1
    if type(scenario) == dict and "links" in scenario.keys() and \
            scenario["links"] is not None and \
            tech in scenario["links"].keys():
        if "mod_capacity" in scenario["links"][tech].keys():
            if (t >= scenario["links"][tech]["mod_capacity"]["start"]) & (t <= scenario["links"][tech]["mod_capacity"]["end"]):
                multiplier = scenario["links"][tech]["mod_capacity"]["multiplier"]
    return multiplier


def update_reliability_dict(reliability_indices, reliability_indices_scenario, prob, config):
    for key_rel in REL_INDICES:
        for n in list(set(list(reliability_indices_scenario.keys())) - set([code_all_system])):
            reliability_indices[n][key_rel] += reliability_indices_scenario[n][key_rel] * prob
            reliability_indices[code_all_system][key_rel] += reliability_indices_scenario[n][key_rel] * prob
    reliability_indices[code_all_system]["objective"] += reliability_indices_scenario[code_all_system]["objective"] * prob


def initialize_reliability_dict(config):
    rel_dict = {f"{n}_{tech}": {rel_code: 0.0 for rel_code in REL_INDICES}
                for n in get_nodes_list(config) if get_node(config, n)["techs"] is not None
```

```
                for tech, vt in get_node(config, n)["techs"].items() if vt["type"] == LOAD_T}
    rel_dict[code_all_system] = {rel_code: 0.0 for rel_code in [*REL_INDICES, "objective"]}
    return rel_dict
```

4.2.3 PDF

4.2.3.1 pdf_creator.py

```
from fpdf import FPDF
import os
import yaml
from __shared__ import variables as v
from pathlib import Path
from PyQt5 import QtWidgets
import copy


root = os.getcwd()

# -- Cells and rows parameters -------------------
t1_h, t1_c, t1_ls, t1_s = 10, 16, 12, 'B'    # Title 1 (height, font_size, line spacing, style)
t2_h, t2_c, t2_ls, t2_s = 4, 10, 12, 'B'     # Title 2 (height, font_size, line spacing, style)
p_h, p_c, p_ls, p_s = 3, 6, 8, ''          # Paragraph (height, font_size, line spacing, style)
i_h, i_c, i_ls, i_s = 3, 6, 8, 'I'         # Captions (height, font_size, line spacing, style)
e_w, p_w, d_w = 35, 5, 17       # Cells width (element, port, data)
# ------------------------------------------------

par = dict()
par['Nodi AC'] = [['Ur'], ['Ur [kV]']]
par['Nodi DC'] = [['Ur'], ['Ur [kV]']]
par['Carichi AC'] = [['P', 'Q', 'S', 'cosPhi', 'I'],
                     ['P [kW]', 'Q [kVA]', 'S [kVA]', 'cosPhi', 'I [A]']]
par['Carichi DC'] = [['P', 'R', 'I'], ['P [kW]', 'R [Ohm]', 'I [A]']]
par['PWM'] = [['Sr', 'Ur', 'S_loss_idle', 'Sw_loss', 'R_loss'],
              ['Sr [kVA]', 'Ur [kV]', 'S loss IDLE [kVA]', 'S loss [kVA/kV]', 'R loss [Ohm]']]
par['Convertitori DC-DC'] = [['Sr'], ['Sr [kVA]']]
par['Trasformatori'] = [['Sr', 'UnHV', 'UnLV', 'URr1', 'Ukr1', 'URr0', 'Ukr0'],
                        ['Sr [kVA]', 'UnHV [kV]', 'UnLV [kV]', 'URr1 [p.u.]', 'Ukr1 [p.u.]', 'URr0 [p.u.]',
                         'Ukr0 [p.u.]']]
par['Fotovoltaico'] = [['Sr'], ['Sr [kVA]']]
par['Eolici AC'] = [['units', 'P', 'Q', 'S', 'cosPhi', 'Sr'],
                    ['unità', 'P [kW]', 'Q [kVA]', 'S [kVA]', 'cosPhi', 'Sr [kVA]']]
par['Eolici DC'] = [['Pr', 'Ir', 'P'], ['Pr [kW]', 'Ir [A]', 'P [kVW]']]
par['Batterie'] = [['cap_en'], ['Capacità [kWh]']]
par['Linee AC'] = [['length', 'lines', 'Irmax', 'R1', 'X1', 'B1', 'R0', 'X0', 'B0'],
                   ['Lunghezza [m]', 'N. linee', 'Ir max [A]', 'R1 [Ohm/km]', 'X1 [Ohm/km]', 'B1 [uS/km]',
                    'R0 [Ohm/km]', 'X0 [Ohm/km]', 'B0 [uS/km]']]
par['Linee DC'] = [['length', 'lines', 'Irmax', 'R', 'L', 'C'],
                   ['Lunghezza [m]', 'N. linee', 'Ir max [A]', 'R [Ohm/km]', 'L [mH/km]', 'C [uF/km]']]

lf = dict()
lf['Nodi'] = [['U', 'Pgen', 'Qgen', 'Pload', 'Qload', 'LimitViolated'],
              ['U [kV]', 'P gen [kW]', 'Q gen [kVA]', 'P load [kW]', 'Q load [kVA]', 'Violazione'], 1]
lf['Periferiche'] = [['P', 'Q', 'S', 'cosPhi', 'I', 'Iangle', 'U', 'LimitViolated'],
                     ['P [kW]', 'Q [kVA]', 'S [kVA]', 'cosPhi', 'I [A]', 'Angolo I [°]', 'U [kV]', 'Violazione'], 1]
lf['Links'] = [['P', 'Q', 'S', 'cosPhi', 'I', 'Iangle', 'U', 'LimitViolated'],
               ['P [kW]', 'Q [kVA]', 'S [kVA]', 'cosPhi', 'I [A]', 'Angolo I [°]', 'U [kV]', 'Violazione'], 2]

prot = [['type', 'cost', 'soglia_I', 'soglia_Vmax', 'soglia_Vmin', 'delay_I', 'delay_Vmax', 'delay_Vmin'],
        ['Tipologia', 'Costo [Euro]', 'Soglia I [A]', 'Soglia Vmax [kV]', 'Soglia Vmin [kV]',
         'Ritardo I [ms]', 'Ritardo Vmax [ms]', 'Ritardo Vmin [ms]']]

rel = [['lambda', 'R', 'MTBF_ore', 'MTBF_anni'], ['lambda [fail/10^6]', 'R', 'MTBF [ore]', 'MTBF [anni]']]

ctrl = [['ris1', 'ris2', 'ris3', 'ti', 'ens', 'ng', 'st', 'gf', 'ri'],
        ['Indice di autonomia [%]', 'Indice di flessibilità [%]', 'Indice di modulazione', 'Tempo (Ti) [min]',
         'Energia non fornita (ENS)', 'Generazione flessibile (Ng)', 'Riserva dello Storage (ST)',
         'Capacità di Grid Forming (GF)', 'Rapporto di inerzia (RI)']]

cat = dict()
cat['AC-Node'] = 'Nodi AC'
cat['DC-Node'] = 'Nodi DC'
cat['AC-Load'] = 'Carichi AC'
cat['DC-Load'] = 'Carichi DC'
cat['PWM'] = 'PWM'
cat['DC-DC_Conv'] = 'Convertitori DC-DC'
cat['2W-Transformer'] = 'Trasformatori'
cat['PV'] = 'Fotovoltaico'
cat['AC-Wind'] = 'Eolici AC'
cat['DC-Wind'] = 'Eolici DC'
cat['Battery'] = 'Batterie'
cat['AC-Line'] = 'Linee AC'
cat['DC-Line'] = 'Linee DC'


class PDF(FPDF):
    def __init__(self):
        super(PDF, self).__init__()

        elements = copy.deepcopy(v.elements)

        self.page_name = ''

        elem_cat = dict()

        for elem in elements:
            if elements[elem]['category'] != 'Ext-Grid':
                if elements[elem]['category'] not in elem_cat.keys():
                    elem_cat[elements[elem]['category']] = dict()
                elem_cat[elements[elem]['category']][elem] = elements[elem]

        self.set_draw_color(226, 226, 226)

        self.cover()
        self.parameters(elem_cat)

        for study in v.executed:
            if study == 'lf':
                self.loadflow(elem_cat=elem_cat)
            elif study == 'rel':
                self.reliability(elem_cat=elem_cat, elements=elements)
            elif study == 'prot':
                self.protections(elem_cat=elem_cat)
            elif study == 'con':
                self.controls()
            elif study == 'ems':
```

```
            self.ems(elem_cat=elem_cat)
        elif study == 'ei':
            self.ems_indexes()

#
def del_PDF(self, filepath='mypdf.pdf'):
    try:
        os.remove(filepath)
    except:
        pass

def header(self):
    if self.page_no() > 1:
        self.image(root + '/_images/SplashScreen/ORAT.png', 8, 8, 24.5, 20)
        self.set_font('Arial', 'B', t1_c)
        self.set_xy(34, 18)
        self.set_text_color(217, 193, 221)
        self.write(0, 'Optimization')
        self.set_font('Arial', '', t1_c)
        self.write(0, ' and')

        self.set_font('Arial', 'B', t1_c)
        self.set_xy(34, 25)
        self.set_text_color(194, 214, 236)
        self.write(0, 'Reliability')
        self.set_text_color(204, 191, 233)
        self.write(0, 'Assessment')
        self.set_text_color(240, 233, 173)
        self.write(0, 'Tool')

        self.set_xy(150, self.get_y() - t1_h)
        self.set_font('Arial', 'B', t1_c)
        self.set_text_color(0, 0, 0)
        self.cell(50, t1_h*2, v.features['name'], 0, 0, 'R')

        self.set_draw_color(0, 0, 0)
        self.set_line_width(0.5)
        self.line(0, 30, 210, 30)

        self.set_y(40)

#
def footer(self):
    if self.page_no() > 1:
        self.set_draw_color(0, 0, 0)
        self.set_line_width(0.5)
        self.line(0, 280, 210, 280)
        self.set_xy(10, 282)
        self.set_font('Arial', 'B', t2_c)
        self.set_text_color(0, 0, 0)
        self.cell(95, t2_h, self.page_name, 0, 0, 'L')
        self.cell(95, t2_h, 'Pagina ' + str(self.page_no()), 0, 0, 'R')

#
def cover(self):
    self.add_page()
    self.set_font('Arial', t1_s, 1.5*t1_c)
    self.image(root + '/_images\SplashScreen/ORAT.png', 55, 80, 100, 82)
    self.set_xy(70, 170)
    self.set_text_color(217, 193, 221)
    self.write(0, 'Optimization')
    self.set_font('Arial', '', 1.5*t1_c)
    self.write(0, ' and')
    self.set_font('Arial', 'B', 1.5*t1_c)
    self.set_xy(50, 180)
    self.set_text_color(194, 214, 236)
    self.write(0, 'Reliability')
    self.set_text_color(204, 191, 233)
    self.write(0, 'Assessment')
    self.set_text_color(240, 233, 173)
    self.write(0, 'Tool')

    self.set_xy(20, 230)
    self.set_text_color(0, 0, 0)
    self.cell(170, 1.5*t1_h, v.features['name'], 0, 0, 'C')

#
def parameters(self, elem_cat):
    self.add_page()

    self.page_name = 'Parametri'

    self.set_font('Arial', t1_s, t1_c)
    self.write(0, 'Parametri')
    self.ln(t1_ls)

    for c in elem_cat:
        line_length = e_w + d_w * len(par[cat[c]][1])
        if 275 < self.get_y() + t2_h + i_h + p_h*len(elem_cat[c].items()):
            self.add_page()
        self.set_font('Arial', t2_s, t2_c)
        self.write(0, cat[c])
        self.ln(t2_h)
        self.set_font('Arial', i_s, i_c)
        self.set_fill_color(226, 226, 226)
        self.cell(e_w, i_h, 'Elemento', 1, 0, 'C', fill=1)
        for e in par[cat[c]][1]:
            self.cell(d_w, i_h, e, 1, 0, 'C', fill=1)
        self.ln(i_h)

        self.set_font('Arial', p_s, p_c)
        for e in elem_cat[c]:
            self.cell(e_w, p_h, e, 0, 0, 'L')
            for item in par[cat[c]][0]:
                self.cell(d_w, p_h, '%.3f' % elem_cat[c][e]['parameters'][item], 0, 0, 'C')
            self.ln(p_h)
            self.line(self.get_x() + 1, self.get_y(), self.get_x() + line_length - 2, self.get_y())
        self.ln(p_ls)

#
def loadflow(self, elem_cat):
    self.add_page()

    self.page_name = 'LoadFlow'
```

```
        self.set_font('Arial', t1_s, t1_c)
        self.write(0, 'LoadFlow')
        self.ln(t1_ls)

        for c in elem_cat:
            if c in ['AC-Node', 'DC-Node']:
                captions = lf['Nodi']
                node, line = '', 1

            elif c in ['2W-Transformer', 'AC-Line', 'DC-Line', 'PWM', 'DC-DC_Conv']:
                captions = lf['Links']
                node, line = 'Nodo', 2
            else:
                captions = lf['Periferiche']
                node, line = '', 1

            if 275 < self.get_y() + t2_h + i_h + p_h * len(elem_cat[c].items()) * line:
                self.add_page()
            self.set_font('Arial', t2_s, t2_c)
            self.write(0, cat[c])
            self.ln(t2_h)
            self.set_font('Arial', i_s, i_c)
            self.set_fill_color(226, 226, 226)
            self.cell(e_w, i_h, 'Elemento', 1, 0, 'C', fill=1)
            self.cell(p_w, i_h, node, 1, 0, 'C', fill=1)

            line_length = e_w + p_w + d_w * len(captions[1])

            for e in captions[1]:
                self.cell(d_w, i_h, e, 1, 0, 'C', fill=1)
            self.ln(i_h)

            self.set_font('Arial', p_s, p_c)
            for e in elem_cat[c]:
                if node != '':
                    self.cell(e_w, 2*p_h, e, 0, 0, 'L')
                    if c in ['2W-Transformer', 'DC-DC_Conv']:
                        nodes = ['HV', 'LV']
                    elif c == 'PWM':
                        nodes = ['AC', 'DC']
                    else:
                        nodes = ['1', '2']

                    for n in [1, 2]:
                        if n == 2:
                            self.cell(e_w, p_h, '', 0, 0, 'C')
                        self.cell(p_w, p_h, nodes[n-1], 0, 0, 'C')
                        for item in captions[0][:len(captions[0])-1]:
                            try:
                                data = '%.3f' % elem_cat[c][e]['results']['Port' + str(n)][item]
                            except:
                                data = ''
                            self.cell(d_w, p_h, data, 0, 0, 'C')
                        if n == 1:
                            self.ln(p_h)

                    self.set_xy(self.get_x(), self.get_y() - p_h)
                    print(e)
                    try:
                        data = str(elem_cat[c][e]['results']['LimitViolated'])
                    except:
                        data = ''
                    self.cell(d_w, 2 * p_h, data, 0, 0, 'C')
                    self.ln(2 * p_h)

                else:
                    self.cell(e_w, p_h, e, 0, 0, 'L')
                    self.cell(p_w, p_h, '', 0, 0, 'C')

                    for item in captions[0]:
                        try:
                            if item in captions[0][:len(captions[0])-1]:
                                data = '%.3f' % elem_cat[c][e]['results'][item]
                            else:
                                data = str(elem_cat[c][e]['results'][item])
                        except:
                            data = ''
                        self.cell(d_w, p_h, data, 0, 0, 'C')
                    self.ln(p_h)
                self.set_draw_color(226, 226, 226)
                self.line(self.get_x() + 1, self.get_y(), self.get_x() + line_length - 2, self.get_y())

            self.ln(p_ls)

    #
    def ems(self, elem_cat):
        self.add_page()
        self.page_name = 'EMS'

        img_path = os.getcwd() + '/_images/SplashScreen/EMS_80x80.png'
        self.set_xy(15, 35)
        self.image(img_path, self.get_x(), self.get_y(), 15, 15)
        self.set_xy(37.5, 44)

        self.set_draw_color(25, 25, 25)
        self.set_line_width(0.3)
        self.line(35, 38, 195, 38)
        self.line(35, 48, 195, 48)

        self.set_font('Arial', t1_s, t1_c)
        self.write(0, 'EMS')
        self.ln(t1_ls)

        img_path = v.project_folder + '/EMS/results'
        images = []
        for file in os.listdir(img_path):
            if file.endswith('.png') and file != 'LoadPie.png' and file != 'SelfConsPie.png':
                images.append(file)
        x, y = 20, 35   # devono essere i margini della pagina

        self.image(img_path + '/LoadPie.png', 45, 35, 120, 120)
        self.image(img_path + '/SelfConsPie.png', 45, 165, 120, 120)

        self.add_page()
        count = 0
```

```
        for img in images:
            if count >= 4:
                self.add_page()
                count = 0
                y = 35
            self.image(img_path + '/' + img, 55, y, 100, 55)
            y = y + 60
            count += 1
        print('ems done')

    # -- Stampa Protezioni ---------
    def protections(self, elem_cat):
        self.add_page()
        self.page_name = 'Protezioni'

        calc_prot = []
        prot_cost = 0
        prot_cost_max = 0
        prot_cost_min = 0

        for c in elem_cat:
            for elem in elem_cat[c]:
                if elem_cat[c][elem]['protections'] != {}:
                    prot_cost += elem_cat[c][elem]['protections']['results']['cost']
                    prot_cost_min += elem_cat[c][elem]['protections']['results']['comparison']['em']['cost']
                    prot_cost_max += elem_cat[c][elem]['protections']['results']['comparison']['el']['cost']
                    if elem_cat[c][elem]['category'] not in calc_prot:
                        calc_prot.append(elem_cat[c][elem]['category'])

        img_path = os.getcwd() + '/_images/SplashScreen/Co_80x80.png'
        self.set_xy(15, 35)
        self.image(img_path, self.get_x(), self.get_y(), 15, 15)
        self.set_xy(37.5, 44)

        self.set_draw_color(25, 25, 25)
        self.set_line_width(0.3)
        self.line(35, 38, 195, 38)
        self.line(35, 48, 195, 48)

        self.set_font('Arial', t1_s, t1_c)
        self.write(0, 'Protezioni')
        self.ln(t1_ls)

        self.set_font('Arial', t2_s, t2_c)
        self.cell(100, t2_h, 'Costo delle Protezioni', fill=1)
        self.ln(t2_h*2)
        self.set_font('Arial', p_s, p_c*1.5)
        self.cell(e_w, p_h, 'Costo minimo: ', 0, 0, 'R')
        self.cell(e_w, p_h, '%.2f Euro' % prot_cost_min, 0, 0, 'R')
        self.ln(p_h*1.5)
        self.cell(e_w, p_h, 'Costo massimo: ', 0, 0, 'R')
        self.cell(e_w, p_h, '%.2f Euro' % prot_cost_max, 0, 0, 'R')
        self.ln(p_h*2)
        self.set_font('Arial', p_s, p_c*2)
        self.cell(e_w, p_h, 'Costo Soluzione: ', 0, 0, 'R')
        self.cell(e_w, p_h, '%.2f Euro' % prot_cost, 0, 0, 'R')
        self.ln(p_h*5)

        self.set_font('Arial', t2_s, t2_c)
        self.write(0, 'Confronto caratteristiche protezioni')
        self.ln(t2_h)
        self.set_font('Arial', i_s, i_c*0.8)
        self.cell(e_w, i_h, '', 1, 0, 'C', fill=1)
        self.cell(d_w * 3, i_h, 'Protezioni elettromeccaniche', 1, 0, 'C', fill=1)
        self.cell(d_w/2, i_h, '', 1, 0, 'C', fill=1)
        self.cell(d_w * 3, i_h, 'Protezioni elettroniche', 1, 0, 'C', fill=1)
        self.ln(i_h)
        self.cell(e_w, i_h, 'elemento', 1, 0, 'C', fill=1)
        for i in range(0, 2):
            self.cell(d_w, i_h, 'Sovracorrente [A]', 1, 0, 'C', fill=1)
            self.cell(d_w, i_h, 'Sovrarensione [kv]', 1, 0, 'C', fill=1)
            self.cell(d_w, i_h, 'Costo [Euro]', 1, 0, 'C', fill=1)
            if i == 0:
                self.cell(d_w/2, i_h, '', 1, 0, 'C', fill=1)
        self.ln(i_h)

        for c in calc_prot:
            for e in elem_cat[c]:
                line_length = e_w + d_w * 6.5
                self.set_font('Arial', i_s, i_c)
                self.cell(e_w, p_h, e, 0, 0, 'L')
                for t in ['em', 'el']:
                    if (elem_cat[c][e]['protections']['results']['type'] == 'Interruttore elettronico' and t == 'el') \
                            or (elem_cat[c][e]['protections']['results']['type'] == 'Interruttore elettromeccanico' and
                                t == 'em'):
                        self.set_font('Arial', 'B', i_c)
                    else:
                        self.set_font('Arial', i_s, i_c)
                    self.cell(d_w, p_h, '%.1f' % elem_cat[c][e]['protections']['results']['comparison'][t]['overcurrent'],
                              0, 0, 'C')
                    self.cell(d_w, p_h, '%.3f' % elem_cat[c][e]['protections']['results']['comparison'][t]['overvoltage'],
                              0, 0, 'C')
                    self.cell(d_w, p_h, '%.2f' % elem_cat[c][e]['protections']['results']['comparison'][t]['cost'], 0, 0, 'C')
                    self.cell(d_w/2, p_h, '', 0, 0, 'C')
                self.ln(p_h)
                self.line(self.get_x() + 1, self.get_y(), self.get_x() + line_length - 2, self.get_y())

        self.add_page()
        for c in calc_prot:
            line_length = e_w + d_w * len(prot[1])
            if 275 < self.get_y() + t2_h + i_h + p_h * len(elem_cat[c].items()):
                self.add_page()
            self.set_font('Arial', t2_s, t2_c)
            self.write(0, cat[c])
            self.ln(t2_h)
            self.set_font('Arial', i_s, i_c)
            self.set_fill_color(226, 226, 226)
            self.cell(e_w, i_h, 'Elemento', 1, 0, 'C', fill=1)
            for e in prot[1]:
                self.cell(d_w, i_h, e, 1, 0, 'C', fill=1)
            self.ln(i_h)
            self.set_font('Arial', p_s, p_c)
            for e in elem_cat[c]:
                self.cell(e_w, p_h, e, 0, 0, 'L')
                self.cell(d_w, p_h, elem_cat[c][e]['protections']['results']['type'].replace('Interruttore ', ''), 0, 0, 'C')
```

```
                for item in prot[0][1:]:
                    self.cell(d_w, p_h, '%.3f' % elem_cat[c][e]['protections']['results'][item], 0, 0, 'C')
                self.ln(p_h)
                self.line(self.get_x() + 1, self.get_y(), self.get_x() + line_length - 2, self.get_y())
            self.ln(p_ls)

    #
    def reliability(self, elem_cat, elements):
        self.add_page()
        self.page_name = "Calcolo dell'Affidabilità"

        img_path = os.getcwd() + '/_images/SplashScreen/Co_80x80.png'
        self.set_xy(15, 35)
        self.image(img_path, self.get_x(), self.get_y(), 15, 15)
        self.set_xy(37.5, 44)

        self.set_draw_color(25, 25, 25)
        self.set_line_width(0.3)
        self.line(35, 38, 195, 38)
        self.line(35, 48, 195, 48)

        self.set_font('Arial', t1_s, t1_c)
        self.write(0, "Calcolo dell'Affidabilità")
        self.ln(t1_ls)

        for c in elem_cat:
            if c not in ['Ext-Grid', 'AC-Node', 'DC-Node']:
                line_length = e_w + d_w * len(rel[1])
                if 275 < self.get_y() + t2_h + i_h + p_h * len(elem_cat[c].items()):
                    self.add_page()
                self.set_font('Arial', t2_s, t2_c)
                self.write(0, cat[c])
                self.ln(t2_h)
                self.set_font('Arial', i_s, i_c)
                self.set_fill_color(226, 226, 226)
                self.cell(e_w, i_h, 'Elemento', 1, 0, 'C', fill=1)
                for e in rel[1]:
                    self.cell(d_w, i_h, e, 1, 0, 'C', fill=1)
                self.ln(i_h)
                self.set_font('Arial', p_s, p_c)
                for e in elem_cat[c]:
                    self.cell(e_w, p_h, e, 0, 0, 'L')
                    for item in rel[0]:
                        if item == 'MTBF_ore' and c in ['PWM', 'DC-DC_Conv', 'DC-Load', 'DC-Wind', 'PV', 'Battery']:
                            val = '%.4e' % (elem_cat[c][e]['reliability']['results'][item] * 1000000)
                        elif elem_cat[c][e]['reliability']['results'][item] < 0.001:
                            val = '%.4e' % elem_cat[c][e]['reliability']['results'][item]
                        elif elem_cat[c][e]['reliability']['results'][item] < 1:
                            val = '%.5f' % elem_cat[c][e]['reliability']['results'][item]
                        else:
                            val = '%.2f' % elem_cat[c][e]['reliability']['results'][item]
                        self.cell(d_w, p_h, val, 0, 0, 'C')
                    self.ln(p_h)
                    self.line(self.get_x() + 1, self.get_y(), self.get_x() + line_length - 2, self.get_y())
                self.ln(p_ls)

        self.add_page()
        img_path = v.project_folder + '/reliability/img'
        images = []
        for file in os.listdir(img_path):
            if file.endswith('.png'):
                images.append(file)
        x, y = 0, 32   # devono essere i margini della pagina

        count = 0
        for img in images:
            self.set_font('Arial', t2_s, t2_c)

            name = img.replace('.png', '')
            if count >= 2:
                self.add_page()
                count = 0
                y = 32
                self.ln(20)
            self.image(img_path + '/' + img, 5, y, 200, 133)

            self.set_y(y+15)
            self.cell(e_w, t2_h, name)
            r = elements[name]['reliability']['results']['load_rel']
            if r < 0.01:
                r_str = 'R(t) = %.4e' % r
            else:
                r_str = 'R(t) = %.5s' % r
            self.cell(e_w, t2_h, '')
            self.cell(e_w, t2_h, r_str)
            y = y + 130
            self.ln(133)
            count += 1

    #
    def controls(self):
        self.add_page()
        self.page_name = 'Controlli'

        img_path = os.getcwd() + '/_images/SplashScreen/Co_80x80.png'
        self.set_xy(15, 35)
        self.image(img_path, self.get_x(), self.get_y(), 15, 15)
        self.set_xy(37.5, 44)

        self.set_draw_color(25, 25, 25)
        self.set_line_width(0.3)
        self.line(35, 38, 195, 38)
        self.line(35, 48, 195, 48)

        self.set_font('Arial', t1_s, t1_c)
        self.write(0, "Controlli")
        self.ln(t1_ls)

        path = str(Path(os.getcwd())) + '/Functionalities/Controls'
        cont_res = yaml.safe_load(open(path + '/results.yml'))

        self.set_font('Arial', i_s, i_c*1.2)
        self.cell(e_w, t2_h, 'Area', 0, 0, 'R')
        self.set_font('Arial', t2_s, t2_c)
        areas = [['PORT', 'RES', 'RS', 'UG'], ['PORT AREA', 'CITY AREA - Settore Residenziale',
```

```
                                                'CITY AREA - Settore Servizi Stradali', 'CITY AREA - Settore Metropolitana']]
        self.cell(4*p_w, t2_h, areas[1][areas[0].index(cont_res['area'])], 0, 0, 'L')
        self.ln(p_ls)

        self.set_font('Arial', i_s, i_c*1.2)
        self.cell(e_w, t2_h, 'Scenario', 0, 0, 'R')
        self.set_font('Arial', t2_s, t2_c)
        self.cell(5*p_w, t2_h, cont_res['scen_year'], 0, 0, 'L')
        if cont_res['scen_year'] != 'Personalizzato':
            self.set_font('Arial', i_s, i_c*1.2)
            self.cell(p_w, t2_h, 'Configurazione', 0, 0, 'R')
            self.set_font('Arial', t2_s, t2_c)
            self.cell(2*p_w, t2_h, cont_res['scen_conf'], 0, 0, 'L')

        self.ln(p_ls)

        self.set_font('Arial', i_s, i_c*1.2)
        self.cell(e_w, t2_h, 'Mese', 0, 0, 'R')
        self.set_font('Arial', t2_s, t2_c)
        months = ['Gennaio', 'Febbraio', 'Marzo', 'Aprile', 'Maggio', 'Giugno', 'Luglio', 'Agosto', 'Settembre', 'Ottobre',
                  'Novembre', 'Dicembre']

        self.cell(5*p_w, t2_h, months[int(cont_res['month']-1)], 0, 0, 'L')

        self.set_font('Arial', i_s, i_c*1.2)
        self.cell(p_w, t2_h, 'Giorno', 0, 0, 'R')
        self.set_font('Arial', t2_s, t2_c)
        days = ['n.d.', 'Lunedì', 'Martedì', 'Mercoeldì', 'Giovedì', 'Venerdì', 'Sabato', 'Domenica']
        self.cell(p_w, t2_h, days[int(cont_res['day'])], 0, 0, 'L')
        self.ln(p_ls)

        self.set_font('Arial', i_s, i_c*1.2)
        self.cell(e_w, t2_h, 'Ora Inizio', 0, 0, 'R')
        self.set_font('Arial', t2_s, t2_c)
        inizio = '%02i:%02i' % (int(cont_res['start']), (cont_res['start']-int(cont_res['start']))*60)
        self.cell(5*p_w, t2_h, inizio, 0, 0, 'L')

        self.set_font('Arial', i_s, i_c*1.2)
        self.cell(p_w, t2_h, 'Ora Fine', 0, 0, 'R')
        self.set_font('Arial', t2_s, t2_c)
        fine = '%02i:%02i' % (int(cont_res['end']), (cont_res['end']-int(cont_res['end']))*60)
        self.cell(p_w, t2_h, fine, 0, 0, 'L')
        self.ln(p_ls)

        self.set_font('Arial', i_s, i_c*1.2)
        self.cell(e_w, t2_h, 'Linee di Backup', 0, 0, 'R')
        self.set_font('Arial', t2_s, t2_c)
        if cont_res['backup_line']:
            backup_line = '%.2f' % cont_res['backup_power']
        else:
            backup_line = 'assente'
        self.cell(5*p_w, t2_h, backup_line, 0, 0, 'L')
        self.ln(2*p_ls)

        count = 0
        for i in ctrl[0]:
            if count >= 3:
                self.ln(p_ls)
                count = 0
            self.set_font('Arial', i_s, i_c * 1.2)
            self.cell(e_w, t2_h, ctrl[1][ctrl[0].index(i)], 0, 0, 'R')
            self.set_font('Arial', t2_s, t2_c)
            self.cell(4 * p_w, t2_h, '%.2f' % cont_res[i], 0, 0, 'L')
            count += 1
        self.ln(p_ls)

        self.set_font('Arial', i_s, i_c*1.2)
        self.cell(e_w, t2_h, 'Commento', 0, 0, 'R')
        self.set_font('Arial', p_s, p_c*1.2)
        self.multi_cell(180-e_w, t2_h, cont_res['log'], 0, 'L')
        # self.cell(190-e_w, t2_h, cont_res['log'], 0, 0, 'L')
        self.ln(2*p_ls)

        y = self.get_y()
        self.image(path + '/' + cont_res['area'] + '.png', 30, y, 150, 120)

        self.add_page()
        y = 32
        for g in ctrl[0]:
            if y > 180:
                self.add_page()
                y = 32
            self.set_font('Arial', t2_s, t2_c)
            self.set_y(y)
            self.image(path + '/RES_' + g + '.png', 30, y, 150, 120)
            y = y + 120

    #
    def ems_indexes(self):
        self.add_page()
        self.page_name = 'EMS Energy Indexes'

        ei = [['ENS', 'ENS_pu', 'ENS_cost', 'ENS_cost_pu'], ['ENS [kW]', 'ENS [p.u.]', 'EIC [Euro]', 'EIC [p.u.]']]
        ei_dict = yaml.safe_load(open(v.project_folder + '/ems_fault_results.yml'))

        img_path = os.getcwd() + '/_images/SplashScreen/Ei_80x80.png'
        self.set_xy(15, 35)
        self.image(img_path, self.get_x(), self.get_y(), 15, 15)
        self.set_xy(37.5, 44)

        self.set_draw_color(25, 25, 25)
        self.set_line_width(0.3)
        self.line(35, 38, 195, 38)
        self.line(35, 48, 195, 48)

        self.set_font('Arial', t1_s, t1_c)
        self.write(20, "Energy Indexes")
        self.ln(2*t1_ls)

        line_length = e_w + 8.5 * d_w

        self.set_draw_color(226, 226, 226)
        self.set_line_width(0.1)

        self.set_font('Arial', t2_s, t2_c)
```

```
        self.write(0, 'Indici Affidabilistici: Guasti stocastici e controllo EMS')
        self.ln(t2_ls)

        self.set_font('Arial', t2_s, t2_c)
        self.cell(2*(e_w+d_w), t2_h, 'Sistema',  0, 0, 'L')
        self.ln(t2_h)
        self.set_font('Arial', i_s, i_c)
        self.cell(e_w + d_w, p_h, 'Caso base', 1, 0, 'L', fill=1)
        self.cell(e_w + d_w, p_h, 'Modello EMS', 1, 0, 'L', fill=1)
        self.ln(t2_h*1.5)

        for i in range(0, len(ei[0])):
            for t in ['Base case', 'EMS model']:
                self.set_font('Arial', 'I', t2_c*0.8)
                self.cell(e_w, p_h, ei[1][i], 0, 0, 'R')
                self.set_font('Arial', t2_s, t2_c)
                if 'p.u' in ei[1][i]:
                    self.cell(d_w, p_h, '%.5f' % ei_dict[t]['ALL_SYSTEM'][ei[0][i]], 0, 0, 'L')
                else:
                    self.cell(d_w, p_h, '%.3f' % ei_dict[t]['ALL_SYSTEM'][ei[0][i]], 0, 0, 'L')
            self.ln(t2_h*1.5)
        self.ln(t2_ls)

        self.set_font('Arial', t2_s, t2_c)
        self.write(0, 'Carichi')
        self.ln(t2_h)
        self.set_font('Arial', i_s, i_c)
        self.cell(e_w, i_h, '', 1, 0, 'C', fill=1)
        self.cell(d_w * 4, i_h, 'Caso Base', 1, 0, 'C', fill=1)
        self.cell(d_w/2, i_h, '', 1, 0, 'C', fill=1)
        self.cell(d_w * 4, i_h, 'Modello EMS', 1, 0, 'C', fill=1)
        self.ln(i_h)
        self.cell(e_w, i_h, 'elemento', 1, 0, 'C', fill=1)
        for i in range(0, 2):
            for e in ei[1]:
                self.cell(d_w, i_h, e, 1, 0, 'C', fill=1)

            if i == 0:
                self.cell(d_w/2, i_h, '', 1, 0, 'C', fill=1)
        self.ln(i_h)

        for e in ei_dict['Base case']:
            if e != 'ALL_SYSTEM':
                self.cell(e_w, p_h, e, 0, 0, 'L')
                for t in ['Base case', 'EMS model']:
                    for p in ei[0]:
                        if p in ['ENS_pu', 'ENS_cost_pu']:
                            self.cell(d_w, p_h, '%.5f' % ei_dict[t][e][p], 0, 0, 'C')
                        else:
                            self.cell(d_w, p_h, '%.3f' % ei_dict[t][e][p], 0, 0, 'C')
                    if t == 'Base case':
                        self.cell(d_w / 2, p_h, '', 0, 0, 'C')
                self.ln(p_h)
                self.line(self.get_x() + 1, self.get_y(), self.get_x() + line_length - 2, self.get_y())

#
def save(self):
    options = QtWidgets.QFileDialog.Options()

    saved = False
    while not saved:
        filename, _ = QtWidgets.QFileDialog.getSaveFileName(QtWidgets.QFileDialog(), "Save File", v.project_folder,
                                                            "PDF File (*.pdf)", options=options)

        if filename:
            if not filename.endswith('.pdf'):
                filename = filename + '.pdf'
            self.del_PDF(filename)
            try:
                self.output(filename, 'F')
                saved = True
            except PermissionError:
                pass
        else:
            saved = True
```

4.2.4 Protections

4.2.4.1 protection.py

```
import yaml
import os
from __shared__ import variables as v
import copy


class Protection:
    def __init__(self, prot_dict, choice):
        for element in prot_dict:
            In = prot_dict[element]['In']
            Vn = prot_dict[element]['Vn'] * 1000
            prot_dict[element]['results'] = dict()
            tot_cost = 0

            type = ''
            cost = 0
            soglia_I = 0
            delay_I = 0
            soglia_Vmax = 0
            soglia_Vmin = 0
            delay_Vmax = 0
            delay_Vmin = 0

            if prot_dict[element]['category'] == "Battery" and choice[2]:
                type = "Interruttore elettronico"
                cost = 35 + (4 * 10 ** -4) * (Vn ** 1.2) * (In ** 1.3)  # [€]
                soglia_I = 1.3 * In
                delay_I = 0.5  # [s]
                soglia_Vmax = 1.1 * Vn  # [V]
                soglia_Vmin = 0.8 * Vn  # [V]
                delay_Vmax = 0.4  # [s]
                delay_Vmin = 1  # [s]

            elif prot_dict[element]['category'] == "PV" and choice[2]:
                type = "Interruttore elettronico"
                cost = 35 + (4 * 10 ** -4) * (Vn ** 1.2) * (In ** 1.3)  # [€]
                soglia_I = 1.3 * In  # [A]
                delay_I = 0.5  # [s]
                soglia_Vmax = 1.1 * Vn  # [V]
                soglia_Vmin = 0.8 * Vn  # [V]
                delay_Vmax = 0.4  # [s]
                delay_Vmin = 1  # [s]

            elif prot_dict[element]['category'] == "DC-Wind" and choice[2]:
                type = "Interruttore elettronico"
                cost = 35 + (4 * 10 ** -4) * (Vn ** 1.2) * (In ** 1.3)  # [€]
                soglia_I = 1.3 * In  # [A]
                delay_I = 0.5  # [s]
                soglia_Vmax = 1.1 * Vn  # [V]
                soglia_Vmin = 0.8 * Vn  # [V]
                delay_Vmax = 0.4  # [s]
                delay_Vmin = 1  # [s]

            elif prot_dict[element]['category'] == "PWM" and choice[3]:
                type = "Interruttore elettromeccanico"
                cost = 15 + (4 * 10 ** -4) * (Vn ** 1.1) * (In ** 1.15)  # [€]
                soglia_I = 2 * In  # [A]
                delay_I = 1  # [s]
                soglia_Vmax = 1.1 * Vn  # [V]
                soglia_Vmin = 0.85 * Vn  # [V]
                delay_Vmax = 0.4  # [s]
                delay_Vmin = 0.5  # [s]

            elif prot_dict[element]['category'] == "DC-DC_Conv" and choice[4]:
                type = "Interruttore elettronico"
                cost = 35 + (4 * 10 ** -4) * (Vn ** 1.2) * (In ** 1.3)  # [€]
                soglia_I = 1.5 * In  # [A]
                delay_I = 1  # [s]
                soglia_Vmax = 1.1 * Vn  # [V]
                soglia_Vmin = 0.8 * Vn  # [V]
                delay_Vmax = 0.6  # [s]
                delay_Vmin = 1.2  # [s]

            elif prot_dict[element]['category'] == "DC-Line" and choice[0]:
                type = "Interruttore elettronico"
                cost = 35 + (4 * 10 ** -4) * (Vn ** 1.2) * (In ** 1.3)  # [€]
                soglia_I = 3 * In  # [A]
                delay_I = 0.5  # [s]
                soglia_Vmax = 1.1 * Vn  # [V]
                soglia_Vmin = 0.8 * Vn  # [V]
                delay_Vmax = 0.5  # [s]
                delay_Vmin = 1.1  # [s]

            elif prot_dict[element]['category'] == "DC-Load" and choice[1]:
                type = "Interruttore elettronico"
                cost = 35 + (4 * 10 ** -4) * (Vn ** 1.2) * (In ** 1.3)  # [€]
                soglia_I = 1.15 * In  # [A]
                delay_I = 0.5  # [s]
                soglia_Vmax = 1.05 * Vn  # [V]
                soglia_Vmin = 0.85 * Vn  # [V]
                delay_Vmax = 0.3  # [s]
                delay_Vmin = 0.1  # [s]

            tot_cost = tot_cost + cost

            if type != '':
                prot_dict[element]['results']['type'] = type
                prot_dict[element]['results']['cost'] = cost
                prot_dict[element]['results']['soglia_I'] = soglia_I
                prot_dict[element]['results']['soglia_Vmax'] = soglia_Vmax / 1000
                prot_dict[element]['results']['soglia_Vmin'] = soglia_Vmin / 1000
                prot_dict[element]['results']['delay_I'] = delay_I
                prot_dict[element]['results']['delay_Vmax'] = delay_Vmax
                prot_dict[element]['results']['delay_Vmin'] = delay_Vmin
                prot_dict[element]['results']['comparison'] = dict()
                prot_dict[element]['results']['comparison']['em'] = dict()
                prot_dict[element]['results']['comparison']['el'] = dict()
                prot_dict[element]['results']['comparison']['em']['overcurrent'] = In * 10
```

```
        prot_dict[element]['results']['comparison']['el']['overcurrent'] = In * 1
        prot_dict[element]['results']['comparison']['em']['overvoltage'] = Vn * 2.3 / 1000
        prot_dict[element]['results']['comparison']['el']['overvoltage'] = Vn * 1.37 / 1000
        prot_dict[element]['results']['comparison']['em']['cost'] = 15 + 0.0004*(Vn**1.1)*(In**1.15)
        prot_dict[element]['results']['comparison']['el']['cost'] = 35 + 0.0004*(Vn**1.2)*(In**1.3)
    v.elements[element]['protections'] = copy.deepcopy(prot_dict[element])

filename = os.path.join(v.project_folder, 'protections.yml')
with open(filename, 'w') as file:
    documents = yaml.dump(prot_dict, file)

filename = os.path.join(v.project_folder, 'elements.yml')
with open(filename, 'w') as file:
    documents = yaml.dump(v.elements, file)
```

4.3 Softwares

4.3.1 Neplan

4.3.1.1 neplan.py

```
import os
from datetime import datetime as dt

import yaml
import copy

from __shared__ import variables as v

from .webservices import Webservices


class Neplan:
    def __init__(self, project):
        self.elementNames, self.elementTypes = dict(), dict()

        self.ID = ""

        self.ws = Webservices()

        self.elementNames, self.elementTypes = self.ws.get_elements(project)

        self.populate()

        self.gen_res_cat = ['Load', 'DCLoad', 'ACDisperseGenerator', 'DCMotor', 'DCPhotoVoltaic', 'DCBattery',
                            'ExternalGrid']
        self.tr_res_cat = ['Trafo2Winding', 'PWM', 'Line', 'DCLine']
        self.node_res_cat = ['Busbar', 'DCNode']

    #
    def close_conn(self):
        del self.ws.neplanservices
        del self.ws

    #
    def load_yaml(self):
        filename = os.getcwd() + '/__shared__/attributes_template.yaml'
        dictionary = yaml.safe_load(open(filename))
        return dictionary

    #
    def populate(self):
        self.elements = dict()
        attributes = self.ws.get_attributes()
        self.conn = self.ws.get_connections()

        base_dict = yaml.safe_load(open(os.getcwd() + '/__shared__/attributes_template.yml'))
        for key in self.elementTypes:

            if self.elementTypes[key] == 'Busbar':
                v.elements[self.elementNames[key]] = copy.deepcopy(base_dict['AC-Node'])
                parameters, connections = self.ACnode_read(attributes[key], self.conn[key])
                for par in parameters:
                    v.elements[self.elementNames[key]]['parameters'][par] = parameters[par]
                v.elements[self.elementNames[key]]['conn'] = connections

            elif self.elementTypes[key] == 'DCNode':
                v.elements[self.elementNames[key]] = copy.deepcopy(base_dict['DC-Node'])
                parameters, connections = self.DCnode_read(attributes[key], self.conn[key])
                for par in parameters:
                    v.elements[self.elementNames[key]]['parameters'][par] = parameters[par]
                v.elements[self.elementNames[key]]['conn'] = connections

        for key in self.elementTypes:
            parameters = dict()
            connections = []
            if self.elementTypes[key] == 'Load':
                v.elements[self.elementNames[key]] = copy.deepcopy(base_dict['AC-Load'])
                parameters, connections = self.ACload_read(attributes[key], self.conn[key])
            elif self.elementTypes[key] == 'DCLoad':
                v.elements[self.elementNames[key]] = copy.deepcopy(base_dict['DC-Load'])
                parameters, connections = self.DCload_read(attributes[key], self.conn[key])
            elif self.elementTypes[key] == 'Trafo2Winding':
                v.elements[self.elementNames[key]] = copy.deepcopy(base_dict['2W-Transformer'])
                parameters, connections = self.transformer_read(attributes[key], self.conn[key])
            elif self.elementTypes[key] == 'PWM':
                v.elements[self.elementNames[key]] = copy.deepcopy(base_dict['PWM'])
                parameters, connections = self.PWM_read(attributes[key], self.conn[key])
            elif self.elementTypes[key] == 'ACDisperseGenerator':
                v.elements[self.elementNames[key]] = copy.deepcopy(base_dict['AC-Wind'])
                parameters, connections = self.ACwind_read(attributes[key], self.conn[key])
            elif self.elementTypes[key] == 'DCMotor':
                v.elements[self.elementNames[key]] = copy.deepcopy(base_dict['DC-Wind'])
                parameters, connections = self.DCwind_read(attributes[key], self.conn[key])
            elif self.elementTypes[key] == 'DCPhotoVoltaic':
                v.elements[self.elementNames[key]] = copy.deepcopy(base_dict['PV'])
                parameters, connections = self.PV_read(attributes[key], self.conn[key])
            elif self.elementTypes[key] == 'DCBattery':
                v.elements[self.elementNames[key]] = copy.deepcopy(base_dict['Battery'])
                parameters, connections = self.battery_read(attributes[key], self.conn[key])
            elif self.elementTypes[key] == 'Line':
                v.elements[self.elementNames[key]] = copy.deepcopy(base_dict['AC-Line'])
                parameters, connections = self.ACline_read(attributes[key], self.conn[key])
            elif self.elementTypes[key] == 'DCLine':
                v.elements[self.elementNames[key]] = copy.deepcopy(base_dict['DC-Line'])
                parameters, connections = self.DCline_read(attributes[key], self.conn[key])
            elif self.elementTypes[key] == 'Busbar':
                pass
            elif self.elementTypes[key] == 'DCNode':
                pass
            elif self.elementTypes[key] == 'ExternalGrid':
                v.elements[self.elementNames[key]] = copy.deepcopy(base_dict['Ext-Grid'])
                parameters, connections = self.extgrid_read(attributes[key], self.conn[key])
                v.ext_grid = self.elementNames[key]
            else:
                pass
```

```
        for par in parameters:
            v.elements[self.elementNames[key]]['parameters'][par] = parameters[par]
        if connections:
            v.elements[self.elementNames[key]]['conn'] = connections

#
def const_curve(self, value):
    profile = dict()
    profile['constant'] = True
    profile['name'] = 'From Origin'
    profile['curve'] = value
    return profile

#
def ACload_read(self, attributes, connections):
    parameters = dict()
    parameters['P'] = attributes['P']
    parameters['Q'] = attributes['Q']
    parameters['S'] = attributes['S']
    parameters['I'] = attributes['I']
    parameters['cosPhi'] = attributes['CosPhi']
    parameters['profile'] = self.const_curve(attributes['ScalingFactorElementP'])
    parameters['control'] = attributes['LfType']
    return parameters, connections

#
def DCload_read(self, attributes, connections):
    parameters = dict()
    parameters['P'] = attributes['Pset'] * 1000
    parameters['I'] = attributes['Iset'] * 1000
    parameters['R'] = attributes['Rset']
    parameters['profile'] = self.const_curve(1)
    parameters['control'] = attributes['Lftype']
    return parameters, connections

#
def transformer_read(self, attributes, connections):
    parameters = dict()
    conn = list(connections.keys())
    conn.remove('h')
    [conn1, conn2, connections['h']] = conn[0], conn[1], conn[0]
    connections['invert'] = False
    if v.elements[conn1]['parameters']['Ur'] < v.elements[conn2]['parameters']['Ur']:
        connections['h'] = conn2
        connections['invert'] = True

    if attributes['Ur1'] > attributes['Ur2']:
        (uH, uL) = (attributes['Ur1'], attributes['Ur2'])
    else:
        (uH, uL) = (attributes['Ur2'], attributes['Ur1'])
    parameters['Sr'] = attributes['Sr'] * 1000
    parameters['UnHV'] = uH
    parameters['UnLV'] = uL
    parameters['URr1'] = attributes['Urr']
    parameters['Ukr1'] = attributes['Ukr']
    parameters['URr0'] = attributes['Urr0']
    parameters['Ukr0'] = attributes['Ukr0']
    return parameters, connections

#
def PWM_read(self, attributes, connections):
    parameters = dict()
    conn = list(connections.keys())
    [conn1, conn2, connections['h']] = conn[0], conn[1], conn[0]
    if v.elements[conn2]['category'] == 'AC-Node':
        connections['h'] = conn2

    parameters['Sr'] = attributes['SN'] * 1000
    parameters['Ur'] = attributes['UN']
    parameters['S_loss_idle'] = attributes['IdleLosses'] * 1000
    parameters['Sw_loss'] = attributes['SwLosses']
    parameters['R_loss'] = attributes['R']
    return parameters, connections

#
def DcDcConv_read(self, attributes, connections):
    parameters = dict()
    return parameters, connections

#
def ACwind_read(self, attributes, connections):
    parameters = dict()
    parameters['Sr'] = attributes['Sr'] * 1000
    parameters['control'] = [0, 0, 0, 1, 0, 2, 0, 0, 0, 0, 0][attributes['TypeLF']]
    parameters['P'] = attributes['Pset']
    parameters['Q'] = attributes['Qset']
    parameters['cosPhi'] = attributes['Cosset']
    parameters['S'] = attributes['Sset']
    parameters['profile'] = self.const_curve(attributes['ScalingFactorElementP'])
    parameters['units'] = attributes['Anzahl']
    return parameters, connections

#
def DCwind_read(self, attributes, connections):
    parameters = dict()

    conn = list(connections.keys())
    parameters['Ur'] = v.elements[conn[0]]['parameters']['Ur']
    parameters['Pr'] = attributes['Ir'] * attributes['Ur'] / 1000
    parameters['P'] = - attributes['Pset'] * 1000
    parameters['profile'] = self.const_curve(1)
    return parameters, connections

#
def PV_read(self, attributes, connections):
    parameters = dict()
    return parameters, connections

#
def battery_read(self, attributes, connections):
    parameters = dict()
    return parameters, connections

#
```

```
def ACline_read(self, attributes, connections):
    parameters = dict()
    parameters['length'] = attributes['Length'] * 1000
    parameters['lines'] = attributes['NumParallel']
    parameters['R1'] = attributes['R1']
    parameters['X1'] = attributes['X1']
    parameters['B1'] = attributes['B1']   # Non legge il valore che gli mando (legge 50...)
    parameters['R0'] = attributes['R0']
    parameters['X0'] = attributes['X0']
    parameters['B0'] = 0    # attributes['B0']   da verificare con NEPLAN
    parameters['Irmax'] = attributes['IrLimit4']
    return parameters, connections

#
def DCline_read(self, attributes, connections):
    parameters = dict()
    parameters['length'] = attributes['Length']   # veridicare con NEPLAN
    parameters['lines'] = attributes['ParallelLines']
    parameters['R'] = attributes['R_pos']
    parameters['L'] = attributes['L']
    parameters['C'] = attributes['C']
    parameters['Irmax'] = attributes['Ir_def']
    return parameters, connections

#
def ACnode_read(self, attributes, connections):
    parameters = dict()
    parameters['Ur'] = attributes['Un']
    return parameters, connections

#
def DCnode_read(self, attributes, connections):
    parameters = dict()
    parameters['Ur'] = attributes['Un']
    return parameters, connections

#
def extgrid_read(self, attributes, connections):
    parameters = dict()
    parameters['control'] = attributes['TypeLF']
    parameters['U'] = attributes['Uoper']
    parameters['Uangle'] = attributes['AngleOper']
    parameters['P'] = attributes['Poper']
    parameters['Q'] = attributes['Qoper']
    parameters['slack'] = attributes['PortionSL']
    return parameters, connections

#
def set_params(self, element):
    attributes = dict()
    IDs = list(self.elementNames.keys())
    self.ID = IDs[list(self.elementNames.values()).index(element)]
    parameters = v.elements[element]['parameters']

    if self.elementTypes[self.ID] == 'Load':
        attributes = self.ACload_set(parameters)
    elif self.elementTypes[self.ID] == 'DCLoad':
        attributes = self.DCload_set(parameters)
    elif self.elementTypes[self.ID] == 'Trafo2Winding':
        attributes = self.transformer_set(parameters)
    elif self.elementTypes[self.ID] == 'PWM':
        attributes = self.PWM_set(parameters)
    elif self.elementTypes[self.ID] == 'ACDisperseGenerator':
        attributes = self.ACwind_set(parameters)
    elif self.elementTypes[self.ID] == 'DCMotor':
        attributes = self.DCwind_set(parameters)
    elif self.elementTypes[self.ID] == 'DCPhotoVoltaic':
        attributes = self.PV_set(parameters)
    elif self.elementTypes[self.ID] == 'DCBattery':
        attributes = self.battery_set(parameters)
    elif self.elementTypes[self.ID] == 'Line':
        attributes = self.ACline_set(parameters)
    elif self.elementTypes[self.ID] == 'DCLine':
        attributes = self.DCline_set(parameters)
    elif self.elementTypes[self.ID] == 'Busbar':
        attributes = self.ACnode_set(parameters)
    elif self.elementTypes[self.ID] == 'DCNode':
        attributes = self.DCnode_set(parameters)
    else:
        print(self.elementNames[self.ID] + ': ' + self.elementTypes[self.ID] + ' non classificabile')

    self.ws.set_attributes_by_ID(self.ID, attributes)

    for port in v.neplan_connections[self.ID]:
        for conn in list(v.elements[element]['conn'].keys()):
            if conn == v.neplan_connections[self.ID][port]['element']:
                v.neplan_connections[self.ID][port]['connected'] = v.elements[element]['conn'][conn]
    self.ws.set_connections(self.ID)

#
def from_profile(self, profile,):
    hour = int((dt.now().hour + dt.now().minute/60)*4)
    if profile['constant']:
        value = profile['curve']
    else:
        value = profile['curve'][int((dt.now().hour + dt.now().minute/60)*4)]
    return value

#
def ACload_set(self, parameters):
    attributes = dict()
    attributes['P'] = parameters['P']
    attributes['Q'] = parameters['Q']
    attributes['S'] = parameters['S']
    attributes['I'] = parameters['I']
    attributes['CosPhi'] = parameters['cosPhi']
    attributes['ScalingFactorElementP'] = self.from_profile(parameters['profile'])
    attributes['ScalingFactorElementQ'] = attributes['ScalingFactorElementP']
    attributes['LfType'] = parameters['control']
    return attributes

#
def DCload_set(self, parameters):
    attributes = dict()
    sf = self.from_profile(parameters['profile'])
```

```
        attributes['Pset'] = parameters['P'] * sf / 1000
        attributes['Iset'] = parameters['I'] * sf / 1000
        attributes['Rset'] = parameters['R'] * sf
        attributes['Lftype'] = parameters['control']
        return attributes

    #
    def transformer_set(self, parameters):  # BIsogna considerare la necessità di inverti rele tensioni
        attributes = dict()
        attributes['Ur1'] = parameters['UnHV']
        attributes['Ur2'] = parameters['UnLV']
        attributes['Urr'] = parameters['URr1']
        attributes['Ukr'] = parameters['Ukr1']
        attributes['UrrO'] = parameters['URrO']
        attributes['UkrO'] = parameters['UkrO']
        attributes['Sr'] = parameters['Sr'] / 1000
        return attributes

    #
    def PWM_set(self, parameters):
        attributes = dict()
        attributes['SN'] = parameters['Sr'] / 1000
        attributes['UN'] = parameters['Ur']
        attributes['IdleLosses'] = parameters['S_loss_idle'] / 1000
        attributes['SwLosses'] = parameters['Sw_loss']
        attributes['R'] = parameters['R_loss']
        return attributes

    #
    def DcDcConv_set(self, parameters):
        attributes = dict()
        return attributes

    #
    def ACwind_set(self, parameters):
        attributes = dict()
        attributes['Sr'] = parameters['Sr'] / 1000
        attributes['Pset'] = parameters['P']
        attributes['Qset'] = parameters['Q']
        attributes['Cosset'] = parameters['cosPhi']
        attributes['TypeLF'] = [2, 3, 5][parameters['control']]
        attributes['ScalingFactorElementP'] = self.from_profile(parameters['profile'])
        attributes['ScalingFactorElementQ'] = attributes['ScalingFactorElementP']
        attributes['Anzahl'] = parameters['units']
        return attributes

    #
    def DCwind_set(self, parameters):
        attributes = dict()
        attributes['Ur'] = parameters['Ur'] * 1000
        attributes['Pr'] = parameters['Pr']
        attributes['Ir'] = parameters['Pr'] / parameters['Ur']
        attributes['Lftype'] = 0
        attributes['Pset'] = - parameters['P'] * self.from_profile(parameters['profile']) / 1000
        return attributes

    #
    def PV_set(self, parameters):
        attributes = dict()
        return attributes

    #
    def battery_set(self, parameters):
        attributes = dict()
        return attributes

    #
    def ACline_set(self, parameters):
        attributes = dict()
        attributes['Length'] = parameters['length'] / 1000
        attributes['NumParallel'] = parameters['lines']
        attributes['R1'] = parameters['R1']
        attributes['X1'] = parameters['X1']
        # attributes['B1'] = parameters['B1']     # tbv
        attributes['R0'] = parameters['R0']
        attributes['X0'] = parameters['X0']
        # attributes['B0'] = parameters['B0']     # tbv
        attributes['IrLimit4'] = parameters['Irmax']
        return attributes

    #
    def DCline_set(self, parameters):
        attributes = dict()
        attributes['Length'] = parameters['length']
        attributes['ParallelLines'] = parameters['lines']
        attributes['R_pos'] = parameters['R']
        attributes['L'] = parameters['L']
        attributes['C'] = parameters['C']
        attributes['Ir_def'] = parameters['Irmax']
        return attributes

    #
    def ACnode_set(self, parameters):
        attributes = dict()
        attributes['Un'] = parameters['Ur']
        return attributes

    #
    def DCnode_set(self, parameters):
        attributes = dict()
        attributes['Un'] = parameters['Ur']
        return attributes

    #
    def results(self, hour):
        results_dict = self.ws.get_results_LF()
        for key in results_dict:
            if not self.elementNames[key] in v.elements.keys():
                v.elements[self.elementNames[key]] = dict()
            if self.elementTypes[key] in self.gen_res_cat:
                v.elements[self.elementNames[key]]['results'] = self.general_results(results_dict[key], 'Port 1')
            elif self.elementTypes[key] in self.tr_res_cat:
                v.elements[self.elementNames[key]]['results'] = self.tr_results(results_dict[key])
            if self.elementTypes[key] in self.node_res_cat:
                v.elements[self.elementNames[key]]['results'] = self.nodes_results(results_dict[key])
```

```
    filename = os.path.join(os.getcwd(), '_data', 'elements.yml')
    with open(filename, 'w') as file:
        documents = yaml.dump(v.elements, file)
    file.close()

    return results_dict

#
def res_dict_build(self, ws_dict, key):
    if self.elementTypes[key] in self.gen_res_cat:
        v.elements[self.elementNames[key]]['results'] = self.general_results(ws_dict, 'Port 1')
    elif self.elementTypes[key] in self.tr_res_cat:
        pass

#
def general_results(self, ws_dict, port):
    elem_res = dict()
    elem_res['I'] = float(ws_dict[port]['I']) * 1000
    elem_res['Iangle'] = float(ws_dict[port]['IAngle'])
    elem_res['P'] = float(ws_dict[port]['P']) * 1000
    elem_res['Q'] = float(ws_dict[port]['Q']) * 1000
    elem_res['S'] = float(ws_dict[port]['S']) * 1000
    elem_res['cosPhi'] = float(ws_dict[port]['PowerFactor'])
    elem_res['U'] = float(ws_dict[port]['Un'])
    if ws_dict[port]['LimitViolated'] == 'true':
        elem_res['LimitViolated'] = True
    else:
        elem_res['LimitViolated'] = False
    return elem_res

#
def tr_results(self, ws_dict):
    elem_res = dict()
    elem_res['Port1'] = self.general_results(ws_dict, 'Port 1')
    elem_res['Port2'] = self.general_results(ws_dict, 'Port 2')
    elem_res['Ploss'] = abs(abs(elem_res['Port1']['P']) - abs(elem_res['Port2']['P']))
    elem_res['Qloss'] = abs(abs(elem_res['Port1']['Q']) - abs(elem_res['Port2']['Q']))
    if elem_res['Port1']['LimitViolated'] or elem_res['Port2']['LimitViolated']:
        elem_res['LimitViolated'] = True
    else:
        elem_res['LimitViolated'] = False
    return elem_res

#
def nodes_results(self, ws_dict):
    elem_res = dict()
    elem_res['Pgen'] = float(ws_dict['Port 0']['PGen']) * 1000
    elem_res['Pload'] = float(ws_dict['Port 0']['PLoad']) * 1000
    elem_res['Qgen'] = float(ws_dict['Port 0']['QGen']) * 1000
    elem_res['Qload'] = float(ws_dict['Port 0']['QLoad']) * 1000
    elem_res['U'] = float(ws_dict['Port 0']['U'])
    elem_res['Up'] = float(ws_dict['Port 0']['Up'])
    elem_res['Un'] = float(ws_dict['Port 0']['Un'])
    if abs(elem_res['Un']-elem_res['U'])/elem_res['Un'] > 0.05:
        elem_res['LimitViolated'] = True
    else:
        elem_res['LimitViolated'] = False
    return elem_res
```

4.3.1.2 webservices.py

```
import hashlib

import uuid

import urllib3

from requests import Session

from zeep import Client
from zeep .wsse import UsernameToken
from zeep.transports import Transport

from __shared__ import variables as v

urllib3.disable_warnings()


class Webservices:
    def __init__(self):
        user_name = "***********"                  #omissis
        password = "***********"                   #omissis

        self.project = vars()
        self.elements = vars()
        self.attributes = vars()

        md5 = hashlib.sha1()
        md5.update(password.encode("utf-8"))
        md5_password = md5.hexdigest()

        self.session = Session()
        self.session.verify = False

        url = "***********"   #omissis
        self.client = Client(url + '?singleWsdl',
                             transport=Transport(session=self.session),
                             wsse=UsernameToken(user_name, password=md5_password))
        self.neplanservices = self.client.create_service(
            '{http://www.neplan.ch/Web/External}BasicHttpBinding_NeplanService', url + '/basic')

        self.client.set_ns_prefix('ns15',
                                  "http://schemas.datacontract.org/2004/07/BCP.Neplan.Web.Services.ServiceManager")

    #
    def __del__(self):
        self.session.__exit__()

    #
    def import_project(self):
        stream_type = self.client.get_type('ns5:StreamBody')

        filename = 'C:/Users/Antonio Ricca/Desktop/testREL2.nep360'
        mystream = stream_type(filename.encode())

        ext_file = self.neplanservices.Nep360Upload(mystream)
        risp = self.neplanservices.ImportFromFile(ext_file, 'Progettoimportato', True, '', False, False)

        ext_file = mystream.BaseStream

    #
    def get_projects(self):  # reading project list
        projects = self.neplanservices.GetProjects()
        return projects

    #
    def open_project(self, proj_name):  # opening "proj_name" project
        self.project = self.neplanservices.GetProject(proj_name, None, None, None)
        return self.project

    #
    def get_elements(self, proj_name):   # Creation of elementNames and elementTypes dictionaries
        proj = self.open_project(proj_name)

        arrayOfKeyValue_type = self.client.get_type('ns3:ArrayOfKeyValueOfstringstring')
        myElementNames = arrayOfKeyValue_type()
        myElementTypes = arrayOfKeyValue_type()
        self.elements = self.neplanservices.GetAllElementsOfProject(proj, myElementNames, myElementTypes)

        elementNames, elementTypes = dict(), dict()
        i = 0
        for element in self.elements.elementNames.KeyValueOfstringstring:
            elementNames[element.Key] = element.Value
            elementTypes[self.elements.elementTypes.KeyValueOfstringstring[i].Key] = \
                self.elements.elementTypes.KeyValueOfstringstring[i].Value
            i += 1
        self.elementNames = elementNames

        for elem in elementTypes:
            if elementTypes[elem] == 'ExternalGrid':
                v.neplan_extgrid = elem
                break

        return elementNames, elementTypes

    # -- Reading attributes of all elements in the project ------------------------------------------
    def get_attributes(self):
        # -- Definition of variable types -------------------------------
        doubleAttribute_type = self.client.get_type('ns15:DoubleAttribute')
        arrayOfAttributeItems_type = self.client.get_type('ns15:ArrayOfAttributeItem')
        techItem_type = self.client.get_type('ns15:TechItem')
        arrayOfTechItems_type = self.client.get_type('ns15:ArrayOfTechItem')
        # ---------------------------------------------------------------

        # -- Collecting all types of elements in the project -----------
        types = []
        for n in self.elements.elementTypes.KeyValueOfstringstring:
            if types.count(n.Value) == 0:
                types.append(n.Value)
        # ---------------------------------------------------------------

        # -- variable "techItems" definition ----------------------------------------
        myTechItems = []
```

```
        for myElement in self.elements.elementTypes.KeyValueOfstringstring:
            # print(myElement.Value)
            myAttributes = self.param_list(myElement.Value)

            # -- Attributes list creation for type ----------------------
            attributesOfType = []
            for a in myAttributes:
                attr = doubleAttribute_type()
                attr.AttributeName = a
                attributesOfType.append(attr)
            attributeItems = arrayOfAttributeItems_type(attributesOfType)
            # -----------------------------------------------------------

            techItem = techItem_type()
            techItem.ElementID = uuid.UUID(str(myElement.Key))
            techItem.Attributes = attributeItems
            myTechItems.append(techItem)

        techItems = arrayOfTechItems_type(myTechItems)
        # ------------------------------------------------------------------------

        self.attributes = self.neplanservices.GetElementAttributes(self.project, techItems)

        attrDict = dict()
        elemDict = dict()
        for n in self.attributes:
            elemDict.clear()
            for nn in n.Attributes.AttributeItem:
                elemDict[nn.AttributeName] = nn.Value
            attrDict[n.ElementID] = elemDict.copy()

        return attrDict

    # Set attributes on a element in the project
    def set_attributes_by_ID(self, ID, attributes):
        for key in attributes:
            self.neplanservices.SetElementAttributeByID(self.project, ID, key, str(attributes[key]))

    # Definition of attributes for each type of element
    def param_list(self, type):
        parLoad = ['IsVariableLoad', 'LfType', 'Unit', 'S', 'P', 'Q', 'I', 'CosPhi', 'E', 'P0', 'Q0',
                   'ScalingFactorElementP', 'ScalingFactorElementQ', 'RelType', 'RelNumCustomersInt', 'InstalledkVA',
                   'RelPriority']

        parTr = ['Sr', 'Ur1', 'Ur2', 'Urr', 'UrrkW', 'Ukr', 'Urr0', 'Urr0kW', 'Ukr0', 'X1R1', 'X0R0', 'Uk01O', 'Uk02O',
                 'I0', 'Pfe', 'IsAutoTrafo', 'IsSwitchable', 'IsRegulated']

        parBusBar = ['NodeType', 'Un', 'Fn', 'Uset', 'Umin', 'Umax', 'Ipmax', 'Ithmax', 'tripTimeDP', 'Ir', 'R_Ground']

        parAsMach = ['Ur', 'Ir', 'Sr', 'Pr', 'Nr', 'SlipRated', 'CosPhi', 'J', 'Eta', 'RsToRr', 'ServiceFactor',
                     'IsToIr', 'T_ac', 'MsTOMr', 'Rmotor', 'MkToMr', 'XtoR', 'LfType', 'Poper', 'Qoper', 'CosPhiOper',
                     'IsCosOperCap', 'ScalingFactorElementP', 'NumParallel']

        parLine = ['Length', 'NumParallel', 'R1', 'X1', 'C1', 'B1', 'G1', 'R0', 'X0', 'C0', 'B0', 'IrLimit1',
                   'IrLimit2', 'IrLimit3', 'IrLimit4']

        parExtGrid = ['TypeLF', 'Uoper', 'AngleOper', 'Poper', 'Qoper', 'PortionSL']

        parPWM = ['SN', 'UN', 'MaxModulation', 'MaxUdc', 'IdleLosses', 'SwLosses', 'R', 'Cntr1', 'SetValue1', 'Cntr2',
                  'SetValue2']

        parDCvolt = ['Uset', 'Ikinput', 'IkSCMin', 'Rb', 'Lb']

        parDCnode = ['Un']

        parDCmotor = ['Ur', 'Ir', 'Pr', 'Mr', 'Nr', 'N0', 'J', 'Rf', 'Lf', 'Rm', 'Lm', 'L0f', 'DecreaseSpeed', 'Pset',
                      'Iset', 'Rset', 'Lftype']

        parDCpv = ['USet']

        parDispGen = ['nProductionType', 'Anzahl', 'Ur', 'Sr', 'Cosr', 'IECPS_Type: 0', 'IECPS_Type: 1',
                      'IECPS_Type: 2', 'Sk2max', 'Sk2min', 'Ik2max', 'Ik2min', 'Ik1max', 'Ik1min', 'R1toX1_max',
                      'R1toX1_min', 'Z0zuZ1_max', 'R0zuX1_max', 'Z2zuZ1_max', 'R2zuX2_max', 'bIEC', 'EOper', 'ScTimeDec',
                      'Srspsg', 'Isolated', 'SCReversible', 'TypeLF', 'Uset', 'Uwset', 'Sset', 'Pset', 'Qset', 'Iset',
                      'Cosset', 'capacitive', 'Sl_anteil', 'IsVariableLoad', 'ScalingFactorElementP',
                      'ScalingFactorElementQ']

        parDCload = ['Lftype', 'Pset', 'Iset', 'Rset']

        parDCbatt = ['Uset', 'Rb', 'Lb']

        parDCline = ['R_pos', 'L', 'C', 'ParallelLines', 'MaxTempAtEnd', 'OperTemp', 'MaxOperTemp', 'RatedTemp', 'Ur',
                     'Ir_min', 'Ir_mid', 'Ir_max', 'Ir_def', 'Q', 'Material', 'Cores', 'Length', 'length']

        attributes = []
        if type == 'Load':
            attributes = parLoad
        elif type == 'Trafo2Winding':
            attributes = parTr
        elif type == 'Busbar':
            attributes = parBusBar
        elif type == 'AsynchronousMachine':
            attributes = parAsMach
        elif type == 'Line':
            attributes = parLine
        elif type == 'ExternalGrid':
            attributes = parExtGrid
        elif type == 'PWM':
            attributes = parPWM
        elif type == 'DCVoltageSource':
            attributes = parDCvolt
        elif type == 'DCNode':
            attributes = parDCnode
        elif type == 'DCMotor':
            attributes = parDCmotor
        elif type == 'DCPhotoVoltaic':
            attributes = parDCpv
        elif type == 'ACDisperseGenerator':
            attributes = parDispGen
        elif type == 'DCLoad':
            attributes = parDCload
        elif type == 'DCBattery':
            attributes = parDCbatt
        elif type == 'DCLine':
            attributes = parDCline
```

```
        return attributes

    # Reading connections for all the elements
    def get_connections(self):
        status = dict()
        i = 0
        for key in self.attributes:
            ID = key.ElementID
            myID = uuid.UUID(str(ID))
            v.neplan_connections[ID] = dict()
            j = 0
            status[ID] = dict()
            for port in self.neplanservices.GetConnectedElementsByElementID(self.project, myID):
                v.neplan_connections[ID][j] = dict()
                connID = self.neplanservices.GetElementAtPortByID(self.project, myID, j)
                if connID:
                    v.neplan_connections[ID][j]['element'] = self.elementNames[connID]
                else:
                    v.neplan_connections[ID][j]['element'] = port
                v.neplan_connections[ID][j]['connected'] = \
                    self.neplanservices.GetSwitchOfElementAtPortByID(self.project, myID, j)
                status[ID][v.neplan_connections[ID][j]['element']] = v.neplan_connections[ID][j]['connected']
                j += 1
            status[ID]['h'] = v.neplan_connections[ID][0]['element']

            i += 1
        return status

    #
    def set_connections(self, ID):
        myID = uuid.UUID(str(ID))
        for port in list(v.neplan_connections[ID].keys()):
            self.neplanservices.SwitchElementAtPortByID(myID, port, v.neplan_connections[ID][port]['connected'])

    #
    def get_results_LF(self):
        self.loadflow()

        mydict = dict()
        a = self.neplanservices.GetAllElementResults(self.project, 'LoadFlow')
        for myVars in self.neplanservices.GetAllElementResults(self.project, 'LoadFlow'):
            port = 'Port ' + str(myVars.portNr + 1)

            if myVars.ElementID not in mydict.keys():
                mydict[myVars.ElementID] = dict()

            if port not in mydict[myVars.ElementID].keys():
                mydict[myVars.ElementID][port] = dict()

            myTag = str(myVars.XMLdata).split('<')[1].split(' ')[0]
            parameters = self.par_LF(myTag)
            for par in parameters:
                start = '<' + par + '>'
                end = '</' + par + '>'
                myvalue = str(myVars.XMLdata).split(start)[1].split(end)[0]
                mydict[myVars.ElementID][port][par] = myvalue
        return mydict

    #
    def loadflow(self):
        analysisID = uuid.uuid4()
        analysisType = 'LoadFlow'
        analysisInfoLocal = self.neplanservices.AnalyseVariant(self.project, str(analysisID), analysisType, None,
                                                                None, None, None)

    #
    def par_LF(self, cat):
        lfPort = ['I', 'IAngle', 'LimitViolated', 'Loading', 'P', 'PowerFactor', 'Q', 'S', 'Un']

        lfPWM = ['Creg', 'MD', 'MQ', 'Pac', 'Teta', 'VmagDC']
        lfLine = ['PLosses', 'QLosses', 'UOpenEndAng', 'UOpenEndMag', 'UOpenEndMagpc']
        lfElement = ['PLosses', 'QLosses']
        lfNode = ['DelUp', 'Distance', 'PGen', 'PLoad', 'PLossSensP', 'PLossSensQ', 'QGen', 'QLoad', 'QShunt', 'U',
                  'UAngle', 'UReg', 'Un', 'UnRef', 'Up']
        lfTrafo2 = ['PLosses', 'QLosses', 'HasTwoTaps', 'IronLosses', 'LoadtoTransformerScaling', 'Priority', 'Ratio',
                    'Ratio2', 'RatioAngle', 'RatioAngle2', 'RatioValid', 'Ratio_SysBase', 'Ratio_SysBase2', 'Tap',
                    'Tap2']

        par = []
        if cat == 'ResultLFPort':
            par = lfPort
        elif cat == 'ResultLFLine':
            par = lfLine
        elif cat == 'ResultLFElement':
            par = lfElement
        elif cat == 'LoadFlowNodeResultSym':
            par = lfNode
        elif cat == 'ResultLFPWM':
            par = lfPWM
        elif cat == 'ResultLFTrafo2':
            par = lfTrafo2

        return par
```

4.3.2 PowerFactory

4.3.2.1 pf_class.py

```
#by Roberto Ciavarella
import os
import sys

from . import pf
from __shared__ import variables as v

path = v.config['pf']['path'].replace('/PowerFactory.exe', '')
sys.path.append(path + '/Python/3.7')

import powerfactory as pfactory


class PowerFactory:
    def __init__(self, pf_dict):
        folder = v.features['pf']['folder_name']
        name = v.features['pf']['project_name']
        study_case = v.features['pf']['study_case_name']
        app = pfactory.GetApplication()

        project = app.ActivateProject(os.path.join(folder, name))
        # activate study case
        study_case_folder = app.GetProjectFolder('study')
        study_case = study_case_folder.GetContents(study_case + '.IntCase')[0]

        ###istanzio la classe PowerFactory_interface. Il dizionario viene riempito
        # con le informazioni lette dal modello di rete presente in powerfactory
        self.P = pf.PowerFactory_interface()

    #
    def set_params(self, element):
        self.P.set_params(element)

    #
    def results(self, hour):
        self.P.results(hour)

    #
    def prof_results(self):
        self.P.profileLF(htot=24)

    #
    def reliability(self, t, Ta):
        for element in v.elements:
            if v.elements[element]['category'] != 'Ext-Grid':
                self.P.Norris_Landzberg(element, t, v.elements[element]['reliability']['T0'], Ta,
                                        v.elements[element]['reliability']['alfa'],
                                        v.elements[element]['reliability']['beta'],
                                        v.elements[element]['reliability']['Pi_E'],
                                        v.elements[element]['reliability']['Pi_Q'])

        rel_path = os.path.join(v.project_folder, 'reliability')
        try:
            os.remove(rel_path)
        except: pass
        try:
            os.mkdir(rel_path)
            os.mkdir(os.path.join(rel_path, 'img'))
        except: pass

        self.P.grafo_rete()
        self.P.RBD(t)
```

4.3.2.2 pf.py

```
import copy
import os
import yaml
import sys
import math
from __shared__ import variables as v
from __shared__.plugins.fiabilipy import Component, System
from sympy import Symbol
import networkx as nx
import matplotlib.pyplot as plt


class PowerFactory_interface:
    def __init__(self):
        path = v.config['pf']['path'].replace('/PowerFactory.exe', '')
        sys.path.append(path + '/Python/3.7')

        import powerfactory as pfactory
        self.app = pfactory.GetApplication()

        FOLDER_NAME = v.features['pf']['folder_name']
        PROJECT_NAME = v.features['pf']['project_name']
        STUDY_CASE_NAME = v.features['pf']['study_case_name']
        self.project = self.app.ActivateProject(os.path.join(FOLDER_NAME, PROJECT_NAME))
        self.ldf = self.app.GetFromStudyCase('ComLdf')

        # activate study case
        study_case_folder = self.app.GetProjectFolder('study')
        study_case = study_case_folder.GetContents(STUDY_CASE_NAME + '.IntCase')[0]
        self.ldf.Execute()

        # SetTime object
        self.oSetTime = self.app.GetFromStudyCase('SetTime')

        self.grid = self.app.GetCalcRelevantObjects('*.ElmXnet')
        self.buss = self.app.GetCalcRelevantObjects('*.ElmTerm')
        self.trasformatori = self.app.GetCalcRelevantObjects('*.ElmTr2') + \
                             self.app.GetCalcRelevantObjects('*.ElmTr3') + \
                             self.app.GetCalcRelevantObjects('*.ElmTr4') + \
                             self.app.GetCalcRelevantObjects('*.ElmTrb') + self.app.GetCalcRelevantObjects('*.ElMdc')
        self.convertitori_dc = self.app.GetCalcRelevantObjects('*.ElmDcdc') + \
                               self.app.GetCalcRelevantObjects('*.ElmDcdcbi')
        self.pwm = self.app.GetCalcRelevantObjects('*.ElmVsc') + self.app.GetCalcRelevantObjects('*.ElmVscmono')
        self.carichi_ac = self.app.GetCalcRelevantObjects('*.ElmLod') + self.app.GetCalcRelevantObjects('*.ElmLodmv') +\
                          self.app.GetCalcRelevantObjects('*.ElmLodlv')
        self.carichi_dc = self.app.GetCalcRelevantObjects('*.ElmLoddc') + \
                          self.app.GetCalcRelevantObjects('*.ElmLoddcbi')
        self.link = self.app.GetCalcRelevantObjects('*.ElmLne')
        self.generatori_stat = self.app.GetCalcRelevantObjects('*.ElmGenstat')
        self.generatori_dc = self.app.GetCalcRelevantObjects('*.ElmDci')

        ##Get Time Characteristics
        self.Timeseries = self.app.GetCalcRelevantObjects('*.ChaTime')
        self.charef = self.app.GetCalcRelevantObjects('*.ChaRef')
        self.ldf =self.app.GetFromStudyCase('ComLdf')
        self.ldf.Execute()
        self.risorse = self.buss + self.trasformatori + self.convertitori_dc + self.pwm + self.carichi_ac + \
                       self.carichi_dc + self.link + self.generatori_stat + self.generatori_dc + self.grid
        self.a = True

        base_dict = yaml.safe_load(open(os.getcwd() + '/__shared__/attributes_template.yml'))

        for elemento in self.risorse:
            v.elements[elemento.loc_name]={}
            v.elements[elemento.loc_name]['category']={}
            v.elements[elemento.loc_name]['conn'] = {}
            v.elements[elemento.loc_name]['parameters'] = {}
            v.elements[elemento.loc_name]['results'] = {}

            if elemento.GetClassName() == 'ElmDci' and ('_PV' in elemento.loc_name):
                pass

            if elemento.GetClassName()!='ElmTerm' and elemento.GetClassName()!='ElmLne' and \
                    elemento.GetClassName() != 'ElmTr2' and elemento.GetClassName() != 'ElmTr3' and\
                    elemento.GetClassName() != 'ElmTr4' and elemento.GetClassName() != 'ElmTrb' and \
                    elemento.GetClassName()!='ElmDcdc' and elemento.GetClassName() != 'ElmVsc' and\
                    elemento.GetClassName() != 'ElmVscmono' and elemento.GetClassName() != 'ElmXnet' and\
                    '_BESS' not in elemento.loc_name and '_PWR' not in elemento.loc_name:
                v.elements[elemento.loc_name]['parameters']['profile'] = {}
                v.elements[elemento.loc_name]['parameters']['profile']['constant'] = self.a
                if elemento.GetClassName() == 'ElmDci':
                    v.elements[elemento.loc_name]['parameters']['profile']['curve'] = - elemento.GetAttribute('e:isetp')
                else:
                    v.elements[elemento.loc_name]['parameters']['profile']['curve'] = elemento.GetAttribute('e:scale0')
                v.elements[elemento.loc_name]['parameters']['profile']['name'] = None
            v.elements[elemento.loc_name]['ems'] = {}
            v.elements[elemento.loc_name]['ems']['profile'] = {}

            if elemento.GetClassName() == 'ElmTerm':
                v.elements[elemento.loc_name]['conn']['h'] = elemento.GetNode(0)
                v.elements[elemento.loc_name]['parameters']['Ur']= elemento.GetAttribute('e:uknom')
            if elemento.GetClassName() == 'ElmDci':
                v.elements[elemento.loc_name]['parameters']['Ir']= elemento.GetAttribute('c:Inom')
            if elemento.GetClassName() == 'ElmTerm' and elemento.GetAttribute('e:systype') == 0:
                v.elements[elemento.loc_name]['category'] = 'AC-Node'

            if elemento.GetClassName() == 'ElmTerm' and elemento.GetAttribute('e:systype') == 1:
                v.elements[elemento.loc_name]['category'] = 'DC-Node'

            if elemento.GetClassName() == 'ElmGenstat' and elemento.GetAttribute('e:cCategory') == 'Wind':
                v.elements[elemento.loc_name]['category'] = 'AC-Wind'
                v.elements[elemento.loc_name]['parameters']['P'] = elemento.GetAttribute('e:pgini')
                v.elements[elemento.loc_name]['parameters']['Q'] = elemento.GetAttribute('e:qgini')
                v.elements[elemento.loc_name]['parameters']['S'] = elemento.GetAttribute('e:sgini')
                v.elements[elemento.loc_name]['parameters']['Sr'] = elemento.GetAttribute('e:sgn')
                v.elements[elemento.loc_name]['parameters']['units'] = elemento.GetAttribute('e:ngnum')
                v.elements[elemento.loc_name]['parameters']['cosPhi'] = elemento.GetAttribute('e:cosn')
                if elemento.GetAttribute('e:mode_inp') == 'PC':
                    v.elements[elemento.loc_name]['parameters']['control'] = 1
                elif elemento.GetAttribute('e:mode_inp') == 'PQ':
                    v.elements[elemento.loc_name]['parameters']['control'] = 0
                elif elemento.GetAttribute('e:mode_inp') == 'SC':
```

```
            v.elements[elemento.loc_name]['parameters']['control'] = 2
        else:
            v.elements[elemento.loc_name]['parameters']['control'] = 1

        v.elements[elemento.loc_name]['parameters']['profile'] = dict()
        v.elements[elemento.loc_name]['parameters']['profile']['curve'] = elemento.GetAttribute('e:scale0')
        v.elements[elemento.loc_name]['parameters']['profile']['name'] = 'From Origin'
        v.elements[elemento.loc_name]['parameters']['profile']['constant'] = True

        h = 0
        for conn_element in elemento.GetConnectedElements(0, 0, 0):
            if conn_element.GetAttribute('e:uknom') > h:
                h = conn_element.GetAttribute('e:uknom')
                v.elements[elemento.loc_name]['conn']['h'] = conn_element.loc_name

    if elemento.GetClassName() == 'ElmTerm' and elemento.GetAttribute('e:systype') == 1:
        v.elements[elemento.loc_name]['category'] = 'DC-Node'

    if elemento.GetClassName() == 'ElmDci' and ('DC-Micro-Wind' in elemento.loc_name or
                                                'DCwind' in elemento.loc_name):
        v.elements[elemento.loc_name]['category'] = 'DC-Wind'

        v.elements[elemento.loc_name]['parameters']['Pr'] = \
            elemento.GetAttribute('e:Inom') * elemento.GetConnectedElements()[0].GetAttribute('e:uknom')
        ##nel nostro caso Pr=P dobbiamo considerare solo Pr

        v.elements[elemento.loc_name]['parameters']['P'] = v.elements[elemento.loc_name]['parameters']['Pr']

    if elemento.GetClassName() == 'ElmDci' and  ('_BESS' in elemento.loc_name or '_PWR' in elemento.loc_name):
        v.elements[elemento.loc_name]['category'] = 'Battery'
        v.elements[elemento.loc_name]['parameters']['cap_en'] = 0
        v.elements[elemento.loc_name]['parameters']['In'] = elemento.GetAttribute('e:Inom')

    if elemento.GetClassName() == 'ElmGenstat' and elemento.GetAttribute('e:cCategory') == 'Storage':
        v.elements[elemento.loc_name]['category'] = 'Battery'
        v.elements[elemento.loc_name]['parameters']['cap_en'] = 0
        v.elements[elemento.loc_name]['protections']['In'] = elemento.GetAttribute('e:Inom')

    if elemento.GetClassName() == 'ElmDci' and  ('_PV' in elemento.loc_name):
        v.elements[elemento.loc_name]['category'] = 'PV'
        v.elements[elemento.loc_name]['parameters']['Sr'] = \
            elemento.GetAttribute('e:Inom') * elemento.GetConnectedElements()[0].GetAttribute('e:uknom')
        #   Per noi Sr=Pr   potenza di targa moltiplicare la corrente X la tensione di busbar

        v.elements[elemento.loc_name]['parameters']['In'] = elemento.GetAttribute('e:Inom')

    if elemento.GetClassName() == 'ElmLod' or  elemento.GetClassName() == 'ElmLodmv' or \
            elemento.GetClassName() == 'ElmLodlv':
        v.elements[elemento.loc_name]['category'] = 'AC-Load'
        v.elements[elemento.loc_name]['parameters']['control'] = {}
        v.elements[elemento.loc_name]['category'] = 'AC-Load'
        v.elements[elemento.loc_name]['parameters']['P'] = elemento.GetAttribute('e:plini') * 1000
        v.elements[elemento.loc_name]['parameters']['Q'] = elemento.GetAttribute('e:qlini') * 1000
        v.elements[elemento.loc_name]['parameters']['S'] = elemento.GetAttribute('e:slini') * 1000
        v.elements[elemento.loc_name]['parameters']['I'] = elemento.GetAttribute('e:ilini') * 1000
        v.elements[elemento.loc_name]['parameters']['cosPhi'] = elemento.GetAttribute('e:coslini')

        if elemento.GetAttribute('e:mode_inp')=='DEF' or elemento.GetAttribute('e:mode_inp')=='PQ':
            v.elements[elemento.loc_name]['parameters']['control'] = 0
        elif elemento.GetAttribute('e:mode_inp') == 'PC':
            v.elements[elemento.loc_name]['parameters']['control'] = 1
        elif elemento.GetAttribute('e:mode_inp') == 'IC':
            v.elements[elemento.loc_name]['parameters']['control'] = 2
        elif elemento.GetAttribute('e:mode_inp')=='IP':
            v.elements[elemento.loc_name]['parameters']['control'] = 3
        elif elemento.GetAttribute('e:mode_inp') == 'SC':
            v.elements[elemento.loc_name]['parameters']['control'] = 4
        else:
            v.elements[elemento.loc_name]['parameters']['control'] = 0

    if elemento.GetClassName() == 'ElmLoddc' or elemento.GetClassName() == 'ElmLoddcbi':
        v.elements[elemento.loc_name]['category'] = 'DC-Load'
        v.elements[elemento.loc_name]['parameters']['control'] = 0
        v.elements[elemento.loc_name]['parameters']['P'] = elemento.GetAttribute('e:plini') * 1000
        v.elements[elemento.loc_name]['parameters']['I'] = 0
        v.elements[elemento.loc_name]['parameters']['R'] = 0

    if elemento.GetClassName() == 'ElmTr2':
        v.elements[elemento.loc_name]['category'] = '2W-Transformer'
    if elemento.GetClassName() == 'ElmTr3':
        v.elements[elemento.loc_name]['category'] = '3W-Transformer'
    if elemento.GetClassName() == 'ElmTr4':
        v.elements[elemento.loc_name]['category'] = '4W-Transformer'
    if elemento.GetClassName() == 'ElmTrb':
        v.elements[elemento.loc_name]['category'] = 'Trb-Transformer'
    if elemento.GetClassName() == 'ElMdc':
        v.elements[elemento.loc_name]['category'] = 'DC-Transformer'

    if elemento.GetClassName() == 'ElmTr2' or elemento.GetClassName() == 'ElmTr3' or \
            elemento.GetClassName() == 'ElmTr4' or elemento.GetClassName() == 'ElmTrb' or \
            elemento.GetClassName() == 'ElMdc' or elemento.GetClassName() == 'ElmDcdc' or \
            elemento.GetClassName() == 'ElmDcdcbi':
        h = 0
        for conn_element in elemento.GetConnectedElements(0, 0, 0):
            if conn_element.GetAttribute('e:uknom') > h:
                h = conn_element.GetAttribute('e:uknom')
                v.elements[elemento.loc_name]['conn']['h'] = conn_element.loc_name

    if elemento.GetClassName() == 'ElmTr2':
        v.elements[elemento.loc_name]['parameters']['Sr'] =elemento.GetAttribute('e:Snom_a') * 1000
        v.elements[elemento.loc_name]['parameters']['UnHV'] = 0
        v.elements[elemento.loc_name]['parameters']['UnLV'] = 0
        v.elements[elemento.loc_name]['parameters']['URr1'] = 0
        v.elements[elemento.loc_name]['parameters']['Ukr1'] = 0
        v.elements[elemento.loc_name]['parameters']['URr0'] = 0
        v.elements[elemento.loc_name]['parameters']['Ukr0'] = 0

    if elemento.GetClassName() == 'ElmVsc' or elemento.GetClassName() == 'ElmVscmono':
        v.elements[elemento.loc_name]['category'] = 'PWM'
        for conn_element in elemento.GetConnectedElements(0, 0, 0):
            if conn_element.GetAttribute('e:systype') == 0:
                v.elements[elemento.loc_name]['conn']['h'] = conn_element.loc_name

        v.elements[elemento.loc_name]['parameters']['Sr'] = elemento.GetAttribute('e:Snom') * 1000
        v.elements[elemento.loc_name]['parameters']['Ur'] = elemento.GetAttribute('e:Unom')
```

```
            v.elements[elemento.loc_name]['parameters']['S_loss_idle'] = 0  # Perdite in IDLE [kW]
            v.elements[elemento.loc_name]['parameters']['Sw_loss'] = 0      # Switching losses [kW/A]
            v.elements[elemento.loc_name]['parameters']['R_loss'] = 0       # Resistive losses [Ohm]

        #
        if elemento.GetClassName() == 'ElmDcdc' or elemento.GetClassName() == 'ElmDcdcbi':
            v.elements[elemento.loc_name]['category'] = 'DC-DC_Conv'
            for conn_element in elemento.GetConnectedElements(0, 0, 0):
                if conn_element.GetAttribute('e:systype') == 0:
                    v.elements[elemento.loc_name]['conn']['h'] = conn_element.loc_name
            v.elements[elemento.loc_name]['parameters']['Sr'] = elemento.GetAttribute('e:Curn') * \
                                                                elemento.GetConnectedElements()[0].\
                                                                    GetAttribute('e:uknom')

        if (elemento.GetClassName() == 'ElmLod' or elemento.GetClassName() == 'ElmLodmv' or
            elemento.GetClassName() == 'ElmLodlv' or elemento.GetClassName() == 'ElmLoddc' or
            elemento.GetClassName() == 'ElmLoddcbi' or elemento.GetClassName() == 'ElmDci') and \
                len(elemento.GetConnectedElements(0, 0, 0))>0:
            v.elements[elemento.loc_name]['conn']['h'] = elemento.GetConnectedElements(0, 0, 0)[0].loc_name

        #
        if elemento.GetClassName() == 'ElmLne' and elemento.GetAttribute('t:systp') == 0:
            v.elements[elemento.loc_name]['parameters']['B0']= elemento.GetAttribute('t:blineO')
            v.elements[elemento.loc_name]['parameters']['B1'] = elemento.GetAttribute('t:bline')
            v.elements[elemento.loc_name]['parameters']['X0'] = elemento.GetAttribute('t:rlineO')
            v.elements[elemento.loc_name]['parameters']['X1'] = elemento.GetAttribute('t:rline')
            v.elements[elemento.loc_name]['parameters']['R0'] = elemento.GetAttribute('t:rlineO')
            v.elements[elemento.loc_name]['parameters']['R1'] = elemento.GetAttribute('t:rline')
            v.elements[elemento.loc_name]['parameters']['Irmax'] = elemento.GetAttribute('e:Inom') * 1000
            v.elements[elemento.loc_name]['parameters']['Ur'] = elemento.GetAttribute('e:Unom')
            v.elements[elemento.loc_name]['parameters']['length'] = elemento.GetAttribute('e:dline') * 1000
            v.elements[elemento.loc_name]['parameters']['lines'] = elemento.GetAttribute('e:nlnum')
            v.elements[elemento.loc_name]['parameters']['sez'] = elemento.typ_id.qurs * math.pow(10, -6)
            v.elements[elemento.loc_name]['category'] = 'AC-Line'
            v.elements[elemento.loc_name]['conn']['h'] = elemento.GetNode(0).loc_name

        #
        if elemento.GetClassName() == 'ElmLne' and elemento.GetAttribute('t:systp') == 1:
            v.elements[elemento.loc_name]['parameters']['C'] = elemento.GetAttribute('t:cline')
            v.elements[elemento.loc_name]['parameters']['R'] = elemento.GetAttribute('t:rline')
            v.elements[elemento.loc_name]['parameters']['L'] = elemento.GetAttribute('t:lline')
            v.elements[elemento.loc_name]['parameters']['Irmax'] = elemento.GetAttribute('t:sline') * 1000
            v.elements[elemento.loc_name]['parameters']['lines'] = elemento.GetAttribute('e:nlnum')
            v.elements[elemento.loc_name]['parameters']['length'] = elemento.GetAttribute('e:dline') * 1000

            v.elements[elemento.loc_name]['category'] = 'DC-Line'
            v.elements[elemento.loc_name]['conn']['h'] = elemento.GetNode(0).loc_name
            v.elements[elemento.loc_name]['parameters']['sez'] = elemento.typ_id.qurs * math.pow(10, -6)

        if elemento.GetClassName() == 'ElmXnet':
            for dictionary in ['parameters', 'ems', 'category', 'results', 'protections']:
                v.elements[elemento.loc_name][dictionary] = base_dict['Ext-Grid'][dictionary]
            v.elements[elemento.loc_name]['conn']['h'] = elemento.GetNode(0).loc_name
            v.ext_grid = elemento.loc_name

        for conn_element in elemento.GetConnectedElements(0, 0, 0):
            v.elements[elemento.loc_name]['conn'][conn_element.loc_name] = (not self.a)
        for conn_element in elemento.GetConnectedElements(1, 1, 1):
            v.elements[elemento.loc_name]['conn'][conn_element.loc_name] = self.a

        for cha in self.charef:
            if cha.GetAttribute('e:fold_id') == elemento:
                v.elements[elemento.loc_name]['parameters']['profile']['constant'] = (not self.a)
                v.elements[elemento.loc_name]['parameters']['profile']['curve'] = cha.GetAttribute('t:vector')
                v.elements[elemento.loc_name]['parameters']['profile']['name'] = cha.loc_name

        v.elements[elemento.loc_name]['reliability'] = \
            base_dict[v.elements[elemento.loc_name]['category']]['reliability']

        self.model_dict = copy.deepcopy(v.elements)

#
def set_params(self, element):
    if 'profile' in v.elements[element]['parameters'].keys():
        if 'name' in v.elements[element]['parameters']['profile'].keys():
            cha_name = v.elements[element]['parameters']['profile']['name']
            for cha in self.charef :
                if cha.loc_name == cha_name and 'curve' in v.elements[element]['parameters']['profile'].keys():
                    cha.SetAttribute('t:vector', v.elements[element]['parameters']['profile']['curve'])
                    print('Vettore caricato per: ' + element)

    # SET AC-LOAD
    if v.elements[element]['category'] == 'AC-Load':
        for obj_element in self.risorse:
            if obj_element.loc_name == element:
                obj_element.SetAttribute('e:plini',v.elements[element]['parameters']['P'] / 1000)
                obj_element.SetAttribute('e:qlini',v.elements[element]['parameters']['Q'] / 1000)
                obj_element.SetAttribute('e:slini',v.elements[element]['parameters']['S'] / 1000)
                obj_element.SetAttribute('e:ilini',v.elements[element]['parameters']['I'] / 1000)
                obj_element.SetAttribute('e:coslini',v.elements[element]['parameters']['cosPhi'])

                if v.elements[element]['parameters']['control'] == 0:
                    obj_element.SetAttribute('e:mode_inp','PQ' )
                elif v.elements[element]['parameters']['control'] == 1:
                    obj_element.SetAttribute('e:mode_inp','PC' )
                elif v.elements[element]['parameters']['control'] == 2:
                    obj_element.SetAttribute('e:mode_inp','IC' )
                elif v.elements[element]['parameters']['control'] == 3:
                    obj_element.SetAttribute('e:mode_inp','IP' )
                elif v.elements[element]['parameters']['control'] == 4:
                    obj_element.SetAttribute('e:mode_inp','SC' )

    #SET DC-LOAD
    if v.elements[element]['category'] == 'DC-Load':
        for  obj_element in self.risorse:
            if obj_element.loc_name == element:
                obj_element.SetAttribute('e:plini',v.elements[element]['parameters']['P'] / 1000)

    #SET AC-Wind
    if v.elements[element]['category'] == 'AC-Wind':
        for obj_element in self.risorse:
            if obj_element.loc_name == element:
                obj_element.SetAttribute('e:pgini',v.elements[element]['parameters']['P'])
                obj_element.SetAttribute('e:qgini',v.elements[element]['parameters']['Q'])
                obj_element.SetAttribute('e:sgini',v.elements[element]['parameters']['S'])
```

```
                obj_element.SetAttribute('c:sgn',v.elements[element]['parameters']['Sr'])
                obj_element.SetAttribute('e:ngnum', v.elements[element]['parameters']['units'])
                obj_element.SetAttribute('c:cosn', v.elements[element]['parameters']['cosPhi'])
                if v.elements[element]['parameters']['control'] == 0:
                    obj_element.SetAttribute('e:av_mode','constv' )
                elif v.elements[element]['parameters']['control'] == 1:
                    obj_element.SetAttribute('e:av_mode','constq' )
                elif v.elements[element]['parameters']['control'] == 2:
                    obj_element.SetAttribute('e:av_mode','constc' )


    #SET AC-LINE
    if v.elements[element]['category'] == 'AC-Line':
        print(element)
        for obj_element in self.risorse:
            if obj_element.loc_name == element:
                # obj_element.SetAttribute('e:B0',v.elements[element]['parameters']['B0'])
                # obj_element.SetAttribute('e:B1',v.elements[element]['parameters']['B1'])
                obj_element.SetAttribute('t:xline0',v.elements[element]['parameters']['X0'])
                obj_element.SetAttribute('t:xline',v.elements[element]['parameters']['X1'])
                obj_element.SetAttribute('t:rline0',v.elements[element]['parameters']['R0'])
                obj_element.SetAttribute('t:rline',v.elements[element]['parameters']['R1'])
                obj_element.SetAttribute('t:sline',v.elements[element]['parameters']['Irmax'] / 1000)
                obj_element.SetAttribute('t:uline',v.elements[element]['parameters']['Ur'])
                obj_element.SetAttribute('e:dline',v.elements[element]['parameters']['length'] / 1000)
                obj_element.typ_id.qurs = v.elements[element]['parameters']['sez']

    if v.elements[element]['category'] == '2W-Transformer':
        pass
    if v.elements[element]['category'] == 'AC-Node':
        pass
    if v.elements[element]['category'] == 'Battery':
        pass
    if v.elements[element]['category'] == 'DC-DC_Conv':
        pass
    if v.elements[element]['category'] == 'DC-Line':
        pass
    if v.elements[element]['category'] == 'DC-node':
        pass
    if v.elements[element]['category'] == 'DC-Wind':
        pass
    if v.elements[element]['category'] == 'PV':
        pass
    if v.elements[element]['category'] == 'PWM':
        pass

    for obj_element in self.risorse:
        if obj_element.loc_name == element and len(v.elements[element]['conn'].keys()) == 2 and \
                obj_element.GetClassName() != 'ElmTerm':
            for elemento in v.elements[element]['conn'].keys():
                if v.elements[element]['conn'][elemento] != self.model_dict[element]['conn'][elemento]:
                    if v.elements[element]['conn'][elemento] == self.a:
                        obj_element.GetCubicle(0).SwitchOn()
                    else:
                        obj_element.GetCubicle(0).SwitchOff()

        if obj_element.loc_name == element and len(v.elements[element]['conn'].keys())>2 and \
                obj_element.GetClassName() != 'ElmTerm' :
            for elemento in v.elements[element]['conn'].keys():
                if v.elements[element]['conn'][elemento] != self.model_dict[element]['conn'][elemento]:
                    # print('Cubicoli di ' + elemento + ' modificato')
                    if v.elements[element]['conn'][elemento] == self.a and \
                            v.elements[element]['conn']['h'] == elemento:
                        obj_element.GetCubicle(0).SwitchOn()
                    elif v.elements[element]['conn'][elemento] == (not self.a) and \
                            v.elements[element]['conn']['h'] == elemento:
                        obj_element.GetCubicle(0).SwitchOff()
                    if v.elements[element]['conn'][elemento] == self.a and \
                            v.elements[element]['conn']['h'] != elemento and \
                            v.elements[element]['conn']['h'] != None:
                        obj_element.GetCubicle(1).SwitchOn()
                    elif v.elements[element]['conn'][elemento] == (not self.a) and \
                            v.elements[element]['conn']['h'] != elemento and \
                            v.elements[element]['conn']['h'] != None:
                        obj_element.GetCubicle(1).SwitchOff()
    self.model_dict = copy.deepcopy(v.elements)

#
def read_dict(self, element):
    self.T0 = v.elements[element]['reliability']['T0']
    self.alfa = v.elements[element]['reliability']['alfa']
    self.beta = v.elements[element]['reliability']['beta']
    self.Pi_E = v.elements[element]['reliability']['Pi_E']
    self.Pi_Q = v.elements[element]['reliability']['Pi_Q']

    return self.T0, self.alfa, self.beta,self.Pi_E, self.Pi_Q

#
def tstep_line(self, h, m, Ta):
    if h == 0 and m == 0:
        iTime = 0
    elif h > 0 and h <= 9 and m == 0:
        iTime = '0' + str(h) + str(m) + '000'
    elif h > 9 and m == 0:
        iTime = str(h) + str(m) + '000'
    elif h > 9 and m != 0:
        iTime = str(h) + str(m * 100)
    else:
        iTime = '0' + str(h) + str(m * 100)
    self.oSetTime.SetAttribute('e:cTime', str(iTime))
    self.ldf.SetAttribute('e:iopt_tem', 3)
    self.ldf.SetAttribute('e:temperature', Ta)
    self.ldf.Execute()

#
def instantLF(self, m):
    h=int(m/60)
    min=round((m/60 - h)*60)
    if h == 0 and min == 0:
        iTime = 0
    elif h > 0 and h <= 9 and min == 0:
        iTime = '0' + str(h) + str(min) + '000'
    elif h > 9 and min == 0:
        iTime = str(h) + str(min) + '000'
    elif h > 9 and m != 0:
```

```
            iTime = str(h) + str(min * 100)
        else:
            iTime = '0' + str(h) + str(min * 100)
        self.oSetTime.SetAttribute('e:cTime', str(iTime))
        self.ldf.Execute()
        for elemento in self.risorse:
            v.elements[elemento.loc_name]['results']['LimitViolated'] = False

            if elemento.GetClassName() == 'ElmLne' or elemento.GetClassName() == 'ElmTr2' or \
                    elemento.GetClassName() == 'ElmTr3' or elemento.GetClassName() == 'ElmTr4' or \
                    elemento.GetClassName() == 'ElmTrb' or elemento.GetClassName() == 'ElmDcdc'  or \
                    elemento.GetClassName() == 'ElmVsc' or elemento.GetClassName() == 'ElmVscmono' :
                v.elements[elemento.loc_name]['results']['Port1'] = {}
                v.elements[elemento.loc_name]['results']['Port2'] = {}

            if (elemento.GetClassName() == 'ElmXnet' or elemento.GetClassName() == 'ElmLod' or
                elemento.GetClassName() == 'ElmGenstat' or elemento.GetClassName() == 'ElmLodmv' or
                elemento.GetClassName() == 'ElmLodlv' or elemento.GetClassName() == 'ElmLoddc' or
                elemento.GetClassName() == 'ElmLoddcbi' or elemento.GetClassName() == 'ElmDci') and \
                    elemento.GetAttribute('e:outserv')==0 :
                v.elements[elemento.loc_name]['results']['I'] = elemento.GetAttribute('m:I:bus1')
                v.elements[elemento.loc_name]['results']['Iangle'] = elemento.GetAttribute('m:phii:bus1')
                v.elements[elemento.loc_name]['results']['P']= elemento.GetAttribute('m:P:bus1')
                v.elements[elemento.loc_name]['results']['Q']= elemento.GetAttribute('m:Q:bus1')
                v.elements[elemento.loc_name]['results']['S']= elemento.GetAttribute('m:S:bus1')
                v.elements[elemento.loc_name]['results']['U']= elemento.GetAttribute('m:U1l:bus1')
                v.elements[elemento.loc_name]['results']['cosPhi']= elemento.GetAttribute('m:cosphi:bus1')

            if elemento.GetClassName() == 'ElmTerm' and elemento.GetAttribute('e:outserv')==0 :
                v.elements[elemento.loc_name]['results']['Pgen'] = elemento.GetAttribute('m:Pgen')
                v.elements[elemento.loc_name]['results']['Pload'] = elemento.GetAttribute('m:Pload')
                v.elements[elemento.loc_name]['results']['Qgen'] = elemento.GetAttribute('m:Qgen')
                v.elements[elemento.loc_name]['results']['Qload'] = elemento.GetAttribute('m:Qload')
                v.elements[elemento.loc_name]['results']['U'] = elemento.GetAttribute('m:Ul')
                v.elements[elemento.loc_name]['results']['Un'] = elemento.GetAttribute('e:uknom')
                v.elements[elemento.loc_name]['results']['Up'] = elemento.GetAttribute('m:u1')

            if (elemento.GetClassName() == 'ElmTr2' or elemento.GetClassName() == 'ElmTr3' or
                elemento.GetClassName() == 'ElmTr4' or elemento.GetClassName() == 'ElmTrb' ) and \
                    elemento.GetAttribute('e:outserv')==0 :
                v.elements[elemento.loc_name]['results']['Ploss'] = elemento.GetAttribute('c:Ploss')
                v.elements[elemento.loc_name]['results']['Qloss'] = elemento.GetAttribute('c:Qloss')
                v.elements[elemento.loc_name]['results']['Port1']['I'] = elemento.GetAttribute('m:I:bushv')
                v.elements[elemento.loc_name]['results']['Port1']['Iangle'] = elemento.GetAttribute('m:phii:bushv')
                v.elements[elemento.loc_name]['results']['Port1']['P'] = elemento.GetAttribute('m:P:bushv')
                v.elements[elemento.loc_name]['results']['Port1']['Q'] = elemento.GetAttribute('m:Q:bushv')
                v.elements[elemento.loc_name]['results']['Port1']['S'] = elemento.GetAttribute('m:S:bushv')
                v.elements[elemento.loc_name]['results']['Port1']['U'] = elemento.GetAttribute('n:Ul:bushv')
                v.elements[elemento.loc_name]['results']['Port1']['cosPhi'] = elemento.GetAttribute('m:cosphi:bushv')
                v.elements[elemento.loc_name]['results']['Port2']['I'] = elemento.GetAttribute('m:I:buslv')
                v.elements[elemento.loc_name]['results']['Port2']['Iangle'] = elemento.GetAttribute('m:phii:buslv')
                v.elements[elemento.loc_name]['results']['Port2']['P'] = elemento.GetAttribute('m:P:buslv')
                v.elements[elemento.loc_name]['results']['Port2']['Q'] = elemento.GetAttribute('m:Q:buslv')
                v.elements[elemento.loc_name]['results']['Port2']['S'] = elemento.GetAttribute('m:S:buslv')
                v.elements[elemento.loc_name]['results']['Port2']['U'] = elemento.GetAttribute('n:Ul:buslv')
                v.elements[elemento.loc_name]['results']['Port2']['cosPhi'] = elemento.GetAttribute('m:cosphi:buslv')

            if (elemento.GetClassName() == 'ElmLne' or elemento.GetClassName() == 'ElmDcdc') and \
                    elemento.GetAttribute('e:outserv')==0 :
                v.elements[elemento.loc_name]['results']['Port1']['I'] = elemento.GetAttribute('m:I:bus1')
                v.elements[elemento.loc_name]['results']['Port1']['Iangle'] = elemento.GetAttribute('m:phii:bus1')
                v.elements[elemento.loc_name]['results']['Port1']['P'] = elemento.GetAttribute('m:P:bus1')
                v.elements[elemento.loc_name]['results']['Port1']['Q'] = elemento.GetAttribute('m:Q:bus1')
                v.elements[elemento.loc_name]['results']['Port1']['S'] = elemento.GetAttribute('m:S:bus1')
                v.elements[elemento.loc_name]['results']['Port1']['U'] = elemento.GetAttribute('m:U1l:bus1')
                v.elements[elemento.loc_name]['results']['Port1']['cosPhi'] = elemento.GetAttribute('m:cosphi:bus1')
                v.elements[elemento.loc_name]['results']['Port2']['I'] = elemento.GetAttribute('m:I:bus2')
                v.elements[elemento.loc_name]['results']['Port2']['Iangle'] = elemento.GetAttribute('m:phii:bus2')
                v.elements[elemento.loc_name]['results']['Port2']['P'] = elemento.GetAttribute('m:P:bus2')
                v.elements[elemento.loc_name]['results']['Port2']['Q'] = elemento.GetAttribute('m:Q:bus2')
                v.elements[elemento.loc_name]['results']['Port2']['S'] = elemento.GetAttribute('m:S:bus2')
                v.elements[elemento.loc_name]['results']['Port2']['U'] = elemento.GetAttribute('m:U1l:bus2')
                v.elements[elemento.loc_name]['results']['Port2']['cosPhi'] = elemento.GetAttribute('m:cosphi:bus2')
                v.elements[elemento.loc_name]['results']['Ploss'] = elemento.GetAttribute('c:Ploss')
                v.elements[elemento.loc_name]['results']['Qloss'] = elemento.GetAttribute('c:Qloss')

            if (elemento.GetClassName() == 'ElmVsc' or elemento.GetClassName() == 'ElmVscmono') and \
                    elemento.GetAttribute('e:outserv') == 0:
                v.elements[elemento.loc_name]['results']['Port1']['I'] = elemento.GetAttribute('m:I:busac')
                v.elements[elemento.loc_name]['results']['Port1']['Iangle'] = elemento.GetAttribute('m:phii:busac')
                v.elements[elemento.loc_name]['results']['Port1']['P'] = elemento.GetAttribute('m:P:busac')
                v.elements[elemento.loc_name]['results']['Port1']['Q'] = elemento.GetAttribute('m:Q:busac')
                v.elements[elemento.loc_name]['results']['Port1']['S'] = elemento.GetAttribute('m:S:busac')
                v.elements[elemento.loc_name]['results']['Port1']['U'] = elemento.GetAttribute('m:U1l:busac')
                v.elements[elemento.loc_name]['results']['Port1']['cosPhi'] = elemento.GetAttribute('m:cosphi:busac')
                v.elements[elemento.loc_name]['results']['Port2']['I'] = elemento.GetAttribute('m:I:busdc')
                v.elements[elemento.loc_name]['results']['Port2']['Iangle'] = elemento.GetAttribute('m:phii:busdc')
                v.elements[elemento.loc_name]['results']['Port2']['P'] = elemento.GetAttribute('m:P:busdc')
                v.elements[elemento.loc_name]['results']['Port2']['Q'] = elemento.GetAttribute('m:Q:busdc')
                v.elements[elemento.loc_name]['results']['Port2']['S'] = elemento.GetAttribute('m:S:busdc')
                v.elements[elemento.loc_name]['results']['Port2']['U'] = elemento.GetAttribute('m:U1:busdc')
                v.elements[elemento.loc_name]['results']['Port2']['cosPhi'] = elemento.GetAttribute('m:cosphi:busdc')
                v.elements[elemento.loc_name]['results']['Ploss'] = elemento.GetAttribute('c:Ploss')
                v.elements[elemento.loc_name]['results']['Qloss'] = elemento.GetAttribute('c:Qloss')

    #
    def profileLF(self, htot):
        for elemento in self.risorse:
            I = []
            Iangle = []
            P = []
            Q = []
            S = []
            U = []
            cosPhi = []
            Pgen = []
            Pload = []
            Qgen = []
            Qload = []
            U_elmterm = []
            Un = []
            Up = []
            Ploss_elmTr = []
            Qloss_elmTr = []
            I_elmTr_port1 = []
```

```
Iangle_elmTr_port1 = []
P_elmTr_port1 = []
Q_elmTr_port1 = []
S_elmTr_port1 = []
U_elmTr_port1 = []
cosPhi_elmTr_port1 = []
I_elmTr_port2 = []
Iangle_elmTr_port2 = []
P_elmTr_port2 = []
Q_elmTr_port2 = []
S_elmTr_port2 = []
U_elmTr_port2 = []
cosPhi_elmTr_port2 = []

I_elmLne_port1 = []
Iangle_elmLne_port1 = []
P_elmLne_port1 = []
Q_elmLne_port1 = []
S_elmLne_port1 = []
U_elmLne_port1 = []
cosPhi_elmLne_port1 = []
I_elmLne_port2 = []
Iangle_elmLne_port2 = []
P_elmLne_port2 = []
Q_elmLne_port2 = []
S_elmLne_port2 = []
U_elmLne_port2 = []
cosPhi_elmLne_port2 = []

I_elmVsc_port1 = []
Iangle_elmVsc_port1 = []
P_elmVsc_port1 = []
Q_elmVsc_port1 = []
S_elmVsc_port1 = []
U_elmVsc_port1 = []
cosPhi_elmVsc_port1 = []
I_elmVsc_port2 = []
Iangle_elmVsc_port2 = []
P_elmVsc_port2 = []
Q_elmVsc_port2 = []
S_elmVsc_port2 = []
U_elmVsc_port2 = []
cosPhi_elmVsc_port2 = []

if elemento.GetClassName() == 'ElmLne' or elemento.GetClassName() == 'ElmTr2' or \
        elemento.GetClassName() == 'ElmTr3' or elemento.GetClassName() == 'ElmTr4' or \
        elemento.GetClassName() == 'ElmTrb' or elemento.GetClassName() == 'ElmDcdc'  or \
        elemento.GetClassName() == 'ElmVsc' or elemento.GetClassName() == 'ElmVscmono' :
    v.elements[elemento.loc_name]['results']['Port1'] = {}
    v.elements[elemento.loc_name]['results']['Port2'] = {}

for h in range(0,htot):
    for m in range(0,60,15):
        if h == 0 and m == 0:
            iTime = 0
        elif h > 0 and h <= 9 and m == 0:
            iTime = '0' + str(h) + str(m) + '000'
        elif h > 9 and m == 0:
            iTime = str(h) + str(m) + '000'
        elif h > 9 and m != 0:
            iTime = str(h) + str(m * 100)
        else:
            iTime = '0' + str(h) + str(m * 100)
        self.oSetTime.SetAttribute('e:cTime', str(iTime))

        if (elemento.GetClassName() == 'ElmLod' or elemento.GetClassName() == 'ElmLodmv' or
            elemento.GetClassName() == 'ElmLodlv' or elemento.GetClassName() == 'ElmLoddc' or
            elemento.GetClassName() == 'ElmLoddcbi' or elemento.GetClassName() == 'ElmDci') and \
                elemento.GetAttribute('e:outserv') == 0:
            I.append(elemento.GetAttribute('m:I:bus1'))
            Iangle.append(elemento.GetAttribute('m:phii:bus1'))
            P.append(elemento.GetAttribute('m:P:bus1'))
            Q.append(elemento.GetAttribute('m:Q:bus1'))
            S.append(elemento.GetAttribute('m:S:bus1'))
            U.append(elemento.GetAttribute('m:cosphi:bus1'))
            cosPhi.append(elemento.GetAttribute('m:I:bus1'))

        if elemento.GetClassName() == 'ElmTerm' and elemento.GetAttribute('e:outserv') == 0:
            Pgen.append(elemento.GetAttribute('m:Pgen'))
            Pload.append(elemento.GetAttribute('m:Pload'))
            Qgen.append(elemento.GetAttribute('m:Qgen'))
            Qload.append(elemento.GetAttribute('m:Qload'))
            U_elmterm.append(elemento.GetAttribute('m:U'))
            Un.append(elemento.GetAttribute('e:uknom'))
            Up.append(elemento.GetAttribute('m:u1'))

        if (elemento.GetClassName() == 'ElmTr2' or elemento.GetClassName() == 'ElmTr3' or
            elemento.GetClassName() == 'ElmTr4' or elemento.GetClassName() == 'ElmTrb') and \
                elemento.GetAttribute('e:outserv') == 0:
            Ploss_elmTr.append(elemento.GetAttribute('c:Ploss'))
            Qloss_elmTr.append(elemento.GetAttribute('c:Qloss'))
            I_elmTr_port1.append(elemento.GetAttribute('m:I:bushv'))
            Iangle_elmTr_port1.append(elemento.GetAttribute('m:phii:bushv'))
            P_elmTr_port1.append(elemento.GetAttribute('m:P:bushv'))
            Q_elmTr_port1.append(elemento.GetAttribute('m:Q:bushv'))
            S_elmTr_port1.append(elemento.GetAttribute('m:S:bushv'))
            U_elmTr_port1.append(elemento.GetAttribute('m:U1:bushv'))
            cosPhi_elmTr_port1.append(elemento.GetAttribute('m:cosphi:bushv'))
            I_elmTr_port2.append(elemento.GetAttribute('m:I:buslv'))
            Iangle_elmTr_port2.append(elemento.GetAttribute('m:phii:buslv'))
            P_elmTr_port2.append(elemento.GetAttribute('m:P:buslv'))
            Q_elmTr_port2.append(elemento.GetAttribute('m:Q:buslv'))
            S_elmTr_port2.append(elemento.GetAttribute('m:S:buslv'))
            U_elmTr_port2.append(elemento.GetAttribute('m:U1:buslv'))
            cosPhi_elmTr_port2.append(elemento.GetAttribute('m:cosphi:buslv'))


        if (elemento.GetClassName() == 'ElmLne' or elemento.GetClassName() == 'ElmDcdc') and \
                elemento.GetAttribute('e:outserv') == 0:
            I_elmLne_port1.append(elemento.GetAttribute('m:I:bus1'))
            Iangle_elmLne_port1.append(elemento.GetAttribute('m:phii:bus1'))
            P_elmLne_port1.append(elemento.GetAttribute('m:P:bus1'))
            Q_elmLne_port1.append(elemento.GetAttribute('m:Q:bus1'))
            S_elmLne_port1.append(elemento.GetAttribute('m:S:bus1'))
            U_elmLne_port1.append(elemento.GetAttribute('m:U1:bus1'))
```

```
                    cosPhi_elmLne_port1.append(elemento.GetAttribute('m:cosphi:bus1'))
                    I_elmLne_port2.append(elemento.GetAttribute('m:I:bus2'))
                    Iangle_elmLne_port2.append(elemento.GetAttribute('m:phii:bus2'))
                    P_elmLne_port2.append(elemento.GetAttribute('m:P:bus2'))
                    Q_elmLne_port2.append(elemento.GetAttribute('m:Q:bus2'))
                    S_elmLne_port2.append(elemento.GetAttribute('m:S:bus2'))
                    U_elmLne_port2.append(elemento.GetAttribute('m:U1:bus2'))
                    cosPhi_elmLne_port2.append(elemento.GetAttribute('m:cosphi:bus2'))

                if (elemento.GetClassName() == 'ElmVsc' or elemento.GetClassName() == 'ElmVscmono') and \
                        lemento.GetAttribute('e:outserv') == 0:
                    I_elmVsc_port1.append(elemento.GetAttribute('m:I:busac'))
                    Iangle_elmVsc_port1.append(elemento.GetAttribute('m:phii:busac'))
                    P_elmVsc_port1.append(elemento.GetAttribute('m:P:busac'))
                    Q_elmVsc_port1.append(elemento.GetAttribute('m:Q:busac'))
                    S_elmVsc_port1.append(elemento.GetAttribute('m:S:busac'))
                    U_elmVsc_port1.append(elemento.GetAttribute('m:U1:busac'))
                    cosPhi_elmVsc_port1.append(elemento.GetAttribute('m:cosphi:busac'))
                    I_elmVsc_port2.append(elemento.GetAttribute('m:I:busdc'))
                    Iangle_elmVsc_port2.append(elemento.GetAttribute('m:phii:busdc'))
                    P_elmVsc_port2.append(elemento.GetAttribute('m:P:busdc'))
                    Q_elmVsc_port2.append(elemento.GetAttribute('m:Q:busdc'))
                    S_elmVsc_port2.append(elemento.GetAttribute('m:S:busdc'))
                    U_elmVsc_port2.append(elemento.GetAttribute('m:U1:busdc'))
                    cosPhi_elmVsc_port2.append(elemento.GetAttribute('m:cosphi:busdc'))

            if (elemento.GetClassName() == 'ElmLod' or elemento.GetClassName() == 'ElmLodmv' or
                elemento.GetClassName() == 'ElmLodlv' or elemento.GetClassName() == 'ElmLoddc' or
                elemento.GetClassName() == 'ElmLoddcbi' or elemento.GetClassName() == 'ElmDci') and \
                    elemento.GetAttribute('e:outserv') == 0:
                v.elements[elemento.loc_name]['results']['I'] = I
                v.elements[elemento.loc_name]['results']['Iangle'] = Iangle
                v.elements[elemento.loc_name]['results']['P'] = P
                v.elements[elemento.loc_name]['results']['Q'] = Q
                v.elements[elemento.loc_name]['results']['S'] = S
                v.elements[elemento.loc_name]['results']['U'] = U
                v.elements[elemento.loc_name]['results']['cosPhi'] = cosPhi

            if elemento.GetClassName() == 'ElmTerm' and elemento.GetAttribute('e:outserv') == 0:
                v.elements[elemento.loc_name]['results']['Pgen'] = Pgen
                v.elements[elemento.loc_name]['results']['Pload'] = Pload
                v.elements[elemento.loc_name]['results']['Qgen'] = Qgen
                v.elements[elemento.loc_name]['results']['Qload'] = Qload
                v.elements[elemento.loc_name]['results']['U'] = U_elmterm
                v.elements[elemento.loc_name]['results']['Un'] = Un
                v.elements[elemento.loc_name]['results']['Up'] = Up

            if (elemento.GetClassName() == 'ElmTr2' or elemento.GetClassName() == 'ElmTr3' or
                elemento.GetClassName() == 'ElmTr4' or elemento.GetClassName() == 'ElmTrb') and \
                    elemento.GetAttribute('e:outserv') == 0:
                v.elements[elemento.loc_name]['results']['Ploss'] = Ploss_elmTr
                v.elements[elemento.loc_name]['results']['Qloss'] = Qloss_elmTr
                v.elements[elemento.loc_name]['results']['Port1']['I'] = I_elmTr_port1
                v.elements[elemento.loc_name]['results']['Port1']['Iangle'] = Iangle_elmTr_port1
                v.elements[elemento.loc_name]['results']['Port1']['P'] = P_elmTr_port1
                v.elements[elemento.loc_name]['results']['Port1']['Q'] = Q_elmTr_port1
                v.elements[elemento.loc_name]['results']['Port1']['S'] = S_elmTr_port1
                v.elements[elemento.loc_name]['results']['Port1']['U'] = U_elmTr_port1
                v.elements[elemento.loc_name]['results']['Port1']['cosPhi'] = cosPhi_elmTr_port1
                v.elements[elemento.loc_name]['results']['Port2']['I'] = I_elmTr_port2
                v.elements[elemento.loc_name]['results']['Port2']['Iangle'] = Iangle_elmTr_port2
                v.elements[elemento.loc_name]['results']['Port2']['P'] = P_elmTr_port2
                v.elements[elemento.loc_name]['results']['Port2']['Q'] = Q_elmTr_port2
                v.elements[elemento.loc_name]['results']['Port2']['S'] = S_elmTr_port2
                v.elements[elemento.loc_name]['results']['Port2']['U'] = U_elmTr_port2
                v.elements[elemento.loc_name]['results']['Port2']['cosPhi'] = cosPhi_elmTr_port2

            if (elemento.GetClassName() == 'ElmLne' or elemento.GetClassName() == 'ElmDcdc') and \
                    elemento.GetAttribute('e:outserv') == 0:
                v.elements[elemento.loc_name]['results']['Port1']['I'] = I_elmLne_port1
                v.elements[elemento.loc_name]['results']['Port1']['Iangle'] = Iangle_elmLne_port1
                v.elements[elemento.loc_name]['results']['Port1']['P'] = P_elmLne_port1
                v.elements[elemento.loc_name]['results']['Port1']['Q'] = Q_elmLne_port1
                v.elements[elemento.loc_name]['results']['Port1']['S'] = S_elmLne_port1
                v.elements[elemento.loc_name]['results']['Port1']['U'] = U_elmLne_port1
                v.elements[elemento.loc_name]['results']['Port1']['cosPhi'] = cosPhi_elmLne_port1
                v.elements[elemento.loc_name]['results']['Port2']['I'] = I_elmLne_port2
                v.elements[elemento.loc_name]['results']['Port2']['Iangle'] = Iangle_elmLne_port2
                v.elements[elemento.loc_name]['results']['Port2']['P'] = P_elmLne_port2
                v.elements[elemento.loc_name]['results']['Port2']['Q'] = Q_elmLne_port2
                v.elements[elemento.loc_name]['results']['Port2']['S'] = S_elmLne_port2
                v.elements[elemento.loc_name]['results']['Port2']['U'] = U_elmLne_port2
                v.elements[elemento.loc_name]['results']['Port2']['cosPhi'] = cosPhi_elmLne_port2

            if (elemento.GetClassName() == 'ElmVsc' or elemento.GetClassName() == 'ElmVscmono') and \
                    elemento.GetAttribute('e:outserv') == 0:
                v.elements[elemento.loc_name]['results']['Port1']['I'] = I_elmVsc_port1
                v.elements[elemento.loc_name]['results']['Port1']['Iangle'] = Iangle_elmVsc_port1
                v.elements[elemento.loc_name]['results']['Port1']['P'] = P_elmVsc_port1
                v.elements[elemento.loc_name]['results']['Port1']['Q'] = Q_elmVsc_port1
                v.elements[elemento.loc_name]['results']['Port1']['S'] = S_elmVsc_port1
                v.elements[elemento.loc_name]['results']['Port1']['U'] = U_elmVsc_port1
                v.elements[elemento.loc_name]['results']['Port1']['cosPhi'] = cosPhi_elmVsc_port1
                v.elements[elemento.loc_name]['results']['Port2']['I'] = I_elmVsc_port2
                v.elements[elemento.loc_name]['results']['Port2']['Iangle'] = Iangle_elmVsc_port2
                v.elements[elemento.loc_name]['results']['Port2']['P'] = P_elmVsc_port2
                v.elements[elemento.loc_name]['results']['Port2']['Q'] = Q_elmVsc_port2
                v.elements[elemento.loc_name]['results']['Port2']['S'] = S_elmVsc_port2
                v.elements[elemento.loc_name]['results']['Port2']['U'] = U_elmVsc_port2
                v.elements[elemento.loc_name]['results']['Port2']['cosPhi'] = cosPhi_elmVsc_port2

#
def tstep_load(self, h, m):
    if h == 0 and m == 0:
        iTime = 0
    elif h > 0 and h <= 9 and m == 0:
        iTime = '0' + str(h) + str(m) + '000'
    elif h > 9 and m == 0:
        iTime = str(h) + str(m) + '000'
    elif h > 9 and m != 0:
        iTime = str(h) + str(m * 100)
    else:
        iTime = '0' + str(h) + str(m * 100)
    self.oSetTime.SetAttribute('e:cTime', str(iTime))
    self.ldf.Execute()
```

```
#
def cicli_termici(self, Ta):
    start = 0
    stop = 1
    start_cre = 0
    stop_cre = 1
    start_decr = 0
    stop_decr = 1
    cicli_cre = {}
    cicli_decre = {}
    ΔTcycling_cre = []
    ΔTcycling_decre = []
    Tmax_cre = []
    Tmax_decr = []
    i = 1
    j = 1
    trend = [b - a for a, b in zip(Ta[start:stop], Ta[start + 1::1])]
    while stop_decr < len(Ta)  and stop_cre < len(Ta):
        if trend[-1] >= 0 or trend[-1] >= -0.05:
            while trend[-1] >= -0.05 and stop_cre <len(Ta)  and stop_decr < len(Ta):
                if stop_decr > start_decr + 1 and stop_cre < stop_decr:
                    start_cre = stop_decr - 1
                    stop_cre = start_cre + 1

                trend = [b - a for a, b in zip(Ta[start_cre:stop_cre], Ta[start_cre + 1::1])]
                stop_cre = stop_cre + 1
            cicli_cre[i] = (start_cre, stop_cre - 2, stop_cre - start_cre - 1,max(Ta[start_cre:stop_cre - 1]) -
                            min(Ta[start_cre:stop_cre - 1]),max(Ta[start_cre:stop_cre - 1]))
            i = i + 1
        if trend[-1] <= 0 or trend[-1] <= 0.05:
            while trend[-1] <= 0.05 and stop_decr < len(Ta) and stop_cre < len(Ta):
                if stop_cre > start_cre + 1 and stop_decr < stop_cre:
                    start_decr = stop_cre - 1
                    stop_decr = start_decr + 1
                trend = [b - a for a, b in zip(Ta[start_decr:stop_decr], Ta[start_decr + 1::1])]
                stop_decr = stop_decr + 1
            cicli_decre[j] = (start_decr, stop_decr - 2, stop_decr - start_decr - 1,
                              max(Ta[start_decr:stop_decr - 1]) - min(Ta[start_decr:stop_decr - 1]),
                              max(Ta[start_decr:stop_decr - 1]))
            j = j + 1

    cicli_cre_tmp = dict(cicli_cre)
    cicli_decre_tmp = dict(cicli_decre)
    for key, value in cicli_cre_tmp.items():
        if value[3] < 2:  ## non considero i cicli con delta inferiore a 2
            del cicli_cre[key]

    for key, value in cicli_decre_tmp.items():
        if value[3] < 2:  ## non considero i cicli con delta inferiore a 2
            del cicli_decre[key]

    for key, value in cicli_cre.items():
        ΔTcycling_cre.append(value[3])
    ΔTcycling_cre_max = max(ΔTcycling_cre)

    for key, value in cicli_cre.items():
        Tmax_cre.append(value[3])
    Tmax1 = max(Tmax_cre)

    for key, value in cicli_decre.items():
        ΔTcycling_decre.append(value[4])
    ΔTcycling_decre_max = max(ΔTcycling_decre)

    for key, value in cicli_decre.items():
        Tmax_decr.append(value[4])
    Tmax2 = max(Tmax_decr)

    Tmax = max(Tmax1, Tmax2)
    ΔTcycling_max = max(ΔTcycling_cre_max, ΔTcycling_decre_max)
    num_cicli = len(cicli_cre)+len(cicli_decre)     # numero cicli in un giorno

    return num_cicli

#t = ore per il calcolo dell'affidabilità'
def Norris_Landzberg(self, element, t, T0, Ta, alfa, beta, Pi_E, Pi_Q):
    ####CALCOLO AFFIDABILITA' DEL COMPONENTE E SCRIVO IL RISULTATO NEL DIZIONARIO ELEMENTS CAMPO RESULTS
    ##For PF>2019
    ###
    tannual=24*365 #[ore]
    #####SARA' UTILIZZATO PER TUTTI I COMPONENTI
    num_cicli = self.cicli_termici(Ta)
    Nannualcy=num_cicli *365
    #calcolo formula Norris-Landzberg:

    if v.elements[element]['category'] == 'AC-Line' or v.elements[element]['category'] == 'DC-Line':
        # Calcolo affidabilità  linee elettriche  aeree_AG

        #Setto orologio Powerfactory a 0
        self.oSetTime.SetAttribute('e:cTime', "0")

        # Lunghezza linee  #in alternativa valore suggerito in rosso solo per rete benchmark
        #lungh_linea1 = 10  # []
        lungh_linea1 = v.elements[element]['parameters']['length']

        ##La sezione del cavo, se non presente, deve essere settata nel modello di rete e riportata nel dizionario
        # A = Sezione cavi  # in alternativa valore suggerito in rosso solo per rete benchmark
        #A = v.elements[element]['parameters']['sez']
        A=0.000012
        # temperatura massima funzionamento cavo
        TMAXO_linea = 90 ##valore preso da catalogo prysmian

        #t va letto nel dizionario ed inserito dall'utente
        # input: parametri da aggiungere al sottodizionario "parameter" del dizionario "linee"
        #resistività rame-> ipotesi di tutti i cavi in rame
        rho_20 = 1.68 * math.pow(10, -8)

        # calore specifico del materiale [J/Kg°C] del rame
        C = 385
        # densita del materilae cavo Kg/m3cdel rame
        density = 8900
        Karm_linee = 1  # ipotizzo lo stesso valore per tutte le linee, in prima approssimazione pari a 1
        Kis_linee = 1  # ipotizzo lo stesso valore per tutte le linee, in prima approssimazione pari a 1

        Airr = 0.5 * A  # l'area irradiata dalla radiazione solare sia pari alla metà della sezione del cavo
```

```
        Epsilon_isolinea = 0.91  # emissività del PVC: essa diminuisce all'aumentare della temperatura
        Sigma_isolinea = pow(10, -10)  # conducibilità elettrica di un polimero
        Deltai = 0
        # Per ogni linea della rete elettrica considerata:
        # Per ogni timestep:
        # inizio ciclo su timestep
        Top_linea_vector = []

        for line in self.link:
            if line.loc_name==element:
                linea=line
        i = 0
        for h in range(0, int(len(Ta)/4)): #sto ipotizzando un timestep di 15 minuti quindi in un'ora ho 4 valori
            for m in range(0, 60, 15):
                self.tstep_line(h,m, Ta[i]) # setto ora e faccio il ldf
                #calcolo rho [Ωm]
                rho = rho_20 * ((234.5 + Ta[i]) / (234.5 + 20))     # resistenza di ogni linea

                # calcolo a partire da ro lunghezza e sezione cavi e resisitività (temeperatura)
                R_linea = rho * lungh_linea1 / A
                Deltaz = TMAX0_linea - Ta[i]  # ver z-Tamb
                Ic = linea.GetAttribute('m:I:bus1')   # [A]

                Iz = linea.typ_id.sline * 1000  # verificare se devo scrivere elemento.typ.....dizionario

                # costante di tempo termica
                T = C * density * Deltaz * pow((A / Iz), 2) / rho

                Deltac = Deltaz * Karm_linee * Kis_linee * pow((Ic / Iz), 2)
                Deltaf = Deltac - (Deltac - Deltai) * math.exp(-t / T)
                Top_linea = Ta[i] + Deltaf # temperatura operativa linea
                Deltai = Deltaf
                Top_linea_vector.append(Top_linea)
                if i<len(Ta):
                    i = i + 1


        Top_MAX_linea = max(Top_linea_vector)
        Top_min_linea = min(Top_linea_vector)
        Delta_Tcycling = Top_MAX_linea - Top_min_linea
        tannual = 24*365  # [ore]

        m = 1  # in prima approssimazione m=1

        # calcolo formula Norris-Landzberg:
        # Ho inserito To pari a 30° (Prysmian)
        lambda_wear_out = (beta / alfa) * pow((t / alfa), (beta - 1))  ### Weibull [guasti/ore]

        # lambda_prot_linea da inserire nel dizionario delle "linee" e di tutti gli altri elementi delle rete
        lambda_prot_linea = 1 # protezione assente; se protezione presente inserire il valore corretto
        m_line=1
        Pi_Si_linea = (12 * Nannualcy / tannual) * pow((Delta_Tcycling / T0),m_line) * \
                      (math.exp(1414 * ((1 / (323)) - (1 / (Top_MAX_linea + 273.15)))))
        Pi_Se_linea = 0

        Lambda = lambda_prot_linea * lambda_wear_out * (Pi_Si_linea + Pi_Se_linea + Pi_E + Pi_Q)  # [guasti/ore]
        MTBF_ore = (1 / Lambda)  # [ore]
        MTBF_anni = MTBF_ore / 8760  # [anni]
        R_comp = math.exp(-Lambda * t)
        v.elements[element]['reliability']['results']['lambda'] = Lambda
        v.elements[element]['reliability']['results']['MTBF_ore'] = MTBF_ore
        v.elements[element]['reliability']['results']['MTBF_anni'] = MTBF_anni
        v.elements[element]['reliability']['results']['R'] = R_comp

    if v.elements[element]['category'] == 'AC-Load':
        # Calcolo affidabilità  linee elettriche  aeree_AG

        #Setto orologio Powerfactory a 0
        self.oSetTime.SetAttribute('e:cTime', "0")

        # temperatura massima funzionamento load
        TMAX0_LOAD = 80 #[°C]INSERITO DA UTENTE E SCRITTO NEL DIZIONARIO
        # Iz_LOAD l'utente deve inserire il valore di corrente massima (ilini) e lo scaling factor.
        Iz_LOAD= 500 #[A] VERIFICARE UNITA DI MISURA IMPOSTATA NEL MODELLO DI RETE
        Karm_LOAD = 1  # ipotizzo lo stesso valore per tutti carichi, in prima approssimazione pari a 1
        Kis_LOAD = 1  # ipotizzo lo stesso valore per tutti i carichi, in prima approssimazione pari a 1

        Karm_linee = 1  # ipotizzo lo stesso valore per tutti i carichi, in prima approssimazione pari a 1
        Kis_linee = 1  # ipotizzo lo stesso valore per tutti i carichi, in prima approssimazione pari a 1

        Top_LOAD_vector = []

        Deltai_LOAD = 0

        for carico in self.carichi_ac:
            if carico.loc_name==element:
                load_ac=carico
        i = 0
        Rth_LOAD = 1;  # Data la difficoltà nel calcolare questo parametro che dipende da diversi material.
        for h in range(0, int(len(Ta)/4)): #sto ipotizzando un timestep di 15 minuti quindi in un'ora ho 4 valori
            for m in range(0, 60, 15):
                self.tstep_line(h,m, Ta[i]) # setto ora e faccio il ldf
                Ploss_LOAD = load_ac.GetAttribute('c:Ploss')
                Top_LOAD = Ta[i] + Ploss_LOAD * Rth_LOAD
                Top_LOAD_vector.append(Top_LOAD)
                if i<len(Ta):
                    i = i + 1

        Top_MAX_LOAD = max(Top_LOAD_vector)
        Top_min_LOAD = min(Top_LOAD_vector)
        Delta_Tcycling_LOAD = Top_MAX_LOAD - Top_min_LOAD

        num_cicli_load = self.cicli_termici(Top_LOAD_vector)
        Nannualcy_load = num_cicli_load * 365
        tannual = 24*365  # [ore]

        m_LOAD = 1  # in prima approssimazione m=1
        # calcolo formula Norris-Landzberg:
        # Ho inserito To pari a 30° (Prysmian)
        lambda_wear_out = (beta / alfa) * pow((t / alfa), (beta - 1))  ### Weibull [guasti/ore]

        # lambda_prot_load da inserire nel dizionario dei carichi e di tutti gli altri elementi delle rete
        #5.5  # caso pessimo: come  se il cavo fosse realizzatc con componenti di bassa qualità.
        lambda_prot_LOAD = 1  # protezione assente; se protezione presente inserire il valore corretto
        ##DA LEGGERE NEL DIZIONARIO
```

```
            #Pi_E_LOAD = self.Pi_E
            # Pi_Q_LOAD = self.Pi_Q
            Pi_E_LOAD = 4  # Ground AG da tabella fattori Pi nella cartella LA1.10
            Pi_Q_LOAD = 5.5
            m_load=1
            Pi_Si_LOAD = (12 * Nannualcy_load / tannual) * pow((Delta_Tcycling_LOAD / T0),m_load) * \
                         (math.exp(1414 * ((1 / (323)) - (1 / (Top_MAX_LOAD + 273.15)))))
            Pi_Se_LOAD = 0

            Lambda = lambda_prot_LOAD * lambda_wear_out * (Pi_Si_LOAD + Pi_Se_LOAD + Pi_E + Pi_Q)  # [guasti/ore]
            MTBF_ore = (1 / Lambda)  # [ore]
            MTBF_anni = MTBF_ore / 8760  # [anni]
            R_comp = math.exp(-Lambda * t)
            v.elements[element]['reliability']['results']['lambda'] = Lambda
            v.elements[element]['reliability']['results']['MTBF_ore'] = MTBF_ore
            v.elements[element]['reliability']['results']['MTBF_anni'] = MTBF_anni
            v.elements[element]['reliability']['results']['R'] = R_comp

        if v.elements[element]['category'] == 'PWM':
        # INVERTER:aprossimando  lambda_DCAC con lambda della sezione switching:

            lambda0_sw = 0.001
            Tmax_pwm = 150 #°C è la temperatura massima degli switch indicata dai costruttori
            Tmin_pwm = 25 #°C è la temperatura minima degli switch indicata dai costruttori
            ΔTcycling_max=Tmax_pwm - Tmin_pwm
            m_pwm=1
            Pi_S_sw_ON = (12 * Nannualcy / tannual) * pow(ΔTcycling_max/T0,m_pwm) * \
                         (math.exp(1414 * ((1 / (313)) - (1 / (Tmax_pwm + 273.15)))))
            Pi_S_sw_OFF = 0
            lambdaON = lambda0_sw * (Pi_S_sw_ON + Pi_E + Pi_Q)
            lambdaOFF = lambda0_sw * (Pi_S_sw_OFF + Pi_E + Pi_Q)
            Lambda = 3 * lambdaON + 3 * lambdaOFF # [failures / 10^6 hours]
            MTBF_ore = (1 / Lambda) # [10 ^ 6 ore]
            MTBF_anni = MTBF_ore*pow(10,6) / 8760 # [anni]
            R_comp = math.exp(-Lambda * t)
            v.elements[element]['reliability']['results']['lambda'] = Lambda
            v.elements[element]['reliability']['results']['MTBF_ore'] = MTBF_ore
            v.elements[element]['reliability']['results']['MTBF_anni'] = MTBF_anni
            v.elements[element]['reliability']['results']['R'] = R_comp

        if v.elements[element]['category'] == 'PV' or v.elements[element]['category'] == 'DC-DC_Conv' or \
                v.elements[element]['category'] == 'Battery' or v.elements[element]['category'] == 'DC-Wind' or \
                v.elements[element]['category'] == 'DC-Load':

            #####DAB PER I CARICHI DC, PV, BATERIA, eolico DC calcolo formula  Norris - Landzberg:
            lambda0_sw = 0.01
            Tmax=150
            Tmin=20
            ΔTcycling_max = Tmax - Tmin
            m=1
            Pi_S_sw_ON = (12 * Nannualcy / tannual) * pow(ΔTcycling_max/ T0, m) * \
                         (math.exp(1414 * ((1 / (333)) - (1 / (Tmax + 273.15)))))
            Pi_S_sw_OFF = 0
            lambdaON = lambda0_sw* (Pi_S_sw_ON + Pi_E + Pi_Q)
            lambdaOFF = lambda0_sw * (Pi_S_sw_OFF + Pi_E + Pi_Q)
            Lambda = 2 * lambdaON + 2 * lambdaOFF
            MTBF_ore = (1 / Lambda) # [10 ^ 6 ore]
            MTBF_anni = MTBF_ore*pow(10,6)/ 8760  # [anni]
            R_comp = math.exp(-Lambda * t)
            v.elements[element]['reliability']['results']['lambda'] = Lambda
            v.elements[element]['reliability']['results']['MTBF_ore'] = MTBF_ore
            v.elements[element]['reliability']['results']['MTBF_anni'] = MTBF_anni
            v.elements[element]['reliability']['results']['R'] = R_comp

        if v.elements[element]['category'] == '2W-Transformer':
            ##TRASFORMATORE
            Tmax_tr=110 #°C dalla normativa dei trasformatori vedere LA1.1 par. 4.1
            Tmin_tr=30 #°C dalla normativa dei trasformatori vedere LA1.1 par. 4.1
            ΔTcycling_max = Tmax_tr - Tmin_tr
            lambda_wear_out = (beta / alfa) * pow((t / alfa), (beta - 1))  ### Weibull [guasti/ore]
            m_tr=1
            Pi_S_sw_ON = (12 * Nannualcy / tannual) * pow(ΔTcycling_max / T0, m_tr) * \
                         (math.exp(1414 * ((1 / (323)) - (1 / (Tmax_tr + 273.15)))))
            Pi_S_sw_OFF = 0
            lambdaON = lambda_wear_out * (Pi_S_sw_ON + Pi_E + Pi_Q)  # [guasti/ore]
            Lambda = lambdaON  # lambdaOFF #[guasti/ore]
            MTBF_ore = (1 / Lambda)  # [ore]
            MTBF_anni = MTBF_ore / 8760  # [ anni]
            R_comp = math.exp(-Lambda * t / beta)
            lambda_wear_out = (beta / alfa) * pow((t / alfa), (beta - 1))  ### Weibull [guasti/ore]
            Pi_S_sw_ON=(12*Nannualcy/tannual) * pow(ΔTcycling_max/T0,m_tr) * \
                       (math.exp(1414*((1/(323)) - (1/(Tmax_tr+273.15)))))
            Pi_S_sw_OFF = 0
            lambdaON = lambda_wear_out*(Pi_S_sw_ON+ Pi_E+ Pi_Q)  #[guasti/ore]
            Lambda= lambdaON # lambdaOFF #[guasti/ore]
            MTBF_ore= (1/Lambda) #[ore]
            MTBF_anni= MTBF_ore/8760 #[ anni]
            R_comp=math.exp(-Lambda*t/beta)
            v.elements[element]['reliability']['results']['lambda'] = Lambda
            v.elements[element]['reliability']['results']['MTBF_ore'] = MTBF_ore
            v.elements[element]['reliability']['results']['MTBF_anni'] = MTBF_anni
            v.elements[element]['reliability']['results']['R'] = R_comp

#
def grafo_rete(self):
    ####creo il grafo a partire dal modello di rete e calcolo tutti
    ### i path tra generatori e carichi#####

    self.tuple_list = []  # contiene gli elementi connessi ai cubicoli
    self.path = []
    self.path_tmp = []
    self.graph = nx.Graph()
    self.cubics = self.app.GetCalcRelevantObjects('StaCubic')
    self.generatori = self.app.GetCalcRelevantObjects('ElmGenstat') + self.app.GetCalcRelevantObjects('ElmDci')
    self.carichi_ac = self.app.GetCalcRelevantObjects('*.ElmLod') + self.app.GetCalcRelevantObjects('*.ElmLodmv') +\
                      self.app.GetCalcRelevantObjects('*.ElmLodlv')
    self.carichi_dc = self.app.GetCalcRelevantObjects('*.ElmLoddc') + \
                      self.app.GetCalcRelevantObjects('*.ElmLoddcbi')
    self.carichi = self.carichi_ac + self.carichi_dc
    self.ldf = self.app.GetFromStudyCase('ComLdf')
    self.ldf.Execute()
    for cubic in self.cubics:
        if cubic.GetConnections(0):
            if len(cubic.GetConnections(1)) > 1:
                self.tuple_list.append([cubic.GetConnections(1)[0], cubic.GetConnections(1)[1]])
```

```
                self.graph.add_edge(cubic.GetConnections(1)[0], cubic.GetConnections(1)[1])

    #
    def RBD(self, time):
        self.id_obj = {}  # contiene gli id dei componenti
        self.extend_list = []
        self.last_node = []
        self.first_node = []
        self.reliability_loads = {}
        self.Rel = []

        t = Symbol('t', positive=True)  # t in ore
        timerange = range(0, 1000, 1)

        for carico in self.carichi:
            print(carico.loc_name)
            self.first_node = []
            if carico in self.graph.nodes() and carico.loc_name not in self.reliability_loads.keys():
                print('inizio ' + carico.loc_name + '...')
                self.S = System()
                self.path = []
                self.path_tmp = []
                self.blocchi = []
                for generatore in self.generatori:
                    if generatore in self.graph.nodes() and generatore.GetAttribute('e:outserv') == 0:
                        self.path.append(nx.shortest_path(self.graph, generatore, carico))

                self.path_tmp = self.path.copy()
                self.blocchi = self.path.copy()  # blocchi del RDB
                for i in range(0, len(self.path)):
                    for j in range(0, len(self.path[i])):
                        self.path_tmp[i][j] = self.path[i][j].loc_name

                for i in range(0, len(self.path)):
                    self.blocchi[i] = [Component(self.path_tmp[i][j],
                                                 v.elements[self.path_tmp[i][j]]['reliability']['results']['lambda'],
                                                 v.elements[self.path_tmp[i][j]]['reliability']['beta'])
                                       for j in range(0, len(self.path_tmp[i]))]

                # creo dizionario con gli id dei componenti
                for componente in self.blocchi:
                    for k in range(0, len(componente)):
                        self.id_obj[id(componente[k])] = componente[k]
                for i in range(0, len(self.blocchi)):
                    if i == 0:
                        self.S['E'] = self.blocchi[i][0]
                        self.first_node.append(self.blocchi[i][0].name)
                    if i > 0 and self.blocchi[i][0].name not in self.first_node:
                        self.S['E'] = self.blocchi[i][0]
                        self.first_node.append(self.blocchi[i][0].name)
                        # self.S['E'] = self.blocchi[i][0]
                    num_nodi = len(self.blocchi[i]) - 1
                    chek1 = 'false'
                    chek2 = 'false'
                    for j in range(0, num_nodi):
                        for componente in self.S.components:
                            if self.blocchi[i][j + 1].name == componente.name and id(self.blocchi[i][j + 1]) != id(
                                    componente):
                                comp_succ = componente
                                chek1 = 'true'

                            if self.blocchi[i][j].name == componente.name and id(self.blocchi[i][j]) != id(componente):
                                if i > 0 and j == 0:
                                    self.S['E'] = componente
                                comp_prec = componente
                                chek2 = 'true'

                        if chek1 == 'false' and chek2 == 'false':
                            self.S[self.blocchi[i][j]] = self.blocchi[i][j + 1]
                        if chek1 == 'true' and chek2 == 'false':
                            self.S[self.blocchi[i][j]] = comp_succ
                            chek1 = 'false'
                        if chek1 == 'false' and chek2 == 'true':
                            self.S[comp_prec] = self.blocchi[i][j + 1]
                            chek2 = 'false'
                        if chek1 == 'true' and chek2 == 'true':
                            if comp_prec != comp_succ:
                                self.S[comp_prec] = comp_succ
                            chek1 = 'false'
                            chek2 = 'false'
                    if i == 0:
                        self.S[self.blocchi[0][len(self.blocchi[0]) - 1]] = 'S'
                        self.last_node.append(self.blocchi[0][len(self.blocchi[0]) - 1].name)
                    if i > 0 and self.blocchi[i][len(self.blocchi[i]) - 1].name not in self.last_node:
                        self.S[self.blocchi[i][len(self.blocchi[i]) - 1]] = 'S'
                        self.last_node.append(self.blocchi[i][len(self.blocchi[i]) - 1].name)

                plt.figure(figsize=(15, 10))
                plt.tick_params(labelsize=3)
                plt.rc('font', size=1)

                self.S.draw()
                filename = os.path.join(v.project_folder, 'reliability', 'img',
                                        carico.loc_name + '.png')

                plt.savefig(filename)
                reliability = self.S.reliability(time)
                v.elements[carico.loc_name]['reliability']['results']['load_rel'] = float(reliability)

    #
    def results(self, hour):
        results_dict = dict()   # forse non necessario
        self.instantLF(hour*60)

        filename = os.path.join(v.project_folder, 'elements.yml')

        with open(filename, 'w') as file:
            documents = yaml.dump(v.elements, file)
        file.close()
        return results_dict

    #
    def result_RDB(self):
        filename = os.path.join(v.project_folder, 'features.yml')
```

```
#
def result(self):
    # Esegue il loadflow. Popola il sotto-dizionario v.elements[element]['results'] con i risultati del LoadFlow.
    filename = os.path.join(v.project_folder, 'elements.yml')

    with open(filename, 'w') as file:
        documents = yaml.dump(v.elements, file)
    file.close()
```

4.4 UI

4.4.1 Main

4.4.1.1 UI.py

```
from PyQt5 import QtGui, QtWidgets
from .MainUI import Ui_form


class UI(QtWidgets.QMainWindow):
    def __init__(self):
        super(UI, self).__init__()
        self.ui = Ui_form()
        self.ui.setupUi(self)

    #
    def table_format(self):
        self.ui.tableWidget.setShowGrid(False)
        self.ui.tableWidget.setStyleSheet('QTableView::item {border-top: 1px solid #333333;}')
        self.ui.tableWidget.verticalHeader().setVisible(False)

        stylesheet = \
            "QHeaderView::section{color:rgb(251,251,251); Background-color:rgb(1,1,1); border - radius: 14 px;}"
        self.ui.tableWidget.horizontalHeader().setStyleSheet(stylesheet)

        for i in range(0, self.ui.tableWidget.rowCount()):
            for j in range(0, self.ui.tableWidget.columnCount()):
                self.ui.tableWidget.item(i, j).setForeground(QtGui.QColor(255, 255, 255))

        self.ui.tableWidget.setColumnWidth(0, 160)
        self.ui.tableWidget.setColumnWidth(1, 150)

    #
    def azione(self):
        self.ui.tableWidget.setRowCount(30)
        for i in range(0, self.ui.tableWidget.rowCount()):
            self.ui.tableWidget.setItem(i, 0, QtWidgets.QTableWidgetItem('Item ' + str(i)))
            self.ui.tableWidget.setItem(i, 1, QtWidgets.QTableWidgetItem('Tipo ' + str(i)))
            self.ui.tableWidget.item(i, 0).setForeground(QtGui.QColor(255, 255, 255))
            self.ui.tableWidget.item(i, 1).setForeground(QtGui.QColor(255, 255, 255))

        self.ui.tableWidget.setShowGrid(False)
        self.ui.tableWidget.setStyleSheet('QTableView::item {border-top: 1px solid #333333;}')

        self.ui.tableWidget.verticalHeader().setVisible(False)
        stylesheet = "QHeaderView::section{Background-color:rgb(1,1,1); border - radius: 14 px;}"

        self.ui.tableWidget.horizontalHeader().setStyleSheet(stylesheet)
        self.ui.tableWidget.horizontalHeaderItem(0).setForeground(QtGui.QColor(255, 255, 255))
        self.ui.tableWidget.horizontalHeaderItem(1).setForeground(QtGui.QColor(255, 255, 255))
```

4.4.1.2 MainUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'MainUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.4
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_form(object):
    def setupUi(self, form):
        form.setObjectName("form")
        form.resize(1800, 1068)
        font = QtGui.QFont()
        font.setPointSize(10)
        form.setFont(font)
        form.setStyleSheet("background-color: rgb(0, 0, 7);")
        self.centralwidget = QtWidgets.QWidget(form)
        self.centralwidget.setObjectName("centralwidget")
        self.tableWidget = QtWidgets.QTableWidget(self.centralwidget)
        self.tableWidget.setGeometry(QtCore.QRect(10, 140, 601, 481))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.tableWidget.setFont(font)
        self.tableWidget.setStyleSheet("")
        self.tableWidget.setFrameShape(QtWidgets.QFrame.NoFrame)
        self.tableWidget.setLineWidth(1)
        self.tableWidget.setMidLineWidth(0)
        self.tableWidget.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)
        self.tableWidget.setObjectName("tableWidget")
        self.tableWidget.setColumnCount(2)
        self.tableWidget.setRowCount(2)
        item = QtWidgets.QTableWidgetItem()
        self.tableWidget.setVerticalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.tableWidget.setVerticalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        self.tableWidget.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.tableWidget.setHorizontalHeaderItem(1, item)
        self.line = QtWidgets.QFrame(self.centralwidget)
        self.line.setGeometry(QtCore.QRect(10, 130, 601, 1))
        self.line.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.line.setFrameShape(QtWidgets.QFrame.HLine)
        self.line.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line.setObjectName("line")
        self.line_2 = QtWidgets.QFrame(self.centralwidget)
        self.line_2.setGeometry(QtCore.QRect(10, 630, 601, 1))
        self.line_2.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.line_2.setFrameShape(QtWidgets.QFrame.HLine)
        self.line_2.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line_2.setObjectName("line_2")
        self.gridname_LBL = QtWidgets.QLabel(self.centralwidget)
        self.gridname_LBL.setGeometry(QtCore.QRect(20, 20, 581, 31))
        font = QtGui.QFont()
        font.setPointSize(16)
        font.setBold(True)
        font.setItalic(False)
        font.setWeight(75)
        self.gridname_LBL.setFont(font)
        self.gridname_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.gridname_LBL.setObjectName("gridname_LBL")
        self.line_3 = QtWidgets.QFrame(self.centralwidget)
        self.line_3.setGeometry(QtCore.QRect(10, 10, 601, 1))
        self.line_3.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.line_3.setFrameShape(QtWidgets.QFrame.HLine)
        self.line_3.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line_3.setObjectName("line_3")
        self.line_4 = QtWidgets.QFrame(self.centralwidget)
        self.line_4.setGeometry(QtCore.QRect(10, 60, 1840, 1))
        self.line_4.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.line_4.setFrameShape(QtWidgets.QFrame.HLine)
        self.line_4.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line_4.setObjectName("line_4")
        self.gridname_LBL_2 = QtWidgets.QLabel(self.centralwidget)
        self.gridname_LBL_2.setGeometry(QtCore.QRect(10, 110, 601, 20))
        self.gridname_LBL_2.setObjectName("gridname_LBL_2")
        self.saveParams_BTN = QtWidgets.QPushButton(self.centralwidget)
        self.saveParams_BTN.setGeometry(QtCore.QRect(490, 650, 111, 23))
        self.saveParams_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.saveParams_BTN.setObjectName("saveParams_BTN")
        self.stackedWidget = QtWidgets.QStackedWidget(self.centralwidget)
        self.stackedWidget.setGeometry(QtCore.QRect(690, 130, 491, 501))
        self.stackedWidget.setStyleSheet("color: rgb(255, 255, 255);")
        self.stackedWidget.setObjectName("stackedWidget")
        self.page = QtWidgets.QWidget()
        self.page.setObjectName("page")
        self.stackedWidget.addWidget(self.page)
        self.ilf_BTN = QtWidgets.QPushButton(self.centralwidget)
        self.ilf_BTN.setGeometry(QtCore.QRect(30, 790, 191, 23))
        self.ilf_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.ilf_BTN.setObjectName("ilf_BTN")
        self.profile_SW = QtWidgets.QStackedWidget(self.centralwidget)
        self.profile_SW.setGeometry(QtCore.QRect(1190, 130, 601, 501))
        self.profile_SW.setStyleSheet("color: rgb(255, 255, 255);")
        self.profile_SW.setObjectName("profile_SW")
        self.prof_page = QtWidgets.QWidget()
        self.prof_page.setObjectName("prof_page")
        self.profile_SW.addWidget(self.prof_page)
        self.plf_BTN = QtWidgets.QPushButton(self.centralwidget)
        self.plf_BTN.setGeometry(QtCore.QRect(340, 980, 81, 23))
        self.plf_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.plf_BTN.setObjectName("plf_BTN")
        self.ems_BTN = QtWidgets.QPushButton(self.centralwidget)
        self.ems_BTN.setGeometry(QtCore.QRect(520, 980, 81, 23))
```

```
        self.ems_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.ems_BTN.setObjectName("ems_BTN")
        self.results_WG = QtWidgets.QStackedWidget(self.centralwidget)
        self.results_WG.setGeometry(QtCore.QRect(690, 690, 1100, 321))
        self.results_WG.setStyleSheet("")
        self.results_WG.setObjectName("results_WG")
        self.results_WGPage1 = QtWidgets.QWidget()
        self.results_WGPage1.setObjectName("results_WGPage1")
        self.results_WG.addWidget(self.results_WGPage1)
        self.log_TBR = QtWidgets.QTextBrowser(self.centralwidget)
        self.log_TBR.setGeometry(QtCore.QRect(250, 780, 351, 181))
        self.log_TBR.setStyleSheet("color: rgb(255, 255, 255);")
        self.log_TBR.setObjectName("log_TBR")
        self.log_LBL = QtWidgets.QLabel(self.centralwidget)
        self.log_LBL.setGeometry(QtCore.QRect(260, 755, 331, 25))
        self.log_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.log_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
        self.log_LBL.setObjectName("log_LBL")
        self.log_dwnLN = QtWidgets.QFrame(self.centralwidget)
        self.log_dwnLN.setGeometry(QtCore.QRect(240, 970, 371, 1))
        self.log_dwnLN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.log_dwnLN.setFrameShape(QtWidgets.QFrame.HLine)
        self.log_dwnLN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.log_dwnLN.setObjectName("log_dwnLN")
        self.log_topLN = QtWidgets.QFrame(self.centralwidget)
        self.log_topLN.setGeometry(QtCore.QRect(240, 750, 371, 1))
        self.log_topLN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.log_topLN.setFrameShape(QtWidgets.QFrame.HLine)
        self.log_topLN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.log_topLN.setObjectName("log_topLN")
        self.reliability_BTN = QtWidgets.QPushButton(self.centralwidget)
        self.reliability_BTN.setGeometry(QtCore.QRect(250, 980, 81, 23))
        self.reliability_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.reliability_BTN.setObjectName("reliability_BTN")
        self.ems_BTN_2 = QtWidgets.QPushButton(self.centralwidget)
        self.ems_BTN_2.setGeometry(QtCore.QRect(430, 980, 81, 23))
        self.ems_BTN_2.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.ems_BTN_2.setObjectName("ems_BTN_2")
        self.calculate_BTN = QtWidgets.QPushButton(self.centralwidget)
        self.calculate_BTN.setGeometry(QtCore.QRect(30, 820, 191, 51))
        self.calculate_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.calculate_BTN.setObjectName("calculate_BTN")
        self.temp_WGT = QtWidgets.QWidget(self.centralwidget)
        self.temp_WGT.setGeometry(QtCore.QRect(10, 690, 601, 40))
        self.temp_WGT.setObjectName("temp_WGT")
        self.temp_prof_LBL = QtWidgets.QLabel(self.temp_WGT)
        self.temp_prof_LBL.setGeometry(QtCore.QRect(140, 0, 331, 30))
        font = QtGui.QFont()
        font.setPointSize(9)
        font.setBold(True)
        font.setWeight(75)
        self.temp_prof_LBL.setFont(font)
        self.temp_prof_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.temp_prof_LBL.setObjectName("temp_prof_LBL")
        self.temp_prof_canc_BTN = QtWidgets.QPushButton(self.temp_WGT)
        self.temp_prof_canc_BTN.setGeometry(QtCore.QRect(480, 4, 51, 23))
        self.temp_prof_canc_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.temp_prof_canc_BTN.setObjectName("temp_prof_canc_BTN")
        self.temp_prof_upLN = QtWidgets.QFrame(self.temp_WGT)
        self.temp_prof_upLN.setGeometry(QtCore.QRect(0, 0, 601, 1))
        self.temp_prof_upLN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.temp_prof_upLN.setFrameShape(QtWidgets.QFrame.HLine)
        self.temp_prof_upLN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.temp_prof_upLN.setObjectName("temp_prof_upLN")
        self.temp_prof_BTN = QtWidgets.QPushButton(self.temp_WGT)
        self.temp_prof_BTN.setGeometry(QtCore.QRect(540, 4, 51, 23))
        self.temp_prof_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.temp_prof_BTN.setObjectName("temp_prof_BTN")
        self.temp_prof_CAP_LBL = QtWidgets.QLabel(self.temp_WGT)
        self.temp_prof_CAP_LBL.setGeometry(QtCore.QRect(10, 0, 121, 30))
        font = QtGui.QFont()
        font.setPointSize(9)
        font.setItalic(True)
        self.temp_prof_CAP_LBL.setFont(font)
        self.temp_prof_CAP_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.temp_prof_CAP_LBL.setObjectName("temp_prof_CAP_LBL")
        self.temp_prof_dwnLN = QtWidgets.QFrame(self.temp_WGT)
        self.temp_prof_dwnLN.setGeometry(QtCore.QRect(0, 30, 601, 1))
        self.temp_prof_dwnLN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.temp_prof_dwnLN.setFrameShape(QtWidgets.QFrame.HLine)
        self.temp_prof_dwnLN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.temp_prof_dwnLN.setObjectName("temp_prof_dwnLN")
        self.temp_prof_LBL.raise_()
        self.temp_prof_canc_BTN.raise_()
        self.temp_prof_BTN.raise_()
        self.temp_prof_CAP_LBL.raise_()
        self.temp_prof_dwnLN.raise_()
        self.temp_prof_upLN.raise_()
        self.multiplelogics_WGT = QtWidgets.QWidget(self.centralwidget)
        self.multiplelogics_WGT.setGeometry(QtCore.QRect(20, 890, 211, 111))
        self.multiplelogics_WGT.setStyleSheet("color: rgb(255, 255, 255);")
        self.multiplelogics_WGT.setObjectName("multiplelogics_WGT")
        self.label = QtWidgets.QLabel(self.multiplelogics_WGT)
        self.label.setGeometry(QtCore.QRect(60, 0, 91, 20))
        self.label.setAlignment(QtCore.Qt.AlignCenter)
        self.label.setObjectName("label")
        self.ml_top_LN = QtWidgets.QFrame(self.multiplelogics_WGT)
        self.ml_top_LN.setGeometry(QtCore.QRect(0, 10, 210, 1))
        self.ml_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ml_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ml_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ml_top_LN.setObjectName("ml_top_LN")
        self.ml_bottom_LN = QtWidgets.QFrame(self.multiplelogics_WGT)
        self.ml_bottom_LN.setGeometry(QtCore.QRect(0, 110, 210, 1))
        self.ml_bottom_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ml_bottom_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ml_bottom_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ml_bottom_LN.setObjectName("ml_bottom_LN")
        self.lm_reliability_BTN = QtWidgets.QPushButton(self.multiplelogics_WGT)
```

```
        self.lm_reliability_BTN.setGeometry(QtCore.QRect(10, 30, 191, 31))
        self.lm_reliability_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.lm_reliability_BTN.setObjectName("lm_reliability_BTN")
        self.lm_protections_BTN = QtWidgets.QPushButton(self.multiplelogics_WGT)
        self.lm_protections_BTN.setGeometry(QtCore.QRect(10, 70, 191, 31))
        self.lm_protections_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.lm_protections_BTN.setObjectName("lm_protections_BTN")
        self.ml_top_LN.raise_()
        self.label.raise_()
        self.ml_bottom_LN.raise_()
        self.lm_reliability_BTN.raise_()
        self.lm_protections_BTN.raise_()
        self.pdf_BTN = QtWidgets.QPushButton(self.centralwidget)
        self.pdf_BTN.setGeometry(QtCore.QRect(30, 750, 191, 23))
        self.pdf_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.pdf_BTN.setObjectName("pdf_BTN")
        self.log_LBL.raise_()
        self.tableWidget.raise_()
        self.line.raise_()
        self.line_2.raise_()
        self.gridname_LBL.raise_()
        self.line_3.raise_()
        self.line_4.raise_()
        self.gridname_LBL_2.raise_()
        self.saveParams_BTN.raise_()
        self.stackedWidget.raise_()
        self.ilf_BTN.raise_()
        self.profile_SW.raise_()
        self.plf_BTN.raise_()
        self.ems_BTN.raise_()
        self.results_WG.raise_()
        self.log_TBR.raise_()
        self.log_dwnLN.raise_()
        self.log_topLN.raise_()
        self.reliability_BTN.raise_()
        self.ems_BTN_2.raise_()
        self.calculate_BTN.raise_()
        self.temp_WGT.raise_()
        self.multiplelogics_WGT.raise_()
        self.pdf_BTN.raise_()
        form.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(form)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1800, 22))
        self.menubar.setObjectName("menubar")
        form.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(form)
        self.statusbar.setObjectName("statusbar")
        form.setStatusBar(self.statusbar)

        self.retranslateUi(form)
        QtCore.QMetaObject.connectSlotsByName(form)

    def retranslateUi(self, form):
        _translate = QtCore.QCoreApplication.translate
        form.setWindowTitle(_translate("form", "ORAtool - Optimization and Reliability Assessment Tool"))
        self.tableWidget.setSortingEnabled(True)
        item = self.tableWidget.verticalHeaderItem(0)
        item.setText(_translate("form", "1"))
        item = self.tableWidget.verticalHeaderItem(1)
        item.setText(_translate("form", "2"))
        item = self.tableWidget.horizontalHeaderItem(0)
        item.setText(_translate("form", "Item"))
        item = self.tableWidget.horizontalHeaderItem(1)
        item.setText(_translate("form", "Col2"))
        self.gridname_LBL.setText(_translate("form", "Grid name"))
        self.gridname_LBL_2.setText(_translate("form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600; font-
style:italic; color:#ffffff;\">Elenco dei componenti</span></p></body></html>"))
        self.saveParams_BTN.setText(_translate("form", "Salva i parametri"))
        self.ilf_BTN.setText(_translate("form", "LoadFlow"))
        self.plf_BTN.setText(_translate("form", "Profile LoadFlow"))
        self.ems_BTN.setText(_translate("form", "EMS"))
        self.log_TBR.setHtml(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8.25pt; font-weight:400; font-style:normal;\">\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-
indent:0px;\"><br /></p></body></html>"))
        self.log_LBL.setText(_translate("form", "Log"))
        self.reliability_BTN.setText(_translate("form", "Calcolo dell\'Affidabilità"))
        self.ems_BTN_2.setText(_translate("form", "Calcolo degli Indici"))
        self.calculate_BTN.setText(_translate("form", "Calcolo"))
        self.temp_prof_LBL.setText(_translate("form", "Temperature Profile"))
        self.temp_prof_canc_BTN.setText(_translate("form", "Cancella"))
        self.temp_prof_BTN.setText(_translate("form", "Vedi"))
        self.temp_prof_CAP_LBL.setText(_translate("form", "Profilo di Temperatura:"))
        self.label.setText(_translate("form", "Aggiungi logica"))
        self.lm_reliability_BTN.setText(_translate("form", "Calcolo dell\'Affidabilità"))
        self.lm_protections_BTN.setText(_translate("form", "Protezioni"))
        self.pdf_BTN.setText(_translate("form", "Report (PDF)"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    form = QtWidgets.QMainWindow()
    ui = Ui_form()
    ui.setupUi(form)
    form.show()
    sys.exit(app.exec_())
```

4.4.2 Controls

4.4.2.1 contrlols.py

```
import os
from Functionalities.Controls.residential import Residential
from Functionalities.Controls.roadservices import RoadServices
from Functionalities.Controls.underground import Underground
from Functionalities.Controls.port import Port
from __shared__ import variables as v
from datetime import datetime
import time
import matplotlib.pyplot as plt
import yaml
from .controlsUI import Ui_form
from PyQt5 import QtWidgets, QtGui, QtCore

t = time.localtime()
now = datetime.now()
path_img = os.getcwd() + '/_images/Controls/'
print('Path ' + path_img + ' exists? ' + str(os.path.exists(path_img)))
path = path_img + '/Scheme'
path_data = os.getcwd() + '/_functionalities/Controls/'


class Controls(QtWidgets.QMainWindow):
    def __init__(self):
        super(Controls, self).__init__()
        self.ui = Ui_form()
        self.ui.setupUi(self)

        self.area = None
        self.scenario = None
        self.cust_scen = False
        self.scen_ready = False
        self.file = None

        self.ui.scenarios_SW.setVisible(False)
        self.ui.calculate_BTN.setVisible(False)
        self.ui.results_WGT.setVisible(False)
        self.ui.gridname_LBL.setText(v.features['name'])

        self.ui.underground_PB.clicked.connect(self.selected_underground)
        self.ui.roadserv_PB.clicked.connect(self.selected_roadservices)
        self.ui.residential_PB.clicked.connect(self.selected_residential)
        self.ui.back_BTN.clicked.connect(self.area_reset)
        self.ui.ev_fix_CB.clicked.connect(self.ev_fix)
        self.ui.backup_CB.clicked.connect(self.backup_lines)

        self.ui.multiplelogics_WGT.setVisible(v.features['pf']['exists'] or v.features['neplan']['exists'])
        self.ui.lm_reliability_BTN.setVisible(v.features['pf']['exists'])
        self.ui.lm_protections_BTN.setVisible(v.features['pf']['exists'] or v.features['neplan']['exists'])
        self.ui.lm_reliability_BTN.clicked.connect(self.ml_reliability)
        self.ui.lm_protections_BTN.clicked.connect(self.ml_protections)
        self.ui.pdf_BTN.clicked.connect(self.pdf_gen)

        self.ui.standard_scen_BTN.clicked.connect(self.scenario_standard)
        self.ui.custom_scen_BTN.clicked.connect(self.scenario_custom)
        for rb in ['y2020', 'y2030', 'y2040', 'bc', 'dec']:
            self.ui.__getattribute__(rb + '_RB').clicked.connect(self.scenario_check)

        self.ui.calculate_BTN.clicked.connect(self.calculate)

        self.area_reset()

    def selected_underground(self):
        self.area = 'UG'
        self.area_selected('Settore Metropolitana')
        self.ui.log_TBr.setText('Settore Metropolitana')
        self.ui.ev_fix_CB.setVisible(False)
        self.ui.backup_CB.setVisible(True)
        self.ui.backup_WGT.setVisible(True)
        self.ui.p_backup_DSB.setValue(30)

    def selected_roadservices(self):
        self.area = 'RS'
        self.area_selected('Settore Servizi Stradali')
        self.ui.log_TBr.setText('Settore Servizi Stradali')
        self.ui.ev_fix_CB.setVisible(True)
        self.ev_fix()
        self.ui.backup_CB.setVisible(True)
        self.ui.backup_WGT.setVisible(True)
        self.ui.p_backup_DSB.setValue(10)

    def selected_residential(self):
        self.area = 'RES'
        self.area_selected('Settore Residenziale')
        self.ui.log_TBr.setText('Settore Residenziale')
        self.ui.ev_fix_CB.setVisible(False)
        self.ui.backup_CB.setVisible(True)
        self.ui.backup_WGT.setVisible(True)
        self.ui.p_backup_DSB.setValue(100)

    def area_reset(self):
        self.ui.backup_CB.setVisible(False)
        self.ui.backup_WGT.setVisible(False)
        self.ui.backup_CB.setVisible(v.features['name'] != 'City Area')
        self.ui.backup_WGT.setVisible(v.features['name'] != 'City Area')
        self.area_buttons(v.features['name'] == 'City Area')
        if v.features['name'] == 'City Area':
            self.ui.area_img_LBL.setPixmap(QtGui.QPixmap(path + '/city_area_801x466.png'))
            self.ui.area_LBL.setText("Seleziona il settore nello schema ")
        else:
            self.ui.area_img_LBL.setPixmap(QtGui.QPixmap(path + '/port_area_801x404.png'))
            self.ui.area_LBL.setText("")
            self.ui.scenarios_SW.setVisible(True)
            self.area = 'PORT'

        self.scenario_reset()
        self.ui.results_WGT.setVisible(False)
        self.ui.log_TBr.clear()
```

```
def area_selected(self, area):
    self.ui.area_LBL.setText(area + ' ')
    self.ui.area_img_LBL.setPixmap(QtGui.QPixmap(path + '/' + self.area + '.png'))
    self.area_buttons(False)
    self.ui.scenarios_SW.setVisible(True)
    self.scenario_check()

def area_buttons(self, visible):
    for btn in ['residential', 'roadserv', 'underground']:
        self.ui.__getattribute__(btn + '_PB').setVisible(visible)

def scenario_reset(self):
    self.scenario_standard()
    self.ui.scenarios_SW.setVisible(v.features['name'] == 'Port Area')
    self.ui.ev_fix_CB.setVisible(v.features['name'] == 'Port Area')
    self.ui.calculate_BTN.setVisible(False)

def scenario_standard(self):
    self.cust_scen = False
    self.scen_ready = False
    self.ui.results_WGT.setVisible(False)
    self.ui.scenarios_SW.setCurrentIndex(1)
    self.scenario_check()

def scenario_custom(self):
    self.scen_ready = False
    self.cust_scen = True
    self.ui.results_WGT.setVisible(False)
    self.ui.scenarios_SW.setCurrentIndex(0)
    areas = ['RES', 'RS', 'UG', 'PORT']
    self.ui.cust_scen_SW.setCurrentIndex(areas.index(self.area))
    self.ui.calculate_BTN.setVisible(True)

def scenario_check(self):
    self.ui.results_WGT.setVisible(False)
    year = ''
    conf = ''
    self.ui.y2030_RB.setVisible(self.area != 'UG')
    if self.ui.y2020_RB.isChecked():
        year = '2020'
    elif self.ui.y2030_RB.isChecked():
        year = '2030'
    elif self.ui.y2040_RB.isChecked():
        year = '2040'

    self.ui.config_WGT.setVisible(year != '' and year != '2020' and self.area != 'UG')

    if year == '2020' or self.area == 'UG':
        conf = ''
    elif self.ui.bc_RB.isChecked():
        conf = 'BC'
    elif self.ui.dec_RB.isChecked():
        conf = 'DEC'

    self.scenario = 'scen_' + year + conf

    self.scen_ready = (year != '' and (conf != '' or self.area == 'UG')) or year == '2020'
    self.ui.calculate_BTN.setVisible(self.scen_ready)

def ev_fix(self):
    if self.ui.ev_fix_CB.isChecked():
        self.ui.nv_rs_DSB.setEnabled(False)
        self.ui.nv_rs_DSB.setValue(10)
    else:
        self.ui.nv_rs_DSB.setEnabled(True)
    self.ui.results_WGT.setVisible(False)

def backup_lines(self):
    self.ui.backup_WGT.setVisible(self.ui.backup_CB.isChecked())

def calculate(self):
    self.ui.results_WGT.setVisible(False)
    scen = []
    dsb_list = []
    software = None
    log = ''
    self.ui.log_TBr.clear()

    if self.area == 'RES':
        dsb_list = ['ccr', 'pcr', 'ppv', 'pwf', 'pess', 'cess', 'nv', 'nphev', 'nev1', 'nev2', 'pphev', 'pev1',
                    'pev2', 'dr']
        software = Residential()
    elif self.area == 'RS':
        dsb_list = ['pc_rs', 'cc_rs', 'ppv_rs', 'pess_rs', 'cess_rs', 'nv_rs', 'pev1_rs', 'pev2_rs', 'cev_rs']
        software = RoadServices()
    elif self.area == 'UG':
        dsb_list = ['cm_ug', 'ppv_ug', 'pess_ug', 'cess_ug']
        software = Underground()
    elif v.features['name'] == 'Port Area':
        dsb_list = ['c_port', 'pc_port', 'cc_port', 'ppv_port', 'pwf_port', 'pcold_port', 'ccold_port', 'ncol_port',
                    'nev1_port', 'nev2_port', 'pev1_port', 'pev2_port', 'cev_port']
        software = Port()

    for dsb in dsb_list:
        scen.append(self.ui.__getattribute__(dsb + '_DSB').value())

    if self.cust_scen:
        self.scenario = scen

    complete = False
    error = False
    e = 1
    while not complete:
        try:
            indexes, failure_time, results, i_sel, log = software.calc(self.cust_scen,
                                                                       self.scenario,
                                                                       lock_ev=self.ui.ev_fix_CB.isChecked(),
                                                                       backup=self.ui.backup_CB.isChecked(),
                                                                       p_backup=self.ui.p_backup_DSB.value())

            # Necessario, in alcuni casi ENS è negativo
            for i in range(0, len(results)):
                for j in range (0, len(results[i])):
                    results[i][j] = max(0, results[i][j])
```

```
                print('calcolo conluso')
                index = indexes[i_sel]
                result = results[i_sel]

                self.ui.ris1_DSB.setValue(index[0])
                self.ui.ris2_DSB.setValue(index[1])
                self.ui.ris3_DSB.setValue(index[2])

                self.ui.failure_ti_DSB.setValue(result[0])
                self.ui.failure_ens_DSB.setValue(result[1])
                self.ui.failure_ng_DSB.setValue(result[2])
                self.ui.failure_st_DSB.setValue(result[3])
                self.ui.failure_gf_DSB.setValue(result[4])
                self.ui.failure_ri_DSB.setValue(result[5])

                months = [None, 'Gennaio', 'Febbraio', 'Marzo', 'Aprile', 'Maggio', 'Giugno', 'Luglio', 'Agosto', 'Settembre',
                          'Ottobre', 'Novembre', 'Dicembre']
                days = [None, 'Lunedì', 'Martedì', 'Mercoledì', 'Giovedì', 'Venerdì', 'Sabato', 'Domenica']

                self.ui.failure_month_LE.setText(months[failure_time[0]])
                self.ui.failure_day_LE.setText(days[failure_time[1]])

                h_start, h_end = int(failure_time[2]), int(failure_time[3])
                m_start = (failure_time[2] - h_start) * 60
                m_end = (failure_time[3] - h_end) * 60
                self.ui.failure_t_start_DSB.setTime(QtCore.QTime(h_start, m_start, 0))
                self.ui.failure_t_end_DSB.setTime(QtCore.QTime(h_end, m_end, 0))

                self.ui.results_fig_LBL.setPixmap(QtGui.QPixmap(path_img + '/' + self.area + '.png'))
                self.ui.results_WGT.setVisible(True)
                complete = True

                all_results = []
                for i in range(0, len(indexes)):
                    all_results.append(indexes[i] + results[i])

                self.results_compile(all_results, i_sel)
                self.ui.results_TW.setCurrentIndex(0)
                res_dict = dict()
                r_i = ['month', 'day', 'start', 'end', 'ris1', 'ris2', 'ris3', 'ti', 'ens', 'ng', 'st', 'gf', 'ri']
                r_r = failure_time + index + result
                for i in range(0, len(r_i)):
                    res_dict[r_i[i]] = float(r_r[i])
                res_dict['area'] = self.area

                if isinstance(self.scenario, list):
                    res_dict['scen_year'] = 'Personalizzato'
                    res_dict['scen_conf'] = 'Personalizzato'
                else:
                    scenario = str(self.scenario).replace('scen_', '')
                    year = scenario[0:4]
                    configurations = [['', 'BC', 'DEC'], ['', 'Base/Centralizzato', 'Decentralizzato']]
                    res_dict['scen_year'] = year
                    res_dict['scen_conf'] = configurations[1][configurations[0].index(scenario.replace(year, ''))]
                    if res_dict['scen_conf'] == '':
                        res_dict['scen_conf'] = '-'

                res_dict['backup_line'] = self.ui.backup_CB.isChecked()
                res_dict['backup_power'] = self.ui.p_backup_DSB.value()
                res_dict['log'] = log

                with open(path_img + 'results.yml', 'w') as file:
                    documents = yaml.dump(res_dict, file)

            except:
                lt = time.localtime()
                self.ui.log_TBr.append(time.strftime('%H:%M:%S', lt) + ': errore nel calcolo n. ' + str(e))
                self.ui.log_TBr.repaint()
                e = e + 1
                if e == 50:     # Limite di errori ammessi in un calcolo
                    complete = True
                    error = True

        lt = time.localtime()
        if error:
            self.ui.log_TBr.append('Calcolo non concluso\n')
        else:
            self.ui.log_TBr.append(time.strftime('%H:%M:%S', lt) + ': Calcolo concluso\n')
            v.executed.append('con')

        print(str(e - 1) + ' tentativi di calcolo falliti')
        self.ui.log_TBr.append(log)

    #
    def results_compile(self, results, i_sel):
        scenarios = ['2020', '2030BC', '2030DEC', '2040BC', '2040DEC', 'Custom']

        graphs = ['ris1', 'ris2', 'ris3', 'ti', 'ens', 'ng', 'st', 'gf', 'ri']
        title = ['Indice di autonomia', 'Indice di flessibilità', 'Indice di modulazione', 'Durata delle interruzioni',
                 'Energia non fornita', 'Generazione flessibile', 'Riserva delo storage', 'Capacità di Grid Forming',
                 'Rapporto di inerzia']
        y_axis = ['Indice di autonomia [%]', 'Indice di flessibilità [%]', 'Indice di modulazione [%]',
                  'Durata delle interruzioni [min]', 'ENS [-]', 'Generazione flessibile [-]',
                  'Riserva delo storage [-]', 'Capacità di Grid Forming [-]', 'Rapporto di inerzia [-]']

        fig = dict()
        ax = dict()
        r = dict()

        for i in range(0, len(graphs)):
            fig[graphs[i]], ax[graphs[i]] = plt.subplots(figsize=(6.6, 5))
            r[graphs[i]] = []

            for j in range(0, len(results)):
                r[graphs[i]].append(results[j][i])
            print(r[graphs[i]])

            color = ['red', 'red', 'red', 'red', 'red', 'red', 'red']
            color[i_sel] = 'green'

            ax[graphs[i]].bar(scenarios[0:len(results)], r[graphs[i]], color=color)
            ax[graphs[i]].set_title(title[i])
            ax[graphs[i]].set_xlabel('Scenario')
            ax[graphs[i]].set_ylabel(y_axis[i])
```

```
        fig[graphs[i]].savefig(path_img + '/RES_' + graphs[i] + '.png')
        self.ui.__getattribute__(graphs[i] + '_fig_LBL').setPixmap(QtGui.QPixmap(path_img + '/RES_' + graphs[i] + '.png'))

#
def ml_reliability(self):
    v.next_ml = True
    v.functionality = 'Reliability'
    self.close()

#
def ml_protections(self):
    v.next_ml = True
    v.functionality = 'Protections'
    self.close()

#
def pdf_gen(self):
    from Functionalities.PDF.pdf_creator import PDF
    pdf = PDF()
    pdf.save()
```

4.4.2.2 controlsUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'controlsUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_form(object):
    def setupUi(self, form):
        form.setObjectName("form")
        form.resize(1800, 1068)
        font = QtGui.QFont()
        font.setPointSize(10)
        form.setFont(font)
        form.setStyleSheet("background-color: rgb(0, 0, 7);")
        self.centralwidget = QtWidgets.QWidget(form)
        self.centralwidget.setObjectName("centralwidget")
        self.area_top_LN = QtWidgets.QFrame(self.centralwidget)
        self.area_top_LN.setGeometry(QtCore.QRect(10, 130, 811, 1))
        self.area_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.area_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.area_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.area_top_LN.setObjectName("area_top_LN")
        self.area_btm_LN = QtWidgets.QFrame(self.centralwidget)
        self.area_btm_LN.setGeometry(QtCore.QRect(10, 960, 811, 1))
        self.area_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.area_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.area_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.area_btm_LN.setObjectName("area_btm_LN")
        self.gridname_LBL = QtWidgets.QLabel(self.centralwidget)
        self.gridname_LBL.setGeometry(QtCore.QRect(20, 20, 581, 31))
        font = QtGui.QFont()
        font.setPointSize(16)
        font.setBold(True)
        font.setItalic(True)
        font.setWeight(75)
        self.gridname_LBL.setFont(font)
        self.gridname_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.gridname_LBL.setObjectName("gridname_LBL")
        self.gridname_top_LN = QtWidgets.QFrame(self.centralwidget)
        self.gridname_top_LN.setGeometry(QtCore.QRect(10, 10, 601, 1))
        self.gridname_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.gridname_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.gridname_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.gridname_top_LN.setObjectName("gridname_top_LN")
        self.gridname_btm_LN = QtWidgets.QFrame(self.centralwidget)
        self.gridname_btm_LN.setGeometry(QtCore.QRect(10, 60, 1840, 1))
        self.gridname_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.gridname_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.gridname_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.gridname_btm_LN.setObjectName("gridname_btm_LN")
        self.area_title_LBL = QtWidgets.QLabel(self.centralwidget)
        self.area_title_LBL.setGeometry(QtCore.QRect(10, 110, 811, 20))
        self.area_title_LBL.setObjectName("area_title_LBL")
        self.calculate_BTN = QtWidgets.QPushButton(self.centralwidget)
        self.calculate_BTN.setGeometry(QtCore.QRect(840, 900, 201, 51))
        self.calculate_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.calculate_BTN.setObjectName("calculate_BTN")
        self.grid_SW = QtWidgets.QStackedWidget(self.centralwidget)
        self.grid_SW.setGeometry(QtCore.QRect(10, 130, 821, 821))
        self.grid_SW.setStyleSheet("color: rgb(255, 255, 255);")
        self.grid_SW.setObjectName("grid_SW")
        self.city_area = QtWidgets.QWidget()
        self.city_area.setObjectName("city_area")
        self.area_img_LBL = QtWidgets.QLabel(self.city_area)
        self.area_img_LBL.setGeometry(QtCore.QRect(0, 30, 801, 801))
        self.area_img_LBL.setStyleSheet("")
        self.area_img_LBL.setText("")
        self.area_img_LBL.setPixmap(QtGui.QPixmap("res/city_area_801x466.png"))
        self.area_img_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.area_img_LBL.setObjectName("area_img_LBL")
        self.underground_PB = QtWidgets.QPushButton(self.city_area)
        self.underground_PB.setGeometry(QtCore.QRect(4, 320, 175, 305))
        self.underground_PB.setStyleSheet("background-color: rgba(255, 255, 255, 0);\n"
"border-color: rgb(255, 0, 0);")
        self.underground_PB.setObjectName("underground_PB")
        self.residential_PB = QtWidgets.QPushButton(self.city_area)
        self.residential_PB.setGeometry(QtCore.QRect(430, 320, 365, 305))
        self.residential_PB.setAcceptDrops(True)
        self.residential_PB.setStyleSheet("background-color: rgba(255, 255, 255, 0);")
        self.residential_PB.setObjectName("residential_PB")
        self.roadserv_PB = QtWidgets.QPushButton(self.city_area)
        self.roadserv_PB.setGeometry(QtCore.QRect(220, 320, 175, 305))
        self.roadserv_PB.setStyleSheet("background-color: rgba(255, 255, 255, 0);")
        self.roadserv_PB.setObjectName("roadserv_PB")
        self.area_LBL = QtWidgets.QLabel(self.city_area)
        self.area_LBL.setGeometry(QtCore.QRect(10, 10, 781, 30))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setItalic(True)
        font.setWeight(75)
        self.area_LBL.setFont(font)
        self.area_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.area_LBL.setObjectName("area_LBL")
        self.back_BTN = QtWidgets.QPushButton(self.city_area)
        self.back_BTN.setGeometry(QtCore.QRect(710, 790, 91, 23))
        self.back_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.back_BTN.setObjectName("back_BTN")
        self.grid_SW.addWidget(self.city_area)
        self.results_WGT = QtWidgets.QWidget(self.centralwidget)
        self.results_WGT.setGeometry(QtCore.QRect(1080, 100, 701, 851))
        self.results_WGT.setObjectName("results_WGT")
        self.results_top_LN = QtWidgets.QFrame(self.results_WGT)
```

```
self.results_top_LN.setGeometry(QtCore.QRect(0, 20, 700, 1))
self.results_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.results_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.results_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.results_top_LN.setObjectName("results_top_LN")
self.results_LBL = QtWidgets.QLabel(self.results_WGT)
self.results_LBL.setGeometry(QtCore.QRect(0, 0, 700, 20))
self.results_LBL.setObjectName("results_LBL")
self.failure_t_start_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_t_start_LBL.setGeometry(QtCore.QRect(440, 600, 71, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_t_start_LBL.setFont(font)
self.failure_t_start_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_t_start_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_t_start_LBL.setObjectName("failure_t_start_LBL")
self.ris2_unit_LBL = QtWidgets.QLabel(self.results_WGT)
self.ris2_unit_LBL.setGeometry(QtCore.QRect(240, 690, 51, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.ris2_unit_LBL.setFont(font)
self.ris2_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ris2_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.ris2_unit_LBL.setObjectName("ris2_unit_LBL")
self.ris2_DSB = QtWidgets.QDoubleSpinBox(self.results_WGT)
self.ris2_DSB.setEnabled(False)
self.ris2_DSB.setGeometry(QtCore.QRect(150, 690, 80, 25))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.ris2_DSB.setFont(font)
self.ris2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.ris2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ris2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.ris2_DSB.setDecimals(2)
self.ris2_DSB.setMaximum(100000.0)
self.ris2_DSB.setProperty("value", 0.0)
self.ris2_DSB.setObjectName("ris2_DSB")
self.ris2_LBL = QtWidgets.QLabel(self.results_WGT)
self.ris2_LBL.setGeometry(QtCore.QRect(0, 690, 141, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.ris2_LBL.setFont(font)
self.ris2_LBL.setAcceptDrops(False)
self.ris2_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ris2_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ris2_LBL.setObjectName("ris2_LBL")
self.ris3_unit_LBL = QtWidgets.QLabel(self.results_WGT)
self.ris3_unit_LBL.setGeometry(QtCore.QRect(240, 730, 51, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.ris3_unit_LBL.setFont(font)
self.ris3_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ris3_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.ris3_unit_LBL.setObjectName("ris3_unit_LBL")
self.ris3_DSB = QtWidgets.QDoubleSpinBox(self.results_WGT)
self.ris3_DSB.setEnabled(False)
self.ris3_DSB.setGeometry(QtCore.QRect(150, 730, 80, 25))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.ris3_DSB.setFont(font)
self.ris3_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.ris3_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ris3_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.ris3_DSB.setDecimals(2)
self.ris3_DSB.setMaximum(100000.0)
self.ris3_DSB.setProperty("value", 0.0)
self.ris3_DSB.setObjectName("ris3_DSB")
self.ris3_LBL = QtWidgets.QLabel(self.results_WGT)
self.ris3_LBL.setGeometry(QtCore.QRect(0, 730, 141, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.ris3_LBL.setFont(font)
self.ris3_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ris3_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ris3_LBL.setObjectName("ris3_LBL")
self.ris1_unit_LBL = QtWidgets.QLabel(self.results_WGT)
self.ris1_unit_LBL.setGeometry(QtCore.QRect(240, 650, 51, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.ris1_unit_LBL.setFont(font)
self.ris1_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ris1_unit_LBL.setText("%")
self.ris1_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.ris1_unit_LBL.setObjectName("ris1_unit_LBL")
self.ris1_DSB = QtWidgets.QDoubleSpinBox(self.results_WGT)
self.ris1_DSB.setEnabled(False)
self.ris1_DSB.setGeometry(QtCore.QRect(150, 650, 80, 25))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.ris1_DSB.setFont(font)
self.ris1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.ris1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ris1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.ris1_DSB.setDecimals(2)
self.ris1_DSB.setMaximum(100000.0)
self.ris1_DSB.setProperty("value", 0.0)
self.ris1_DSB.setObjectName("ris1_DSB")
self.ris1_LBL = QtWidgets.QLabel(self.results_WGT)
self.ris1_LBL.setGeometry(QtCore.QRect(0, 650, 141, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.ris1_LBL.setFont(font)
self.ris1_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ris1_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ris1_LBL.setObjectName("ris1_LBL")
self.failure_t_start_DSB = QtWidgets.QTimeEdit(self.results_WGT)
self.failure_t_start_DSB.setEnabled(False)
self.failure_t_start_DSB.setGeometry(QtCore.QRect(520, 600, 50, 25))
font = QtGui.QFont()
```

```
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.failure_t_start_DSB.setFont(font)
self.failure_t_start_DSB.setToolTip("")
self.failure_t_start_DSB.setStatusTip("")
self.failure_t_start_DSB.setWhatsThis("")
self.failure_t_start_DSB.setAccessibleName("")
self.failure_t_start_DSB.setAccessibleDescription("")
self.failure_t_start_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_t_start_DSB.setAlignment(QtCore.Qt.AlignCenter)
self.failure_t_start_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.failure_t_start_DSB.setSpecialValueText("")
self.failure_t_start_DSB.setObjectName("failure_t_start_DSB")
self.failure_t_end_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_t_end_LBL.setGeometry(QtCore.QRect(570, 600, 11, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_t_end_LBL.setFont(font)
self.failure_t_end_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_t_end_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.failure_t_end_LBL.setObjectName("failure_t_end_LBL")
self.failure_t_end_DSB = QtWidgets.QTimeEdit(self.results_WGT)
self.failure_t_end_DSB.setEnabled(False)
self.failure_t_end_DSB.setGeometry(QtCore.QRect(580, 600, 50, 25))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.failure_t_end_DSB.setFont(font)
self.failure_t_end_DSB.setToolTip("")
self.failure_t_end_DSB.setStatusTip("")
self.failure_t_end_DSB.setWhatsThis("")
self.failure_t_end_DSB.setAccessibleName("")
self.failure_t_end_DSB.setAccessibleDescription("")
self.failure_t_end_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_t_end_DSB.setAlignment(QtCore.Qt.AlignCenter)
self.failure_t_end_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.failure_t_end_DSB.setSpecialValueText("")
self.failure_t_end_DSB.setObjectName("failure_t_end_DSB")
self.failure_month_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_month_LBL.setGeometry(QtCore.QRect(40, 600, 71, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_month_LBL.setFont(font)
self.failure_month_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_month_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_month_LBL.setObjectName("failure_month_LBL")
self.failure_month_LE = QtWidgets.QLineEdit(self.results_WGT)
self.failure_month_LE.setEnabled(False)
self.failure_month_LE.setGeometry(QtCore.QRect(120, 600, 110, 25))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.failure_month_LE.setFont(font)
self.failure_month_LE.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_month_LE.setAlignment(QtCore.Qt.AlignCenter)
self.failure_month_LE.setObjectName("failure_month_LE")
self.failure_day_LE = QtWidgets.QLineEdit(self.results_WGT)
self.failure_day_LE.setEnabled(False)
self.failure_day_LE.setGeometry(QtCore.QRect(320, 600, 110, 25))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.failure_day_LE.setFont(font)
self.failure_day_LE.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_day_LE.setAlignment(QtCore.Qt.AlignCenter)
self.failure_day_LE.setObjectName("failure_day_LE")
self.failure_day_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_day_LBL.setGeometry(QtCore.QRect(240, 600, 71, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_day_LBL.setFont(font)
self.failure_day_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_day_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_day_LBL.setObjectName("failure_day_LBL")
self.failure_ti_unit_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_ti_unit_LBL.setGeometry(QtCore.QRect(440, 650, 51, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_ti_unit_LBL.setFont(font)
self.failure_ti_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_ti_unit_LBL.setText("min")
self.failure_ti_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.failure_ti_unit_LBL.setObjectName("failure_ti_unit_LBL")
self.failure_ti_DSB = QtWidgets.QDoubleSpinBox(self.results_WGT)
self.failure_ti_DSB.setEnabled(False)
self.failure_ti_DSB.setGeometry(QtCore.QRect(350, 650, 80, 25))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.failure_ti_DSB.setFont(font)
self.failure_ti_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_ti_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_ti_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.failure_ti_DSB.setDecimals(2)
self.failure_ti_DSB.setMaximum(100000.0)
self.failure_ti_DSB.setProperty("value", 0.0)
self.failure_ti_DSB.setObjectName("failure_ti_DSB")
self.failure_ti_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_ti_LBL.setGeometry(QtCore.QRect(300, 650, 41, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_ti_LBL.setFont(font)
self.failure_ti_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_ti_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_ti_LBL.setObjectName("failure_ti_LBL")
self.failure_ens_unit_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_ens_unit_LBL.setGeometry(QtCore.QRect(440, 690, 51, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_ens_unit_LBL.setFont(font)
```

```
self.failure_ens_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_ens_unit_LBL.setText("-")
self.failure_ens_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.failure_ens_unit_LBL.setObjectName("failure_ens_unit_LBL")
self.failure_ens_DSB = QtWidgets.QDoubleSpinBox(self.results_WGT)
self.failure_ens_DSB.setEnabled(False)
self.failure_ens_DSB.setGeometry(QtCore.QRect(350, 690, 80, 25))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.failure_ens_DSB.setFont(font)
self.failure_ens_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_ens_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_ens_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.failure_ens_DSB.setDecimals(4)
self.failure_ens_DSB.setMaximum(100000.0)
self.failure_ens_DSB.setProperty("value", 0.0)
self.failure_ens_DSB.setObjectName("failure_ens_DSB")
self.failure_ens_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_ens_LBL.setGeometry(QtCore.QRect(300, 690, 41, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_ens_LBL.setFont(font)
self.failure_ens_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_ens_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_ens_LBL.setObjectName("failure_ens_LBL")
self.failure_ng_unit_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_ng_unit_LBL.setGeometry(QtCore.QRect(440, 730, 51, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_ng_unit_LBL.setFont(font)
self.failure_ng_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_ng_unit_LBL.setText("-")
self.failure_ng_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.failure_ng_unit_LBL.setObjectName("failure_ng_unit_LBL")
self.failure_ng_DSB = QtWidgets.QDoubleSpinBox(self.results_WGT)
self.failure_ng_DSB.setEnabled(False)
self.failure_ng_DSB.setGeometry(QtCore.QRect(350, 730, 80, 25))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.failure_ng_DSB.setFont(font)
self.failure_ng_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_ng_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_ng_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.failure_ng_DSB.setDecimals(4)
self.failure_ng_DSB.setMaximum(100000.0)
self.failure_ng_DSB.setProperty("value", 0.0)
self.failure_ng_DSB.setObjectName("failure_ng_DSB")
self.failure_ng_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_ng_LBL.setGeometry(QtCore.QRect(300, 730, 41, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_ng_LBL.setFont(font)
self.failure_ng_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_ng_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_ng_LBL.setObjectName("failure_ng_LBL")
self.failure_st__unit_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_st__unit_LBL.setGeometry(QtCore.QRect(640, 650, 51, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_st__unit_LBL.setFont(font)
self.failure_st__unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_st__unit_LBL.setText("-")
self.failure_st__unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.failure_st__unit_LBL.setObjectName("failure_st__unit_LBL")
self.failure_st_DSB = QtWidgets.QDoubleSpinBox(self.results_WGT)
self.failure_st_DSB.setEnabled(False)
self.failure_st_DSB.setGeometry(QtCore.QRect(550, 650, 80, 25))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.failure_st_DSB.setFont(font)
self.failure_st_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_st_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_st_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.failure_st_DSB.setDecimals(4)
self.failure_st_DSB.setMaximum(100000.0)
self.failure_st_DSB.setProperty("value", 0.0)
self.failure_st_DSB.setObjectName("failure_st_DSB")
self.failure_st_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_st_LBL.setGeometry(QtCore.QRect(500, 650, 41, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_st_LBL.setFont(font)
self.failure_st_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_st_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_st_LBL.setObjectName("failure_st_LBL")
self.failure_gf__LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_gf__LBL.setGeometry(QtCore.QRect(500, 690, 41, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_gf__LBL.setFont(font)
self.failure_gf__LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_gf__LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_gf__LBL.setObjectName("failure_gf__LBL")
self.failure_gf_unit_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_gf_unit_LBL.setGeometry(QtCore.QRect(640, 690, 51, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_gf_unit_LBL.setFont(font)
self.failure_gf_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_gf_unit_LBL.setText("-")
self.failure_gf_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.failure_gf_unit_LBL.setObjectName("failure_gf_unit_LBL")
self.failure_gf_DSB = QtWidgets.QDoubleSpinBox(self.results_WGT)
self.failure_gf_DSB.setEnabled(False)
self.failure_gf_DSB.setGeometry(QtCore.QRect(550, 690, 80, 25))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
```

```
self.failure_gf_DSB.setFont(font)
self.failure_gf_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_gf_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_gf_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.failure_gf_DSB.setDecimals(4)
self.failure_gf_DSB.setMaximum(100000.0)
self.failure_gf_DSB.setProperty("value", 0.0)
self.failure_gf_DSB.setObjectName("failure_gf_DSB")
self.failure_ri_unit_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_ri_unit_LBL.setGeometry(QtCore.QRect(640, 730, 51, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_ri_unit_LBL.setFont(font)
self.failure_ri_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_ri_unit_LBL.setText("-")
self.failure_ri_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.failure_ri_unit_LBL.setObjectName("failure_ri_unit_LBL")
self.failure_ri_DSB = QtWidgets.QDoubleSpinBox(self.results_WGT)
self.failure_ri_DSB.setEnabled(False)
self.failure_ri_DSB.setGeometry(QtCore.QRect(550, 730, 80, 25))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.failure_ri_DSB.setFont(font)
self.failure_ri_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_ri_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_ri_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.failure_ri_DSB.setDecimals(4)
self.failure_ri_DSB.setMaximum(100000.0)
self.failure_ri_DSB.setProperty("value", 0.0)
self.failure_ri_DSB.setObjectName("failure_ri_DSB")
self.failure_ri_LBL = QtWidgets.QLabel(self.results_WGT)
self.failure_ri_LBL.setGeometry(QtCore.QRect(500, 730, 41, 25))
font = QtGui.QFont()
font.setPointSize(10)
self.failure_ri_LBL.setFont(font)
self.failure_ri_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.failure_ri_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.failure_ri_LBL.setObjectName("failure_ri_LBL")
self.results_TW = QtWidgets.QTabWidget(self.results_WGT)
self.results_TW.setGeometry(QtCore.QRect(20, 30, 671, 531))
self.results_TW.setObjectName("results_TW")
self.graph_TAB = QtWidgets.QWidget()
self.graph_TAB.setObjectName("graph_TAB")
self.results_fig_LBL = QtWidgets.QLabel(self.graph_TAB)
self.results_fig_LBL.setGeometry(QtCore.QRect(10, 10, 641, 481))
self.results_fig_LBL.setObjectName("results_fig_LBL")
self.results_TW.addTab(self.graph_TAB, "")
self.ris1_TAB = QtWidgets.QWidget()
self.ris1_TAB.setObjectName("ris1_TAB")
self.ris1_fig_LBL = QtWidgets.QLabel(self.ris1_TAB)
self.ris1_fig_LBL.setGeometry(QtCore.QRect(10, 10, 641, 481))
self.ris1_fig_LBL.setObjectName("ris1_fig_LBL")
self.results_TW.addTab(self.ris1_TAB, "")
self.ris2_tAB = QtWidgets.QWidget()
self.ris2_tAB.setObjectName("ris2_tAB")
self.ris2_fig_LBL = QtWidgets.QLabel(self.ris2_tAB)
self.ris2_fig_LBL.setGeometry(QtCore.QRect(10, 10, 641, 481))
self.ris2_fig_LBL.setObjectName("ris2_fig_LBL")
self.results_TW.addTab(self.ris2_tAB, "")
self.ris3_TAB = QtWidgets.QWidget()
self.ris3_TAB.setObjectName("ris3_TAB")
self.ris3_fig_LBL = QtWidgets.QLabel(self.ris3_TAB)
self.ris3_fig_LBL.setGeometry(QtCore.QRect(10, 10, 641, 481))
self.ris3_fig_LBL.setObjectName("ris3_fig_LBL")
self.results_TW.addTab(self.ris3_TAB, "")
self.ti_TAB = QtWidgets.QWidget()
self.ti_TAB.setObjectName("ti_TAB")
self.ti_fig_LBL = QtWidgets.QLabel(self.ti_TAB)
self.ti_fig_LBL.setGeometry(QtCore.QRect(10, 10, 641, 481))
self.ti_fig_LBL.setObjectName("ti_fig_LBL")
self.results_TW.addTab(self.ti_TAB, "")
self.ens_TAB = QtWidgets.QWidget()
self.ens_TAB.setObjectName("ens_TAB")
self.ens_fig_LBL = QtWidgets.QLabel(self.ens_TAB)
self.ens_fig_LBL.setGeometry(QtCore.QRect(10, 10, 641, 481))
self.ens_fig_LBL.setObjectName("ens_fig_LBL")
self.results_TW.addTab(self.ens_TAB, "")
self.ng_TAB = QtWidgets.QWidget()
self.ng_TAB.setObjectName("ng_TAB")
self.ng_fig_LBL = QtWidgets.QLabel(self.ng_TAB)
self.ng_fig_LBL.setGeometry(QtCore.QRect(10, 10, 641, 481))
self.ng_fig_LBL.setObjectName("ng_fig_LBL")
self.results_TW.addTab(self.ng_TAB, "")
self.st_TAB = QtWidgets.QWidget()
self.st_TAB.setObjectName("st_TAB")
self.st_fig_LBL = QtWidgets.QLabel(self.st_TAB)
self.st_fig_LBL.setGeometry(QtCore.QRect(10, 10, 641, 481))
self.st_fig_LBL.setObjectName("st_fig_LBL")
self.results_TW.addTab(self.st_TAB, "")
self.gf_TAB = QtWidgets.QWidget()
self.gf_TAB.setObjectName("gf_TAB")
self.gf_fig_LBL = QtWidgets.QLabel(self.gf_TAB)
self.gf_fig_LBL.setGeometry(QtCore.QRect(10, 10, 641, 481))
self.gf_fig_LBL.setObjectName("gf_fig_LBL")
self.results_TW.addTab(self.gf_TAB, "")
self.ri_TAB = QtWidgets.QWidget()
self.ri_TAB.setObjectName("ri_TAB")
self.ri_fig_LBL = QtWidgets.QLabel(self.ri_TAB)
self.ri_fig_LBL.setGeometry(QtCore.QRect(10, 10, 641, 481))
self.ri_fig_LBL.setObjectName("ri_fig_LBL")
self.results_TW.addTab(self.ri_TAB, "")
self.multiplelogics_WGT = QtWidgets.QWidget(self.results_WGT)
self.multiplelogics_WGT.setGeometry(QtCore.QRect(20, 780, 341, 81))
self.multiplelogics_WGT.setStyleSheet("color: rgb(255, 255, 255);")
self.multiplelogics_WGT.setObjectName("multiplelogics_WGT")
self.label = QtWidgets.QLabel(self.multiplelogics_WGT)
self.label.setGeometry(QtCore.QRect(120, 0, 100, 20))
self.label.setAlignment(QtCore.Qt.AlignCenter)
self.label.setObjectName("label")
self.ml_top_LN = QtWidgets.QFrame(self.multiplelogics_WGT)
self.ml_top_LN.setGeometry(QtCore.QRect(0, 10, 340, 1))
self.ml_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.ml_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
```

```
        self.ml_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ml_top_LN.setObjectName("ml_top_LN")
        self.ml_bottom_LN = QtWidgets.QFrame(self.multiplelogics_WGT)
        self.ml_bottom_LN.setGeometry(QtCore.QRect(0, 70, 340, 1))
        self.ml_bottom_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ml_bottom_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ml_bottom_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ml_bottom_LN.setObjectName("ml_bottom_LN")
        self.lm_protections_BTN = QtWidgets.QPushButton(self.multiplelogics_WGT)
        self.lm_protections_BTN.setGeometry(QtCore.QRect(10, 30, 151, 31))
        self.lm_protections_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.lm_protections_BTN.setObjectName("lm_protections_BTN")
        self.lm_reliability_BTN = QtWidgets.QPushButton(self.multiplelogics_WGT)
        self.lm_reliability_BTN.setGeometry(QtCore.QRect(180, 30, 151, 31))
        self.lm_reliability_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.lm_reliability_BTN.setObjectName("lm_reliability_BTN")
        self.ml_top_LN.raise_()
        self.label.raise_()
        self.ml_bottom_LN.raise_()
        self.lm_protections_BTN.raise_()
        self.lm_reliability_BTN.raise_()
        self.pdf_BTN = QtWidgets.QPushButton(self.results_WGT)
        self.pdf_BTN.setGeometry(QtCore.QRect(540, 810, 151, 31))
        self.pdf_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.pdf_BTN.setObjectName("pdf_BTN")
        self.failure_t_end_LBL.raise_()
        self.results_top_LN.raise_()
        self.results_LBL.raise_()
        self.failure_t_start_LBL.raise_()
        self.ris2_unit_LBL.raise_()
        self.ris2_DSB.raise_()
        self.ris2_LBL.raise_()
        self.ris3_unit_LBL.raise_()
        self.ris3_DSB.raise_()
        self.ris3_LBL.raise_()
        self.ris1_unit_LBL.raise_()
        self.ris1_DSB.raise_()
        self.ris1_LBL.raise_()
        self.failure_t_start_DSB.raise_()
        self.failure_t_end_DSB.raise_()
        self.failure_month_LBL.raise_()
        self.failure_month_LE.raise_()
        self.failure_day_LE.raise_()
        self.failure_day_LBL.raise_()
        self.failure_ti_unit_LBL.raise_()
        self.failure_ti_DSB.raise_()
        self.failure_ti_LBL.raise_()
        self.failure_ens_unit_LBL.raise_()
        self.failure_ens_DSB.raise_()
        self.failure_ens_LBL.raise_()
        self.failure_ng_unit_LBL.raise_()
        self.failure_ng_DSB.raise_()
        self.failure_ng_LBL.raise_()
        self.failure_st__unit_LBL.raise_()
        self.failure_st_DSB.raise_()
        self.failure_st_LBL.raise_()
        self.failure_gf__LBL.raise_()
        self.failure_gf_unit_LBL.raise_()
        self.failure_gf_DSB.raise_()
        self.failure_ri_unit_LBL.raise_()
        self.failure_ri_DSB.raise_()
        self.failure_ri_LBL.raise_()
        self.results_TW.raise_()
        self.multiplelogics_WGT.raise_()
        self.pdf_BTN.raise_()
        self.scenarios_SW = QtWidgets.QStackedWidget(self.centralwidget)
        self.scenarios_SW.setGeometry(QtCore.QRect(840, 130, 191, 481))
        self.scenarios_SW.setObjectName("scenarios_SW")
        self.cust_scen_WGT = QtWidgets.QWidget()
        self.cust_scen_WGT.setObjectName("cust_scen_WGT")
        self.cust_scen_SW = QtWidgets.QStackedWidget(self.cust_scen_WGT)
        self.cust_scen_SW.setGeometry(QtCore.QRect(10, 10, 171, 431))
        self.cust_scen_SW.setObjectName("cust_scen_SW")
        self.custom_RES_scen_WGT = QtWidgets.QWidget()
        self.custom_RES_scen_WGT.setObjectName("custom_RES_scen_WGT")
        self.ccr_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
        self.ccr_LBL.setGeometry(QtCore.QRect(0, 10, 51, 21))
        self.ccr_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.ccr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.ccr_LBL.setObjectName("ccr_LBL")
        self.ccr_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
        self.ccr_DSB.setGeometry(QtCore.QRect(60, 10, 62, 21))
        self.ccr_DSB.setStatusTip("")
        self.ccr_DSB.setWhatsThis("")
        self.ccr_DSB.setAccessibleName("")
        self.ccr_DSB.setAccessibleDescription("")
        self.ccr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
        self.ccr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.ccr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
        self.ccr_DSB.setSpecialValueText("")
        self.ccr_DSB.setDecimals(2)
        self.ccr_DSB.setMaximum(100000.0)
        self.ccr_DSB.setProperty("value", 3696.0)
        self.ccr_DSB.setObjectName("ccr_DSB")
        self.ccr_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
        self.ccr_unit_LBL.setGeometry(QtCore.QRect(130, 10, 41, 21))
        self.ccr_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.ccr_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
        self.ccr_unit_LBL.setObjectName("ccr_unit_LBL")
        self.pcr_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
        self.pcr_unit_LBL.setGeometry(QtCore.QRect(130, 40, 41, 21))
        self.pcr_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.pcr_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
        self.pcr_unit_LBL.setObjectName("pcr_unit_LBL")
        self.pcr_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
        self.pcr_LBL.setGeometry(QtCore.QRect(0, 40, 51, 21))
        self.pcr_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.pcr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.pcr_LBL.setObjectName("pcr_LBL")
        self.pcr_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
        self.pcr_DSB.setGeometry(QtCore.QRect(60, 40, 62, 21))
        self.pcr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
```

```
self.pcr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pcr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pcr_DSB.setDecimals(2)
self.pcr_DSB.setMaximum(100000.0)
self.pcr_DSB.setProperty("value", 280.0)
self.pcr_DSB.setObjectName("pcr_DSB")
self.pwf_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.pwf_unit_LBL.setGeometry(QtCore.QRect(130, 100, 41, 21))
self.pwf_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pwf_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pwf_unit_LBL.setObjectName("pwf_unit_LBL")
self.ppv_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.ppv_unit_LBL.setGeometry(QtCore.QRect(130, 70, 41, 21))
self.ppv_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ppv_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.ppv_unit_LBL.setObjectName("ppv_unit_LBL")
self.ppv_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.ppv_LBL.setGeometry(QtCore.QRect(0, 70, 51, 21))
self.ppv_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ppv_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ppv_LBL.setObjectName("ppv_LBL")
self.pwf_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
self.pwf_DSB.setGeometry(QtCore.QRect(60, 100, 62, 21))
self.pwf_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pwf_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pwf_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pwf_DSB.setDecimals(2)
self.pwf_DSB.setMaximum(100000.0)
self.pwf_DSB.setProperty("value", 33.0)
self.pwf_DSB.setObjectName("pwf_DSB")
self.pwf_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.pwf_LBL.setGeometry(QtCore.QRect(0, 100, 51, 21))
self.pwf_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pwf_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pwf_LBL.setObjectName("pwf_LBL")
self.ppv_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
self.ppv_DSB.setGeometry(QtCore.QRect(60, 70, 62, 21))
self.ppv_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.ppv_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ppv_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.ppv_DSB.setDecimals(2)
self.ppv_DSB.setMaximum(100000.0)
self.ppv_DSB.setProperty("value", 33.0)
self.ppv_DSB.setObjectName("ppv_DSB")
self.pess_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.pess_unit_LBL.setGeometry(QtCore.QRect(130, 130, 41, 21))
self.pess_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pess_unit_LBL.setText("kW")
self.pess_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pess_unit_LBL.setObjectName("pess_unit_LBL")
self.nphev_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.nphev_unit_LBL.setGeometry(QtCore.QRect(130, 220, 41, 21))
self.nphev_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.nphev_unit_LBL.setText("%")
self.nphev_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.nphev_unit_LBL.setObjectName("nphev_unit_LBL")
self.pess_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.pess_LBL.setGeometry(QtCore.QRect(0, 130, 51, 21))
self.pess_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pess_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pess_LBL.setObjectName("pess_LBL")
self.nphev_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
self.nphev_DSB.setGeometry(QtCore.QRect(60, 220, 62, 21))
self.nphev_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.nphev_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.nphev_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.nphev_DSB.setMaximum(100.0)
self.nphev_DSB.setProperty("value", 2.0)
self.nphev_DSB.setObjectName("nphev_DSB")
self.nphev_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.nphev_LBL.setGeometry(QtCore.QRect(0, 220, 51, 21))
self.nphev_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.nphev_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.nphev_LBL.setObjectName("nphev_LBL")
self.nv_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
self.nv_DSB.setGeometry(QtCore.QRect(60, 190, 62, 21))
self.nv_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.nv_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.nv_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.nv_DSB.setDecimals(0)
self.nv_DSB.setMaximum(100000.0)
self.nv_DSB.setProperty("value", 10.0)
self.nv_DSB.setObjectName("nv_DSB")
self.cess_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
self.cess_DSB.setGeometry(QtCore.QRect(60, 160, 62, 21))
self.cess_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cess_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cess_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cess_DSB.setDecimals(2)
self.cess_DSB.setMaximum(100000.0)
self.cess_DSB.setProperty("value", 540.0)
self.cess_DSB.setObjectName("cess_DSB")
self.cess_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.cess_LBL.setGeometry(QtCore.QRect(0, 160, 51, 21))
self.cess_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.cess_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cess_LBL.setObjectName("cess_LBL")
self.pess_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
self.pess_DSB.setGeometry(QtCore.QRect(60, 130, 62, 21))
self.pess_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pess_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pess_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pess_DSB.setDecimals(2)
self.pess_DSB.setMaximum(100000.0)
self.pess_DSB.setProperty("value", 270.0)
self.pess_DSB.setObjectName("pess_DSB")
self.nv_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.nv_LBL.setGeometry(QtCore.QRect(0, 190, 51, 21))
self.nv_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.nv_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.nv_LBL.setObjectName("nv_LBL")
self.nv_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.nv_unit_LBL.setGeometry(QtCore.QRect(130, 190, 41, 21))
self.nv_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.nv_unit_LBL.setText("-")
```

```
self.nv_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.nv_unit_LBL.setObjectName("nv_unit_LBL")
self.cess_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.cess_unit_LBL.setGeometry(QtCore.QRect(130, 160, 41, 21))
self.cess_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.cess_unit_LBL.setText("kWh")
self.cess_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cess_unit_LBL.setObjectName("cess_unit_LBL")
self.nev1_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.nev1_LBL.setGeometry(QtCore.QRect(0, 250, 51, 21))
self.nev1_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.nev1_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.nev1_LBL.setObjectName("nev1_LBL")
self.pev1_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.pev1_unit_LBL.setGeometry(QtCore.QRect(130, 340, 41, 21))
self.pev1_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pev1_unit_LBL.setText("kW")
self.pev1_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pev1_unit_LBL.setObjectName("pev1_unit_LBL")
self.pev2_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.pev2_unit_LBL.setGeometry(QtCore.QRect(130, 370, 41, 21))
self.pev2_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pev2_unit_LBL.setText("kW")
self.pev2_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pev2_unit_LBL.setObjectName("pev2_unit_LBL")
self.nev1_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.nev1_unit_LBL.setGeometry(QtCore.QRect(130, 250, 41, 21))
self.nev1_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.nev1_unit_LBL.setText("%")
self.nev1_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.nev1_unit_LBL.setObjectName("nev1_unit_LBL")
self.pphev_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.pphev_unit_LBL.setGeometry(QtCore.QRect(130, 310, 41, 21))
self.pphev_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pphev_unit_LBL.setText("kW")
self.pphev_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pphev_unit_LBL.setObjectName("pphev_unit_LBL")
self.pev2_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.pev2_LBL.setGeometry(QtCore.QRect(0, 370, 51, 21))
self.pev2_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pev2_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pev2_LBL.setObjectName("pev2_LBL")
self.pev1_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.pev1_LBL.setGeometry(QtCore.QRect(0, 340, 51, 21))
self.pev1_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pev1_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pev1_LBL.setObjectName("pev1_LBL")
self.nev2_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.nev2_LBL.setGeometry(QtCore.QRect(0, 280, 51, 21))
self.nev2_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.nev2_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.nev2_LBL.setObjectName("nev2_LBL")
self.nev1_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
self.nev1_DSB.setGeometry(QtCore.QRect(60, 250, 62, 21))
self.nev1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.nev1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.nev1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.nev1_DSB.setMaximum(100.0)
self.nev1_DSB.setProperty("value", 4.0)
self.nev1_DSB.setObjectName("nev1_DSB")
self.pev2_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
self.pev2_DSB.setGeometry(QtCore.QRect(60, 370, 62, 21))
self.pev2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pev2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pev2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pev2_DSB.setDecimals(2)
self.pev2_DSB.setMaximum(100000.0)
self.pev2_DSB.setProperty("value", 3.0)
self.pev2_DSB.setObjectName("pev2_DSB")
self.pphev_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.pphev_LBL.setGeometry(QtCore.QRect(0, 310, 51, 21))
self.pphev_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pphev_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pphev_LBL.setObjectName("pphev_LBL")
self.pev1_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
self.pev1_DSB.setGeometry(QtCore.QRect(60, 340, 62, 21))
self.pev1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pev1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pev1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pev1_DSB.setDecimals(2)
self.pev1_DSB.setMaximum(100000.0)
self.pev1_DSB.setProperty("value", 10.0)
self.pev1_DSB.setObjectName("pev1_DSB")
self.nev2_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
self.nev2_DSB.setGeometry(QtCore.QRect(60, 280, 62, 21))
self.nev2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.nev2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.nev2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.nev2_DSB.setMaximum(100.0)
self.nev2_DSB.setProperty("value", 4.0)
self.nev2_DSB.setObjectName("nev2_DSB")
self.nev2_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.nev2_unit_LBL.setGeometry(QtCore.QRect(130, 280, 41, 21))
self.nev2_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.nev2_unit_LBL.setText("%")
self.nev2_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.nev2_unit_LBL.setObjectName("nev2_unit_LBL")
self.pphev_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
self.pphev_DSB.setGeometry(QtCore.QRect(60, 310, 62, 21))
self.pphev_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pphev_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pphev_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pphev_DSB.setDecimals(2)
self.pphev_DSB.setMaximum(100000.0)
self.pphev_DSB.setProperty("value", 3.0)
self.pphev_DSB.setObjectName("pphev_DSB")
self.dr_unit_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.dr_unit_LBL.setGeometry(QtCore.QRect(130, 400, 41, 21))
self.dr_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.dr_unit_LBL.setText("-")
self.dr_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.dr_unit_LBL.setObjectName("dr_unit_LBL")
self.dr_DSB = QtWidgets.QDoubleSpinBox(self.custom_RES_scen_WGT)
self.dr_DSB.setGeometry(QtCore.QRect(60, 400, 62, 21))
self.dr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
```

```
self.dr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.dr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.dr_DSB.setDecimals(4)
self.dr_DSB.setMaximum(1.0)
self.dr_DSB.setProperty("value", 0.0)
self.dr_DSB.setObjectName("dr_DSB")
self.dr_LBL = QtWidgets.QLabel(self.custom_RES_scen_WGT)
self.dr_LBL.setGeometry(QtCore.QRect(0, 400, 51, 21))
self.dr_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.dr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.dr_LBL.setObjectName("dr_LBL")
self.cust_scen_SW.addWidget(self.custom_RES_scen_WGT)
self.custom_RS_scen_WGT = QtWidgets.QWidget()
self.custom_RS_scen_WGT.setObjectName("custom_RS_scen_WGT")
self.pev2_unit_LBL_2 = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.pev2_unit_LBL_2.setGeometry(QtCore.QRect(130, 220, 41, 21))
self.pev2_unit_LBL_2.setStyleSheet("color: rgb(255, 255, 255);")
self.pev2_unit_LBL_2.setText("kW")
self.pev2_unit_LBL_2.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pev2_unit_LBL_2.setObjectName("pev2_unit_LBL_2")
self.cc_rs_unit_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.cc_rs_unit_LBL.setGeometry(QtCore.QRect(130, 40, 41, 21))
self.cc_rs_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.cc_rs_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cc_rs_unit_LBL.setObjectName("cc_rs_unit_LBL")
self.pc_rs_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.pc_rs_LBL.setGeometry(QtCore.QRect(0, 10, 51, 21))
self.pc_rs_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pc_rs_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pc_rs_LBL.setObjectName("pc_rs_LBL")
self.pess_rs_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.pess_rs_LBL.setGeometry(QtCore.QRect(0, 100, 51, 21))
self.pess_rs_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pess_rs_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pess_rs_LBL.setObjectName("pess_rs_LBL")
self.cev_rs_DSB = QtWidgets.QDoubleSpinBox(self.custom_RS_scen_WGT)
self.cev_rs_DSB.setGeometry(QtCore.QRect(60, 250, 62, 21))
self.cev_rs_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cev_rs_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cev_rs_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cev_rs_DSB.setDecimals(2)
self.cev_rs_DSB.setMaximum(100000.0)
self.cev_rs_DSB.setProperty("value", 40.0)
self.cev_rs_DSB.setObjectName("cev_rs_DSB")
self.pess_rs_unit_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.pess_rs_unit_LBL.setGeometry(QtCore.QRect(130, 100, 41, 21))
self.pess_rs_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pess_rs_unit_LBL.setText("kW")
self.pess_rs_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pess_rs_unit_LBL.setObjectName("pess_rs_unit_LBL")
self.nv_rs_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.nv_rs_LBL.setGeometry(QtCore.QRect(0, 160, 51, 21))
self.nv_rs_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.nv_rs_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.nv_rs_LBL.setObjectName("nv_rs_LBL")
self.pev2_ev_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.pev2_ev_LBL.setGeometry(QtCore.QRect(0, 220, 51, 21))
self.pev2_ev_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pev2_ev_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pev2_ev_LBL.setObjectName("pev2_ev_LBL")
self.cc_rs_DSB = QtWidgets.QDoubleSpinBox(self.custom_RS_scen_WGT)
self.cc_rs_DSB.setGeometry(QtCore.QRect(60, 40, 62, 21))
self.cc_rs_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cc_rs_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cc_rs_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cc_rs_DSB.setDecimals(2)
self.cc_rs_DSB.setMaximum(100000.0)
self.cc_rs_DSB.setProperty("value", 635.5)
self.cc_rs_DSB.setObjectName("cc_rs_DSB")
self.ppv_rs_unit_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.ppv_rs_unit_LBL.setGeometry(QtCore.QRect(130, 70, 41, 21))
self.ppv_rs_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ppv_rs_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.ppv_rs_unit_LBL.setObjectName("ppv_rs_unit_LBL")
self.pess_rs_DSB = QtWidgets.QDoubleSpinBox(self.custom_RS_scen_WGT)
self.pess_rs_DSB.setGeometry(QtCore.QRect(60, 100, 62, 21))
self.pess_rs_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pess_rs_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pess_rs_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pess_rs_DSB.setDecimals(2)
self.pess_rs_DSB.setMaximum(100000.0)
self.pess_rs_DSB.setProperty("value", 25.0)
self.pess_rs_DSB.setObjectName("pess_rs_DSB")
self.cess_rs_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.cess_rs_LBL.setGeometry(QtCore.QRect(0, 130, 51, 21))
self.cess_rs_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.cess_rs_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cess_rs_LBL.setObjectName("cess_rs_LBL")
self.pev1_rs_DSB = QtWidgets.QDoubleSpinBox(self.custom_RS_scen_WGT)
self.pev1_rs_DSB.setGeometry(QtCore.QRect(60, 190, 62, 21))
self.pev1_rs_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pev1_rs_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pev1_rs_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pev1_rs_DSB.setDecimals(2)
self.pev1_rs_DSB.setMaximum(100000.0)
self.pev1_rs_DSB.setProperty("value", 10.0)
self.pev1_rs_DSB.setObjectName("pev1_rs_DSB")
self.cc_rs_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.cc_rs_LBL.setGeometry(QtCore.QRect(0, 40, 51, 21))
self.cc_rs_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.cc_rs_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cc_rs_LBL.setObjectName("cc_rs_LBL")
self.ppv_rs_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.ppv_rs_LBL.setGeometry(QtCore.QRect(0, 70, 51, 21))
self.ppv_rs_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ppv_rs_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ppv_rs_LBL.setObjectName("ppv_rs_LBL")
self.cev_rs_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.cev_rs_LBL.setGeometry(QtCore.QRect(0, 250, 51, 21))
self.cev_rs_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.cev_rs_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cev_rs_LBL.setObjectName("cev_rs_LBL")
self.cess_rs_unit_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.cess_rs_unit_LBL.setGeometry(QtCore.QRect(130, 130, 41, 21))
self.cess_rs_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
```

```
self.cess_rs_unit_LBL.setText("kWh")
self.cess_rs_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cess_rs_unit_LBL.setObjectName("cess_rs_unit_LBL")
self.nv_rs_unit_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.nv_rs_unit_LBL.setGeometry(QtCore.QRect(130, 160, 41, 21))
self.nv_rs_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.nv_rs_unit_LBL.setText("-")
self.nv_rs_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.nv_rs_unit_LBL.setObjectName("nv_rs_unit_LBL")
self.pc_rs_DSB = QtWidgets.QDoubleSpinBox(self.custom_RS_scen_WGT)
self.pc_rs_DSB.setGeometry(QtCore.QRect(60, 10, 62, 21))
self.pc_rs_DSB.setStatusTip("")
self.pc_rs_DSB.setWhatsThis("")
self.pc_rs_DSB.setAccessibleName("")
self.pc_rs_DSB.setAccessibleDescription("")
self.pc_rs_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pc_rs_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pc_rs_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pc_rs_DSB.setSpecialValueText("")
self.pc_rs_DSB.setDecimals(2)
self.pc_rs_DSB.setMaximum(100000.0)
self.pc_rs_DSB.setProperty("value", 19.0)
self.pc_rs_DSB.setObjectName("pc_rs_DSB")
self.nv_rs_DSB = QtWidgets.QDoubleSpinBox(self.custom_RS_scen_WGT)
self.nv_rs_DSB.setGeometry(QtCore.QRect(60, 160, 62, 21))
self.nv_rs_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.nv_rs_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.nv_rs_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.nv_rs_DSB.setDecimals(0)
self.nv_rs_DSB.setMaximum(100000.0)
self.nv_rs_DSB.setProperty("value", 10.0)
self.nv_rs_DSB.setObjectName("nv_rs_DSB")
self.ppv_rs_DSB = QtWidgets.QDoubleSpinBox(self.custom_RS_scen_WGT)
self.ppv_rs_DSB.setGeometry(QtCore.QRect(60, 70, 62, 21))
self.ppv_rs_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.ppv_rs_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ppv_rs_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.ppv_rs_DSB.setDecimals(2)
self.ppv_rs_DSB.setMaximum(100000.0)
self.ppv_rs_DSB.setProperty("value", 20.0)
self.ppv_rs_DSB.setObjectName("ppv_rs_DSB")
self.pev1_rs_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.pev1_rs_LBL.setGeometry(QtCore.QRect(0, 190, 51, 21))
self.pev1_rs_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pev1_rs_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pev1_rs_LBL.setObjectName("pev1_rs_LBL")
self.pev1_rs_unit_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.pev1_rs_unit_LBL.setGeometry(QtCore.QRect(130, 190, 41, 21))
self.pev1_rs_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pev1_rs_unit_LBL.setText("kW")
self.pev1_rs_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pev1_rs_unit_LBL.setObjectName("pev1_rs_unit_LBL")
self.cev_rs_unit_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.cev_rs_unit_LBL.setGeometry(QtCore.QRect(130, 250, 41, 21))
self.cev_rs_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.cev_rs_unit_LBL.setText("kWh")
self.cev_rs_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cev_rs_unit_LBL.setObjectName("cev_rs_unit_LBL")
self.pc_rs_unit_LBL = QtWidgets.QLabel(self.custom_RS_scen_WGT)
self.pc_rs_unit_LBL.setGeometry(QtCore.QRect(130, 10, 41, 21))
self.pc_rs_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pc_rs_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pc_rs_unit_LBL.setObjectName("pc_rs_unit_LBL")
self.pev2_rs_DSB = QtWidgets.QDoubleSpinBox(self.custom_RS_scen_WGT)
self.pev2_rs_DSB.setGeometry(QtCore.QRect(60, 220, 62, 21))
self.pev2_rs_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pev2_rs_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pev2_rs_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pev2_rs_DSB.setDecimals(2)
self.pev2_rs_DSB.setMaximum(100000.0)
self.pev2_rs_DSB.setProperty("value", 3.0)
self.pev2_rs_DSB.setObjectName("pev2_rs_DSB")
self.cess_rs_DSB = QtWidgets.QDoubleSpinBox(self.custom_RS_scen_WGT)
self.cess_rs_DSB.setGeometry(QtCore.QRect(60, 130, 62, 21))
self.cess_rs_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cess_rs_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cess_rs_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cess_rs_DSB.setDecimals(2)
self.cess_rs_DSB.setMaximum(100000.0)
self.cess_rs_DSB.setProperty("value", 50.0)
self.cess_rs_DSB.setObjectName("cess_rs_DSB")
self.cust_scen_SW.addWidget(self.custom_RS_scen_WGT)
self.custom_UG_scen_WGT = QtWidgets.QWidget()
self.custom_UG_scen_WGT.setObjectName("custom_UG_scen_WGT")
self.ppv_ug_LBL = QtWidgets.QLabel(self.custom_UG_scen_WGT)
self.ppv_ug_LBL.setGeometry(QtCore.QRect(0, 40, 51, 21))
self.ppv_ug_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ppv_ug_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ppv_ug_LBL.setObjectName("ppv_ug_LBL")
self.pess_ug_unit_LBL = QtWidgets.QLabel(self.custom_UG_scen_WGT)
self.pess_ug_unit_LBL.setGeometry(QtCore.QRect(130, 70, 41, 21))
self.pess_ug_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pess_ug_unit_LBL.setText("kW")
self.pess_ug_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pess_ug_unit_LBL.setObjectName("pess_ug_unit_LBL")
self.cess_ug_unit_LBL = QtWidgets.QLabel(self.custom_UG_scen_WGT)
self.cess_ug_unit_LBL.setGeometry(QtCore.QRect(130, 100, 41, 21))
self.cess_ug_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.cess_ug_unit_LBL.setText("kWh")
self.cess_ug_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cess_ug_unit_LBL.setObjectName("cess_ug_unit_LBL")
self.ppv_ug_unit_LBL = QtWidgets.QLabel(self.custom_UG_scen_WGT)
self.ppv_ug_unit_LBL.setGeometry(QtCore.QRect(130, 40, 41, 21))
self.ppv_ug_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ppv_ug_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.ppv_ug_unit_LBL.setObjectName("ppv_ug_unit_LBL")
self.ppv_ug_DSB = QtWidgets.QDoubleSpinBox(self.custom_UG_scen_WGT)
self.ppv_ug_DSB.setGeometry(QtCore.QRect(60, 40, 62, 21))
self.ppv_ug_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.ppv_ug_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ppv_ug_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.ppv_ug_DSB.setDecimals(2)
self.ppv_ug_DSB.setMaximum(100000.0)
self.ppv_ug_DSB.setProperty("value", 109.0)
self.ppv_ug_DSB.setObjectName("ppv_ug_DSB")
```

```
self.cm_ug_unit_LBL = QtWidgets.QLabel(self.custom_UG_scen_WGT)
self.cm_ug_unit_LBL.setGeometry(QtCore.QRect(130, 10, 41, 21))
self.cm_ug_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.cm_ug_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cm_ug_unit_LBL.setObjectName("cm_ug_unit_LBL")
self.cm_ug_DSB = QtWidgets.QDoubleSpinBox(self.custom_UG_scen_WGT)
self.cm_ug_DSB.setGeometry(QtCore.QRect(60, 10, 62, 21))
self.cm_ug_DSB.setStatusTip("")
self.cm_ug_DSB.setWhatsThis("")
self.cm_ug_DSB.setAccessibleName("")
self.cm_ug_DSB.setAccessibleDescription("")
self.cm_ug_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cm_ug_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cm_ug_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cm_ug_DSB.setSpecialValueText("")
self.cm_ug_DSB.setDecimals(2)
self.cm_ug_DSB.setMaximum(100000.0)
self.cm_ug_DSB.setProperty("value", 3694.0)
self.cm_ug_DSB.setObjectName("cm_ug_DSB")
self.pess_ug_LBL = QtWidgets.QLabel(self.custom_UG_scen_WGT)
self.pess_ug_LBL.setGeometry(QtCore.QRect(0, 70, 51, 21))
self.pess_ug_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pess_ug_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pess_ug_LBL.setObjectName("pess_ug_LBL")
self.cm_ug_LBL = QtWidgets.QLabel(self.custom_UG_scen_WGT)
self.cm_ug_LBL.setGeometry(QtCore.QRect(0, 10, 51, 21))
self.cm_ug_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.cm_ug_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cm_ug_LBL.setObjectName("cm_ug_LBL")
self.cess_ug_LBL = QtWidgets.QLabel(self.custom_UG_scen_WGT)
self.cess_ug_LBL.setGeometry(QtCore.QRect(0, 100, 51, 21))
self.cess_ug_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.cess_ug_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cess_ug_LBL.setObjectName("cess_ug_LBL")
self.cess_ug_DSB = QtWidgets.QDoubleSpinBox(self.custom_UG_scen_WGT)
self.cess_ug_DSB.setGeometry(QtCore.QRect(60, 100, 62, 21))
self.cess_ug_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cess_ug_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cess_ug_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cess_ug_DSB.setDecimals(2)
self.cess_ug_DSB.setMaximum(100000.0)
self.cess_ug_DSB.setProperty("value", 500.0)
self.cess_ug_DSB.setObjectName("cess_ug_DSB")
self.pess_ug_DSB = QtWidgets.QDoubleSpinBox(self.custom_UG_scen_WGT)
self.pess_ug_DSB.setGeometry(QtCore.QRect(60, 70, 62, 21))
self.pess_ug_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pess_ug_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pess_ug_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pess_ug_DSB.setDecimals(2)
self.pess_ug_DSB.setMaximum(100000.0)
self.pess_ug_DSB.setProperty("value", 250.0)
self.pess_ug_DSB.setObjectName("pess_ug_DSB")
self.cust_scen_SW.addWidget(self.custom_UG_scen_WGT)
self.custom_PORT_scen_WGT = QtWidgets.QWidget()
self.custom_PORT_scen_WGT.setObjectName("custom_PORT_scen_WGT")
self.ppv_port_unit_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.ppv_port_unit_LBL.setGeometry(QtCore.QRect(130, 100, 41, 21))
self.ppv_port_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ppv_port_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.ppv_port_unit_LBL.setObjectName("ppv_port_unit_LBL")
self.pwf_port_unit_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.pwf_port_unit_LBL.setGeometry(QtCore.QRect(130, 130, 41, 21))
self.pwf_port_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pwf_port_unit_LBL.setText("kW")
self.pwf_port_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pwf_port_unit_LBL.setObjectName("pwf_port_unit_LBL")
self.cc_port_unit_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.cc_port_unit_LBL.setGeometry(QtCore.QRect(130, 70, 41, 21))
self.cc_port_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.cc_port_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cc_port_unit_LBL.setObjectName("cc_port_unit_LBL")
self.nev1_port_DSB = QtWidgets.QDoubleSpinBox(self.custom_PORT_scen_WGT)
self.nev1_port_DSB.setGeometry(QtCore.QRect(60, 250, 62, 21))
self.nev1_port_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.nev1_port_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.nev1_port_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.nev1_port_DSB.setDecimals(0)
self.nev1_port_DSB.setMaximum(999999.0)
self.nev1_port_DSB.setProperty("value", 5.0)
self.nev1_port_DSB.setObjectName("nev1_port_DSB")
self.pev2_port_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.pev2_port_LBL.setGeometry(QtCore.QRect(0, 340, 51, 21))
self.pev2_port_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pev2_port_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pev2_port_LBL.setObjectName("pev2_port_LBL")
self.ncol_port_DSB = QtWidgets.QDoubleSpinBox(self.custom_PORT_scen_WGT)
self.ncol_port_DSB.setGeometry(QtCore.QRect(60, 220, 62, 21))
self.ncol_port_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.ncol_port_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ncol_port_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.ncol_port_DSB.setDecimals(0)
self.ncol_port_DSB.setMaximum(999999.0)
self.ncol_port_DSB.setProperty("value", 10.0)
self.ncol_port_DSB.setObjectName("ncol_port_DSB")
self.c_port_unit_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.c_port_unit_LBL.setGeometry(QtCore.QRect(130, 10, 41, 21))
self.c_port_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.c_port_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.c_port_unit_LBL.setObjectName("c_port_unit_LBL")
self.pev2_port_DSB = QtWidgets.QDoubleSpinBox(self.custom_PORT_scen_WGT)
self.pev2_port_DSB.setGeometry(QtCore.QRect(60, 340, 62, 21))
self.pev2_port_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pev2_port_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pev2_port_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pev2_port_DSB.setDecimals(2)
self.pev2_port_DSB.setMaximum(999999.99)
self.pev2_port_DSB.setProperty("value", 3.0)
self.pev2_port_DSB.setObjectName("pev2_port_DSB")
self.pev1_port_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.pev1_port_LBL.setGeometry(QtCore.QRect(0, 310, 51, 21))
self.pev1_port_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pev1_port_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pev1_port_LBL.setObjectName("pev1_port_LBL")
self.pev1_port_unit_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.pev1_port_unit_LBL.setGeometry(QtCore.QRect(130, 310, 41, 21))
```

```
self.pev1_port_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pev1_port_unit_LBL.setText("kW")
self.pev1_port_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pev1_port_unit_LBL.setObjectName("pev1_port_unit_LBL")
self.ncol_port_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.ncol_port_LBL.setGeometry(QtCore.QRect(0, 220, 51, 21))
self.ncol_port_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ncol_port_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ncol_port_LBL.setObjectName("ncol_port_LBL")
self.c_port_DSB = QtWidgets.QDoubleSpinBox(self.custom_PORT_scen_WGT)
self.c_port_DSB.setGeometry(QtCore.QRect(60, 10, 62, 21))
self.c_port_DSB.setStatusTip("")
self.c_port_DSB.setWhatsThis("")
self.c_port_DSB.setAccessibleName("")
self.c_port_DSB.setAccessibleDescription("")
self.c_port_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.c_port_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.c_port_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.c_port_DSB.setSpecialValueText("")
self.c_port_DSB.setDecimals(2)
self.c_port_DSB.setMaximum(999999.99)
self.c_port_DSB.setProperty("value", 48559.1)
self.c_port_DSB.setObjectName("c_port_DSB")
self.nev1_port_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.nev1_port_LBL.setGeometry(QtCore.QRect(0, 250, 51, 21))
self.nev1_port_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.nev1_port_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.nev1_port_LBL.setObjectName("nev1_port_LBL")
self.pcold_port__LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.pcold_port__LBL.setGeometry(QtCore.QRect(0, 160, 51, 21))
self.pcold_port__LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pcold_port__LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pcold_port__LBL.setObjectName("pcold_port__LBL")
self.ppv_port_DSB = QtWidgets.QDoubleSpinBox(self.custom_PORT_scen_WGT)
self.ppv_port_DSB.setGeometry(QtCore.QRect(60, 100, 62, 21))
self.ppv_port_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.ppv_port_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ppv_port_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.ppv_port_DSB.setDecimals(2)
self.ppv_port_DSB.setMaximum(999999.99)
self.ppv_port_DSB.setProperty("value", 320.0)
self.ppv_port_DSB.setObjectName("ppv_port_DSB")
self.pcold_port_unit_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.pcold_port_unit_LBL.setGeometry(QtCore.QRect(130, 160, 41, 21))
self.pcold_port_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pcold_port_unit_LBL.setText("kW")
self.pcold_port_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pcold_port_unit_LBL.setObjectName("pcold_port_unit_LBL")
self.ppv_port_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.ppv_port_LBL.setGeometry(QtCore.QRect(0, 100, 51, 21))
self.ppv_port_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ppv_port_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ppv_port_LBL.setObjectName("ppv_port_LBL")
self.pc_port_DSB = QtWidgets.QDoubleSpinBox(self.custom_PORT_scen_WGT)
self.pc_port_DSB.setGeometry(QtCore.QRect(60, 40, 62, 21))
self.pc_port_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pc_port_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pc_port_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pc_port_DSB.setDecimals(2)
self.pc_port_DSB.setMaximum(999999.99)
self.pc_port_DSB.setProperty("value", 4000.0)
self.pc_port_DSB.setObjectName("pc_port_DSB")
self.pev1_port_DSB = QtWidgets.QDoubleSpinBox(self.custom_PORT_scen_WGT)
self.pev1_port_DSB.setGeometry(QtCore.QRect(60, 310, 62, 21))
self.pev1_port_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pev1_port_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pev1_port_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pev1_port_DSB.setDecimals(2)
self.pev1_port_DSB.setMaximum(999999.99)
self.pev1_port_DSB.setProperty("value", 10.0)
self.pev1_port_DSB.setObjectName("pev1_port_DSB")
self.ncol_port_unit_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.ncol_port_unit_LBL.setGeometry(QtCore.QRect(130, 220, 41, 21))
self.ncol_port_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ncol_port_unit_LBL.setText("-")
self.ncol_port_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.ncol_port_unit_LBL.setObjectName("ncol_port_unit_LBL")
self.ccold_port_unit_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.ccold_port_unit_LBL.setGeometry(QtCore.QRect(130, 190, 41, 21))
self.ccold_port_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.ccold_port_unit_LBL.setText("kWh")
self.ccold_port_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.ccold_port_unit_LBL.setObjectName("ccold_port_unit_LBL")
self.pcold_port_DSB = QtWidgets.QDoubleSpinBox(self.custom_PORT_scen_WGT)
self.pcold_port_DSB.setGeometry(QtCore.QRect(60, 160, 62, 21))
self.pcold_port_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.pcold_port_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pcold_port_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pcold_port_DSB.setDecimals(2)
self.pcold_port_DSB.setMaximum(999999.99)
self.pcold_port_DSB.setProperty("value", 2000.0)
self.pcold_port_DSB.setObjectName("pcold_port_DSB")
self.pwf_port_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.pwf_port_LBL.setGeometry(QtCore.QRect(0, 130, 51, 21))
self.pwf_port_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.pwf_port_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pwf_port_LBL.setObjectName("pwf_port_LBL")
self.nev2_port_unit_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.nev2_port_unit_LBL.setGeometry(QtCore.QRect(130, 280, 41, 21))
self.nev2_port_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.nev2_port_unit_LBL.setText("-")
self.nev2_port_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.nev2_port_unit_LBL.setObjectName("nev2_port_unit_LBL")
self.cev_port_DSB = QtWidgets.QDoubleSpinBox(self.custom_PORT_scen_WGT)
self.cev_port_DSB.setGeometry(QtCore.QRect(60, 370, 62, 21))
self.cev_port_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cev_port_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cev_port_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cev_port_DSB.setDecimals(2)
self.cev_port_DSB.setMaximum(999999.99)
self.cev_port_DSB.setProperty("value", 40.0)
self.cev_port_DSB.setObjectName("cev_port_DSB")
self.pc_port_unit_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
self.pc_port_unit_LBL.setGeometry(QtCore.QRect(130, 40, 41, 21))
self.pc_port_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
```

```
        self.pc_port_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
        self.pc_port_unit_LBL.setObjectName("pc_port_unit_LBL")
        self.pev2_port_unit_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
        self.pev2_port_unit_LBL.setGeometry(QtCore.QRect(130, 340, 41, 21))
        self.pev2_port_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.pev2_port_unit_LBL.setText("kW")
        self.pev2_port_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
        self.pev2_port_unit_LBL.setObjectName("pev2_port_unit_LBL")
        self.pwf_port_DSB = QtWidgets.QDoubleSpinBox(self.custom_PORT_scen_WGT)
        self.pwf_port_DSB.setGeometry(QtCore.QRect(60, 130, 62, 21))
        self.pwf_port_DSB.setStyleSheet("color: rgb(255, 255, 255);")
        self.pwf_port_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.pwf_port_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
        self.pwf_port_DSB.setDecimals(2)
        self.pwf_port_DSB.setMaximum(999999.99)
        self.pwf_port_DSB.setProperty("value", 550.0)
        self.pwf_port_DSB.setObjectName("pwf_port_DSB")
        self.cev_port_unit_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
        self.cev_port_unit_LBL.setGeometry(QtCore.QRect(130, 370, 41, 21))
        self.cev_port_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.cev_port_unit_LBL.setText("kWh")
        self.cev_port_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
        self.cev_port_unit_LBL.setObjectName("cev_port_unit_LBL")
        self.nev2_port_DSB = QtWidgets.QDoubleSpinBox(self.custom_PORT_scen_WGT)
        self.nev2_port_DSB.setGeometry(QtCore.QRect(60, 280, 62, 21))
        self.nev2_port_DSB.setStyleSheet("color: rgb(255, 255, 255);")
        self.nev2_port_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.nev2_port_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
        self.nev2_port_DSB.setDecimals(0)
        self.nev2_port_DSB.setMaximum(999999.0)
        self.nev2_port_DSB.setProperty("value", 5.0)
        self.nev2_port_DSB.setObjectName("nev2_port_DSB")
        self.pc_port_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
        self.pc_port_LBL.setGeometry(QtCore.QRect(0, 40, 51, 21))
        self.pc_port_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.pc_port_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.pc_port_LBL.setObjectName("pc_port_LBL")
        self.cc_port_DSB = QtWidgets.QDoubleSpinBox(self.custom_PORT_scen_WGT)
        self.cc_port_DSB.setGeometry(QtCore.QRect(60, 70, 62, 21))
        self.cc_port_DSB.setStyleSheet("color: rgb(255, 255, 255);")
        self.cc_port_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.cc_port_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
        self.cc_port_DSB.setDecimals(2)
        self.cc_port_DSB.setMaximum(999999.99)
        self.cc_port_DSB.setProperty("value", 200.0)
        self.cc_port_DSB.setObjectName("cc_port_DSB")
        self.ccold_port_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
        self.ccold_port_LBL.setGeometry(QtCore.QRect(0, 190, 51, 21))
        self.ccold_port_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.ccold_port_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.ccold_port_LBL.setObjectName("ccold_port_LBL")
        self.cev_port_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
        self.cev_port_LBL.setGeometry(QtCore.QRect(0, 370, 51, 21))
        self.cev_port_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.cev_port_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.cev_port_LBL.setObjectName("cev_port_LBL")
        self.nev2_port_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
        self.nev2_port_LBL.setGeometry(QtCore.QRect(0, 280, 51, 21))
        self.nev2_port_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.nev2_port_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.nev2_port_LBL.setObjectName("nev2_port_LBL")
        self.nev1_port_unit_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
        self.nev1_port_unit_LBL.setGeometry(QtCore.QRect(130, 250, 41, 21))
        self.nev1_port_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.nev1_port_unit_LBL.setText("-")
        self.nev1_port_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
        self.nev1_port_unit_LBL.setObjectName("nev1_port_unit_LBL")
        self.ccold_port_DSB = QtWidgets.QDoubleSpinBox(self.custom_PORT_scen_WGT)
        self.ccold_port_DSB.setGeometry(QtCore.QRect(60, 190, 62, 21))
        self.ccold_port_DSB.setStyleSheet("color: rgb(255, 255, 255);")
        self.ccold_port_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.ccold_port_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
        self.ccold_port_DSB.setDecimals(2)
        self.ccold_port_DSB.setMaximum(999999.99)
        self.ccold_port_DSB.setProperty("value", 4000.0)
        self.ccold_port_DSB.setObjectName("ccold_port_DSB")
        self.c_port_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
        self.c_port_LBL.setGeometry(QtCore.QRect(0, 10, 51, 21))
        self.c_port_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.c_port_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.c_port_LBL.setObjectName("c_port_LBL")
        self.cc_port_LBL = QtWidgets.QLabel(self.custom_PORT_scen_WGT)
        self.cc_port_LBL.setGeometry(QtCore.QRect(0, 70, 51, 21))
        self.cc_port_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.cc_port_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.cc_port_LBL.setObjectName("cc_port_LBL")
        self.cust_scen_SW.addWidget(self.custom_PORT_scen_WGT)
        self.standard_scen_BTN = QtWidgets.QPushButton(self.cust_scen_WGT)
        self.standard_scen_BTN.setGeometry(QtCore.QRect(30, 460, 121, 23))
        self.standard_scen_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.standard_scen_BTN.setObjectName("standard_scen_BTN")
        self.scenarios_SW.addWidget(self.cust_scen_WGT)
        self.stand_scen_WGT = QtWidgets.QWidget()
        self.stand_scen_WGT.setObjectName("stand_scen_WGT")
        self.config_WGT = QtWidgets.QWidget(self.stand_scen_WGT)
        self.config_WGT.setGeometry(QtCore.QRect(10, 240, 171, 171))
        self.config_WGT.setStyleSheet("")
        self.config_WGT.setObjectName("config_WGT")
        self.config_top_LN = QtWidgets.QFrame(self.config_WGT)
        self.config_top_LN.setGeometry(QtCore.QRect(0, 0, 171, 3))
        self.config_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.config_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.config_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.config_top_LN.setObjectName("config_top_LN")
        self.config_mid_LN = QtWidgets.QFrame(self.config_WGT)
        self.config_mid_LN.setGeometry(QtCore.QRect(0, 30, 171, 1))
        self.config_mid_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.config_mid_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.config_mid_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.config_mid_LN.setObjectName("config_mid_LN")
        self.config_bottom_LN = QtWidgets.QFrame(self.config_WGT)
        self.config_bottom_LN.setGeometry(QtCore.QRect(0, 160, 171, 3))
        self.config_bottom_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.config_bottom_LN.setFrameShape(QtWidgets.QFrame.HLine)
```

```
        self.config_bottom_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.config_bottom_LN.setObjectName("config_bottom_LN")
        self.config_LBL = QtWidgets.QLabel(self.config_WGT)
        self.config_LBL.setGeometry(QtCore.QRect(0, 0, 171, 31))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setItalic(True)
        self.config_LBL.setFont(font)
        self.config_LBL.setStyleSheet("color: rgb(220, 233, 255);")
        self.config_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.config_LBL.setObjectName("config_LBL")
        self.bc_RB = QtWidgets.QRadioButton(self.config_WGT)
        self.bc_RB.setGeometry(QtCore.QRect(10, 50, 151, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.bc_RB.setFont(font)
        self.bc_RB.setStyleSheet("color: rgb(220, 233, 255);")
        self.bc_RB.setObjectName("bc_RB")
        self.dec_RB = QtWidgets.QRadioButton(self.config_WGT)
        self.dec_RB.setGeometry(QtCore.QRect(10, 90, 151, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.dec_RB.setFont(font)
        self.dec_RB.setStyleSheet("color: rgb(220, 233, 255);")
        self.dec_RB.setObjectName("dec_RB")
        self.config_LBL.raise_()
        self.config_mid_LN.raise_()
        self.config_bottom_LN.raise_()
        self.bc_RB.raise_()
        self.dec_RB.raise_()
        self.config_top_LN.raise_()
        self.year_WGT = QtWidgets.QWidget(self.stand_scen_WGT)
        self.year_WGT.setGeometry(QtCore.QRect(10, 30, 171, 171))
        self.year_WGT.setStyleSheet("")
        self.year_WGT.setObjectName("year_WGT")
        self.year_top_LN = QtWidgets.QFrame(self.year_WGT)
        self.year_top_LN.setGeometry(QtCore.QRect(0, 0, 171, 3))
        self.year_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.year_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.year_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.year_top_LN.setObjectName("year_top_LN")
        self.year_mid_LN = QtWidgets.QFrame(self.year_WGT)
        self.year_mid_LN.setGeometry(QtCore.QRect(0, 30, 171, 1))
        self.year_mid_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.year_mid_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.year_mid_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.year_mid_LN.setObjectName("year_mid_LN")
        self.year_bottom_LN = QtWidgets.QFrame(self.year_WGT)
        self.year_bottom_LN.setGeometry(QtCore.QRect(0, 160, 171, 3))
        self.year_bottom_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.year_bottom_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.year_bottom_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.year_bottom_LN.setObjectName("year_bottom_LN")
        self.year_LBL = QtWidgets.QLabel(self.year_WGT)
        self.year_LBL.setGeometry(QtCore.QRect(0, 0, 171, 31))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setItalic(True)
        self.year_LBL.setFont(font)
        self.year_LBL.setStyleSheet("color: rgb(220, 233, 255);")
        self.year_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.year_LBL.setObjectName("year_LBL")
        self.y2020_RB = QtWidgets.QRadioButton(self.year_WGT)
        self.y2020_RB.setGeometry(QtCore.QRect(10, 40, 80, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.y2020_RB.setFont(font)
        self.y2020_RB.setStyleSheet("color: rgb(220, 233, 255);")
        self.y2020_RB.setObjectName("y2020_RB")
        self.y2030_RB = QtWidgets.QRadioButton(self.year_WGT)
        self.y2030_RB.setGeometry(QtCore.QRect(10, 80, 80, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.y2030_RB.setFont(font)
        self.y2030_RB.setStyleSheet("color: rgb(220, 233, 255);")
        self.y2030_RB.setObjectName("y2030_RB")
        self.y2040_RB = QtWidgets.QRadioButton(self.year_WGT)
        self.y2040_RB.setGeometry(QtCore.QRect(10, 120, 80, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.y2040_RB.setFont(font)
        self.y2040_RB.setStyleSheet("color: rgb(220, 233, 255);")
        self.y2040_RB.setObjectName("y2040_RB")
        self.year_LBL.raise_()
        self.year_top_LN.raise_()
        self.year_mid_LN.raise_()
        self.year_bottom_LN.raise_()
        self.y2020_RB.raise_()
        self.y2030_RB.raise_()
        self.y2040_RB.raise_()
        self.custom_scen_BTN = QtWidgets.QPushButton(self.stand_scen_WGT)
        self.custom_scen_BTN.setGeometry(QtCore.QRect(30, 460, 121, 23))
        self.custom_scen_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.custom_scen_BTN.setObjectName("custom_scen_BTN")
        self.scenarios_SW.addWidget(self.stand_scen_WGT)
        self.scenario_LBL = QtWidgets.QLabel(self.centralwidget)
        self.scenario_LBL.setGeometry(QtCore.QRect(840, 110, 201, 20))
        self.scenario_LBL.setObjectName("scenario_LBL")
        self.year_mid_LN_2 = QtWidgets.QFrame(self.centralwidget)
        self.year_mid_LN_2.setGeometry(QtCore.QRect(840, 130, 191, 1))
        self.year_mid_LN_2.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.year_mid_LN_2.setFrameShape(QtWidgets.QFrame.HLine)
        self.year_mid_LN_2.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.year_mid_LN_2.setObjectName("year_mid_LN_2")
```

```
        self.log_TBr = QtWidgets.QTextBrowser(self.centralwidget)
        self.log_TBr.setGeometry(QtCore.QRect(830, 730, 221, 141))
        self.log_TBr.setStyleSheet("color: rgb(255, 255, 255);")
        self.log_TBr.setObjectName("log_TBr")
        self.log_LBL = QtWidgets.QLabel(self.centralwidget)
        self.log_LBL.setGeometry(QtCore.QRect(830, 710, 221, 20))
        self.log_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.log_LBL.setObjectName("log_LBL")
        self.ev_fix_CB = QtWidgets.QCheckBox(self.centralwidget)
        self.ev_fix_CB.setGeometry(QtCore.QRect(830, 660, 221, 31))
        self.ev_fix_CB.setStyleSheet("color: rgb(255, 255, 255);")
        self.ev_fix_CB.setChecked(True)
        self.ev_fix_CB.setObjectName("ev_fix_CB")
        self.backup_CB = QtWidgets.QCheckBox(self.centralwidget)
        self.backup_CB.setGeometry(QtCore.QRect(830, 630, 101, 21))
        self.backup_CB.setStyleSheet("color: rgb(255, 255, 255);")
        self.backup_CB.setChecked(False)
        self.backup_CB.setObjectName("backup_CB")
        self.backup_WGT = QtWidgets.QWidget(self.centralwidget)
        self.backup_WGT.setGeometry(QtCore.QRect(940, 630, 111, 21))
        self.backup_WGT.setObjectName("backup_WGT")
        self.p_backup_unit_LBL = QtWidgets.QLabel(self.backup_WGT)
        self.p_backup_unit_LBL.setGeometry(QtCore.QRect(91, 0, 16, 21))
        self.p_backup_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.p_backup_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
        self.p_backup_unit_LBL.setObjectName("p_backup_unit_LBL")
        self.p_backup_DSB = QtWidgets.QDoubleSpinBox(self.backup_WGT)
        self.p_backup_DSB.setGeometry(QtCore.QRect(31, 0, 51, 21))
        self.p_backup_DSB.setStyleSheet("color: rgb(255, 255, 255);")
        self.p_backup_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.p_backup_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
        self.p_backup_DSB.setDecimals(2)
        self.p_backup_DSB.setMaximum(100000.0)
        self.p_backup_DSB.setProperty("value", 30.0)
        self.p_backup_DSB.setObjectName("p_backup_DSB")
        self.p_backup_LBL = QtWidgets.QLabel(self.backup_WGT)
        self.p_backup_LBL.setGeometry(QtCore.QRect(0, 0, 21, 21))
        self.p_backup_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.p_backup_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.p_backup_LBL.setObjectName("p_backup_LBL")
        self.grid_SW.raise_()
        self.area_top_LN.raise_()
        self.area_btm_LN.raise_()
        self.gridname_LBL.raise_()
        self.gridname_top_LN.raise_()
        self.gridname_btm_LN.raise_()
        self.area_title_LBL.raise_()
        self.calculate_BTN.raise_()
        self.results_WGT.raise_()
        self.scenarios_SW.raise_()
        self.scenario_LBL.raise_()
        self.year_mid_LN_2.raise_()
        self.log_TBr.raise_()
        self.log_LBL.raise_()
        self.ev_fix_CB.raise_()
        self.backup_CB.raise_()
        self.backup_WGT.raise_()
        form.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(form)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1800, 21))
        self.menubar.setObjectName("menubar")
        form.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(form)
        self.statusbar.setObjectName("statusbar")
        form.setStatusBar(self.statusbar)

        self.retranslateUi(form)
        self.results_TW.setCurrentIndex(3)
        self.scenarios_SW.setCurrentIndex(0)
        self.cust_scen_SW.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(form)
        form.setTabOrder(self.underground_PB, self.roadserv_PB)
        form.setTabOrder(self.roadserv_PB, self.residential_PB)
        form.setTabOrder(self.residential_PB, self.back_BTN)
        form.setTabOrder(self.back_BTN, self.ccr_DSB)
        form.setTabOrder(self.ccr_DSB, self.pcr_DSB)
        form.setTabOrder(self.pcr_DSB, self.ppv_DSB)
        form.setTabOrder(self.ppv_DSB, self.pwf_DSB)
        form.setTabOrder(self.pwf_DSB, self.pess_DSB)
        form.setTabOrder(self.pess_DSB, self.cess_DSB)
        form.setTabOrder(self.cess_DSB, self.nv_DSB)
        form.setTabOrder(self.nv_DSB, self.nphev_DSB)
        form.setTabOrder(self.nphev_DSB, self.nev1_DSB)
        form.setTabOrder(self.nev1_DSB, self.nev2_DSB)
        form.setTabOrder(self.nev2_DSB, self.pphev_DSB)
        form.setTabOrder(self.pphev_DSB, self.pev1_DSB)
        form.setTabOrder(self.pev1_DSB, self.pev2_DSB)
        form.setTabOrder(self.pev2_DSB, self.dr_DSB)
        form.setTabOrder(self.dr_DSB, self.pc_rs_DSB)
        form.setTabOrder(self.pc_rs_DSB, self.cc_rs_DSB)
        form.setTabOrder(self.cc_rs_DSB, self.ppv_rs_DSB)
        form.setTabOrder(self.ppv_rs_DSB, self.pess_rs_DSB)
        form.setTabOrder(self.pess_rs_DSB, self.cess_rs_DSB)
        form.setTabOrder(self.cess_rs_DSB, self.nv_rs_DSB)
        form.setTabOrder(self.nv_rs_DSB, self.pev1_rs_DSB)
        form.setTabOrder(self.pev1_rs_DSB, self.pev2_rs_DSB)
        form.setTabOrder(self.pev2_rs_DSB, self.cev_rs_DSB)
        form.setTabOrder(self.cev_rs_DSB, self.cm_ug_DSB)
        form.setTabOrder(self.cm_ug_DSB, self.ppv_ug_DSB)
        form.setTabOrder(self.ppv_ug_DSB, self.pess_ug_DSB)
        form.setTabOrder(self.pess_ug_DSB, self.cess_ug_DSB)
        form.setTabOrder(self.cess_ug_DSB, self.standard_scen_BTN)
        form.setTabOrder(self.standard_scen_BTN, self.y2020_RB)
        form.setTabOrder(self.y2020_RB, self.y2030_RB)
        form.setTabOrder(self.y2030_RB, self.y2040_RB)
        form.setTabOrder(self.y2040_RB, self.bc_RB)
        form.setTabOrder(self.bc_RB, self.dec_RB)
        form.setTabOrder(self.dec_RB, self.custom_scen_BTN)
        form.setTabOrder(self.custom_scen_BTN, self.calculate_BTN)
        form.setTabOrder(self.calculate_BTN, self.failure_t_start_DSB)
        form.setTabOrder(self.failure_t_start_DSB, self.ris1_DSB)
        form.setTabOrder(self.ris1_DSB, self.ris2_DSB)
        form.setTabOrder(self.ris2_DSB, self.ris3_DSB)

    def retranslateUi(self, form):
```

```
        _translate = QtCore.QCoreApplication.translate
        form.setWindowTitle(_translate("form", "MainWindow"))
        self.gridname_LBL.setText(_translate("form", "Grid Name"))
        self.area_title_LBL.setText(_translate("form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600; font-
style:italic; color:#ffffff;\">Area di riferimento</span></p></body></html>"))
        self.calculate_BTN.setText(_translate("form", "Calcola"))
        self.underground_PB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" font-size:12pt; font-weight:600; color:#00007f;\">Settore Metropolitana</span></p></body></html>"))
        self.underground_PB.setText(_translate("form", "PushButton"))
        self.residential_PB.setToolTip(_translate("form", "<html><head/><body><p><span style=\" font-size:12pt; font-weight:600;
color:#00007f;\">Settore Residenziale</span></p></body></html>"))
        self.residential_PB.setText(_translate("form", "PushButton"))
        self.roadserv_PB.setToolTip(_translate("form", "<html><head/><body><p><span style=\" font-size:12pt; font-weight:600;
color:#00007f;\">Settore Servizi Stradali</span></p></body></html>"))
        self.roadserv_PB.setText(_translate("form", "PushButton"))
        self.area_LBL.setText(_translate("form", "Seleziona l\'area nello schema"))
        self.back_BTN.setText(_translate("form", "< Indetro"))
        self.results_LBL.setText(_translate("form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600; font-
style:italic; color:#ffffff;\">Risultati</span></p></body></html>"))
        self.failure_t_start_LBL.setText(_translate("form", "Ora"))
        self.ris2_unit_LBL.setText(_translate("form", "%"))
        self.ris2_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Indice di flessibilità</span></p></body></html>"))
        self.ris2_LBL.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Indice di flessibilità</span></p></body></html>"))
        self.ris2_LBL.setText(_translate("form", "Indice di Flessibilità"))
        self.ris3_unit_LBL.setText(_translate("form", "-"))
        self.ris3_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Indice di capacità di modulazione del prfilo di potenza in un tempo
convenzionale</span></p></body></html>"))
        self.ris3_LBL.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Indice di capacità di modulazione del prfilo di potenza in un tempo
convenzionale</span></p></body></html>"))
        self.ris3_LBL.setText(_translate("form", "Indice di Modulaziome"))
        self.ris1_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Indice di autonomia della microrete</span></p></body></html>"))
        self.ris1_LBL.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Indice di autonomia della microrete</span></p></body></html>"))
        self.ris1_LBL.setText(_translate("form", "Indice di Autonomia"))
        self.failure_t_end_LBL.setText(_translate("form", ":"))
        self.failure_month_LBL.setText(_translate("form", "Mese"))
        self.failure_month_LE.setText(_translate("form", "Settembre"))
        self.failure_day_LE.setText(_translate("form", "Domenica"))
        self.failure_day_LBL.setText(_translate("form", "Giorno"))
        self.failure_ti_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Durata delle interruzioni</span></p></body></html>"))
        self.failure_ti_LBL.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Durata delle interruzioni</span></p></body></html>"))
        self.failure_ti_LBL.setText(_translate("form", "Ti"))
        self.failure_ens_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Energia non fornita</span></p></body></html>"))
        self.failure_ens_LBL.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Energia non fornita</span></p></body></html>"))
        self.failure_ens_LBL.setText(_translate("form", "Ens"))
        self.failure_ng_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
```

```
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Generazione flessibile</span></p></body></html>"))
        self.failure_ng_LBL.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Generazione flessibile</span></p></body></html>"))
        self.failure_ng_LBL.setText(_translate("form", "Ng"))
        self.failure_st_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n<html><head><meta name=\"qrichtext\" content=\"1\" />\n"
"<style type=\"text/css\">\np, li { white-space: pre-wrap; }\n</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-
size:8pt; font-weight:400; font-style:normal;\">\n\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Riserva dello storage</span></p></body></html>"))
        self.failure_st_LBL.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n<html><head><meta name=\"qrichtext\" content=\"1\" />\n"
"<style type=\"text/css\">\np, li { white-space: pre-wrap; }\n</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-
size:8pt; font-weight:400; font-style:normal;\">\n\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Riserva dello storage</span></p></body></html>"))
        self.failure_st_LBL.setText(_translate("form", "ST"))
        self.failure_gf__LBL.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Capacità di Grid Forming</span></p></body></html>"))
        self.failure_gf__LBL.setText(_translate("form", "GF"))
        self.failure_gf_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Capacità di Grid Forming</span></p></body></html>"))
        self.failure_ri_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Rapporto di inerzia</span></p></body></html>"))
        self.failure_ri_LBL.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Rapporto di inerzia</span></p></body></html>"))
        self.failure_ri_LBL.setText(_translate("form", "RI"))
        self.results_fig_LBL.setText(_translate("form", "TextLabel"))
        self.results_TW.setTabText(self.results_TW.indexOf(self.graph_TAB), _translate("form", "Grafico"))
        self.ris1_fig_LBL.setText(_translate("form", "TextLabel"))
        self.results_TW.setTabText(self.results_TW.indexOf(self.ris1_TAB), _translate("form", "Indice di Autonomia"))
        self.ris2_fig_LBL.setText(_translate("form", "TextLabel"))
        self.results_TW.setTabText(self.results_TW.indexOf(self.ris2_tAB), _translate("form", "Indice di Flessibilità"))
        self.ris3_fig_LBL.setText(_translate("form", "TextLabel"))
        self.results_TW.setTabText(self.results_TW.indexOf(self.ris3_TAB), _translate("form", "Indice di Modulazione"))
        self.ti_fig_LBL.setText(_translate("form", "TextLabel"))
        self.results_TW.setTabText(self.results_TW.indexOf(self.ti_TAB), _translate("form", "Ti"))
        self.ens_fig_LBL.setText(_translate("form", "TextLabel"))
        self.results_TW.setTabText(self.results_TW.indexOf(self.ens_TAB), _translate("form", "Ens"))
        self.ng_fig_LBL.setText(_translate("form", "TextLabel"))
        self.results_TW.setTabText(self.results_TW.indexOf(self.ng_TAB), _translate("form", "Ng"))
        self.st_fig_LBL.setText(_translate("form", "TextLabel"))
        self.results_TW.setTabText(self.results_TW.indexOf(self.st_TAB), _translate("form", "ST"))
        self.gf_fig_LBL.setText(_translate("form", "TextLabel"))
        self.results_TW.setTabText(self.results_TW.indexOf(self.gf_TAB), _translate("form", "GF"))
        self.ri_fig_LBL.setText(_translate("form", "TextLabel"))
        self.results_TW.setTabText(self.results_TW.indexOf(self.ri_TAB), _translate("form", "RI"))
        self.label.setText(_translate("form", "Aggiungi logica"))
        self.lm_protections_BTN.setText(_translate("form", "Protezioni"))
        self.lm_reliability_BTN.setText(_translate("form", "Calcolo dell\'Affidabilità"))
        self.pdf_BTN.setText(_translate("form", "Report (PDF)"))
        self.ccr_LBL.setText(_translate("form", "Cap CR"))
        self.ccr_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Capacità della Cabina Residenziale</span></p></body></html>"))
        self.ccr_unit_LBL.setText(_translate("form", "kWh"))
        self.pcr_unit_LBL.setText(_translate("form", "kW"))
        self.pcr_LBL.setText(_translate("form", "Pow CR"))
        self.pcr_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza della Cabina Residenziale</span></p></body></html>"))
        self.pwf_unit_LBL.setText(_translate("form", "kW"))
        self.ppv_unit_LBL.setText(_translate("form", "kW"))
        self.ppv_LBL.setText(_translate("form", "P PV"))
        self.pwf_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza nominale degli impianti eolici</span></p></body></html>"))
        self.pwf_LBL.setText(_translate("form", "P WF"))
        self.ppv_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza nominale degli impianti fotovoltaici</span></p></body></html>"))
        self.pess_LBL.setText(_translate("form", "P storage"))
```

```
        self.nphev_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Percentuale dei veicoli elettrici Plug-In Hybrid rispetto al totale</span></p></body></html>"))
        self.nphev_LBL.setText(_translate("form", "% HEV"))
        self.nv_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Numero dei veicoli elettrici</span></p></body></html>"))
        self.cess_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Capacità nominale degli Storage</span></p></body></html>"))
        self.cess_LBL.setText(_translate("form", "C storage"))
        self.pess_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza nomnale degli Storage</span></p></body></html>"))
        self.nv_LBL.setText(_translate("form", "N. EV"))
        self.nev1_LBL.setText(_translate("form", "% fastEV"))
        self.pev2_LBL.setText(_translate("form", "P slowEV"))
        self.pev1_LBL.setText(_translate("form", "P fastPV"))
        self.nev2_LBL.setText(_translate("form", "% slowEV"))
        self.nev1_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Percentuale dei veicoli elettrici a ricarica rapida rispetto al totale</span></p></body></html>"))
        self.pev2_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza di ricarica dei veicoli elettrici a ricarica lenta</span></p></body></html>"))
        self.pphev_LBL.setText(_translate("form", "P HEV"))
        self.pev1_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza di ricarica dei veicoli elettrici a ricarica rapida</span></p></body></html>"))
        self.nev2_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Percentuale dei veicoli elettric a ricarica lenta rispetto al totale</span></p></body></html>"))
        self.pphev_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza di ricarica dei veicoli elettrici Plug-In Hybrid</span></p></body></html>"))
        self.dr_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Frazione di utenti che partecipano alla Demand Response</span></p></body></html>"))
        self.dr_LBL.setText(_translate("form", "DR"))
        self.cc_rs_unit_LBL.setText(_translate("form", "kWh"))
        self.pc_rs_LBL.setText(_translate("form", "P picco"))
        self.pess_rs_LBL.setText(_translate("form", "P storage"))
        self.cev_rs_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Capacità di stoccaggio dei veicoli elettrici</span></p></body></html>"))
        self.nv_rs_LBL.setText(_translate("form", "N. EV"))
        self.pev2_ev_LBL.setText(_translate("form", "P slowEV"))
        self.cc_rs_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Consumo</span></p></body></html>"))
        self.ppv_rs_unit_LBL.setText(_translate("form", "kW"))
        self.pess_rs_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza nomnale degli Storage</span></p></body></html>"))
        self.cess_rs_LBL.setText(_translate("form", "C storage"))
        self.pev1_rs_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza di ricarica dei veicoli elettrici a ricarica rapida</span></p></body></html>"))
        self.cc_rs_LBL.setText(_translate("form", "Consumo"))
```

```
        self.ppv_rs_LBL.setText(_translate("form", "P PV"))
        self.cev_rs_LBL.setText(_translate("form", "Cap. EV"))
        self.pc_rs_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza di picco</span></p></body></html>"))
        self.nv_rs_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Numero dei veicoli elettrici</span></p></body></html>"))
        self.ppv_rs_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza nominale degli impianti fotovoltaici</span></p></body></html>"))
        self.pev1_rs_LBL.setText(_translate("form", "P fastPV"))
        self.pc_rs_unit_LBL.setText(_translate("form", "kW"))
        self.pev2_rs_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza di ricarica dei veicoli elettrici a ricarica lenta</span></p></body></html>"))
        self.cess_rs_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Capacità nominale degli Storage</span></p></body></html>"))
        self.ppv_ug_LBL.setText(_translate("form", "P PV"))
        self.ppv_ug_unit_LBL.setText(_translate("form", "kW"))
        self.ppv_ug_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza nominale degli impianti fotovoltaici</span></p></body></html>"))
        self.cm_ug_unit_LBL.setText(_translate("form", "kWh"))
        self.cm_ug_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Capacità della Metropolitana</span></p></body></html>"))
        self.pess_ug_LBL.setText(_translate("form", "P storage"))
        self.cm_ug_LBL.setText(_translate("form", "Capacità"))
        self.cess_ug_LBL.setText(_translate("form", "C storage"))
        self.cess_ug_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Capacità nominale degli Storage</span></p></body></html>"))
        self.pess_ug_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza nomnale degli Storage</span></p></body></html>"))
        self.ppv_port_unit_LBL.setText(_translate("form", "kW"))
        self.cc_port_unit_LBL.setText(_translate("form", "kW"))
        self.nev1_port_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Percentuale dei veicoli elettrici a ricarica rapida rispetto al totale</span></p></body></html>"))
        self.pev2_port_LBL.setText(_translate("form", "P slowPV"))
        self.ncol_port_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Percentuale dei veicoli elettrici Plug-In Hybrid rispetto al totale</span></p></body></html>"))
        self.c_port_unit_LBL.setText(_translate("form", "kWh"))
        self.pev2_port_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza di ricarica dei veicoli elettrici a ricarica rapida</span></p></body></html>"))
        self.pev1_port_LBL.setText(_translate("form", "P fastEV"))
        self.ncol_port_LBL.setText(_translate("form", "N. veicoli"))
        self.c_port_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Capacità della Cabina Residenziale</span></p></body></html>"))
        self.nev1_port_LBL.setText(_translate("form", "N. fastEV"))
        self.pcold_port__LBL.setText(_translate("form", "P Cold Ir."))
        self.ppv_port_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
```

```
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza nominale degli impianti eolici</span></p></body></html>"))
        self.ppv_port_LBL.setText(_translate("form", "P PV"))
        self.pc_port_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza della Cabina Residenziale</span></p></body></html>"))
        self.pev1_port_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza di ricarica dei veicoli elettrici Plug-In Hybrid</span></p></body></html>"))
        self.pcold_port_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Capacità nominale degli Storage</span></p></body></html>"))
        self.pwf_port_LBL.setText(_translate("form", "P WF"))
        self.cev_port_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Frazione di utenti che partecipano alla Demand Response</span></p></body></html>"))
        self.pc_port_unit_LBL.setText(_translate("form", "kW"))
        self.pwf_port_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza nominale degli Storage</span></p></body></html>"))
        self.nev2_port_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Percentuale dei veicoli elettric a ricarica lenta rispetto al totale</span></p></body></html>"))
        self.pc_port_LBL.setText(_translate("form", "Pot. N."))
        self.cc_port_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza nominale degli impianti fotovoltaici</span></p></body></html>"))
        self.ccold_port_LBL.setText(_translate("form", "C Cold Ir."))
        self.cev_port_LBL.setText(_translate("form", "Capac. EV"))
        self.nev2_port_LBL.setText(_translate("form", "N. slowEV"))
        self.ccold_port_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Numero dei veicoli elettrici</span></p></body></html>"))
        self.c_port_LBL.setText(_translate("form", "Capacity"))
        self.cc_port_LBL.setText(_translate("form", "Contr. P"))
        self.standard_scen_BTN.setText(_translate("form", "Standard..."))
        self.config_LBL.setText(_translate("form", "Configurazione"))
        self.bc_RB.setText(_translate("form", "Base / Centralized"))
        self.dec_RB.setText(_translate("form", "Decentralized"))
        self.year_LBL.setText(_translate("form", "Anno"))
        self.y2020_RB.setText(_translate("form", "2020"))
        self.y2030_RB.setText(_translate("form", "2030"))
        self.y2040_RB.setText(_translate("form", "2040"))
        self.custom_scen_BTN.setText(_translate("form", "Personalizza..."))
        self.scenario_LBL.setText(_translate("form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600; font-
style:italic; color:#ffffff;\">Scenario</span></p></body></html>"))
        self.log_TBr.setHtml(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8.25pt; font-weight:400; font-style:normal;\">\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-
indent:0px;\"><br /></p></body></html>"))
        self.log_LBL.setText(_translate("form", "log:"))
        self.ev_fix_CB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Se selezionato, il numero di EV sarà costante nel confronto tra gli scenari;<br />Se non selezionato, il
numero di EV assumerà valori casuali diversi per ogni scenario</span></p></body></html>"))
        self.ev_fix_CB.setText(_translate("form", "Uguale numero di EV in carica\n"
"in confronto scenari"))
        self.backup_CB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Selezionare per indicare la presenza di linee di backup</span></p></body></html>"))
        self.backup_CB.setText(_translate("form", "Linee di backup"))
        self.p_backup_unit_LBL.setText(_translate("form", "kW"))
        self.p_backup_DSB.setToolTip(_translate("form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Potenza complessiva dele linee di backup</span></p></body></html> "))
        self.p_backup_LBL.setText(_translate("form", "P"))
```

```
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    form = QtWidgets.QMainWindow()
    ui = Ui_form()
    ui.setupUi(form)
    form.show()
    sys.exit(app.exec_())
```

4.4.3 Elements

4.4.3.1 ACLine

4.4.3.1.1 acline.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .aclineUI import Ui_Form
from __shared__ import variables as v
import copy


class ACLine(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(ACLine, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")

        for box in ['cap_pwr', 'etainout', 'etaoutin']:
            self.ui.__getattribute__(box + '_DSB').setStyleSheet("color: rgb(127, 127, 127);")
            self.ui.__getattribute__(box + '_DSB').setEnabled(False)

        self.ui.tabWidget.setCurrentIndex(0)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])

        conn_list = list(v.elements[element]['conn'].keys())
        try:
            self.bb1 = v.elements[element]['conn']['h']
        except KeyError:
            self.bb1 = conn_list[0]
        conn_list.remove(self.bb1)
        conn_list.remove('h')
        self.bb2 = conn_list[0]

        self.cubicle1 = v.elements[self.element]['conn'][self.bb1]
        self.cubicle2 = v.elements[self.element]['conn'][self.bb2]

        self.u = v.elements[self.bb1]['parameters']['Ur']
        self.ui.bb_in_LBL.setText(self.bb1)
        self.ui.bb_out_LBL.setText(self.bb2)

        self.ui.symbol_LBL.setPixmap(QtGui.QPixmap("_images/Elements/ACLine/element.png"))
        self.switch_draw()
        self.fill()

        for attr in ['length', 'lines', 'R1', 'Imax']:
            self.ui.__getattribute__(attr + '_DSB').valueChanged.connect(self.calculate)
        self.ui.cub_in_LBL.mouseDoubleClickEvent = self.cub1_switch
        self.ui.cub_out_LBL.mouseDoubleClickEvent = self.cub2_switch

    #
    def store(self):
        self.par['length'] = self.ui.length_DSB.value()
        self.par['lines'] = int(self.ui.lines_DSB.value())
        self.par['R1'] = self.ui.R1_DSB.value()
        self.par['X1'] = self.ui.X1_DSB.value()
        self.par['B1'] = self.ui.B1_DSB.value()
        self.par['R0'] = self.ui.R0_DSB.value()
        self.par['X0'] = self.ui.X0_DSB.value()
        self.par['B0'] = self.ui.B0_DSB.value()
        self.par['Irmax'] = self.ui.Imax_DSB.value()
        v.elements[self.element]['parameters'] = copy.deepcopy(self.par)

        v.elements[self.element]['conn'][self.bb1] = self.cubicle1
        v.elements[self.element]['conn'][self.bb2] = self.cubicle2

        self.ems['in'] = self.bb1
        self.ems['out'] = self.bb2
        for par in ['cap_pwr', 'max_outin', 'max_inout', 'etaoutin', 'etainout']:
            self.ems[par] = self.ui.__getattribute__(par + '_DSB').value()
        v.elements[self.element]['ems'] = copy.deepcopy(self.ems)

        for par in ['T0', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)

    #
    def fill(self):
        self.ui.length_DSB.setValue(self.par['length'])
        self.ui.lines_DSB.setValue(self.par['lines'])
        self.ui.R1_DSB.setValue(self.par['R1'])
        self.ui.X1_DSB.setValue(self.par['X1'])
        self.ui.B1_DSB.setValue(self.par['B1'])
        self.ui.R0_DSB.setValue(self.par['R0'])
        self.ui.X0_DSB.setValue(self.par['X0'])
        self.ui.B0_DSB.setValue(self.par['B0'])
        self.ui.Imax_DSB.setValue(self.par['Irmax'])

        for par in ['max_outin', 'max_inout']:
            self.ui.__getattribute__(par + '_DSB').setValue(self.ems[par])

        self.calculate()
        # for par in ['cap_pwr', 'max_outin', 'max_inout', 'etaoutin', 'etainout']:
        #     self.ui.__getattribute__(par + '_DSB').setValue(self.ems[par])
        if self.res != {}:
            self.fill_results()
        self.ui.tabWidget.setTabVisible(3, self.res != {})
        self.fill_reliability()
```

```
#
def fill_reliability(self):
    for par in ['T0', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
        self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
    for par in self.rel['results']:
        try:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
        except:
            if self.rel['results'][par] == 0:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('0.0')
            elif self.rel['results'][par] < 0.01:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
            else:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])

#
def fill_results(self):
    results = ['I', 'Iangle', 'P', 'Q', 'S', 'cosPhi', 'U']
    ports = ['P1', 'P2']
    ports2 = ['Port1', 'Port2']
    for port in ports:
        p = ports2[ports.index(port)]
        for result in results:
            self.ui.__getattribute__('res_' + result + '_' + port + '_DSB').setValue(self.res[p][result])
    self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])
    self.ui.res_Ploss_DSB.setValue(self.res['Ploss'])
    self.ui.res_Qloss_DSB.setValue(self.res['Qloss'])

#
def switch_draw(self):
    if self.cubicle1:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
    else:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))
    if self.cubicle2:
        self.ui.cub_out_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
    else:
        self.ui.cub_out_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))

#
def cub1_switch(self, event):
    self.cubicle1 = not self.cubicle1
    self.switch_draw()

#
def cub2_switch(self, event):
    self.cubicle2 = not self.cubicle2
    self.switch_draw()

#
def calculate(self):
    i = self.ui.Imax_DSB.value()    # sarebbe dovuto essere la corrente reale, non la massima
    r = self.ui.R1_DSB.value() * self.ui.length_DSB.value() / 1000
    u = self.u * 1000

    if u > 0:
        eta = 1 - (3**0.5 * r * i / u)
    else:
        eta = 1

    self.ui.cap_pwr_DSB.setValue(u / 1000 * self.ui.Imax_DSB.value() * self.ui.lines_DSB.value())
    for par in ['etainout', 'etaoutin']:
        self.ui.__getattribute__(par + '_DSB').setValue(eta)
```

4.4.3.1.2 aclineUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'aclineUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(494, 502)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        font = QtGui.QFont()
        font.setPointSize(9)
        self.widget.setFont(font)
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("../../../../../../.designer/backup/res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("../../../../../../.designer/backup/res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("../../../../../../.designer/backup/res/2W_Tr.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.bb_in_LBL.setFont(font)
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
```

```
self.bb_out_LN.setObjectName("bb_out_LN")
self.bb_out_LBL = QtWidgets.QLabel(self.widget)
self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
self.bb_out_LBL.setFont(font)
self.bb_out_LBL.setStyleSheet("")
self.bb_out_LBL.setLineWidth(4)
self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.store_BTN = QtWidgets.QPushButton(self.widget)
self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.store_BTN.setFont(font)
self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.store_BTN.setObjectName("store_BTN")
self.cancel_BTN = QtWidgets.QPushButton(self.widget)
self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.cancel_BTN.setFont(font)
self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.cancel_BTN.setObjectName("cancel_BTN")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.R1_LBL = QtWidgets.QLabel(self.Parameters)
self.R1_LBL.setGeometry(QtCore.QRect(60, 70, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.R1_LBL.setFont(font)
self.R1_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.R1_LBL.setObjectName("R1_LBL")
self.R0_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.R0_DSB.setGeometry(QtCore.QRect(160, 160, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.R0_DSB.setFont(font)
self.R0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.R0_DSB.setReadOnly(False)
self.R0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.R0_DSB.setDecimals(5)
self.R0_DSB.setMaximum(999.99999)
self.R0_DSB.setSingleStep(0.1)
self.R0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.R0_DSB.setProperty("value", 0.0)
self.R0_DSB.setObjectName("R0_DSB")
self.length_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.length_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.length_DSB.setFont(font)
self.length_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.length_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.length_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.length_DSB.setDecimals(1)
self.length_DSB.setMaximum(1000000.0)
self.length_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.length_DSB.setProperty("value", 0.0)
self.length_DSB.setObjectName("length_DSB")
self.lines_unit = QtWidgets.QLabel(self.Parameters)
self.lines_unit.setGeometry(QtCore.QRect(240, 40, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.lines_unit.setFont(font)
self.lines_unit.setText("")
self.lines_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.lines_unit.setObjectName("lines_unit")
self.length_unit = QtWidgets.QLabel(self.Parameters)
self.length_unit.setGeometry(QtCore.QRect(240, 10, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.length_unit.setFont(font)
self.length_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.length_unit.setObjectName("length_unit")
self.lines_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.lines_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.lines_DSB.setFont(font)
self.lines_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.lines_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.lines_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.lines_DSB.setDecimals(0)
self.lines_DSB.setMaximum(1000000.0)
self.lines_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.lines_DSB.setProperty("value", 0.0)
self.lines_DSB.setObjectName("lines_DSB")
self.R0_unit = QtWidgets.QLabel(self.Parameters)
self.R0_unit.setGeometry(QtCore.QRect(240, 160, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.R0_unit.setFont(font)
self.R0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
```

```
self.R0_unit.setObjectName("R0_unit")
self.X1_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.X1_DSB.setGeometry(QtCore.QRect(160, 100, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.X1_DSB.setFont(font)
self.X1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.X1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.X1_DSB.setDecimals(5)
self.X1_DSB.setMinimum(0.0)
self.X1_DSB.setMaximum(999.99999)
self.X1_DSB.setSingleStep(0.1)
self.X1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.X1_DSB.setProperty("value", 0.0)
self.X1_DSB.setObjectName("X1_DSB")
self.R1_unit = QtWidgets.QLabel(self.Parameters)
self.R1_unit.setGeometry(QtCore.QRect(240, 70, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.R1_unit.setFont(font)
self.R1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.R1_unit.setObjectName("R1_unit")
self.R0_LBL = QtWidgets.QLabel(self.Parameters)
self.R0_LBL.setGeometry(QtCore.QRect(60, 160, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.R0_LBL.setFont(font)
self.R0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.R0_LBL.setObjectName("R0_LBL")
self.B1_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.B1_DSB.setGeometry(QtCore.QRect(160, 130, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.B1_DSB.setFont(font)
self.B1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.B1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.B1_DSB.setDecimals(5)
self.B1_DSB.setMaximum(999.99999)
self.B1_DSB.setSingleStep(0.1)
self.B1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.B1_DSB.setProperty("value", 0.0)
self.B1_DSB.setObjectName("B1_DSB")
self.B1_LBL = QtWidgets.QLabel(self.Parameters)
self.B1_LBL.setGeometry(QtCore.QRect(60, 130, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.B1_LBL.setFont(font)
self.B1_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.B1_LBL.setObjectName("B1_LBL")
self.X1_LBL = QtWidgets.QLabel(self.Parameters)
self.X1_LBL.setGeometry(QtCore.QRect(60, 100, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.X1_LBL.setFont(font)
self.X1_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.X1_LBL.setObjectName("X1_LBL")
self.X1_unit = QtWidgets.QLabel(self.Parameters)
self.X1_unit.setGeometry(QtCore.QRect(240, 100, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.X1_unit.setFont(font)
self.X1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.X1_unit.setObjectName("X1_unit")
self.B1_unit = QtWidgets.QLabel(self.Parameters)
self.B1_unit.setGeometry(QtCore.QRect(240, 130, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.B1_unit.setFont(font)
self.B1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.B1_unit.setObjectName("B1_unit")
self.lines_LBL = QtWidgets.QLabel(self.Parameters)
self.lines_LBL.setGeometry(QtCore.QRect(60, 40, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.lines_LBL.setFont(font)
self.lines_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.lines_LBL.setObjectName("lines_LBL")
self.R1_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.R1_DSB.setGeometry(QtCore.QRect(160, 70, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.R1_DSB.setFont(font)
self.R1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.R1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.R1_DSB.setDecimals(5)
self.R1_DSB.setMaximum(999999.999)
self.R1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.R1_DSB.setProperty("value", 0.0)
self.R1_DSB.setObjectName("R1_DSB")
self.length_LBL = QtWidgets.QLabel(self.Parameters)
self.length_LBL.setGeometry(QtCore.QRect(60, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
```

```
self.length_LBL.setFont(font)
self.length_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.length_LBL.setObjectName("length_LBL")
self.B0_unit = QtWidgets.QLabel(self.Parameters)
self.B0_unit.setGeometry(QtCore.QRect(240, 220, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.B0_unit.setFont(font)
self.B0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.B0_unit.setObjectName("B0_unit")
self.X0_LBL = QtWidgets.QLabel(self.Parameters)
self.X0_LBL.setGeometry(QtCore.QRect(60, 190, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.X0_LBL.setFont(font)
self.X0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.X0_LBL.setObjectName("X0_LBL")
self.X0_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.X0_DSB.setGeometry(QtCore.QRect(160, 190, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.X0_DSB.setFont(font)
self.X0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.X0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.X0_DSB.setDecimals(5)
self.X0_DSB.setMaximum(999.99999)
self.X0_DSB.setSingleStep(0.1)
self.X0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.X0_DSB.setProperty("value", 0.0)
self.X0_DSB.setObjectName("X0_DSB")
self.Imax_unit = QtWidgets.QLabel(self.Parameters)
self.Imax_unit.setGeometry(QtCore.QRect(240, 250, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.Imax_unit.setFont(font)
self.Imax_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.Imax_unit.setObjectName("Imax_unit")
self.Imax_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.Imax_DSB.setGeometry(QtCore.QRect(160, 250, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.Imax_DSB.setFont(font)
self.Imax_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.Imax_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.Imax_DSB.setDecimals(3)
self.Imax_DSB.setMaximum(1000.0)
self.Imax_DSB.setSingleStep(0.1)
self.Imax_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.Imax_DSB.setProperty("value", 0.0)
self.Imax_DSB.setObjectName("Imax_DSB")
self.Imax_LBL = QtWidgets.QLabel(self.Parameters)
self.Imax_LBL.setGeometry(QtCore.QRect(60, 250, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Imax_LBL.setFont(font)
self.Imax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.Imax_LBL.setObjectName("Imax_LBL")
self.B0_LBL = QtWidgets.QLabel(self.Parameters)
self.B0_LBL.setGeometry(QtCore.QRect(60, 220, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.B0_LBL.setFont(font)
self.B0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.B0_LBL.setObjectName("B0_LBL")
self.X0_unit = QtWidgets.QLabel(self.Parameters)
self.X0_unit.setGeometry(QtCore.QRect(240, 190, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.X0_unit.setFont(font)
self.X0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.X0_unit.setObjectName("X0_unit")
self.B0_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.B0_DSB.setGeometry(QtCore.QRect(160, 220, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.B0_DSB.setFont(font)
self.B0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.B0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.B0_DSB.setDecimals(5)
self.B0_DSB.setMaximum(999.99999)
self.B0_DSB.setSingleStep(0.1)
self.B0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.B0_DSB.setProperty("value", 0.0)
self.B0_DSB.setObjectName("B0_DSB")
self.tabWidget.addTab(self.Parameters, "")
self.LoadFlow = QtWidgets.QWidget()
self.LoadFlow.setObjectName("LoadFlow")
self.res_P_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_P_LBL.setGeometry(QtCore.QRect(0, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_P_LBL.setFont(font)
self.res_P_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_LBL.setObjectName("res_P_LBL")
self.res_I_P1_unit = QtWidgets.QLabel(self.LoadFlow)
```

```
self.res_I_P1_unit.setGeometry(QtCore.QRect(170, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_unit.setFont(font)
self.res_I_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P1_unit.setObjectName("res_I_P1_unit")
self.res_Ploss_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Ploss_DSB.setEnabled(False)
self.res_Ploss_DSB.setGeometry(QtCore.QRect(90, 270, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Ploss_DSB.setFont(font)
self.res_Ploss_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Ploss_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Ploss_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Ploss_DSB.setDecimals(3)
self.res_Ploss_DSB.setMaximum(999999.999)
self.res_Ploss_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Ploss_DSB.setProperty("value", 0.0)
self.res_Ploss_DSB.setObjectName("res_Ploss_DSB")
self.res_Iangle_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P1_unit.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_unit.setFont(font)
self.res_Iangle_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_unit.setObjectName("res_Iangle_P1_unit")
self.res_S_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P1_DSB.setEnabled(False)
self.res_S_P1_DSB.setGeometry(QtCore.QRect(90, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_DSB.setFont(font)
self.res_S_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P1_DSB.setDecimals(3)
self.res_S_P1_DSB.setMaximum(999999.999)
self.res_S_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P1_DSB.setProperty("value", 0.0)
self.res_S_P1_DSB.setObjectName("res_S_P1_DSB")
self.res_cosPhi_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P2_DSB.setEnabled(False)
self.res_cosPhi_P2_DSB.setGeometry(QtCore.QRect(220, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P2_DSB.setFont(font)
self.res_cosPhi_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P2_DSB.setDecimals(3)
self.res_cosPhi_P2_DSB.setMaximum(999999.999)
self.res_cosPhi_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P2_DSB.setProperty("value", 0.0)
self.res_cosPhi_P2_DSB.setObjectName("res_cosPhi_P2_DSB")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_U_LBL.setGeometry(QtCore.QRect(0, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.res_cosPhi_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P1_unit.setGeometry(QtCore.QRect(170, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_unit.setFont(font)
self.res_cosPhi_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_unit.setObjectName("res_cosPhi_P1_unit")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(0, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_I_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_I_LBL.setGeometry(QtCore.QRect(0, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_I_LBL.setFont(font)
self.res_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_LBL.setObjectName("res_I_LBL")
self.res_P_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P2_DSB.setEnabled(False)
self.res_P_P2_DSB.setGeometry(QtCore.QRect(220, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P2_DSB.setFont(font)
self.res_P_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P2_DSB.setDecimals(3)
self.res_P_P2_DSB.setMinimum(-999999.999)
self.res_P_P2_DSB.setMaximum(999999.999)
```

```
self.res_P_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P2_DSB.setProperty("value", 0.0)
self.res_P_P2_DSB.setObjectName("res_P_P2_DSB")
self.Por1_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Por1_LBL.setGeometry(QtCore.QRect(90, 10, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Por1_LBL.setFont(font)
self.Por1_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Por1_LBL.setObjectName("Por1_LBL")
self.res_Q_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P1_DSB.setEnabled(False)
self.res_Q_P1_DSB.setGeometry(QtCore.QRect(90, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_DSB.setFont(font)
self.res_Q_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P1_DSB.setDecimals(3)
self.res_Q_P1_DSB.setMinimum(-999999.999)
self.res_Q_P1_DSB.setMaximum(999999.999)
self.res_Q_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P1_DSB.setProperty("value", 0.0)
self.res_Q_P1_DSB.setObjectName("res_Q_P1_DSB")
self.res_I_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P2_DSB.setEnabled(False)
self.res_I_P2_DSB.setGeometry(QtCore.QRect(220, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P2_DSB.setFont(font)
self.res_I_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P2_DSB.setDecimals(3)
self.res_I_P2_DSB.setMaximum(999999.999)
self.res_I_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P2_DSB.setProperty("value", 0.0)
self.res_I_P2_DSB.setObjectName("res_I_P2_DSB")
self.res_Q_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P1_unit.setGeometry(QtCore.QRect(170, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_unit.setFont(font)
self.res_Q_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P1_unit.setObjectName("res_Q_P1_unit")
self.res_U_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P2_DSB.setEnabled(False)
self.res_U_P2_DSB.setGeometry(QtCore.QRect(220, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P2_DSB.setFont(font)
self.res_U_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P2_DSB.setDecimals(3)
self.res_U_P2_DSB.setMaximum(999999.999)
self.res_U_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P2_DSB.setProperty("value", 0.0)
self.res_U_P2_DSB.setObjectName("res_U_P2_DSB")
self.res_I_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P1_DSB.setEnabled(False)
self.res_I_P1_DSB.setGeometry(QtCore.QRect(90, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_DSB.setFont(font)
self.res_I_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P1_DSB.setDecimals(3)
self.res_I_P1_DSB.setMaximum(999999.999)
self.res_I_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P1_DSB.setProperty("value", 0.0)
self.res_I_P1_DSB.setObjectName("res_I_P1_DSB")
self.res_P_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P1_DSB.setEnabled(False)
self.res_P_P1_DSB.setGeometry(QtCore.QRect(90, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_DSB.setFont(font)
self.res_P_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P1_DSB.setDecimals(3)
self.res_P_P1_DSB.setMinimum(-999999.999)
self.res_P_P1_DSB.setMaximum(999999.999)
self.res_P_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P1_DSB.setProperty("value", 0.0)
self.res_P_P1_DSB.setObjectName("res_P_P1_DSB")
self.res_Qloss_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Qloss_unit.setGeometry(QtCore.QRect(170, 300, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qloss_unit.setFont(font)
self.res_Qloss_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Qloss_unit.setObjectName("res_Qloss_unit")
self.res_S_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_S_LBL.setGeometry(QtCore.QRect(0, 160, 81, 21))
font = QtGui.QFont()
```

```
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_S_LBL.setFont(font)
self.res_S_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_LBL.setObjectName("res_S_LBL")
self.res_cosPhi_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_LBL.setGeometry(QtCore.QRect(0, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_cosPhi_LBL.setFont(font)
self.res_cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_LBL.setObjectName("res_cosPhi_LBL")
self.res_Ploss_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Ploss_LBL.setGeometry(QtCore.QRect(0, 270, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Ploss_LBL.setFont(font)
self.res_Ploss_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Ploss_LBL.setObjectName("res_Ploss_LBL")
self.res_Iangle_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P1_DSB.setEnabled(False)
self.res_Iangle_P1_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_DSB.setFont(font)
self.res_Iangle_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P1_DSB.setDecimals(3)
self.res_Iangle_P1_DSB.setMinimum(-999999.999)
self.res_Iangle_P1_DSB.setMaximum(999999.999)
self.res_Iangle_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P1_DSB.setProperty("value", 0.0)
self.res_Iangle_P1_DSB.setObjectName("res_Iangle_P1_DSB")
self.res_Qloss_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Qloss_DSB.setEnabled(False)
self.res_Qloss_DSB.setGeometry(QtCore.QRect(90, 300, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qloss_DSB.setFont(font)
self.res_Qloss_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Qloss_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qloss_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Qloss_DSB.setDecimals(3)
self.res_Qloss_DSB.setMaximum(999999.999)
self.res_Qloss_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Qloss_DSB.setProperty("value", 0.0)
self.res_Qloss_DSB.setObjectName("res_Qloss_DSB")
self.res_S_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P2_DSB.setEnabled(False)
self.res_S_P2_DSB.setGeometry(QtCore.QRect(220, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P2_DSB.setFont(font)
self.res_S_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P2_DSB.setDecimals(3)
self.res_S_P2_DSB.setMaximum(999999.999)
self.res_S_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P2_DSB.setProperty("value", 0.0)
self.res_S_P2_DSB.setObjectName("res_S_P2_DSB")
self.res_cosPhi_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P1_DSB.setEnabled(False)
self.res_cosPhi_P1_DSB.setGeometry(QtCore.QRect(90, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_DSB.setFont(font)
self.res_cosPhi_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P1_DSB.setDecimals(3)
self.res_cosPhi_P1_DSB.setMaximum(999999.999)
self.res_cosPhi_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P1_DSB.setProperty("value", 0.0)
self.res_cosPhi_P1_DSB.setObjectName("res_cosPhi_P1_DSB")
self.res_Iangle_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Iangle_LBL.setFont(font)
self.res_Iangle_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_LBL.setObjectName("res_Iangle_LBL")
self.Port2_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Port2_LBL.setGeometry(QtCore.QRect(220, 10, 101, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Port2_LBL.setFont(font)
self.Port2_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Port2_LBL.setObjectName("Port2_LBL")
self.res_Iangle_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P2_DSB.setEnabled(False)
self.res_Iangle_P2_DSB.setGeometry(QtCore.QRect(220, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P2_DSB.setFont(font)
```

```
self.res_Iangle_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P2_DSB.setDecimals(3)
self.res_Iangle_P2_DSB.setMinimum(-999999.999)
self.res_Iangle_P2_DSB.setMaximum(999999.999)
self.res_Iangle_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P2_DSB.setProperty("value", 0.0)
self.res_Iangle_P2_DSB.setObjectName("res_Iangle_P2_DSB")
self.res_U_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P1_unit.setGeometry(QtCore.QRect(170, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_unit.setFont(font)
self.res_U_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P1_unit.setObjectName("res_U_P1_unit")
self.res_Ploss_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Ploss_unit.setGeometry(QtCore.QRect(170, 270, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Ploss_unit.setFont(font)
self.res_Ploss_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Ploss_unit.setObjectName("res_Ploss_unit")
self.res_S_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P1_unit.setGeometry(QtCore.QRect(170, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_unit.setFont(font)
self.res_S_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P1_unit.setObjectName("res_S_P1_unit")
self.res_U_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P1_DSB.setEnabled(False)
self.res_U_P1_DSB.setGeometry(QtCore.QRect(90, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_DSB.setFont(font)
self.res_U_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P1_DSB.setDecimals(3)
self.res_U_P1_DSB.setMaximum(999999.999)
self.res_U_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P1_DSB.setProperty("value", 0.0)
self.res_U_P1_DSB.setObjectName("res_U_P1_DSB")
self.res_Qloss_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Qloss_LBL.setGeometry(QtCore.QRect(0, 300, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Qloss_LBL.setFont(font)
self.res_Qloss_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qloss_LBL.setObjectName("res_Qloss_LBL")
self.res_P_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P1_unit.setGeometry(QtCore.QRect(170, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_unit.setFont(font)
self.res_P_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P1_unit.setObjectName("res_P_P1_unit")
self.res_Q_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_LBL.setGeometry(QtCore.QRect(0, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Q_LBL.setFont(font)
self.res_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_LBL.setObjectName("res_Q_LBL")
self.res_Q_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P2_DSB.setEnabled(False)
self.res_Q_P2_DSB.setGeometry(QtCore.QRect(220, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P2_DSB.setFont(font)
self.res_Q_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P2_DSB.setDecimals(3)
self.res_Q_P2_DSB.setMinimum(-999999.999)
self.res_Q_P2_DSB.setMaximum(999999.999)
self.res_Q_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P2_DSB.setProperty("value", 0.0)
self.res_Q_P2_DSB.setObjectName("res_Q_P2_DSB")
self.res_Q_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P2_unit.setGeometry(QtCore.QRect(300, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P2_unit.setFont(font)
self.res_Q_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P2_unit.setObjectName("res_Q_P2_unit")
self.res_I_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P2_unit.setGeometry(QtCore.QRect(300, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P2_unit.setFont(font)
self.res_I_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P2_unit.setObjectName("res_I_P2_unit")
self.res_cosPhi_P2_unit = QtWidgets.QLabel(self.LoadFlow)
```

```
self.res_cosPhi_P2_unit.setGeometry(QtCore.QRect(300, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P2_unit.setFont(font)
self.res_cosPhi_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P2_unit.setObjectName("res_cosPhi_P2_unit")
self.res_P_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P2_unit.setGeometry(QtCore.QRect(300, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P2_unit.setFont(font)
self.res_P_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P2_unit.setObjectName("res_P_P2_unit")
self.res_S_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P2_unit.setGeometry(QtCore.QRect(300, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P2_unit.setFont(font)
self.res_S_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P2_unit.setObjectName("res_S_P2_unit")
self.res_U_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P2_unit.setGeometry(QtCore.QRect(300, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P2_unit.setFont(font)
self.res_U_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P2_unit.setObjectName("res_U_P2_unit")
self.res_Iangle_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P2_unit.setGeometry(QtCore.QRect(300, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P2_unit.setFont(font)
self.res_Iangle_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P2_unit.setObjectName("res_Iangle_P2_unit")
self.tabWidget.addTab(self.LoadFlow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.cap_pwr_LBL = QtWidgets.QLabel(self.EMS)
self.cap_pwr_LBL.setGeometry(QtCore.QRect(10, 10, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.cap_pwr_LBL.setFont(font)
self.cap_pwr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_LBL.setObjectName("cap_pwr_LBL")
self.cap_pwr_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.cap_pwr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_DSB.setFont(font)
self.cap_pwr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cap_pwr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cap_pwr_DSB.setDecimals(3)
self.cap_pwr_DSB.setMaximum(999999.999)
self.cap_pwr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cap_pwr_DSB.setProperty("value", 0.0)
self.cap_pwr_DSB.setObjectName("cap_pwr_DSB")
self.cap_pwr_unit = QtWidgets.QLabel(self.EMS)
self.cap_pwr_unit.setGeometry(QtCore.QRect(240, 10, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_unit.setFont(font)
self.cap_pwr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cap_pwr_unit.setObjectName("cap_pwr_unit")
self.max_outin_unit = QtWidgets.QLabel(self.EMS)
self.max_outin_unit.setGeometry(QtCore.QRect(240, 40, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_outin_unit.setFont(font)
self.max_outin_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_outin_unit.setObjectName("max_outin_unit")
self.max_outin_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.max_outin_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_outin_DSB.setFont(font)
self.max_outin_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_outin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_outin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_outin_DSB.setDecimals(3)
self.max_outin_DSB.setMaximum(999999.999)
self.max_outin_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_outin_DSB.setProperty("value", 0.0)
self.max_outin_DSB.setObjectName("max_outin_DSB")
self.max_outin_LBL = QtWidgets.QLabel(self.EMS)
self.max_outin_LBL.setGeometry(QtCore.QRect(10, 40, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_outin_LBL.setFont(font)
self.max_outin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_outin_LBL.setObjectName("max_outin_LBL")
self.max_inout_unit = QtWidgets.QLabel(self.EMS)
self.max_inout_unit.setGeometry(QtCore.QRect(240, 70, 61, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_inout_unit.setFont(font)
self.max_inout_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_inout_unit.setObjectName("max_inout_unit")
self.max_inout_LBL = QtWidgets.QLabel(self.EMS)
self.max_inout_LBL.setGeometry(QtCore.QRect(10, 70, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_inout_LBL.setFont(font)
self.max_inout_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_inout_LBL.setObjectName("max_inout_LBL")
self.max_inout_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.max_inout_DSB.setGeometry(QtCore.QRect(160, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_inout_DSB.setFont(font)
self.max_inout_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_inout_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_inout_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_inout_DSB.setDecimals(3)
self.max_inout_DSB.setMaximum(999999.999)
self.max_inout_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_inout_DSB.setProperty("value", 0.0)
self.max_inout_DSB.setObjectName("max_inout_DSB")
self.etainout_unit = QtWidgets.QLabel(self.EMS)
self.etainout_unit.setGeometry(QtCore.QRect(240, 130, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etainout_unit.setFont(font)
self.etainout_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.etainout_unit.setObjectName("etainout_unit")
self.etaoutin_LBL = QtWidgets.QLabel(self.EMS)
self.etaoutin_LBL.setGeometry(QtCore.QRect(10, 100, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.etaoutin_LBL.setFont(font)
self.etaoutin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etaoutin_LBL.setObjectName("etaoutin_LBL")
self.etainout_LBL = QtWidgets.QLabel(self.EMS)
self.etainout_LBL.setGeometry(QtCore.QRect(10, 130, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.etainout_LBL.setFont(font)
self.etainout_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etainout_LBL.setObjectName("etainout_LBL")
self.etaoutin_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.etaoutin_DSB.setGeometry(QtCore.QRect(160, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etaoutin_DSB.setFont(font)
self.etaoutin_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.etaoutin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etaoutin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.etaoutin_DSB.setDecimals(3)
self.etaoutin_DSB.setMaximum(999999.999)
self.etaoutin_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.etaoutin_DSB.setProperty("value", 0.0)
self.etaoutin_DSB.setObjectName("etaoutin_DSB")
self.etainout_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.etainout_DSB.setGeometry(QtCore.QRect(160, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etainout_DSB.setFont(font)
self.etainout_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.etainout_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etainout_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.etainout_DSB.setDecimals(3)
self.etainout_DSB.setMaximum(999999.999)
self.etainout_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.etainout_DSB.setProperty("value", 0.0)
self.etainout_DSB.setObjectName("etainout_DSB")
self.etaoutin_unit = QtWidgets.QLabel(self.EMS)
self.etaoutin_unit.setGeometry(QtCore.QRect(240, 100, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etaoutin_unit.setFont(font)
self.etaoutin_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.etaoutin_unit.setObjectName("etaoutin_unit")
self.tabWidget.addTab(self.EMS, "")
self.Reliability = QtWidgets.QWidget()
self.Reliability.setObjectName("Reliability")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(160, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
```

```
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_unit = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_unit.setGeometry(QtCore.QRect(240, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_beta_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_beta_LBL.setGeometry(QtCore.QRect(0, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(0, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(160, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_T0_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_T0_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(240, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(160, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(240, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(0, 170, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Reliability)
```

```
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(0, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(160, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_R_unit = QtWidgets.QLabel(self.Reliability)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_R_unit.setObjectName("rel_R_unit")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_anni_DSB.setEnabled(False)
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.rel_R_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_R_LBL.setGeometry(QtCore.QRect(160, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_lambda_unit = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(160, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.bottom_LN = QtWidgets.QFrame(self.Reliability)
self.bottom_LN.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN.setObjectName("bottom_LN")
self.rel_results_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 65, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_T0_unit = QtWidgets.QLabel(self.Reliability)
self.rel_T0_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_beta_unit = QtWidgets.QLabel(self.Reliability)
self.rel_beta_unit.setGeometry(QtCore.QRect(240, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_R_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.tabWidget.addTab(self.Reliability, "")
self.bb_out_LBL.raise_()
self.bb_in_LBL.raise_()
self.cub_out_LBL.raise_()
self.top_LN.raise_()
self.elem_name_LN.raise_()
self.elem_name_LBL.raise_()
self.type_LBL.raise_()
self.type_cap_LBL.raise_()
self.cub_in_LBL.raise_()
self.symbol_LBL.raise_()
self.ot_Frame_LN.raise_()
self.vdx_frame_LN.raise_()
self.vsx_frame_LN.raise_()
self.ob_frame_LN.raise_()
self.bb_in_LN.raise_()
self.bb_out_LN.raise_()
self.store_BTN.raise_()
self.cancel_BTN.raise_()
self.tabWidget.raise_()

self.retranslateUi(Form)
self.tabWidget.setCurrentIndex(3)
QtCore.QMetaObject.connectSlotsByName(Form)
Form.setTabOrder(self.tabWidget, self.length_DSB)
Form.setTabOrder(self.length_DSB, self.lines_DSB)
Form.setTabOrder(self.lines_DSB, self.R1_DSB)
Form.setTabOrder(self.R1_DSB, self.X1_DSB)
Form.setTabOrder(self.X1_DSB, self.B1_DSB)
Form.setTabOrder(self.B1_DSB, self.R0_DSB)
Form.setTabOrder(self.R0_DSB, self.X0_DSB)
Form.setTabOrder(self.X0_DSB, self.B0_DSB)
Form.setTabOrder(self.B0_DSB, self.Imax_DSB)
Form.setTabOrder(self.Imax_DSB, self.cap_pwr_DSB)
Form.setTabOrder(self.cap_pwr_DSB, self.max_outin_DSB)
Form.setTabOrder(self.max_outin_DSB, self.max_inout_DSB)
Form.setTabOrder(self.max_inout_DSB, self.etaoutin_DSB)
Form.setTabOrder(self.etaoutin_DSB, self.etainout_DSB)
Form.setTabOrder(self.etainout_DSB, self.rel_T0_DSB)
Form.setTabOrder(self.rel_T0_DSB, self.rel_alfa_DSB)
Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_I_P1_DSB)
Form.setTabOrder(self.res_I_P1_DSB, self.res_Iangle_P1_DSB)
Form.setTabOrder(self.res_Iangle_P1_DSB, self.res_P_P1_DSB)
Form.setTabOrder(self.res_P_P1_DSB, self.res_Q_P1_DSB)
Form.setTabOrder(self.res_Q_P1_DSB, self.res_S_P1_DSB)
Form.setTabOrder(self.res_S_P1_DSB, self.res_cosPhi_P1_DSB)
Form.setTabOrder(self.res_cosPhi_P1_DSB, self.res_U_P1_DSB)
Form.setTabOrder(self.res_U_P1_DSB, self.res_I_P2_DSB)
Form.setTabOrder(self.res_I_P2_DSB, self.res_Iangle_P2_DSB)
```

```
        Form.setTabOrder(self.res_Iangle_P2_DSB, self.res_P_P2_DSB)
        Form.setTabOrder(self.res_P_P2_DSB, self.res_Q_P2_DSB)
        Form.setTabOrder(self.res_Q_P2_DSB, self.res_S_P2_DSB)
        Form.setTabOrder(self.res_S_P2_DSB, self.res_cosPhi_P2_DSB)
        Form.setTabOrder(self.res_cosPhi_P2_DSB, self.res_U_P2_DSB)
        Form.setTabOrder(self.res_U_P2_DSB, self.res_Ploss_DSB)
        Form.setTabOrder(self.res_Ploss_DSB, self.res_Qloss_DSB)
        Form.setTabOrder(self.res_Qloss_DSB, self.store_BTN)
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria:"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.store_BTN.setText(_translate("Form", "Salva"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))
        self.R1_LBL.setText(_translate("Form", "R(1)"))
        self.length_unit.setText(_translate("Form", "m"))
        self.R0_unit.setText(_translate("Form", "Ohm/km"))
        self.R1_unit.setText(_translate("Form", "Ohm/km"))
        self.R0_LBL.setText(_translate("Form", "R(0)"))
        self.B1_LBL.setText(_translate("Form", "B(1)"))
        self.X1_LBL.setText(_translate("Form", "X(1)"))
        self.X1_unit.setText(_translate("Form", "Ohm/km"))
        self.B1_unit.setText(_translate("Form", "µS/km"))
        self.lines_LBL.setText(_translate("Form", "N. lines"))
        self.length_LBL.setText(_translate("Form", "Length"))
        self.B0_unit.setText(_translate("Form", "µS/km"))
        self.X0_LBL.setText(_translate("Form", "X(0)"))
        self.Imax_unit.setText(_translate("Form", "A"))
        self.Imax_LBL.setText(_translate("Form", "I max"))
        self.B0_LBL.setText(_translate("Form", "B(0)"))
        self.X0_unit.setText(_translate("Form", "Ohm/km"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Parameters), _translate("Form", "Parametri"))
        self.res_P_LBL.setText(_translate("Form", "P"))
        self.res_I_P1_unit.setText(_translate("Form", "A"))
        self.res_Iangle_P1_unit.setText(_translate("Form", "°"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.res_cosPhi_P1_unit.setText(_translate("Form", "-"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMITI VIOLATI"))
        self.res_I_LBL.setText(_translate("Form", " I"))
        self.Port1_LBL.setText(_translate("Form", "Nodo 1"))
        self.res_Q_P1_unit.setText(_translate("Form", "kVA"))
        self.res_Qloss_unit.setText(_translate("Form", "kVA"))
        self.res_S_LBL.setText(_translate("Form", "S"))
        self.res_cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.res_Ploss_LBL.setText(_translate("Form", "P loss"))
        self.res_Iangle_LBL.setText(_translate("Form", "Angollo I"))
        self.Port2_LBL.setText(_translate("Form", "Nodo 2"))
        self.res_U_P1_unit.setText(_translate("Form", "kV"))
        self.res_Ploss_unit.setText(_translate("Form", "kW"))
        self.res_S_P1_unit.setText(_translate("Form", "kVA"))
        self.res_Qloss_LBL.setText(_translate("Form", "Q loss"))
        self.res_P_P1_unit.setText(_translate("Form", "kW"))
        self.res_Q_LBL.setText(_translate("Form", "Q"))
        self.res_Q_P2_unit.setText(_translate("Form", "kVA"))
        self.res_I_P2_unit.setText(_translate("Form", "A"))
        self.res_cosPhi_P2_unit.setText(_translate("Form", "-"))
        self.res_P_P2_unit.setText(_translate("Form", "kW"))
        self.res_S_P2_unit.setText(_translate("Form", "kVA"))
        self.res_U_P2_unit.setText(_translate("Form", "kV"))
        self.res_Iangle_P2_unit.setText(_translate("Form", "°"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFlow), _translate("Form", "LoadFlow"))
        self.cap_pwr_LBL.setText(_translate("Form", "Potenza Nominale"))
        self.cap_pwr_unit.setText(_translate("Form", "kVA"))
        self.max_outin_unit.setText(_translate("Form", "p.u."))
        self.max_outin_LBL.setText(_translate("Form", "Potenza Massima IN"))
        self.max_inout_unit.setText(_translate("Form", "p.u."))
        self.max_inout_LBL.setText(_translate("Form", "Potenza Massima OUT"))
        self.etainout_unit.setText(_translate("Form", "p.u."))
        self.etaoutin_LBL.setText(_translate("Form", "Efficienza OUT-to-IN"))
        self.etainout_LBL.setText(_translate("Form", "Efficienza IN-to-OUT"))
        self.etaoutin_unit.setText(_translate("Form", "p.u."))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_T0_unit.setText(_translate("Form", "°C"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
```

```
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Reliability), _translate("Form", "Affidabilità"))

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.3.2 ACLoads

4.4.3.2.1 acload.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .acloadUI import Ui_Form
from __shared__ import variables as v
import copy


class ACLoad(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(ACLoad, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.bb_out_LN.setVisible(False)
        self.ui.bb_out_LBL.setVisible(False)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")
        self.ui.tabWidget.setCurrentIndex(0)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])

        self.bb = v.elements[element]['conn']['h']
        self.cubicle = v.elements[self.element]['conn'][self.bb]
        self.u = v.elements[self.bb]['parameters']['Ur']
        self.ui.bb_in_LBL.setText(self.bb)

        self.ui.symbol_LBL.setPixmap(QtGui.QPixmap("_images/Elements/ACLoads/element.png"))
        self.switch_draw()
        self.fill()

        for attr in ['p_DSB', 'q_DSB', 's_DSB', 'i_DSB', 'cosPhi_DSB']:
            self.ui.__getattribute__(attr).valueChanged.connect(self.calculate)
        self.ui.cub_in_LBL.mouseDoubleClickEvent = self.cub1_switch

    #
    def calculate(self):
        p = self.ui.p_DSB.value()
        q = self.ui.q_DSB.value()
        s = self.ui.s_DSB.value()
        i = self.ui.i_DSB.value()
        cosPhi = self.ui.cosPhi_DSB.value()

        if self.ui.control_CB.currentIndex() == 0:
            s = (p ** 2 + q ** 2) ** 0.5
            cosPhi = p / s
            i = 0
            self.ui.s_DSB.setValue(s)
            self.ui.cosPhi_DSB.setValue(cosPhi)
            self.ui.i_DSB.setValue(i)
        elif self.ui.control_CB.currentIndex() == 1:
            s = p / cosPhi
            q = (s ** 2 - p ** 2) ** 0.5
            self.ui.q_DSB.setValue(q)
            self.ui.s_DSB.setValue(s)
            self.ui.i_DSB.setValue(i)

        elif self.ui.control_CB.currentIndex() == 2:
            s = self.u * i
            p = s * cosPhi
            q = (s ** 2 - p ** 2) ** 0.5
            self.ui.p_DSB.setValue(p)
            self.ui.q_DSB.setValue(q)
            self.ui.s_DSB.setValue(s)
        elif self.ui.control_CB.currentIndex() == 3:
            q = (3**0.5) * self.u * i
            s = (p**2 + q ** 2)**0.5
            cosPhi = p / s
            self.ui.q_DSB.setValue(q)
            self.ui.s_DSB.setValue(s)
            self.ui.cosPhi_DSB.setValue(cosPhi)
        else:
            p = s*cosPhi
            q = (s ** 2 - p ** 2) ** 0.5
            self.ui.p_DSB.setValue(p)
            self.ui.q_DSB.setValue(q)
            self.ui.i_DSB.setValue(i)

    #
    def store(self): # Inutile?? ------------------------------------------------------------------------------
        self.par['P'] = self.ui.p_DSB.value()
        self.par['Q'] = self.ui.q_DSB.value()
        self.par['S'] = self.ui.s_DSB.value()
        self.par['I'] = self.ui.i_DSB.value()
        self.par['cosPhi'] = self.ui.cosPhi_DSB.value()
        self.par['control'] = self.ui.control_CB.currentIndex()
        v.elements[self.element]['parameters'] = copy.deepcopy(self.par)

        v.elements[self.element]['conn'][self.bb] = self.cubicle

        self.ems['unmet_cost'] = self.ui.unmet_cost_DSB.value()
        v.elements[self.element]['ems'] = copy.deepcopy(self.ems)

        for par in ['T0', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)

    #
    def fill(self):
        self.ui.p_DSB.setValue(self.par['P'])
        self.ui.q_DSB.setValue(self.par['Q'])
        self.ui.s_DSB.setValue(self.par['S'])
```

```
    self.ui.i_DSB.setValue(self.par['I'])
    self.ui.cosPhi_DSB.setValue(self.par['cosPhi'])
    self.ui.control_CB.setCurrentIndex(self.par['control'])

    self.control_mode(self.ui)

    self.ui.unmet_cost_DSB.setValue(self.ems['unmet_cost'])

    if self.res != {}:
        self.fill_results()
    self.ui.tabWidget.setTabVisible(3, self.res != {})
    self.fill_reliability()

#
def fill_reliability(self):
    for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
        self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
    for par in ['lambda', 'R', 'MTBF_ore', 'MTBF_anni']:
        try:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
        except:
            if self.rel['results'][par] == 0:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('0')
            elif self.rel['results'][par] < 0.01:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
            else:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])

#
def fill_results(self):
    results = ['I', 'Iangle', 'P', 'Q', 'S', 'cosPhi', 'U']
    ports = ['P1']

    for port in ports:
        for result in results:
            self.ui.__getattribute__('res_' + result + '_' + port + '_DSB').setValue(self.res[result])
    self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])

#
def control_mode(self, ui):
    enabled = [True, True, False, False, False]
    if ui.control_CB.currentIndex() == 0:
        enabled = [True, True, False, False, False]
    elif ui.control_CB.currentIndex() == 1:
        enabled = [True, False, False, False, True]
        ui.cosPhi_DSB.setMinimum(0.00001)
    elif ui.control_CB.currentIndex() == 2:
        enabled = [False, False, False, True, True]
    elif ui.control_CB.currentIndex() == 3:
        enabled = [True, False, False, True, False]
    elif ui.control_CB.currentIndex() == 4:
        enabled = [False, False, True, False, True]

    i = 0
    for obj in [ui.p_DSB, ui.q_DSB, ui.s_DSB, ui.i_DSB, ui.cosPhi_DSB]:
        if enabled[i]:
            obj.setStyleSheet("color: rgb(255, 255, 255);")
        else:
            obj.setStyleSheet("color: rgb(127, 127, 127);")
        obj.setEnabled(enabled[i])
        i += 1

def switch_draw(self):
    if self.cubicle:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
    else:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))

def cub1_switch(self, event):
    self.cubicle = not self.cubicle
    self.switch_draw()

def calc_profile(self, pu_profile):
    profile = []
    for data in pu_profile:
        profile.append(data * self.par['P'])
    return profile
```

4.4.3.2.2 acloadUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'acloadUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(497, 505)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.widget.setFont(font)
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("res/Load.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.bb_in_LBL.setFont(font)
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
```

```
self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bb_out_LN.setObjectName("bb_out_LN")
self.bb_out_LBL = QtWidgets.QLabel(self.widget)
self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
self.bb_out_LBL.setFont(font)
self.bb_out_LBL.setStyleSheet("")
self.bb_out_LBL.setLineWidth(4)
self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.sf_unit = QtWidgets.QLabel(self.widget)
self.sf_unit.setGeometry(QtCore.QRect(340, 300, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sf_unit.setFont(font)
self.sf_unit.setText("")
self.sf_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.sf_unit.setObjectName("sf_unit")
self.store_BTN = QtWidgets.QPushButton(self.widget)
self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.store_BTN.setFont(font)
self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.store_BTN.setObjectName("store_BTN")
self.cancel_BTN = QtWidgets.QPushButton(self.widget)
self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.cancel_BTN.setFont(font)
self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.cancel_BTN.setObjectName("cancel_BTN")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.cosPhi_unit = QtWidgets.QLabel(self.Parameters)
self.cosPhi_unit.setGeometry(QtCore.QRect(240, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cosPhi_unit.setFont(font)
self.cosPhi_unit.setText("")
self.cosPhi_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cosPhi_unit.setObjectName("cosPhi_unit")
self.sf_profile_RB = QtWidgets.QRadioButton(self.Parameters)
self.sf_profile_RB.setGeometry(QtCore.QRect(240, 210, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
self.sf_profile_RB.setFont(font)
self.sf_profile_RB.setObjectName("sf_profile_RB")
self.sf_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.sf_DSB.setGeometry(QtCore.QRect(160, 190, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sf_DSB.setFont(font)
self.sf_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sf_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.sf_DSB.setDecimals(5)
self.sf_DSB.setMaximum(1.0)
self.sf_DSB.setSingleStep(0.01)
self.sf_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.sf_DSB.setProperty("value", 0.0)
self.sf_DSB.setObjectName("sf_DSB")
self.cosPhi_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.cosPhi_DSB.setGeometry(QtCore.QRect(160, 100, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cosPhi_DSB.setFont(font)
self.cosPhi_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cosPhi_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cosPhi_DSB.setDecimals(5)
self.cosPhi_DSB.setMinimum(-1.0)
self.cosPhi_DSB.setMaximum(1.0)
self.cosPhi_DSB.setSingleStep(0.01)
self.cosPhi_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cosPhi_DSB.setProperty("value", 0.0)
self.cosPhi_DSB.setObjectName("cosPhi_DSB")
self.s_unit = QtWidgets.QLabel(self.Parameters)
self.s_unit.setGeometry(QtCore.QRect(240, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.s_unit.setFont(font)
self.s_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.s_unit.setObjectName("s_unit")
self.i_unit = QtWidgets.QLabel(self.Parameters)
self.i_unit.setGeometry(QtCore.QRect(240, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.i_unit.setFont(font)
self.i_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.i_unit.setObjectName("i_unit")
self.sf_const_RB = QtWidgets.QRadioButton(self.Parameters)
self.sf_const_RB.setGeometry(QtCore.QRect(240, 190, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
self.sf_const_RB.setFont(font)
self.sf_const_RB.setObjectName("sf_const_RB")
```

```
self.sfi_LBL = QtWidgets.QLabel(self.Parameters)
self.sfi_LBL.setGeometry(QtCore.QRect(60, 190, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.sfi_LBL.setFont(font)
self.sfi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sfi_LBL.setObjectName("sfi_LBL")
self.s_LBL = QtWidgets.QLabel(self.Parameters)
self.s_LBL.setGeometry(QtCore.QRect(60, 130, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.s_LBL.setFont(font)
self.s_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.s_LBL.setObjectName("s_LBL")
self.i_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.i_DSB.setGeometry(QtCore.QRect(160, 160, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.i_DSB.setFont(font)
self.i_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.i_DSB.setReadOnly(False)
self.i_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.i_DSB.setDecimals(3)
self.i_DSB.setMaximum(999999.999)
self.i_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.i_DSB.setProperty("value", 0.0)
self.i_DSB.setObjectName("i_DSB")
self.cosPhi_LBL = QtWidgets.QLabel(self.Parameters)
self.cosPhi_LBL.setGeometry(QtCore.QRect(60, 100, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.cosPhi_LBL.setFont(font)
self.cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cosPhi_LBL.setObjectName("cosPhi_LBL")
self.s_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.s_DSB.setGeometry(QtCore.QRect(160, 130, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.s_DSB.setFont(font)
self.s_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.s_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.s_DSB.setDecimals(3)
self.s_DSB.setMaximum(999999.999)
self.s_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.s_DSB.setProperty("value", 0.0)
self.s_DSB.setObjectName("s_DSB")
self.i_LBL = QtWidgets.QLabel(self.Parameters)
self.i_LBL.setGeometry(QtCore.QRect(60, 160, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.i_LBL.setFont(font)
self.i_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.i_LBL.setObjectName("i_LBL")
self.q_unit = QtWidgets.QLabel(self.Parameters)
self.q_unit.setGeometry(QtCore.QRect(240, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.q_unit.setFont(font)
self.q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.q_unit.setObjectName("q_unit")
self.q_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.q_DSB.setGeometry(QtCore.QRect(160, 70, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.q_DSB.setFont(font)
self.q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.q_DSB.setDecimals(3)
self.q_DSB.setMaximum(999999.999)
self.q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.q_DSB.setProperty("value", 0.0)
self.q_DSB.setObjectName("q_DSB")
self.q_LBL = QtWidgets.QLabel(self.Parameters)
self.q_LBL.setGeometry(QtCore.QRect(60, 70, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.q_LBL.setFont(font)
self.q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.q_LBL.setObjectName("q_LBL")
self.control_CB = QtWidgets.QComboBox(self.Parameters)
self.control_CB.setGeometry(QtCore.QRect(160, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.control_CB.setFont(font)
self.control_CB.setObjectName("control_CB")
self.control_CB.addItem("")
self.control_CB.addItem("")
self.control_CB.addItem("")
self.control_CB.addItem("")
self.control_CB.addItem("")
self.p_LBL = QtWidgets.QLabel(self.Parameters)
self.p_LBL.setGeometry(QtCore.QRect(60, 40, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
```

```
font.setWeight(75)
self.p_LBL.setFont(font)
self.p_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.p_LBL.setObjectName("p_LBL")
self.p_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.p_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.p_DSB.setFont(font)
self.p_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.p_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.p_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.p_DSB.setDecimals(3)
self.p_DSB.setMaximum(999999.999)
self.p_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.p_DSB.setProperty("value", 0.0)
self.p_DSB.setObjectName("p_DSB")
self.lftype_LBL = QtWidgets.QLabel(self.Parameters)
self.lftype_LBL.setGeometry(QtCore.QRect(60, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.lftype_LBL.setFont(font)
self.lftype_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.lftype_LBL.setObjectName("lftype_LBL")
self.p_unit = QtWidgets.QLabel(self.Parameters)
self.p_unit.setGeometry(QtCore.QRect(240, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.p_unit.setFont(font)
self.p_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.p_unit.setObjectName("p_unit")
self.tabWidget.addTab(self.Parameters, "")
self.LoadFLow = QtWidgets.QWidget()
self.LoadFLow.setObjectName("LoadFLow")
self.res_P_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_P_P1_DSB.setEnabled(False)
self.res_P_P1_DSB.setGeometry(QtCore.QRect(90, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_DSB.setFont(font)
self.res_P_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P1_DSB.setDecimals(3)
self.res_P_P1_DSB.setMinimum(-999999.999)
self.res_P_P1_DSB.setMaximum(999999.999)
self.res_P_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P1_DSB.setProperty("value", 0.0)
self.res_P_P1_DSB.setObjectName("res_P_P1_DSB")
self.res_Iangle_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_Iangle_P1_DSB.setEnabled(False)
self.res_Iangle_P1_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_DSB.setFont(font)
self.res_Iangle_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P1_DSB.setDecimals(3)
self.res_Iangle_P1_DSB.setMinimum(-999999.999)
self.res_Iangle_P1_DSB.setMaximum(999999.999)
self.res_Iangle_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P1_DSB.setProperty("value", 0.0)
self.res_Iangle_P1_DSB.setObjectName("res_Iangle_P1_DSB")
self.res_Iangle_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_Iangle_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Iangle_LBL.setFont(font)
self.res_Iangle_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_LBL.setObjectName("res_Iangle_LBL")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_U_LBL.setGeometry(QtCore.QRect(0, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.Por1_LBL = QtWidgets.QLabel(self.LoadFLow)
self.Por1_LBL.setGeometry(QtCore.QRect(90, 10, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Por1_LBL.setFont(font)
self.Por1_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Por1_LBL.setObjectName("Por1_LBL")
self.res_U_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_U_P1_DSB.setEnabled(False)
self.res_U_P1_DSB.setGeometry(QtCore.QRect(90, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_DSB.setFont(font)
self.res_U_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P1_DSB.setDecimals(3)
self.res_U_P1_DSB.setMaximum(999999.999)
self.res_U_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
```

```
self.res_U_P1_DSB.setProperty("value", 0.0)
self.res_U_P1_DSB.setObjectName("res_U_P1_DSB")
self.res_Iangle_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_Iangle_P1_unit.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_unit.setFont(font)
self.res_Iangle_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_unit.setObjectName("res_Iangle_P1_unit")
self.res_I_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_I_LBL.setGeometry(QtCore.QRect(0, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_I_LBL.setFont(font)
self.res_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_LBL.setObjectName("res_I_LBL")
self.res_S_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_S_P1_DSB.setEnabled(False)
self.res_S_P1_DSB.setGeometry(QtCore.QRect(90, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_DSB.setFont(font)
self.res_S_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P1_DSB.setDecimals(3)
self.res_S_P1_DSB.setMaximum(999999.999)
self.res_S_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P1_DSB.setProperty("value", 0.0)
self.res_S_P1_DSB.setObjectName("res_S_P1_DSB")
self.res_P_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_P_P1_unit.setGeometry(QtCore.QRect(170, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_unit.setFont(font)
self.res_P_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P1_unit.setObjectName("res_P_P1_unit")
self.res_S_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_S_LBL.setGeometry(QtCore.QRect(0, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_S_LBL.setFont(font)
self.res_S_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_LBL.setObjectName("res_S_LBL")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(0, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_I_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_I_P1_unit.setGeometry(QtCore.QRect(170, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_unit.setFont(font)
self.res_I_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P1_unit.setObjectName("res_I_P1_unit")
self.res_cosPhi_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_cosPhi_P1_unit.setGeometry(QtCore.QRect(170, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_unit.setFont(font)
self.res_cosPhi_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_unit.setObjectName("res_cosPhi_P1_unit")
self.res_Q_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_Q_P1_unit.setGeometry(QtCore.QRect(170, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_unit.setFont(font)
self.res_Q_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P1_unit.setObjectName("res_Q_P1_unit")
self.res_cosPhi_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_cosPhi_LBL.setGeometry(QtCore.QRect(0, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_cosPhi_LBL.setFont(font)
self.res_cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_LBL.setObjectName("res_cosPhi_LBL")
self.res_Q_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_Q_P1_DSB.setEnabled(False)
self.res_Q_P1_DSB.setGeometry(QtCore.QRect(90, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_DSB.setFont(font)
self.res_Q_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P1_DSB.setDecimals(3)
self.res_Q_P1_DSB.setMinimum(-999999.999)
self.res_Q_P1_DSB.setMaximum(999999.999)
self.res_Q_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P1_DSB.setProperty("value", 0.0)
```

```
self.res_Q_P1_DSB.setObjectName("res_Q_P1_DSB")
self.res_P_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_P_LBL.setGeometry(QtCore.QRect(0, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_P_LBL.setFont(font)
self.res_P_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_LBL.setObjectName("res_P_LBL")
self.res_cosPhi_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_cosPhi_P1_DSB.setEnabled(False)
self.res_cosPhi_P1_DSB.setGeometry(QtCore.QRect(90, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_DSB.setFont(font)
self.res_cosPhi_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P1_DSB.setDecimals(3)
self.res_cosPhi_P1_DSB.setMaximum(999999.999)
self.res_cosPhi_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P1_DSB.setProperty("value", 0.0)
self.res_cosPhi_P1_DSB.setObjectName("res_cosPhi_P1_DSB")
self.res_S_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_S_P1_unit.setGeometry(QtCore.QRect(170, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_unit.setFont(font)
self.res_S_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P1_unit.setObjectName("res_S_P1_unit")
self.res_Q_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_Q_LBL.setGeometry(QtCore.QRect(0, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Q_LBL.setFont(font)
self.res_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_LBL.setObjectName("res_Q_LBL")
self.res_U_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_U_P1_unit.setGeometry(QtCore.QRect(170, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_unit.setFont(font)
self.res_U_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P1_unit.setObjectName("res_U_P1_unit")
self.res_I_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_I_P1_DSB.setEnabled(False)
self.res_I_P1_DSB.setGeometry(QtCore.QRect(90, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_DSB.setFont(font)
self.res_I_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P1_DSB.setDecimals(3)
self.res_I_P1_DSB.setMaximum(999999.999)
self.res_I_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P1_DSB.setProperty("value", 0.0)
self.res_I_P1_DSB.setObjectName("res_I_P1_DSB")
self.tabWidget.addTab(self.LoadFLow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.unmet_cost_LBL = QtWidgets.QLabel(self.EMS)
self.unmet_cost_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.unmet_cost_LBL.setFont(font)
self.unmet_cost_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.unmet_cost_LBL.setObjectName("unmet_cost_LBL")
self.unmet_cost_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.unmet_cost_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.unmet_cost_DSB.setFont(font)
self.unmet_cost_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.unmet_cost_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.unmet_cost_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.unmet_cost_DSB.setDecimals(3)
self.unmet_cost_DSB.setMaximum(999999.999)
self.unmet_cost_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.unmet_cost_DSB.setProperty("value", 0.0)
self.unmet_cost_DSB.setObjectName("unmet_cost_DSB")
self.unmet_cost_unit = QtWidgets.QLabel(self.EMS)
self.unmet_cost_unit.setGeometry(QtCore.QRect(240, 10, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.unmet_cost_unit.setFont(font)
self.unmet_cost_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.unmet_cost_unit.setObjectName("unmet_cost_unit")
self.tabWidget.addTab(self.EMS, "")
self.Reliability = QtWidgets.QWidget()
self.Reliability.setObjectName("Reliability")
self.rel_results_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 65, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
```

```
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_alfa_unit = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_unit.setGeometry(QtCore.QRect(240, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_R_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_R_LBL.setGeometry(QtCore.QRect(160, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_lambda_unit = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_beta_unit = QtWidgets.QLabel(self.Reliability)
self.rel_beta_unit.setGeometry(QtCore.QRect(240, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(0, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
self.rel_T0_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_T0_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(160, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(160, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.bottom_LN_2 = QtWidgets.QFrame(self.Reliability)
self.bottom_LN_2.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN_2.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_2.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_2.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_2.setObjectName("bottom_LN_2")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(160, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
```

```
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(240, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_beta_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_beta_LBL.setGeometry(QtCore.QRect(0, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(240, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_T0_unit = QtWidgets.QLabel(self.Reliability)
self.rel_T0_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(0, 170, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(160, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(160, 140, 71, 21))
font = QtGui.QFont()
```

```
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_anni_DSB.setEnabled(False)
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(0, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_R_unit = QtWidgets.QLabel(self.Reliability)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_R_unit.setObjectName("rel_R_unit")
self.rel_R_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
self.rel_R_LE.setFont(font)
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
self.rel_lambda_LE.setFont(font)
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.rel_alfa_unit.raise_()
self.rel_R_LBL.raise_()
self.rel_lambda_unit.raise_()
self.rel_beta_unit.raise_()
self.rel_alfa_LBL.raise_()
self.rel_T0_LBL.raise_()
self.rel_beta_DSB.raise_()
self.rel_alfa_DSB.raise_()
self.rel_lambda_LBL.raise_()
self.bottom_LN_2.raise_()
self.rel_Pi_Q_DSB.raise_()
self.rel_T0_DSB.raise_()
self.rel_MTBF_ore_unit.raise_()
self.rel_MTBF_ore_DSB.raise_()
self.rel_Pi_Q_unit.raise_()
self.rel_MTBF_ore_LBL.raise_()
self.rel_beta_LBL.raise_()
self.rel_Pi_E_unit.raise_()
self.rel_T0_unit.raise_()
self.rel_Pi_Q_LBL.raise_()
self.rel_MTBF_anni_LBL.raise_()
self.rel_Pi_E_DSB.raise_()
self.rel_MTBF_anni_DSB.raise_()
self.rel_Pi_E_LBL.raise_()
self.rel_R_unit.raise_()
self.rel_R_LE.raise_()
self.rel_lambda_LE.raise_()
self.rel_results_LBL.raise_()
self.rel_MTBF_anni_unit.raise_()
self.tabWidget.addTab(self.Reliability, "")
self.bb_out_LBL.raise_()
self.bb_in_LBL.raise_()
self.cub_out_LBL.raise_()
self.top_LN.raise_()
self.elem_name_LN.raise_()
self.elem_name_LBL.raise_()
self.type_LBL.raise_()
self.type_cap_LBL.raise_()
self.cub_in_LBL.raise_()
self.symbol_LBL.raise_()
```

```
        self.ot_Frame_LN.raise_()
        self.vdx_frame_LN.raise_()
        self.vsx_frame_LN.raise_()
        self.ob_frame_LN.raise_()
        self.bb_in_LN.raise_()
        self.bb_out_LN.raise_()
        self.sf_unit.raise_()
        self.store_BTN.raise_()
        self.cancel_BTN.raise_()
        self.tabWidget.raise_()

        self.retranslateUi(Form)
        self.tabWidget.setCurrentIndex(3)
        QtCore.QMetaObject.connectSlotsByName(Form)
        Form.setTabOrder(self.tabWidget, self.control_CB)
        Form.setTabOrder(self.control_CB, self.p_DSB)
        Form.setTabOrder(self.p_DSB, self.q_DSB)
        Form.setTabOrder(self.q_DSB, self.cosPhi_DSB)
        Form.setTabOrder(self.cosPhi_DSB, self.s_DSB)
        Form.setTabOrder(self.s_DSB, self.i_DSB)
        Form.setTabOrder(self.i_DSB, self.sf_DSB)
        Form.setTabOrder(self.sf_DSB, self.sf_const_RB)
        Form.setTabOrder(self.sf_const_RB, self.sf_profile_RB)
        Form.setTabOrder(self.sf_profile_RB, self.unmet_cost_DSB)
        Form.setTabOrder(self.unmet_cost_DSB, self.rel_T0_DSB)
        Form.setTabOrder(self.rel_T0_DSB, self.rel_alfa_DSB)
        Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
        Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
        Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
        Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
        Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
        Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_I_P1_DSB)
        Form.setTabOrder(self.res_I_P1_DSB, self.res_Iangle_P1_DSB)
        Form.setTabOrder(self.res_Iangle_P1_DSB, self.res_P_P1_DSB)
        Form.setTabOrder(self.res_P_P1_DSB, self.res_Q_P1_DSB)
        Form.setTabOrder(self.res_Q_P1_DSB, self.res_S_P1_DSB)
        Form.setTabOrder(self.res_S_P1_DSB, self.res_cosPhi_P1_DSB)
        Form.setTabOrder(self.res_cosPhi_P1_DSB, self.res_U_P1_DSB)
        Form.setTabOrder(self.res_U_P1_DSB, self.store_BTN)
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria:"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.store_BTN.setText(_translate("Form", "Salva"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))
        self.sf_profile_RB.setText(_translate("Form", "Profilo"))
        self.s_unit.setText(_translate("Form", "kVA"))
        self.i_unit.setText(_translate("Form", "A"))
        self.sf_const_RB.setText(_translate("Form", "Costante"))
        self.sfi_LBL.setText(_translate("Form", "scala"))
        self.s_LBL.setText(_translate("Form", "S"))
        self.cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.i_LBL.setText(_translate("Form", "I"))
        self.q_unit.setText(_translate("Form", "kVA"))
        self.q_LBL.setText(_translate("Form", "Q"))
        self.control_CB.setItemText(0, _translate("Form", "P, Q"))
        self.control_CB.setItemText(1, _translate("Form", "P, cosPhi"))
        self.control_CB.setItemText(2, _translate("Form", "I, cosPhi"))
        self.control_CB.setItemText(3, _translate("Form", "P, I"))
        self.control_CB.setItemText(4, _translate("Form", "S, cosPhi"))
        self.p_LBL.setText(_translate("Form", "P"))
        self.lftype_LBL.setText(_translate("Form", "Parametri LF"))
        self.p_unit.setText(_translate("Form", "kW"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Parameters), _translate("Form", "Parametri"))
        self.res_Iangle_LBL.setText(_translate("Form", "I angle"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.Port1_LBL.setText(_translate("Form", "Port 1"))
        self.res_Iangle_P1_unit.setText(_translate("Form", "°"))
        self.res_I_LBL.setText(_translate("Form", " I"))
        self.res_P_P1_unit.setText(_translate("Form", "kW"))
        self.res_S_LBL.setText(_translate("Form", "S"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMIT VIOLATED"))
        self.res_I_P1_unit.setText(_translate("Form", "A"))
        self.res_cosPhi_P1_unit.setText(_translate("Form", "-"))
        self.res_Q_P1_unit.setText(_translate("Form", "kVA"))
        self.res_cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.res_P_LBL.setText(_translate("Form", "P"))
        self.res_S_P1_unit.setText(_translate("Form", "kVA"))
        self.res_Q_LBL.setText(_translate("Form", "Q"))
        self.res_U_P1_unit.setText(_translate("Form", "kV"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFLow), _translate("Form", "LoadFlow"))
        self.unmet_cost_LBL.setText(_translate("Form", "Costo energia non fornita"))
        self.unmet_cost_unit.setText(_translate("Form", "€/kWh"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
```

```
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_TQ_unit.setText(_translate("Form", "°C"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Reliability), _translate("Form", "Affidabilità"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.3.3 ACnode

4.4.3.3.1 acnode.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .acnodeUI import Ui_Form
from __shared__ import variables as v
import copy


class ACnode(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(ACnode, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.bb_in_LBL.setVisible(False)
        self.ui.bb_in_LN.setVisible(False)
        self.ui.bb_out_LN.setVisible(False)
        self.ui.bb_out_LBL.setVisible(False)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")
        self.ui.tabWidget.setCurrentIndex(0)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])

        self.fill()

    #
    def store(self):
        self.par['Ur'] = self.ui.u_DSB.value()

        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()

        v.elements[self.element]['parameters'] = copy.deepcopy(self.par)
        v.elements[self.element]['ems'] = copy.deepcopy(self.ems)
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)
        v.elements[self.element]['results'] = copy.deepcopy(self.res)

    #
    def fill(self):
        self.ui.u_DSB.setValue(self.par['Ur'])

        if self.res != {}:
            self.fill_results()
        self.ui.tabWidget.setTabVisible(3, self.res != {})
        self.fill_reliability()

    #
    def fill_reliability(self):
        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
        for par in self.rel['results']:
            try:
                self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
            except:
                if self.rel['results'][par] == 0:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('0.0')
                elif self.rel['results'][par] < 0.01:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
                else:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])

    #
    def fill_results(self):
        results = ['Pgen', 'Pload', 'Qgen', 'Qload', 'U', 'Un', 'Up']
        for result in results:
            self.ui.__getattribute__('res_' + result + '_DSB').setValue(self.res[result])
        self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])
```

4.4.3.3.2 acnodeUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'acnodeUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(492, 505)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("res/Load.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.bb_in_LBL.setFont(font)
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_out_LN.setObjectName("bb_out_LN")
        self.bb_out_LBL = QtWidgets.QLabel(self.widget)
        self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
self.bb_out_LBL.setFont(font)
self.bb_out_LBL.setStyleSheet("")
self.bb_out_LBL.setLineWidth(4)
self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.store_BTN = QtWidgets.QPushButton(self.widget)
self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.store_BTN.setFont(font)
self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.store_BTN.setObjectName("store_BTN")
self.cancel_BTN = QtWidgets.QPushButton(self.widget)
self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.cancel_BTN.setFont(font)
self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.cancel_BTN.setObjectName("cancel_BTN")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.u_LBL = QtWidgets.QLabel(self.Parameters)
self.u_LBL.setGeometry(QtCore.QRect(60, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.u_LBL.setFont(font)
self.u_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.u_LBL.setObjectName("u_LBL")
self.u_unit = QtWidgets.QLabel(self.Parameters)
self.u_unit.setGeometry(QtCore.QRect(240, 10, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.u_unit.setFont(font)
self.u_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.u_unit.setObjectName("u_unit")
self.u_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.u_DSB.setGeometry(QtCore.QRect(160, 10, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.u_DSB.setFont(font)
self.u_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.u_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.u_DSB.setDecimals(3)
self.u_DSB.setMaximum(999999.999)
self.u_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.u_DSB.setProperty("value", 0.0)
self.u_DSB.setObjectName("u_DSB")
self.tabWidget.addTab(self.Parameters, "")
self.LoadFlow = QtWidgets.QWidget()
self.LoadFlow.setObjectName("LoadFlow")
self.res_Pload_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Pload_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Pload_LBL.setFont(font)
self.res_Pload_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Pload_LBL.setObjectName("res_Pload_LBL")
self.res_Qgen_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Qgen_LBL.setGeometry(QtCore.QRect(0, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Qgen_LBL.setFont(font)
self.res_Qgen_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qgen_LBL.setObjectName("res_Qgen_LBL")
self.res_U_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_DSB.setEnabled(False)
self.res_U_DSB.setGeometry(QtCore.QRect(90, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_DSB.setFont(font)
self.res_U_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_DSB.setDecimals(3)
self.res_U_DSB.setMaximum(999999.999)
self.res_U_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_DSB.setProperty("value", 0.0)
self.res_U_DSB.setObjectName("res_U_DSB")
self.res_Pload_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Pload_DSB.setEnabled(False)
self.res_Pload_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Pload_DSB.setFont(font)
self.res_Pload_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Pload_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Pload_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Pload_DSB.setDecimals(3)
self.res_Pload_DSB.setMinimum(-999999.99)
self.res_Pload_DSB.setMaximum(999999.999)
self.res_Pload_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Pload_DSB.setProperty("value", 0.0)
```

```
self.res_Pload_DSB.setObjectName("res_Pload_DSB")
self.res_U_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_unit.setGeometry(QtCore.QRect(170, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_unit.setFont(font)
self.res_U_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_unit.setObjectName("res_U_unit")
self.res_Pload_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Pload_unit.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Pload_unit.setFont(font)
self.res_Pload_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Pload_unit.setObjectName("res_Pload_unit")
self.res_Pgen_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Pgen_LBL.setGeometry(QtCore.QRect(0, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Pgen_LBL.setFont(font)
self.res_Pgen_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Pgen_LBL.setObjectName("res_Pgen_LBL")
self.res_Up_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Up_unit.setGeometry(QtCore.QRect(170, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Up_unit.setFont(font)
self.res_Up_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Up_unit.setObjectName("res_Up_unit")
self.res_Un_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Un_LBL.setGeometry(QtCore.QRect(0, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Un_LBL.setFont(font)
self.res_Un_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Un_LBL.setObjectName("res_Un_LBL")
self.res_Pgen_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Pgen_unit.setGeometry(QtCore.QRect(170, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Pgen_unit.setFont(font)
self.res_Pgen_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Pgen_unit.setObjectName("res_Pgen_unit")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_U_LBL.setGeometry(QtCore.QRect(0, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.sf_unit = QtWidgets.QLabel(self.LoadFlow)
self.sf_unit.setGeometry(QtCore.QRect(168, 215, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sf_unit.setFont(font)
self.sf_unit.setText("")
self.sf_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.sf_unit.setObjectName("sf_unit")
self.res_Qgen_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Qgen_unit.setGeometry(QtCore.QRect(170, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qgen_unit.setFont(font)
self.res_Qgen_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Qgen_unit.setObjectName("res_Qgen_unit")
self.res_Pgen_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Pgen_DSB.setEnabled(False)
self.res_Pgen_DSB.setGeometry(QtCore.QRect(90, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Pgen_DSB.setFont(font)
self.res_Pgen_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Pgen_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Pgen_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Pgen_DSB.setDecimals(3)
self.res_Pgen_DSB.setMinimum(-999999.99)
self.res_Pgen_DSB.setMaximum(999999.999)
self.res_Pgen_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Pgen_DSB.setProperty("value", 0.0)
self.res_Pgen_DSB.setObjectName("res_Pgen_DSB")
self.res_Up_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Up_LBL.setGeometry(QtCore.QRect(0, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Up_LBL.setFont(font)
self.res_Up_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Up_LBL.setObjectName("res_Up_LBL")
self.res_Qload_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Qload_LBL.setGeometry(QtCore.QRect(0, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
```

```
self.res_Qload_LBL.setFont(font)
self.res_Qload_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qload_LBL.setObjectName("res_Qload_LBL")
self.Por1_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Por1_LBL.setGeometry(QtCore.QRect(90, 10, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Por1_LBL.setFont(font)
self.Por1_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Por1_LBL.setObjectName("Por1_LBL")
self.res_Un_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Un_unit.setGeometry(QtCore.QRect(170, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Un_unit.setFont(font)
self.res_Un_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Un_unit.setObjectName("res_Un_unit")
self.res_Qload_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Qload_DSB.setEnabled(False)
self.res_Qload_DSB.setGeometry(QtCore.QRect(90, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qload_DSB.setFont(font)
self.res_Qload_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Qload_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qload_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Qload_DSB.setDecimals(3)
self.res_Qload_DSB.setMinimum(-999999.99)
self.res_Qload_DSB.setMaximum(999999.999)
self.res_Qload_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Qload_DSB.setProperty("value", 0.0)
self.res_Qload_DSB.setObjectName("res_Qload_DSB")
self.res_Qgen_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Qgen_DSB.setEnabled(False)
self.res_Qgen_DSB.setGeometry(QtCore.QRect(90, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qgen_DSB.setFont(font)
self.res_Qgen_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Qgen_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qgen_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Qgen_DSB.setDecimals(3)
self.res_Qgen_DSB.setMinimum(-999999.99)
self.res_Qgen_DSB.setMaximum(999999.999)
self.res_Qgen_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Qgen_DSB.setProperty("value", 0.0)
self.res_Qgen_DSB.setObjectName("res_Qgen_DSB")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(0, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_Up_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Up_DSB.setEnabled(False)
self.res_Up_DSB.setGeometry(QtCore.QRect(90, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Up_DSB.setFont(font)
self.res_Up_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Up_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Up_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Up_DSB.setDecimals(3)
self.res_Up_DSB.setMaximum(999999.999)
self.res_Up_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Up_DSB.setProperty("value", 0.0)
self.res_Up_DSB.setObjectName("res_Up_DSB")
self.res_Un_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Un_DSB.setEnabled(False)
self.res_Un_DSB.setGeometry(QtCore.QRect(90, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Un_DSB.setFont(font)
self.res_Un_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Un_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Un_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Un_DSB.setDecimals(3)
self.res_Un_DSB.setMaximum(999999.999)
self.res_Un_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Un_DSB.setProperty("value", 0.0)
self.res_Un_DSB.setObjectName("res_Un_DSB")
self.res_Qload_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Qload_unit.setGeometry(QtCore.QRect(170, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qload_unit.setFont(font)
self.res_Qload_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Qload_unit.setObjectName("res_Qload_unit")
self.tabWidget.addTab(self.LoadFlow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.tabWidget.addTab(self.EMS, "")
self.Reliability = QtWidgets.QWidget()
self.Reliability.setObjectName("Reliability")
self.rel_results_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 65, 21))
font = QtGui.QFont()
font.setPointSize(8)
```

```
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_alfa_unit = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_unit.setGeometry(QtCore.QRect(240, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_R_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_R_LBL.setGeometry(QtCore.QRect(160, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_lambda_unit = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_beta_unit = QtWidgets.QLabel(self.Reliability)
self.rel_beta_unit.setGeometry(QtCore.QRect(240, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(0, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
self.rel_T0_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_T0_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(160, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(160, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.bottom_LN_2 = QtWidgets.QFrame(self.Reliability)
self.bottom_LN_2.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN_2.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_2.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_2.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_2.setObjectName("bottom_LN_2")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(160, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
```

```
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(240, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_beta_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_beta_LBL.setGeometry(QtCore.QRect(0, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(240, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_T0_unit = QtWidgets.QLabel(self.Reliability)
self.rel_T0_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(0, 170, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(160, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
```

```
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(160, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_anni_DSB.setEnabled(False)
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(0, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_R_unit = QtWidgets.QLabel(self.Reliability)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_R_unit.setObjectName("rel_R_unit")
self.rel_R_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.rel_alfa_unit.raise_()
self.rel_R_LBL.raise_()
self.rel_lambda_unit.raise_()
self.rel_beta_unit.raise_()
self.rel_alfa_LBL.raise_()
self.rel_T0_LBL.raise_()
self.rel_beta_DSB.raise_()
self.rel_alfa_DSB.raise_()
self.rel_lambda_LBL.raise_()
self.bottom_LN_2.raise_()
self.rel_Pi_Q_DSB.raise_()
self.rel_T0_DSB.raise_()
self.rel_MTBF_ore_unit.raise_()
self.rel_MTBF_ore_DSB.raise_()
self.rel_Pi_Q_unit.raise_()
self.rel_MTBF_ore_LBL.raise_()
self.rel_beta_LBL.raise_()
self.rel_Pi_E_unit.raise_()
self.rel_T0_unit.raise_()
self.rel_Pi_Q_LBL.raise_()
self.rel_MTBF_anni_LBL.raise_()
self.rel_Pi_E_DSB.raise_()
self.rel_MTBF_anni_DSB.raise_()
self.rel_Pi_E_LBL.raise_()
self.rel_R_unit.raise_()
self.rel_R_LE.raise_()
self.rel_lambda_LE.raise_()
self.rel_results_LBL.raise_()
self.rel_MTBF_anni_unit.raise_()
self.tabWidget.addTab(self.Reliability, "")
self.bb_out_LBL.raise_()
self.bb_in_LBL.raise_()
self.cub_out_LBL.raise_()
self.top_LN.raise_()
self.elem_name_LN.raise_()
self.elem_name_LBL.raise_()
self.type_LBL.raise_()
self.type_cap_LBL.raise_()
self.cub_in_LBL.raise_()
self.symbol_LBL.raise_()
self.ot_Frame_LN.raise_()
self.vdx_frame_LN.raise_()
self.vsx_frame_LN.raise_()
```

```
        self.ob_frame_LN.raise_()
        self.bb_in_LN.raise_()
        self.bb_out_LN.raise_()
        self.store_BTN.raise_()
        self.cancel_BTN.raise_()
        self.tabWidget.raise_()

        self.retranslateUi(Form)
        self.tabWidget.setCurrentIndex(3)
        QtCore.QMetaObject.connectSlotsByName(Form)
        Form.setTabOrder(self.tabWidget, self.u_DSB)
        Form.setTabOrder(self.u_DSB, self.rel_T0_DSB)
        Form.setTabOrder(self.rel_T0_DSB, self.rel_alfa_DSB)
        Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
        Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
        Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
        Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
        Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
        Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_Pgen_DSB)
        Form.setTabOrder(self.res_Pgen_DSB, self.res_Pload_DSB)
        Form.setTabOrder(self.res_Pload_DSB, self.res_Qgen_DSB)
        Form.setTabOrder(self.res_Qgen_DSB, self.res_Qload_DSB)
        Form.setTabOrder(self.res_Qload_DSB, self.res_U_DSB)
        Form.setTabOrder(self.res_U_DSB, self.res_Un_DSB)
        Form.setTabOrder(self.res_Un_DSB, self.res_Up_DSB)
        Form.setTabOrder(self.res_Up_DSB, self.store_BTN)
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria:"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.store_BTN.setText(_translate("Form", "Salva"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))
        self.u_LBL.setText(_translate("Form", "Un"))
        self.u_unit.setText(_translate("Form", "kV"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Parameters), _translate("Form", "Parametri"))
        self.res_Pload_LBL.setText(_translate("Form", "P load"))
        self.res_Qgen_LBL.setText(_translate("Form", "Q gen"))
        self.res_U_unit.setText(_translate("Form", "kV"))
        self.res_Pload_unit.setText(_translate("Form", "kW"))
        self.res_Pgen_LBL.setText(_translate("Form", "P gen"))
        self.res_Up_unit.setText(_translate("Form", "-"))
        self.res_Un_LBL.setText(_translate("Form", "Un"))
        self.res_Pgen_unit.setText(_translate("Form", "kW"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.res_Qgen_unit.setText(_translate("Form", "kVA"))
        self.res_Up_LBL.setText(_translate("Form", "Up"))
        self.res_Qload_LBL.setText(_translate("Form", "Q load"))
        self.Por1_LBL.setText(_translate("Form", "Nodo 1"))
        self.res_Un_unit.setText(_translate("Form", "kV"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMITI VIOLATI"))
        self.res_Qload_unit.setText(_translate("Form", "kVA"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFlow), _translate("Form", "LoadFlow"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_T0_unit.setText(_translate("Form", "°C"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Reliability), _translate("Form", "Affidabilità"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.3.4 ACwind

4.4.3.4.1 acwind.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .acwindUI import Ui_Form
from __shared__ import variables as v
import copy


class ACwind(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(ACwind, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.bb_out_LN.setVisible(False)
        self.ui.bb_out_LBL.setVisible(False)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")

        for box in ['cap_pwr']:
            self.ui.__getattribute__(box + '_DSB').setStyleSheet("color: rgb(127, 127, 127);")
            self.ui.__getattribute__(box + '_DSB').setEnabled(False)

        self.ui.tabWidget.setCurrentIndex(0)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])

        self.bb = v.elements[element]['conn']['h']
        self.cubicle = v.elements[self.element]['conn'][self.bb]
        self.u = v.elements[self.bb]['parameters']['Ur']
        self.ui.bb_in_LBL.setText(self.bb)

        self.ui.symbol_LBL.setPixmap(QtGui.QPixmap("_images/Elements/ACwind/element.png"))
        self.switch_draw()
        self.fill()
        self.calculate()

        for attr in ['p', 'q', 's', 'cosPhi', 'sr']:
            self.ui.__getattribute__(attr + '_DSB').valueChanged.connect(self.calculate)
        self.ui.cub_in_LBL.mouseDoubleClickEvent = self.cub1_switch

    #
    def calculate(self):
        p = self.ui.p_DSB.value()
        q = self.ui.q_DSB.value()
        s = self.ui.s_DSB.value()
        cosPhi = self.ui.cosPhi_DSB.value()

        if self.ui.control_CB.currentIndex() == 0:
            s = (p ** 2 + q ** 2) ** 0.5
            cosPhi = p / s
            self.ui.s_DSB.setValue(s)
            self.ui.cosPhi_DSB.setValue(cosPhi)
        elif self.ui.control_CB.currentIndex() == 1:
            s = p / cosPhi
            q = (s ** 2 - p ** 2) ** 0.5
            self.ui.q_DSB.setValue(q)
            self.ui.s_DSB.setValue(s)
        else:
            p = s*cosPhi
            q = (s ** 2 - p ** 2) ** 0.5
            self.ui.p_DSB.setValue(p)
            self.ui.q_DSB.setValue(q)

        self.ui.cap_pwr_DSB.setValue(self.ui.sr_DSB.value())

    #
    def store(self):
        self.par['P'] = self.ui.p_DSB.value()
        self.par['Q'] = self.ui.q_DSB.value()
        self.par['S'] = self.ui.s_DSB.value()
        self.par['cosPhi'] = self.ui.cosPhi_DSB.value()
        self.par['Sr'] = self.ui.sr_DSB.value()
        self.par['control'] = self.ui.control_CB.currentIndex()
        self.par['units'] = int(self.ui.units_DSB.value())
        v.elements[self.element]['parameters'] = copy.deepcopy(self.par)

        v.elements[self.element]['conn'][self.bb] = self.cubicle

        self.ems['cap_pwr'] = self.ui.cap_pwr_DSB.value()
        v.elements[self.element]['ems'] = copy.deepcopy(self.ems)

        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)

    #
    def fill(self):
        self.ui.p_DSB.setValue(self.par['P'])
        self.ui.q_DSB.setValue(self.par['Q'])
        self.ui.s_DSB.setValue(self.par['S'])
        self.ui.cosPhi_DSB.setValue(self.par['cosPhi'])
        self.ui.sr_DSB.setValue(self.par['Sr'])
        self.ui.control_CB.setCurrentIndex(self.par['control'])

        self.ui.units_DSB.setValue(self.par['units'])
        self.control_mode(self.ui)

        self.calculate()

        if self.res != {}:
            self.fill_results()
```

```
    self.ui.tabWidget.setTabVisible(3, self.res != {})
    self.fill_reliability()

#
def fill_reliability(self):
    for par in ['T0', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
        self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
    for par in self.rel['results']:
        try:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
        except:
            if self.rel['results'][par] == 0:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('0')
            elif self.rel['results'][par] < 0.01:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
            else:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])

#
def fill_results(self):
    results = ['I', 'Iangle', 'P', 'Q', 'S', 'cosPhi', 'U']
    ports = ['P1']
    for port in ports:
        for result in results:
            self.ui.__getattribute__('res_' + result + '_' + port + '_DSB').setValue(self.res[result])
    self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])

#
def control_mode(self, ui):
    enabled = [True, True, False, False]
    if ui.control_CB.currentIndex() == 0:
        enabled = [True, True, False, False]
    elif ui.control_CB.currentIndex() == 1:
        enabled = [True, False, False, True]
        ui.cosPhi_DSB.setMinimum(0.00001)
    elif ui.control_CB.currentIndex() == 2:
        enabled = [False, False, True, True]

    i = 0
    for obj in [ui.p_DSB, ui.q_DSB, ui.s_DSB, ui.cosPhi_DSB]:
        if enabled[i]:
            obj.setStyleSheet("color: rgb(255, 255, 255);")
        else:
            obj.setStyleSheet("color: rgb(127, 127, 127);")
        obj.setEnabled(enabled[i])
        i += 1

#
def switch_draw(self):
    if self.cubicle:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
    else:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))

#
def cub1_switch(self, event):
    self.cubicle = not self.cubicle
    self.switch_draw()

#
def calc_profile(self, pu_profile):
    profile = []
    for data in pu_profile:
        profile.append(data * 1)
    return profile
```

4.4.3.4.2 acwindUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'acwindUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(494, 502)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("res/Load.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.bb_in_LBL.setFont(font)
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_out_LN.setObjectName("bb_out_LN")
        self.bb_out_LBL = QtWidgets.QLabel(self.widget)
        self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
self.bb_out_LBL.setFont(font)
self.bb_out_LBL.setStyleSheet("")
self.bb_out_LBL.setLineWidth(4)
self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.store_BTN = QtWidgets.QPushButton(self.widget)
self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.store_BTN.setFont(font)
self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.store_BTN.setObjectName("store_BTN")
self.cancel_BTN = QtWidgets.QPushButton(self.widget)
self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.cancel_BTN.setFont(font)
self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.cancel_BTN.setObjectName("cancel_BTN")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.cosPhi_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.cosPhi_DSB.setGeometry(QtCore.QRect(160, 190, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cosPhi_DSB.setFont(font)
self.cosPhi_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cosPhi_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cosPhi_DSB.setDecimals(5)
self.cosPhi_DSB.setMinimum(-1.0)
self.cosPhi_DSB.setMaximum(1.0)
self.cosPhi_DSB.setSingleStep(0.01)
self.cosPhi_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cosPhi_DSB.setProperty("value", 0.0)
self.cosPhi_DSB.setObjectName("cosPhi_DSB")
self.sf_profile_RB = QtWidgets.QRadioButton(self.Parameters)
self.sf_profile_RB.setGeometry(QtCore.QRect(240, 240, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
self.sf_profile_RB.setFont(font)
self.sf_profile_RB.setObjectName("sf_profile_RB")
self.units_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.units_DSB.setGeometry(QtCore.QRect(160, 10, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.units_DSB.setFont(font)
self.units_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.units_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.units_DSB.setDecimals(0)
self.units_DSB.setMaximum(1000.0)
self.units_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.units_DSB.setProperty("value", 0.0)
self.units_DSB.setObjectName("units_DSB")
self.s_LBL = QtWidgets.QLabel(self.Parameters)
self.s_LBL.setGeometry(QtCore.QRect(60, 160, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.s_LBL.setFont(font)
self.s_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.s_LBL.setObjectName("s_LBL")
self.control_CB = QtWidgets.QComboBox(self.Parameters)
self.control_CB.setGeometry(QtCore.QRect(160, 70, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.control_CB.setFont(font)
self.control_CB.setObjectName("control_CB")
self.control_CB.addItem("")
self.control_CB.addItem("")
self.control_CB.addItem("")
self.sr_LBL = QtWidgets.QLabel(self.Parameters)
self.sr_LBL.setGeometry(QtCore.QRect(60, 40, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.sr_LBL.setFont(font)
self.sr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sr_LBL.setObjectName("sr_LBL")
self.p_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.p_DSB.setGeometry(QtCore.QRect(160, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.p_DSB.setFont(font)
self.p_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.p_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.p_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.p_DSB.setDecimals(3)
self.p_DSB.setMaximum(999999.999)
self.p_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.p_DSB.setProperty("value", 0.0)
self.p_DSB.setObjectName("p_DSB")
self.sr_unit = QtWidgets.QLabel(self.Parameters)
self.sr_unit.setGeometry(QtCore.QRect(240, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
```

```
font.setWeight(50)
self.sr_unit.setFont(font)
self.sr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.sr_unit.setObjectName("sr_unit")
self.s_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.s_DSB.setGeometry(QtCore.QRect(160, 160, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.s_DSB.setFont(font)
self.s_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.s_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.s_DSB.setDecimals(3)
self.s_DSB.setMaximum(999999.999)
self.s_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.s_DSB.setProperty("value", 0.0)
self.s_DSB.setObjectName("s_DSB")
self.q_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.q_DSB.setGeometry(QtCore.QRect(160, 130, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.q_DSB.setFont(font)
self.q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.q_DSB.setDecimals(3)
self.q_DSB.setMaximum(999999.999)
self.q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.q_DSB.setProperty("value", 0.0)
self.q_DSB.setObjectName("q_DSB")
self.sr_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.sr_DSB.setGeometry(QtCore.QRect(160, 40, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sr_DSB.setFont(font)
self.sr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.sr_DSB.setDecimals(3)
self.sr_DSB.setMaximum(999999.999)
self.sr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.sr_DSB.setProperty("value", 0.0)
self.sr_DSB.setObjectName("sr_DSB")
self.sfi_LBL = QtWidgets.QLabel(self.Parameters)
self.sfi_LBL.setGeometry(QtCore.QRect(60, 220, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.sfi_LBL.setFont(font)
self.sfi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sfi_LBL.setObjectName("sfi_LBL")
self.units_unit = QtWidgets.QLabel(self.Parameters)
self.units_unit.setGeometry(QtCore.QRect(240, 10, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.units_unit.setFont(font)
self.units_unit.setText("")
self.units_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.units_unit.setObjectName("units_unit")
self.q_unit = QtWidgets.QLabel(self.Parameters)
self.q_unit.setGeometry(QtCore.QRect(240, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.q_unit.setFont(font)
self.q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.q_unit.setObjectName("q_unit")
self.cosPhi_unit = QtWidgets.QLabel(self.Parameters)
self.cosPhi_unit.setGeometry(QtCore.QRect(240, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cosPhi_unit.setFont(font)
self.cosPhi_unit.setText("")
self.cosPhi_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cosPhi_unit.setObjectName("cosPhi_unit")
self.s_unit = QtWidgets.QLabel(self.Parameters)
self.s_unit.setGeometry(QtCore.QRect(240, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.s_unit.setFont(font)
self.s_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.s_unit.setObjectName("s_unit")
self.p_LBL = QtWidgets.QLabel(self.Parameters)
self.p_LBL.setGeometry(QtCore.QRect(60, 100, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.p_LBL.setFont(font)
self.p_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.p_LBL.setObjectName("p_LBL")
self.sf_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.sf_DSB.setGeometry(QtCore.QRect(160, 220, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sf_DSB.setFont(font)
self.sf_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sf_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.sf_DSB.setDecimals(5)
self.sf_DSB.setMaximum(1.0)
self.sf_DSB.setSingleStep(0.01)
self.sf_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
```

```
self.sf_DSB.setProperty("value", 0.0)
self.sf_DSB.setObjectName("sf_DSB")
self.sf_const_RB = QtWidgets.QRadioButton(self.Parameters)
self.sf_const_RB.setGeometry(QtCore.QRect(240, 220, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
self.sf_const_RB.setFont(font)
self.sf_const_RB.setObjectName("sf_const_RB")
self.control_LBL = QtWidgets.QLabel(self.Parameters)
self.control_LBL.setGeometry(QtCore.QRect(60, 70, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.control_LBL.setFont(font)
self.control_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.control_LBL.setObjectName("control_LBL")
self.unts_LBL = QtWidgets.QLabel(self.Parameters)
self.unts_LBL.setGeometry(QtCore.QRect(60, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.unts_LBL.setFont(font)
self.unts_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.unts_LBL.setObjectName("unts_LBL")
self.p_unit = QtWidgets.QLabel(self.Parameters)
self.p_unit.setGeometry(QtCore.QRect(240, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.p_unit.setFont(font)
self.p_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.p_unit.setObjectName("p_unit")
self.q_LBL = QtWidgets.QLabel(self.Parameters)
self.q_LBL.setGeometry(QtCore.QRect(60, 130, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.q_LBL.setFont(font)
self.q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.q_LBL.setObjectName("q_LBL")
self.cosPhi_LBL = QtWidgets.QLabel(self.Parameters)
self.cosPhi_LBL.setGeometry(QtCore.QRect(60, 190, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.cosPhi_LBL.setFont(font)
self.cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cosPhi_LBL.setObjectName("cosPhi_LBL")
self.tabWidget.addTab(self.Parameters, "")
self.LoadFlow = QtWidgets.QWidget()
self.LoadFlow.setObjectName("LoadFlow")
self.res_Q_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P1_unit.setGeometry(QtCore.QRect(170, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_unit.setFont(font)
self.res_Q_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P1_unit.setObjectName("res_Q_P1_unit")
self.res_P_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P1_DSB.setEnabled(False)
self.res_P_P1_DSB.setGeometry(QtCore.QRect(90, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_DSB.setFont(font)
self.res_P_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P1_DSB.setDecimals(3)
self.res_P_P1_DSB.setMinimum(-999999.999)
self.res_P_P1_DSB.setMaximum(999999.999)
self.res_P_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P1_DSB.setProperty("value", 0.0)
self.res_P_P1_DSB.setObjectName("res_P_P1_DSB")
self.res_Iangle_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P1_DSB.setEnabled(False)
self.res_Iangle_P1_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_DSB.setFont(font)
self.res_Iangle_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P1_DSB.setDecimals(3)
self.res_Iangle_P1_DSB.setMinimum(-999999.999)
self.res_Iangle_P1_DSB.setMaximum(999999.999)
self.res_Iangle_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P1_DSB.setProperty("value", 0.0)
self.res_Iangle_P1_DSB.setObjectName("res_Iangle_P1_DSB")
self.Por1_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Por1_LBL.setGeometry(QtCore.QRect(90, 10, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Por1_LBL.setFont(font)
self.Por1_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Por1_LBL.setObjectName("Por1_LBL")
self.res_P_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_P_LBL.setGeometry(QtCore.QRect(0, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_P_LBL.setFont(font)
```

```
self.res_P_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_LBL.setObjectName("res_P_LBL")
self.res_cosPhi_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_LBL.setGeometry(QtCore.QRect(0, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_cosPhi_LBL.setFont(font)
self.res_cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_LBL.setObjectName("res_cosPhi_LBL")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_U_LBL.setGeometry(QtCore.QRect(0, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.res_cosPhi_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P1_unit.setGeometry(QtCore.QRect(170, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_unit.setFont(font)
self.res_cosPhi_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_unit.setObjectName("res_cosPhi_P1_unit")
self.res_U_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P1_unit.setGeometry(QtCore.QRect(170, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_unit.setFont(font)
self.res_U_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P1_unit.setObjectName("res_U_P1_unit")
self.res_S_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_S_LBL.setGeometry(QtCore.QRect(0, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_S_LBL.setFont(font)
self.res_S_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_LBL.setObjectName("res_S_LBL")
self.res_Q_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_LBL.setGeometry(QtCore.QRect(0, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Q_LBL.setFont(font)
self.res_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_LBL.setObjectName("res_Q_LBL")
self.sf_unit = QtWidgets.QLabel(self.LoadFlow)
self.sf_unit.setGeometry(QtCore.QRect(168, 215, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sf_unit.setFont(font)
self.sf_unit.setText("")
self.sf_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.sf_unit.setObjectName("sf_unit")
self.res_Q_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P1_DSB.setEnabled(False)
self.res_Q_P1_DSB.setGeometry(QtCore.QRect(90, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_DSB.setFont(font)
self.res_Q_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P1_DSB.setDecimals(3)
self.res_Q_P1_DSB.setMinimum(-999999.999)
self.res_Q_P1_DSB.setMaximum(999999.999)
self.res_Q_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P1_DSB.setProperty("value", 0.0)
self.res_Q_P1_DSB.setObjectName("res_Q_P1_DSB")
self.res_Iangle_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Iangle_LBL.setFont(font)
self.res_Iangle_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_LBL.setObjectName("res_Iangle_LBL")
self.res_U_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P1_DSB.setEnabled(False)
self.res_U_P1_DSB.setGeometry(QtCore.QRect(90, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_DSB.setFont(font)
self.res_U_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P1_DSB.setDecimals(3)
self.res_U_P1_DSB.setMaximum(999999.999)
self.res_U_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P1_DSB.setProperty("value", 0.0)
self.res_U_P1_DSB.setObjectName("res_U_P1_DSB")
self.res_P_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P1_unit.setGeometry(QtCore.QRect(170, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_unit.setFont(font)
```

```
self.res_P_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P1_unit.setObjectName("res_P_P1_unit")
self.res_I_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_I_LBL.setGeometry(QtCore.QRect(0, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_I_LBL.setFont(font)
self.res_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_LBL.setObjectName("res_I_LBL")
self.res_S_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P1_unit.setGeometry(QtCore.QRect(170, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_unit.setFont(font)
self.res_S_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P1_unit.setObjectName("res_S_P1_unit")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(0, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_S_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P1_DSB.setEnabled(False)
self.res_S_P1_DSB.setGeometry(QtCore.QRect(90, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_DSB.setFont(font)
self.res_S_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P1_DSB.setDecimals(3)
self.res_S_P1_DSB.setMaximum(999999.999)
self.res_S_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P1_DSB.setProperty("value", 0.0)
self.res_S_P1_DSB.setObjectName("res_S_P1_DSB")
self.res_Iangle_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P1_unit.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_unit.setFont(font)
self.res_Iangle_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_unit.setObjectName("res_Iangle_P1_unit")
self.res_cosPhi_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P1_DSB.setEnabled(False)
self.res_cosPhi_P1_DSB.setGeometry(QtCore.QRect(90, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_DSB.setFont(font)
self.res_cosPhi_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P1_DSB.setDecimals(3)
self.res_cosPhi_P1_DSB.setMaximum(999999.999)
self.res_cosPhi_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P1_DSB.setProperty("value", 0.0)
self.res_cosPhi_P1_DSB.setObjectName("res_cosPhi_P1_DSB")
self.res_I_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P1_unit.setGeometry(QtCore.QRect(170, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_unit.setFont(font)
self.res_I_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P1_unit.setObjectName("res_I_P1_unit")
self.res_I_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P1_DSB.setEnabled(False)
self.res_I_P1_DSB.setGeometry(QtCore.QRect(90, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_DSB.setFont(font)
self.res_I_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P1_DSB.setDecimals(3)
self.res_I_P1_DSB.setMaximum(999999.999)
self.res_I_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P1_DSB.setProperty("value", 0.0)
self.res_I_P1_DSB.setObjectName("res_I_P1_DSB")
self.tabWidget.addTab(self.LoadFlow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.cap_pwr_LBL = QtWidgets.QLabel(self.EMS)
self.cap_pwr_LBL.setGeometry(QtCore.QRect(10, 10, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.cap_pwr_LBL.setFont(font)
self.cap_pwr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_LBL.setObjectName("cap_pwr_LBL")
self.cap_pwr_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.cap_pwr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_DSB.setFont(font)
self.cap_pwr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
```

```
self.cap_pwr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cap_pwr_DSB.setDecimals(3)
self.cap_pwr_DSB.setMaximum(999999.999)
self.cap_pwr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cap_pwr_DSB.setProperty("value", 0.0)
self.cap_pwr_DSB.setObjectName("cap_pwr_DSB")
self.cap_pwr_unit = QtWidgets.QLabel(self.EMS)
self.cap_pwr_unit.setGeometry(QtCore.QRect(240, 10, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_unit.setFont(font)
self.cap_pwr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cap_pwr_unit.setObjectName("cap_pwr_unit")
self.tabWidget.addTab(self.EMS, "")
self.Reliability = QtWidgets.QWidget()
self.Reliability.setObjectName("Reliability")
self.rel_results_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 65, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_alfa_unit = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_unit.setGeometry(QtCore.QRect(240, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_R_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_R_LBL.setGeometry(QtCore.QRect(160, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_lambda_unit = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_beta_unit = QtWidgets.QLabel(self.Reliability)
self.rel_beta_unit.setGeometry(QtCore.QRect(240, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(0, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
self.rel_T0_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_T0_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(160, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(160, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Reliability)
```

```
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.bottom_LN_2 = QtWidgets.QFrame(self.Reliability)
self.bottom_LN_2.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN_2.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_2.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_2.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_2.setObjectName("bottom_LN_2")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(160, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(240, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_beta_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_beta_LBL.setGeometry(QtCore.QRect(0, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(240, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_T0_unit = QtWidgets.QLabel(self.Reliability)
self.rel_T0_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
```

```
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(0, 170, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(160, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(160, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_anni_DSB.setEnabled(False)
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(0, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_R_unit = QtWidgets.QLabel(self.Reliability)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_R_unit.setObjectName("rel_R_unit")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_R_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.rel_alfa_unit.raise_()
self.rel_R_LBL.raise_()
self.rel_lambda_unit.raise_()
self.rel_beta_unit.raise_()
self.rel_alfa_LBL.raise_()
self.rel_T0_LBL.raise_()
self.rel_beta_DSB.raise_()
self.rel_alfa_DSB.raise_()
self.rel_lambda_LBL.raise_()
self.bottom_LN_2.raise_()
self.rel_Pi_Q_DSB.raise_()
self.rel_T0_DSB.raise_()
self.rel_MTBF_ore_unit.raise_()
self.rel_MTBF_ore_DSB.raise_()
self.rel_Pi_Q_unit.raise_()
self.rel_MTBF_ore_LBL.raise_()
self.rel_beta_LBL.raise_()
self.rel_Pi_E_unit.raise_()
self.rel_T0_unit.raise_()
self.rel_Pi_Q_LBL.raise_()
```

```
        self.rel_MTBF_anni_LBL.raise_()
        self.rel_Pi_E_DSB.raise_()
        self.rel_MTBF_anni_DSB.raise_()
        self.rel_Pi_E_LBL.raise_()
        self.rel_R_unit.raise_()
        self.rel_lambda_LE.raise_()
        self.rel_R_LE.raise_()
        self.rel_results_LBL.raise_()
        self.rel_MTBF_anni_unit.raise_()
        self.tabWidget.addTab(self.Reliability, "")
        self.bb_out_LBL.raise_()
        self.bb_in_LBL.raise_()
        self.cub_out_LBL.raise_()
        self.top_LN.raise_()
        self.elem_name_LN.raise_()
        self.elem_name_LBL.raise_()
        self.type_LBL.raise_()
        self.type_cap_LBL.raise_()
        self.cub_in_LBL.raise_()
        self.symbol_LBL.raise_()
        self.ot_Frame_LN.raise_()
        self.vdx_frame_LN.raise_()
        self.vsx_frame_LN.raise_()
        self.ob_frame_LN.raise_()
        self.bb_in_LN.raise_()
        self.bb_out_LN.raise_()
        self.store_BTN.raise_()
        self.cancel_BTN.raise_()
        self.tabWidget.raise_()

        self.retranslateUi(Form)
        self.tabWidget.setCurrentIndex(3)
        QtCore.QMetaObject.connectSlotsByName(Form)
        Form.setTabOrder(self.tabWidget, self.units_DSB)
        Form.setTabOrder(self.units_DSB, self.sr_DSB)
        Form.setTabOrder(self.sr_DSB, self.control_CB)
        Form.setTabOrder(self.control_CB, self.p_DSB)
        Form.setTabOrder(self.p_DSB, self.q_DSB)
        Form.setTabOrder(self.q_DSB, self.s_DSB)
        Form.setTabOrder(self.s_DSB, self.cosPhi_DSB)
        Form.setTabOrder(self.cosPhi_DSB, self.sf_DSB)
        Form.setTabOrder(self.sf_DSB, self.sf_const_RB)
        Form.setTabOrder(self.sf_const_RB, self.sf_profile_RB)
        Form.setTabOrder(self.sf_profile_RB, self.cap_pwr_DSB)
        Form.setTabOrder(self.cap_pwr_DSB, self.rel_TO_DSB)
        Form.setTabOrder(self.rel_TO_DSB, self.rel_alfa_DSB)
        Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
        Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
        Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
        Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
        Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
        Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_I_P1_DSB)
        Form.setTabOrder(self.res_I_P1_DSB, self.res_Iangle_P1_DSB)
        Form.setTabOrder(self.res_Iangle_P1_DSB, self.res_P_P1_DSB)
        Form.setTabOrder(self.res_P_P1_DSB, self.res_Q_P1_DSB)
        Form.setTabOrder(self.res_Q_P1_DSB, self.res_S_P1_DSB)
        Form.setTabOrder(self.res_S_P1_DSB, self.res_cosPhi_P1_DSB)
        Form.setTabOrder(self.res_cosPhi_P1_DSB, self.res_U_P1_DSB)
        Form.setTabOrder(self.res_U_P1_DSB, self.store_BTN)
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria:"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.store_BTN.setText(_translate("Form", "Salva"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))
        self.sf_profile_RB.setText(_translate("Form", "Profilo"))
        self.s_LBL.setText(_translate("Form", "S"))
        self.control_CB.setItemText(0, _translate("Form", "P, Q"))
        self.control_CB.setItemText(1, _translate("Form", "P, cosPhi"))
        self.control_CB.setItemText(2, _translate("Form", "S, cosPhi"))
        self.sr_LBL.setText(_translate("Form", "Sr"))
        self.sr_unit.setText(_translate("Form", "kVA"))
        self.sfi_LBL.setText(_translate("Form", "scala"))
        self.q_unit.setText(_translate("Form", "kVA"))
        self.s_unit.setText(_translate("Form", "kVA"))
        self.p_LBL.setText(_translate("Form", "P"))
        self.sf_const_RB.setText(_translate("Form", "Costante"))
        self.control_LBL.setText(_translate("Form", "Controllo"))
        self.unts_LBL.setText(_translate("Form", "N. Unità"))
        self.p_unit.setText(_translate("Form", "kW"))
        self.q_LBL.setText(_translate("Form", "Q"))
        self.cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Parameters), _translate("Form", "Parametri"))
        self.res_Q_P1_unit.setText(_translate("Form", "kVA"))
        self.Por1_LBL.setText(_translate("Form", "Nodo 1"))
        self.res_P_LBL.setText(_translate("Form", "P"))
        self.res_cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.res_cosPhi_P1_unit.setText(_translate("Form", "-"))
        self.res_U_P1_unit.setText(_translate("Form", "kV"))
        self.res_S_LBL.setText(_translate("Form", "S"))
        self.res_Q_LBL.setText(_translate("Form", "Q"))
        self.res_Iangle_LBL.setText(_translate("Form", "I angle"))
        self.res_P_P1_unit.setText(_translate("Form", "kW"))
        self.res_I_LBL.setText(_translate("Form", "  I"))
        self.res_S_P1_unit.setText(_translate("Form", "kVA"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMITI VIOLATI"))
        self.res_Iangle_P1_unit.setText(_translate("Form", "°"))
        self.res_I_P1_unit.setText(_translate("Form", "A"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFlow), _translate("Form", "LoadFlow"))
        self.cap_pwr_LBL.setText(_translate("Form", "Potenza Nominale"))
        self.cap_pwr_unit.setText(_translate("Form", "kVA"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
```

```
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_T0_unit.setText(_translate("Form", "°C"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Reliability), _translate("Form", "Affidabilità"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.3.5 Battery

4.4.3.5.1 battery.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .batteryUI import Ui_Form
from __shared__ import variables as v
import copy


class Battery(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(Battery, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.bb_out_LN.setVisible(False)
        self.ui.bb_out_LBL.setVisible(False)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")
        self.ui.tabWidget.setCurrentIndex(0)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])
        self.prot = copy.deepcopy(v.elements[element]['protections'])

        self.bb = v.elements[element]['conn']['h']
        self.cubicle = v.elements[self.element]['conn'][self.bb]
        self.u = v.elements[self.bb]['parameters']['Ur']
        self.ui.bb_in_LBL.setText(self.bb)

        if v.software == 'neplan':
            bb_conns = list(v.elements[self.bb]['conn'].keys())
            bb_conns.remove(element)
            bb_conns.remove('h')
            pwm = bb_conns[0]
            self.Pn = v.elements[pwm]['parameters']['Sr']
            self.In = self.Pn / self.u
        else:
            self.In = self.par['In']    # In powerfactory questo parametro non cambia
            self.Pn = self.In * self.u

        self.ui.symbol_LBL.setPixmap(QtGui.QPixmap("_images/Elements/Battery/element.png"))
        self.switch_draw()
        self.fill()

        self.ui.cub_in_LBL.mouseDoubleClickEvent = self.cub1_switch

    #
    def store(self):
        self.par['cap_en'] = self.ui.cap_en_DSB.value()
        v.elements[self.element]['parameters'] = copy.deepcopy(self.par)

        v.elements[self.element]['conn'][self.bb] = self.cubicle

        self.ems['cap_en'] = self.ui.cap_en_DSB.value()
        self.ems['init_SOC'] = self.ui.init_SOC_DSB.value()
        self.ems['max_SOC'] = self.ui.max_SOC_DSB.value()
        self.ems['min_SOC'] = self.ui.min_SOC_DSB.value()
        self.ems['final_SOC'] = self.ui.final_SOC_DSB.value()
        self.ems['max_dch'] = self.ui.max_dch_DSB.value()
        self.ems['max_ch'] = self.ui.max_ch_DSB.value()
        self.ems['cost_dch'] = self.ui.cost_dch_DSB.value()
        self.ems['cost_ch'] = self.ui.cost_ch_DSB.value()
        self.ems['eta_dch'] = self.ui.eta_dch_DSB.value()
        self.ems['eta_ch'] = self.ui.eta_ch_DSB.value()
        v.elements[self.element]['ems'] = copy.deepcopy(self.ems)

        for par in ['T0', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)

        self.protections_par()

    #
    def protections_par(self):
        self.prot['Pn'] = self.Pn
        self.prot['Vn'] = self.u
        self.prot['In'] = self.In
        v.elements[self.element]['protections'] = copy.deepcopy(self.prot)

    #
    def fill(self):
        self.ui.cap_en_DSB.setValue(self.par['cap_en'])
        self.ui.cap_en_DSB.setValue(self.ems['cap_en'])
        self.ui.init_SOC_DSB.setValue(self.ems['init_SOC'])
        self.ui.max_SOC_DSB.setValue(self.ems['max_SOC'])
        self.ui.min_SOC_DSB.setValue(self.ems['min_SOC'])
        self.ui.final_SOC_DSB.setValue(self.ems['final_SOC'])
        self.ui.max_dch_DSB.setValue(self.ems['max_dch'])
        self.ui.max_ch_DSB.setValue(self.ems['max_ch'])
        self.ui.cost_dch_DSB.setValue(self.ems['cost_dch'])
        self.ui.cost_ch_DSB.setValue(self.ems['cost_ch'])
        self.ui.eta_dch_DSB.setValue(self.ems['eta_dch'])
        self.ui.eta_ch_DSB.setValue(self.ems['eta_ch'])

        if self.res != {}:
            self.fill_results()
        self.ui.tabWidget.setTabVisible(3, self.res != {})
        self.fill_reliability()
        if self.prot != {}:
            if self.prot['results'] != {} and v.protections:
                self.fill_protections()

    #
```

```
def fill_reliability(self):
    for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
        self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
    for par in self.rel['results']:
        try:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
        except:
            if self.rel['results'][par] == 0:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('0')
            elif self.rel['results'][par] < 0.01:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
            else:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])

#
def fill_protections(self):
    self.ui.prot_type_LE.setText(self.prot['results']['type'])
    self.ui.prot_cost_DSB.setValue(self.prot['results']['cost'])
    self.ui.prot_cal_I_val_DSB.setValue(self.prot['results']['soglia_I'])
    self.ui.prot_cal_I_pu_DSB.setValue(self.prot['results']['soglia_I']/self.prot['In'] * 100)
    self.ui.prot_cal_Vmax_val_DSB.setValue(self.prot['results']['soglia_Vmax'])
    self.ui.prot_cal_Vmax_pu_DSB.setValue(self.prot['results']['soglia_Vmax'] / self.prot['Vn'] * 100)
    self.ui.prot_cal_Vmin_val_DSB.setValue(self.prot['results']['soglia_Vmin'])
    self.ui.prot_cal_Vmin_pu_DSB.setValue(self.prot['results']['soglia_Vmin'] / self.prot['Vn'] * 100)
    self.ui.prot_delay_I_DSB.setValue(self.prot['results']['delay_I'])
    self.ui.prot_delay_Vmax_DSB.setValue(self.prot['results']['delay_Vmax'])
    self.ui.prot_delay_Vmin_DSB.setValue(self.prot['results']['delay_Vmin'])

#
def fill_results(self):
    results = ['I', 'Iangle', 'P', 'Q', 'S', 'cosPhi', 'U']
    ports = ['P1']

    for port in ports:
        for result in results:
            self.ui.__getattribute__('res_' + result + '_' + port + '_DSB').setValue(self.res[result])
    self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])

#
def switch_draw(self):
    if self.cubicle:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
    else:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))

def cub1_switch(self, event):
    self.cubicle = not self.cubicle
    self.switch_draw()

def calculate(self):
    pass
```

4.4.3.5.2 batteryUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'batteryUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(497, 505)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("res/Load.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.bb_in_LBL.setFont(font)
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_out_LN.setObjectName("bb_out_LN")
        self.bb_out_LBL = QtWidgets.QLabel(self.widget)
        self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
self.bb_out_LBL.setFont(font)
self.bb_out_LBL.setStyleSheet("")
self.bb_out_LBL.setLineWidth(4)
self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.store_BTN = QtWidgets.QPushButton(self.widget)
self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.store_BTN.setFont(font)
self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.store_BTN.setObjectName("store_BTN")
self.cancel_BTN = QtWidgets.QPushButton(self.widget)
self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.cancel_BTN.setFont(font)
self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.cancel_BTN.setObjectName("cancel_BTN")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.init_SOC_LBL = QtWidgets.QLabel(self.Parameters)
self.init_SOC_LBL.setGeometry(QtCore.QRect(10, 40, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.init_SOC_LBL.setFont(font)
self.init_SOC_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.init_SOC_LBL.setObjectName("init_SOC_LBL")
self.init_SOC_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.init_SOC_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.init_SOC_DSB.setFont(font)
self.init_SOC_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.init_SOC_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.init_SOC_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.init_SOC_DSB.setDecimals(5)
self.init_SOC_DSB.setMaximum(2.0)
self.init_SOC_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.init_SOC_DSB.setProperty("value", 0.0)
self.init_SOC_DSB.setObjectName("init_SOC_DSB")
self.cap_en_LBL = QtWidgets.QLabel(self.Parameters)
self.cap_en_LBL.setGeometry(QtCore.QRect(10, 10, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.cap_en_LBL.setFont(font)
self.cap_en_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_en_LBL.setObjectName("cap_en_LBL")
self.cap_en_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.cap_en_DSB.setGeometry(QtCore.QRect(160, 10, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_en_DSB.setFont(font)
self.cap_en_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_en_DSB.setReadOnly(False)
self.cap_en_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cap_en_DSB.setDecimals(3)
self.cap_en_DSB.setMaximum(999999.999)
self.cap_en_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cap_en_DSB.setProperty("value", 0.0)
self.cap_en_DSB.setObjectName("cap_en_DSB")
self.cap_en_unit = QtWidgets.QLabel(self.Parameters)
self.cap_en_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_en_unit.setFont(font)
self.cap_en_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cap_en_unit.setObjectName("cap_en_unit")
self.init_SOC_unit = QtWidgets.QLabel(self.Parameters)
self.init_SOC_unit.setGeometry(QtCore.QRect(240, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.init_SOC_unit.setFont(font)
self.init_SOC_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.init_SOC_unit.setObjectName("init_SOC_unit")
self.max_SOC_unit = QtWidgets.QLabel(self.Parameters)
self.max_SOC_unit.setGeometry(QtCore.QRect(240, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_SOC_unit.setFont(font)
self.max_SOC_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_SOC_unit.setObjectName("max_SOC_unit")
self.max_SOC_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.max_SOC_DSB.setGeometry(QtCore.QRect(160, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_SOC_DSB.setFont(font)
self.max_SOC_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_SOC_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_SOC_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
```

```
self.max_SOC_DSB.setDecimals(5)
self.max_SOC_DSB.setMaximum(2.0)
self.max_SOC_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_SOC_DSB.setProperty("value", 0.0)
self.max_SOC_DSB.setObjectName("max_SOC_DSB")
self.max_SOC_LBL = QtWidgets.QLabel(self.Parameters)
self.max_SOC_LBL.setGeometry(QtCore.QRect(10, 70, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_SOC_LBL.setFont(font)
self.max_SOC_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_SOC_LBL.setObjectName("max_SOC_LBL")
self.min_SOC_LBL = QtWidgets.QLabel(self.Parameters)
self.min_SOC_LBL.setGeometry(QtCore.QRect(10, 100, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.min_SOC_LBL.setFont(font)
self.min_SOC_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.min_SOC_LBL.setObjectName("min_SOC_LBL")
self.min_SOC_unit = QtWidgets.QLabel(self.Parameters)
self.min_SOC_unit.setGeometry(QtCore.QRect(240, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.min_SOC_unit.setFont(font)
self.min_SOC_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.min_SOC_unit.setObjectName("min_SOC_unit")
self.min_SOC_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.min_SOC_DSB.setGeometry(QtCore.QRect(160, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.min_SOC_DSB.setFont(font)
self.min_SOC_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.min_SOC_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.min_SOC_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.min_SOC_DSB.setDecimals(5)
self.min_SOC_DSB.setMaximum(2.0)
self.min_SOC_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.min_SOC_DSB.setProperty("value", 0.0)
self.min_SOC_DSB.setObjectName("min_SOC_DSB")
self.tabWidget.addTab(self.Parameters, "")
self.LoadFlow = QtWidgets.QWidget()
self.LoadFlow.setObjectName("LoadFlow")
self.res_Iangle_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P1_DSB.setEnabled(False)
self.res_Iangle_P1_DSB.setGeometry(QtCore.QRect(100, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_DSB.setFont(font)
self.res_Iangle_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P1_DSB.setDecimals(3)
self.res_Iangle_P1_DSB.setMinimum(-999999.999)
self.res_Iangle_P1_DSB.setMaximum(999999.999)
self.res_Iangle_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P1_DSB.setProperty("value", 0.0)
self.res_Iangle_P1_DSB.setObjectName("res_Iangle_P1_DSB")
self.res_U_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P1_unit.setGeometry(QtCore.QRect(180, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_unit.setFont(font)
self.res_U_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P1_unit.setObjectName("res_U_P1_unit")
self.res_cosPhi_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P1_unit.setGeometry(QtCore.QRect(180, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_unit.setFont(font)
self.res_cosPhi_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_unit.setObjectName("res_cosPhi_P1_unit")
self.res_S_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_S_LBL.setGeometry(QtCore.QRect(10, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_S_LBL.setFont(font)
self.res_S_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_LBL.setObjectName("res_S_LBL")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(10, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_I_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_I_LBL.setGeometry(QtCore.QRect(10, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_I_LBL.setFont(font)
self.res_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_LBL.setObjectName("res_I_LBL")
self.res_cosPhi_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_LBL.setGeometry(QtCore.QRect(10, 190, 81, 21))
font = QtGui.QFont()
```

```
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_cosPhi_LBL.setFont(font)
self.res_cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_LBL.setObjectName("res_cosPhi_LBL")
self.res_Iangle_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_LBL.setGeometry(QtCore.QRect(10, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Iangle_LBL.setFont(font)
self.res_Iangle_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_LBL.setObjectName("res_Iangle_LBL")
self.res_Q_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P1_DSB.setEnabled(False)
self.res_Q_P1_DSB.setGeometry(QtCore.QRect(100, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_DSB.setFont(font)
self.res_Q_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P1_DSB.setDecimals(3)
self.res_Q_P1_DSB.setMinimum(-999999.999)
self.res_Q_P1_DSB.setMaximum(999999.999)
self.res_Q_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P1_DSB.setProperty("value", 0.0)
self.res_Q_P1_DSB.setObjectName("res_Q_P1_DSB")
self.res_Iangle_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P1_unit.setGeometry(QtCore.QRect(180, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_unit.setFont(font)
self.res_Iangle_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_unit.setObjectName("res_Iangle_P1_unit")
self.res_S_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P1_DSB.setEnabled(False)
self.res_S_P1_DSB.setGeometry(QtCore.QRect(100, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_DSB.setFont(font)
self.res_S_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P1_DSB.setDecimals(3)
self.res_S_P1_DSB.setMaximum(999999.999)
self.res_S_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P1_DSB.setProperty("value", 0.0)
self.res_S_P1_DSB.setObjectName("res_S_P1_DSB")
self.Por1_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Por1_LBL.setGeometry(QtCore.QRect(100, 10, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Por1_LBL.setFont(font)
self.Por1_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Por1_LBL.setObjectName("Por1_LBL")
self.res_P_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P1_unit.setGeometry(QtCore.QRect(180, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_unit.setFont(font)
self.res_P_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P1_unit.setObjectName("res_P_P1_unit")
self.res_I_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P1_unit.setGeometry(QtCore.QRect(180, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_unit.setFont(font)
self.res_I_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P1_unit.setObjectName("res_I_P1_unit")
self.sf_unit = QtWidgets.QLabel(self.LoadFlow)
self.sf_unit.setGeometry(QtCore.QRect(178, 195, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sf_unit.setFont(font)
self.sf_unit.setText("")
self.sf_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.sf_unit.setObjectName("sf_unit")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_U_LBL.setGeometry(QtCore.QRect(10, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.res_S_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P1_unit.setGeometry(QtCore.QRect(180, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_unit.setFont(font)
self.res_S_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P1_unit.setObjectName("res_S_P1_unit")
self.res_I_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P1_DSB.setEnabled(False)
self.res_I_P1_DSB.setGeometry(QtCore.QRect(100, 40, 71, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_DSB.setFont(font)
self.res_I_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P1_DSB.setDecimals(3)
self.res_I_P1_DSB.setMaximum(999999.999)
self.res_I_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P1_DSB.setProperty("value", 0.0)
self.res_I_P1_DSB.setObjectName("res_I_P1_DSB")
self.res_P_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P1_DSB.setEnabled(False)
self.res_P_P1_DSB.setGeometry(QtCore.QRect(100, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_DSB.setFont(font)
self.res_P_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P1_DSB.setDecimals(3)
self.res_P_P1_DSB.setMinimum(-999999.999)
self.res_P_P1_DSB.setMaximum(999999.999)
self.res_P_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P1_DSB.setProperty("value", 0.0)
self.res_P_P1_DSB.setObjectName("res_P_P1_DSB")
self.res_Q_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_LBL.setGeometry(QtCore.QRect(10, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Q_LBL.setFont(font)
self.res_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_LBL.setObjectName("res_Q_LBL")
self.res_cosPhi_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P1_DSB.setEnabled(False)
self.res_cosPhi_P1_DSB.setGeometry(QtCore.QRect(100, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_DSB.setFont(font)
self.res_cosPhi_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P1_DSB.setDecimals(3)
self.res_cosPhi_P1_DSB.setMaximum(999999.999)
self.res_cosPhi_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P1_DSB.setProperty("value", 0.0)
self.res_cosPhi_P1_DSB.setObjectName("res_cosPhi_P1_DSB")
self.res_P_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_P_LBL.setGeometry(QtCore.QRect(10, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_P_LBL.setFont(font)
self.res_P_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_LBL.setObjectName("res_P_LBL")
self.res_Q_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P1_unit.setGeometry(QtCore.QRect(180, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_unit.setFont(font)
self.res_Q_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P1_unit.setObjectName("res_Q_P1_unit")
self.res_U_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P1_DSB.setEnabled(False)
self.res_U_P1_DSB.setGeometry(QtCore.QRect(100, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_DSB.setFont(font)
self.res_U_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P1_DSB.setDecimals(3)
self.res_U_P1_DSB.setMaximum(999999.999)
self.res_U_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P1_DSB.setProperty("value", 0.0)
self.res_U_P1_DSB.setObjectName("res_U_P1_DSB")
self.tabWidget.addTab(self.LoadFlow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.final_SOC_unit = QtWidgets.QLabel(self.EMS)
self.final_SOC_unit.setGeometry(QtCore.QRect(240, 10, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.final_SOC_unit.setFont(font)
self.final_SOC_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.final_SOC_unit.setObjectName("final_SOC_unit")
self.final_SOC_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.final_SOC_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.final_SOC_DSB.setFont(font)
self.final_SOC_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.final_SOC_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.final_SOC_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.final_SOC_DSB.setDecimals(5)
self.final_SOC_DSB.setMaximum(2.0)
self.final_SOC_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.final_SOC_DSB.setProperty("value", 0.0)
```

```
self.final_SOC_DSB.setObjectName("final_SOC_DSB")
self.final_SOC_LBL = QtWidgets.QLabel(self.EMS)
self.final_SOC_LBL.setGeometry(QtCore.QRect(10, 10, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.final_SOC_LBL.setFont(font)
self.final_SOC_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.final_SOC_LBL.setObjectName("final_SOC_LBL")
self.max_dch_unit = QtWidgets.QLabel(self.EMS)
self.max_dch_unit.setGeometry(QtCore.QRect(240, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_dch_unit.setFont(font)
self.max_dch_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_dch_unit.setObjectName("max_dch_unit")
self.max_dch_LBL = QtWidgets.QLabel(self.EMS)
self.max_dch_LBL.setGeometry(QtCore.QRect(10, 40, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_dch_LBL.setFont(font)
self.max_dch_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_dch_LBL.setObjectName("max_dch_LBL")
self.max_dch_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.max_dch_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_dch_DSB.setFont(font)
self.max_dch_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_dch_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_dch_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_dch_DSB.setDecimals(3)
self.max_dch_DSB.setMaximum(99999.0)
self.max_dch_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_dch_DSB.setProperty("value", 0.0)
self.max_dch_DSB.setObjectName("max_dch_DSB")
self.max_ch_unit = QtWidgets.QLabel(self.EMS)
self.max_ch_unit.setGeometry(QtCore.QRect(240, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_ch_unit.setFont(font)
self.max_ch_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_ch_unit.setObjectName("max_ch_unit")
self.max_ch_LBL = QtWidgets.QLabel(self.EMS)
self.max_ch_LBL.setGeometry(QtCore.QRect(10, 70, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_ch_LBL.setFont(font)
self.max_ch_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_ch_LBL.setObjectName("max_ch_LBL")
self.max_ch_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.max_ch_DSB.setGeometry(QtCore.QRect(160, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_ch_DSB.setFont(font)
self.max_ch_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_ch_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_ch_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_ch_DSB.setDecimals(3)
self.max_ch_DSB.setMaximum(99999.0)
self.max_ch_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_ch_DSB.setProperty("value", 0.0)
self.max_ch_DSB.setObjectName("max_ch_DSB")
self.cost_dch_unit = QtWidgets.QLabel(self.EMS)
self.cost_dch_unit.setGeometry(QtCore.QRect(240, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cost_dch_unit.setFont(font)
self.cost_dch_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cost_dch_unit.setObjectName("cost_dch_unit")
self.cost_dch_LBL = QtWidgets.QLabel(self.EMS)
self.cost_dch_LBL.setGeometry(QtCore.QRect(10, 100, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.cost_dch_LBL.setFont(font)
self.cost_dch_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cost_dch_LBL.setObjectName("cost_dch_LBL")
self.cost_dch_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.cost_dch_DSB.setGeometry(QtCore.QRect(160, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cost_dch_DSB.setFont(font)
self.cost_dch_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cost_dch_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cost_dch_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cost_dch_DSB.setDecimals(3)
self.cost_dch_DSB.setMaximum(99999.0)
self.cost_dch_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cost_dch_DSB.setProperty("value", 0.0)
self.cost_dch_DSB.setObjectName("cost_dch_DSB")
self.cost_ch_LBL = QtWidgets.QLabel(self.EMS)
self.cost_ch_LBL.setGeometry(QtCore.QRect(10, 130, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
```

```
self.cost_ch_LBL.setFont(font)
self.cost_ch_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cost_ch_LBL.setObjectName("cost_ch_LBL")
self.cost_ch_unit = QtWidgets.QLabel(self.EMS)
self.cost_ch_unit.setGeometry(QtCore.QRect(240, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cost_ch_unit.setFont(font)
self.cost_ch_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cost_ch_unit.setObjectName("cost_ch_unit")
self.cost_ch_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.cost_ch_DSB.setGeometry(QtCore.QRect(160, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cost_ch_DSB.setFont(font)
self.cost_ch_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cost_ch_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cost_ch_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cost_ch_DSB.setDecimals(3)
self.cost_ch_DSB.setMaximum(99999.0)
self.cost_ch_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cost_ch_DSB.setProperty("value", 0.0)
self.cost_ch_DSB.setObjectName("cost_ch_DSB")
self.eta_dch_unit = QtWidgets.QLabel(self.EMS)
self.eta_dch_unit.setGeometry(QtCore.QRect(240, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.eta_dch_unit.setFont(font)
self.eta_dch_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.eta_dch_unit.setObjectName("eta_dch_unit")
self.eta_dch_LBL = QtWidgets.QLabel(self.EMS)
self.eta_dch_LBL.setGeometry(QtCore.QRect(10, 160, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.eta_dch_LBL.setFont(font)
self.eta_dch_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.eta_dch_LBL.setObjectName("eta_dch_LBL")
self.eta_dch_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.eta_dch_DSB.setGeometry(QtCore.QRect(160, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.eta_dch_DSB.setFont(font)
self.eta_dch_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.eta_dch_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.eta_dch_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.eta_dch_DSB.setDecimals(5)
self.eta_dch_DSB.setMaximum(2.0)
self.eta_dch_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.eta_dch_DSB.setProperty("value", 0.0)
self.eta_dch_DSB.setObjectName("eta_dch_DSB")
self.eta_ch_unit = QtWidgets.QLabel(self.EMS)
self.eta_ch_unit.setGeometry(QtCore.QRect(240, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.eta_ch_unit.setFont(font)
self.eta_ch_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.eta_ch_unit.setObjectName("eta_ch_unit")
self.eta_ch_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.eta_ch_DSB.setGeometry(QtCore.QRect(160, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.eta_ch_DSB.setFont(font)
self.eta_ch_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.eta_ch_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.eta_ch_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.eta_ch_DSB.setDecimals(5)
self.eta_ch_DSB.setMaximum(2.0)
self.eta_ch_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.eta_ch_DSB.setProperty("value", 0.0)
self.eta_ch_DSB.setObjectName("eta_ch_DSB")
self.eta_ch_LBL = QtWidgets.QLabel(self.EMS)
self.eta_ch_LBL.setGeometry(QtCore.QRect(10, 190, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.eta_ch_LBL.setFont(font)
self.eta_ch_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.eta_ch_LBL.setObjectName("eta_ch_LBL")
self.tabWidget.addTab(self.EMS, "")
self.Reliability = QtWidgets.QWidget()
self.Reliability.setObjectName("Reliability")
self.rel_results_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 65, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_alfa_unit = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_unit.setGeometry(QtCore.QRect(240, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_R_LBL = QtWidgets.QLabel(self.Reliability)
```

```
self.rel_R_LBL.setGeometry(QtCore.QRect(190, 320, 31, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_lambda_unit = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_beta_unit = QtWidgets.QLabel(self.Reliability)
self.rel_beta_unit.setGeometry(QtCore.QRect(240, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(0, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
self.rel_T0_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_T0_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(160, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(160, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.bottom_LN_3 = QtWidgets.QFrame(self.Reliability)
self.bottom_LN_3.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN_3.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_3.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_3.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_3.setObjectName("bottom_LN_3")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(160, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
```

```
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(240, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_beta_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_beta_LBL.setGeometry(QtCore.QRect(0, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(240, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_T0_unit = QtWidgets.QLabel(self.Reliability)
self.rel_T0_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(0, 170, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(160, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(160, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
```

```
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_anni_DSB.setEnabled(False)
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(0, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_R_unit = QtWidgets.QLabel(self.Reliability)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_R_unit.setObjectName("rel_R_unit")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_R_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.rel_alfa_unit.raise_()
self.rel_R_LBL.raise_()
self.rel_lambda_unit.raise_()
self.rel_beta_unit.raise_()
self.rel_alfa_LBL.raise_()
self.rel_T0_LBL.raise_()
self.rel_beta_DSB.raise_()
self.rel_alfa_DSB.raise_()
self.rel_lambda_LBL.raise_()
self.bottom_LN_3.raise_()
self.rel_Pi_Q_DSB.raise_()
self.rel_T0_DSB.raise_()
self.rel_MTBF_ore_unit.raise_()
self.rel_MTBF_ore_DSB.raise_()
self.rel_Pi_Q_unit.raise_()
self.rel_MTBF_ore_LBL.raise_()
self.rel_beta_LBL.raise_()
self.rel_Pi_E_unit.raise_()
self.rel_T0_unit.raise_()
self.rel_Pi_Q_LBL.raise_()
self.rel_MTBF_anni_LBL.raise_()
self.rel_Pi_E_DSB.raise_()
self.rel_MTBF_anni_DSB.raise_()
self.rel_Pi_E_LBL.raise_()
self.rel_R_unit.raise_()
self.rel_lambda_LE.raise_()
self.rel_R_LE.raise_()
self.rel_results_LBL.raise_()
self.rel_MTBF_anni_unit.raise_()
self.tabWidget.addTab(self.Reliability, "")
self.Protections = QtWidgets.QWidget()
self.Protections.setObjectName("Protections")
self.bottom_LN_2 = QtWidgets.QFrame(self.Protections)
self.bottom_LN_2.setGeometry(QtCore.QRect(0, 500, 490, 1))
self.bottom_LN_2.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_2.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_2.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_2.setObjectName("bottom_LN_2")
self.prot_cal_I_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_val_DSB.setGeometry(QtCore.QRect(90, 150, 71, 21))
self.prot_cal_I_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_val_DSB.setDecimals(1)
self.prot_cal_I_val_DSB.setMaximum(99999999.0)
self.prot_cal_I_val_DSB.setObjectName("prot_cal_I_val_DSB")
self.prot_cal_Vmax_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_sep_LBL.setGeometry(QtCore.QRect(190, 180, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_sep_LBL.setFont(font)
self.prot_cal_Vmax_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_sep_LBL.setObjectName("prot_cal_Vmax_sep_LBL")
self.prot_cost_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cost_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
self.prot_cost_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cost_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
```

```
self.prot_cost_DSB.setDecimals(2)
self.prot_cost_DSB.setMaximum(99999999.0)
self.prot_cost_DSB.setObjectName("prot_cost_DSB")
self.prot_cal_Vmin_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_LBL.setGeometry(QtCore.QRect(0, 210, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_LBL.setFont(font)
self.prot_cal_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_LBL.setObjectName("prot_cal_Vmin_LBL")
self.prot_delay_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_LBL.setGeometry(QtCore.QRect(0, 320, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_Vmax_LBL.setFont(font)
self.prot_delay_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_LBL.setObjectName("prot_delay_Vmax_LBL")
self.prot_cal_Vmax_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_val_DSB.setGeometry(QtCore.QRect(90, 180, 71, 21))
self.prot_cal_Vmax_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_val_DSB.setDecimals(3)
self.prot_cal_Vmax_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_val_DSB.setObjectName("prot_cal_Vmax_val_DSB")
self.prot_delay_Vmax_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_unit_LBL.setGeometry(QtCore.QRect(170, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmax_unit_LBL.setFont(font)
self.prot_delay_Vmax_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_unit_LBL.setObjectName("prot_delay_Vmax_unit_LBL")
self.prot_delay_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_top_LN.setGeometry(QtCore.QRect(10, 270, 305, 1))
self.prot_delay_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_top_LN.setObjectName("prot_delay_top_LN")
self.prot_cost_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cost_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cost_LBL.setFont(font)
self.prot_cost_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cost_LBL.setObjectName("prot_cost_LBL")
self.prot_calib_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_btm_LN.setGeometry(QtCore.QRect(10, 240, 305, 1))
self.prot_calib_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_calib_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_btm_LN.setObjectName("prot_calib_btm_LN")
self.prot_delay_Vmin_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_LBL.setGeometry(QtCore.QRect(0, 350, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_Vmin_LBL.setFont(font)
self.prot_delay_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_LBL.setObjectName("prot_delay_Vmin_LBL")
self.prot_cost_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cost_unit_LBL.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cost_unit_LBL.setFont(font)
self.prot_cost_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cost_unit_LBL.setObjectName("prot_cost_unit_LBL")
self.prot_type_LE = QtWidgets.QLineEdit(self.Protections)
self.prot_type_LE.setGeometry(QtCore.QRect(90, 40, 201, 20))
self.prot_type_LE.setText("")
self.prot_type_LE.setObjectName("prot_type_LE")
self.prot_cal_Vmin_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_val_unit_LBL.setGeometry(QtCore.QRect(170, 210, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_val_unit_LBL.setFont(font)
self.prot_cal_Vmin_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_unit_LBL.setObjectName("prot_cal_Vmin_val_unit_LBL")
self.prot_cal_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_LBL.setGeometry(QtCore.QRect(0, 180, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_LBL.setFont(font)
self.prot_cal_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_LBL.setObjectName("prot_cal_Vmax_LBL")
self.prot_cal_Vmin_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_val_DSB.setGeometry(QtCore.QRect(90, 210, 71, 21))
self.prot_cal_Vmin_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_val_DSB.setDecimals(3)
self.prot_cal_Vmin_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_val_DSB.setObjectName("prot_cal_Vmin_val_DSB")
self.prot_delay_I_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_I_DSB.setGeometry(QtCore.QRect(90, 290, 71, 21))
self.prot_delay_I_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_I_DSB.setDecimals(3)
self.prot_delay_I_DSB.setMaximum(99999999.0)
self.prot_delay_I_DSB.setObjectName("prot_delay_I_DSB")
self.prot_calib_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_top_LN.setGeometry(QtCore.QRect(10, 130, 305, 1))
self.prot_calib_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
```

```
self.prot_calib_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_top_LN.setObjectName("prot_calib_top_LN")
self.prot_cal_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_LBL.setGeometry(QtCore.QRect(0, 150, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_LBL.setFont(font)
self.prot_cal_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_LBL.setObjectName("prot_cal_I_LBL")
self.prot_char_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_btm_LN.setGeometry(QtCore.QRect(10, 100, 305, 1))
self.prot_char_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_btm_LN.setObjectName("prot_char_btm_LN")
self.prot_delay_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_LBL.setGeometry(QtCore.QRect(0, 290, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_I_LBL.setFont(font)
self.prot_delay_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_LBL.setObjectName("prot_delay_I_LBL")
self.prot_cal_I_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_sep_LBL.setGeometry(QtCore.QRect(190, 150, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_sep_LBL.setFont(font)
self.prot_cal_I_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_sep_LBL.setObjectName("prot_cal_I_sep_LBL")
self.prot_delay_I_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_unit_LBL.setGeometry(QtCore.QRect(170, 290, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_I_unit_LBL.setFont(font)
self.prot_delay_I_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_I_unit_LBL.setObjectName("prot_delay_I_unit_LBL")
self.prot_cal_I_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_pu_DSB.setGeometry(QtCore.QRect(220, 150, 71, 21))
self.prot_cal_I_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_pu_DSB.setDecimals(1)
self.prot_cal_I_pu_DSB.setMaximum(99999999.0)
self.prot_cal_I_pu_DSB.setObjectName("prot_cal_I_pu_DSB")
self.prot_delay_Vmax_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmax_DSB.setGeometry(QtCore.QRect(90, 320, 71, 21))
self.prot_delay_Vmax_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmax_DSB.setDecimals(3)
self.prot_delay_Vmax_DSB.setMaximum(99999999.0)
self.prot_delay_Vmax_DSB.setObjectName("prot_delay_Vmax_DSB")
self.prot_cal_Vmin_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_sep_LBL.setGeometry(QtCore.QRect(190, 210, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_sep_LBL.setFont(font)
self.prot_cal_Vmin_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_sep_LBL.setObjectName("prot_cal_Vmin_sep_LBL")
self.prot_calib_LBL = QtWidgets.QLabel(self.Protections)
self.prot_calib_LBL.setGeometry(QtCore.QRect(110, 120, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_calib_LBL.setFont(font)
self.prot_calib_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_calib_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_calib_LBL.setObjectName("prot_calib_LBL")
self.prot_cal_Vmax_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_pu_DSB.setGeometry(QtCore.QRect(220, 180, 71, 21))
self.prot_cal_Vmax_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_pu_DSB.setDecimals(1)
self.prot_cal_Vmax_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_pu_DSB.setObjectName("prot_cal_Vmax_pu_DSB")
self.prot_char_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_top_LN.setGeometry(QtCore.QRect(10, 20, 305, 1))
self.prot_char_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_top_LN.setObjectName("prot_char_top_LN")
self.prot_type_LBL = QtWidgets.QLabel(self.Protections)
self.prot_type_LBL.setGeometry(QtCore.QRect(0, 40, 81, 20))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_type_LBL.setFont(font)
self.prot_type_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_type_LBL.setObjectName("prot_type_LBL")
self.prot_cal_Vmax_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_val_unit_LBL.setGeometry(QtCore.QRect(170, 180, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmax_val_unit_LBL.setFont(font)
self.prot_cal_Vmax_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_val_unit_LBL.setObjectName("prot_cal_Vmax_val_unit_LBL")
self.prot_delay_Vmin_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmin_DSB.setGeometry(QtCore.QRect(90, 350, 71, 21))
self.prot_delay_Vmin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmin_DSB.setDecimals(3)
self.prot_delay_Vmin_DSB.setMaximum(99999999.0)
```

```
self.prot_delay_Vmin_DSB.setObjectName("prot_delay_Vmin_DSB")
self.prot_charact_LBL = QtWidgets.QLabel(self.Protections)
self.prot_charact_LBL.setGeometry(QtCore.QRect(60, 10, 211, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_charact_LBL.setFont(font)
self.prot_charact_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_charact_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_charact_LBL.setObjectName("prot_charact_LBL")
self.prot_cal_Vmin_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_pu_DSB.setGeometry(QtCore.QRect(220, 210, 71, 21))
self.prot_cal_Vmin_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_pu_DSB.setDecimals(1)
self.prot_cal_Vmin_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_pu_DSB.setObjectName("prot_cal_Vmin_pu_DSB")
self.prot_delay_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_btm_LN.setGeometry(QtCore.QRect(10, 380, 305, 1))
self.prot_delay_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_btm_LN.setObjectName("prot_delay_btm_LN")
self.prot_cal_I_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_val_unit_LBL.setGeometry(QtCore.QRect(170, 150, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_val_unit_LBL.setFont(font)
self.prot_cal_I_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_unit_LBL.setObjectName("prot_cal_I_val_unit_LBL")
self.prot_cdelay_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cdelay_LBL.setGeometry(QtCore.QRect(110, 260, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cdelay_LBL.setFont(font)
self.prot_cdelay_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_cdelay_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_cdelay_LBL.setObjectName("prot_cdelay_LBL")
self.prot_delay_Vmin_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_unit_LBL.setGeometry(QtCore.QRect(170, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmin_unit_LBL.setFont(font)
self.prot_delay_Vmin_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_unit_LBL.setObjectName("prot_delay_Vmin_unit_LBL")
self.prot_cal_Vmax_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_pu_unit_LBL.setGeometry(QtCore.QRect(300, 180, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmax_pu_unit_LBL.setFont(font)
self.prot_cal_Vmax_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_unit_LBL.setObjectName("prot_cal_Vmax_pu_unit_LBL")
self.prot_cal_I_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_pu_unit_LBL.setGeometry(QtCore.QRect(300, 150, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_pu_unit_LBL.setFont(font)
self.prot_cal_I_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_unit_LBL.setObjectName("prot_cal_I_pu_unit_LBL")
self.prot_cal_Vmin_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_pu_unit_LBL.setGeometry(QtCore.QRect(300, 210, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_pu_unit_LBL.setFont(font)
self.prot_cal_Vmin_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_unit_LBL.setObjectName("prot_cal_Vmin_pu_unit_LBL")
self.tabWidget.addTab(self.Protections, "")
self.bb_out_LBL.raise_()
self.bb_in_LBL.raise_()
self.cub_out_LBL.raise_()
self.top_LN.raise_()
self.elem_name_LN.raise_()
self.elem_name_LBL.raise_()
self.type_LBL.raise_()
self.type_cap_LBL.raise_()
self.cub_in_LBL.raise_()
self.symbol_LBL.raise_()
self.ot_Frame_LN.raise_()
self.vdx_frame_LN.raise_()
self.vsx_frame_LN.raise_()
self.ob_frame_LN.raise_()
self.bb_in_LN.raise_()
self.bb_out_LN.raise_()
self.store_BTN.raise_()
self.cancel_BTN.raise_()
self.tabWidget.raise_()

self.retranslateUi(Form)
self.tabWidget.setCurrentIndex(3)
QtCore.QMetaObject.connectSlotsByName(Form)
Form.setTabOrder(self.tabWidget, self.cap_en_DSB)
Form.setTabOrder(self.cap_en_DSB, self.init_SOC_DSB)
Form.setTabOrder(self.init_SOC_DSB, self.max_SOC_DSB)
Form.setTabOrder(self.max_SOC_DSB, self.min_SOC_DSB)
Form.setTabOrder(self.min_SOC_DSB, self.final_SOC_DSB)
Form.setTabOrder(self.final_SOC_DSB, self.max_dch_DSB)
Form.setTabOrder(self.max_dch_DSB, self.max_ch_DSB)
Form.setTabOrder(self.max_ch_DSB, self.cost_dch_DSB)
Form.setTabOrder(self.cost_dch_DSB, self.cost_ch_DSB)
Form.setTabOrder(self.cost_ch_DSB, self.eta_dch_DSB)
Form.setTabOrder(self.eta_dch_DSB, self.eta_ch_DSB)
Form.setTabOrder(self.eta_ch_DSB, self.rel_TO_DSB)
```

```
        Form.setTabOrder(self.rel_T0_DSB, self.rel_alfa_DSB)
        Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
        Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
        Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
        Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
        Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
        Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_I_P1_DSB)
        Form.setTabOrder(self.res_I_P1_DSB, self.res_Iangle_P1_DSB)
        Form.setTabOrder(self.res_Iangle_P1_DSB, self.res_P_P1_DSB)
        Form.setTabOrder(self.res_P_P1_DSB, self.res_Q_P1_DSB)
        Form.setTabOrder(self.res_Q_P1_DSB, self.res_S_P1_DSB)
        Form.setTabOrder(self.res_S_P1_DSB, self.res_cosPhi_P1_DSB)
        Form.setTabOrder(self.res_cosPhi_P1_DSB, self.res_U_P1_DSB)
        Form.setTabOrder(self.res_U_P1_DSB, self.store_BTN)
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria:"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.store_BTN.setText(_translate("Form", "Salva"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))
        self.init_SOC_LBL.setText(_translate("Form", "SOC Iniziale"))
        self.cap_en_LBL.setText(_translate("Form", "Capacità"))
        self.cap_en_unit.setText(_translate("Form", "kWh"))
        self.init_SOC_unit.setText(_translate("Form", "p.u."))
        self.max_SOC_unit.setText(_translate("Form", "p.u."))
        self.max_SOC_LBL.setText(_translate("Form", "SOC max"))
        self.min_SOC_LBL.setText(_translate("Form", "SOC min"))
        self.min_SOC_unit.setText(_translate("Form", "p.u."))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Parameters), _translate("Form", "Parametri"))
        self.res_U_P1_unit.setText(_translate("Form", "kV"))
        self.res_cosPhi_P1_unit.setText(_translate("Form", "-"))
        self.res_S_LBL.setText(_translate("Form", "S"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMITI VIOLATI"))
        self.res_I_LBL.setText(_translate("Form", " I"))
        self.res_cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.res_Iangle_LBL.setText(_translate("Form", "Angolo I"))
        self.res_Iangle_P1_unit.setText(_translate("Form", "°"))
        self.Por1_LBL.setText(_translate("Form", "Nodo 1"))
        self.res_P_P1_unit.setText(_translate("Form", "kW"))
        self.res_I_P1_unit.setText(_translate("Form", "A"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.res_S_P1_unit.setText(_translate("Form", "kVA"))
        self.res_Q_LBL.setText(_translate("Form", "Q"))
        self.res_P_LBL.setText(_translate("Form", "P"))
        self.res_Q_P1_unit.setText(_translate("Form", "kVA"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFlow), _translate("Form", "LoadFlow"))
        self.final_SOC_unit.setText(_translate("Form", "p.u."))
        self.final_SOC_LBL.setText(_translate("Form", "SOC Finale"))
        self.max_dch_unit.setText(_translate("Form", "kW/kWh"))
        self.max_dch_LBL.setText(_translate("Form", "Velocità di scarica max"))
        self.max_ch_unit.setText(_translate("Form", "kW/kWh"))
        self.max_ch_LBL.setText(_translate("Form", "Velocità di carica max"))
        self.cost_dch_unit.setText(_translate("Form", "€/kWh"))
        self.cost_dch_LBL.setText(_translate("Form", "Costo degrad. in scarica"))
        self.cost_ch_LBL.setText(_translate("Form", "Costo degrad. in carica"))
        self.cost_ch_unit.setText(_translate("Form", "€/kWh"))
        self.eta_dch_unit.setText(_translate("Form", "p.u."))
        self.eta_dch_LBL.setText(_translate("Form", "Efficienza in scarica"))
        self.eta_ch_unit.setText(_translate("Form", "p.u."))
        self.eta_ch_LBL.setText(_translate("Form", "Efficienza in carica"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_T0_unit.setText(_translate("Form", "°C"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Reliability), _translate("Form", "Affidabilità"))
        self.prot_cal_Vmax_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cal_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_delay_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_delay_Vmax_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_cost_LBL.setText(_translate("Form", "Costo"))
        self.prot_delay_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_cost_unit_LBL.setText(_translate("Form", "Euro"))
        self.prot_cal_Vmin_val_unit_LBL.setText(_translate("Form", "kV"))
        self.prot_cal_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_cal_I_LBL.setText(_translate("Form", "Corrente"))
```

```
        self.prot_delay_I_LBL.setText(_translate("Form", "Corrente"))
        self.prot_cal_I_sep_LBL.setText(_translate("Form", "-"))
        self.prot_delay_I_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_cal_Vmin_sep_LBL.setText(_translate("Form", "-"))
        self.prot_calib_LBL.setText(_translate("Form", "Soglie di taratura"))
        self.prot_type_LBL.setText(_translate("Form", "Tipologia"))
        self.prot_cal_Vmax_val_unit_LBL.setText(_translate("Form", "kV"))
        self.prot_charact_LBL.setText(_translate("Form", "Caratterisitiche dell\'interruttore"))
        self.prot_cal_I_val_unit_LBL.setText(_translate("Form", "A"))
        self.prot_cdelay_LBL.setText(_translate("Form", "Tempi di ritardo"))
        self.prot_delay_Vmin_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_cal_Vmax_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_I_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_Vmin_pu_unit_LBL.setText(_translate("Form", "%"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Protections), _translate("Form", "Protezioni"))

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.3.6 DCDC_converter

4.4.3.6.1 dcdc_conv.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .dcdc_convUI import Ui_Form
from __shared__ import variables as v
import copy


class DCDC_conv(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(DCDC_conv, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")
        self.ui.tabWidget.setCurrentIndex(0)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])
        self.prot = copy.deepcopy(v.elements[element]['protections'])

        conn_list = list(v.elements[element]['conn'].keys())
        self.bb1 = v.elements[element]['conn']['h']
        conn_list.remove(self.bb1)
        conn_list.remove('h')
        self.bb2 = conn_list[0]

        self.cubicle1 = v.elements[self.element]['conn'][self.bb1]
        self.cubicle2 = v.elements[self.element]['conn'][self.bb2]

        self.ui.bb_in_LBL.setText(self.bb1)
        self.ui.bb_out_LBL.setText(self.bb2)

        self.ui.symbol_LBL.setPixmap(QtGui.QPixmap("_images/Elements/DCDC_converter/element.png"))
        self.switch_draw()
        self.fill()
        self.tab_activation()

        self.ui.cub_in_LBL.mouseDoubleClickEvent = self.cub1_switch
        self.ui.cub_out_LBL.mouseDoubleClickEvent = self.cub2_switch
        self.ui.tabWidget.currentChanged.connect(self.test)

    #
    def tab_activation(self):
        self.ui.tabWidget.setTabVisible(2, v.functionality == 'EMS')
        self.ui.tabWidget.setTabVisible(3, v.functionality == 'Reliability')
        self.ui.tabWidget.setTabVisible(4, v.protections and self.prot != {})

    #
    def store(self):
        self.par['Sr'] = self.ui.sr_DSB.value()
        v.elements[self.element]['parameters'] = copy.deepcopy(self.par)

        v.elements[self.element]['conn'][self.bb1] = self.cubicle1
        v.elements[self.element]['conn'][self.bb2] = self.cubicle2

        self.ems['in'] = self.bb1
        self.ems['out'] = self.bb2
        for par in ['cap_pwr', 'max_outin', 'max_inout', 'etaoutin', 'etainout']:
            self.ems[par] = self.ui.__getattribute__(par + '_DSB').value()
        v.elements[self.element]['ems'] = copy.deepcopy(self.ems)

        for par in ['T0', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)

        self.protections_par()

    #
    def protections_par(self):
        self.prot['Pn'] = self.par['Sr']
        self.prot['Vn'] = v.elements[self.bb2]['parameters']['Ur']
        self.prot['In'] = self.prot['Pn'] / self.prot['Vn']
        v.elements[self.element]['protections'] = copy.deepcopy(self.prot)

    #
    def fill(self):
        self.ui.sr_DSB.setValue(self.par['Sr'])

        for par in ['cap_pwr', 'max_outin', 'max_inout', 'etaoutin', 'etainout']:
            self.ui.__getattribute__(par + '_DSB').setValue(self.ems[par])

        self.calculate()

        if self.res != {}:
            self.fill_results()
            self.ui.tabWidget.setTabVisible(3, self.res != {})
        self.fill_reliability()
        if self.prot != {}:
            if self.prot['results'] != {} and v.protections:
                self.fill_protections()

    #
    def fill_reliability(self):
        for par in ['T0', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
        for par in self.rel['results']:
            try:
                self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
            except:
                if self.rel['results'][par] == 0:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('0')
                elif self.rel['results'][par] < 0.01:
```

```
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
            else:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])

#
def fill_protections(self):
    self.ui.prot_type_LE.setText(self.prot['results']['type'])
    self.ui.prot_cost_DSB.setValue(self.prot['results']['cost'])
    self.ui.prot_cal_I_val_DSB.setValue(self.prot['results']['soglia_I'])
    self.ui.prot_cal_I_pu_DSB.setValue(self.prot['results']['soglia_I']/self.prot['In'] * 100)
    self.ui.prot_cal_Vmax_val_DSB.setValue(self.prot['results']['soglia_Vmax'])
    self.ui.prot_cal_Vmax_pu_DSB.setValue(self.prot['results']['soglia_Vmax'] / self.prot['Vn'] * 100)
    self.ui.prot_cal_Vmin_val_DSB.setValue(self.prot['results']['soglia_Vmin'])
    self.ui.prot_cal_Vmin_pu_DSB.setValue(self.prot['results']['soglia_Vmin'] / self.prot['Vn'] * 100)
    self.ui.prot_delay_I_DSB.setValue(self.prot['results']['delay_I'])
    self.ui.prot_delay_Vmax_DSB.setValue(self.prot['results']['delay_Vmax'])
    self.ui.prot_delay_Vmin_DSB.setValue(self.prot['results']['delay_Vmin'])

#
def fill_results(self):
    results = ['I', 'Iangle', 'P', 'Q', 'S', 'cosPhi', 'U']
    ports = ['P1', 'P2']
    ports2 = ['Port1', 'Port2']
    for port in ports:
        p = ports2[ports.index(port)]
        for result in results:
            self.ui.__getattribute__('res_' + result + '_' + port + '_DSB').setValue(self.res[p][result])
    self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])
    self.ui.res_Ploss_DSB.setValue(self.res['Ploss'])
    self.ui.res_Qloss_DSB.setValue(self.res['Qloss'])

#
def control_mode(self, ui):
    pass

#
def switch_draw(self):
    if self.cubicle1:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
    else:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))
    if self.cubicle2:
        self.ui.cub_out_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
    else:
        self.ui.cub_out_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))

#
def cub1_switch(self, event):
    self.cubicle1 = not self.cubicle1
    self.switch_draw()

#
def cub2_switch(self, event):
    self.cubicle2 = not self.cubicle2
    self.switch_draw()

#
def calculate(self):
    self.ui.cap_pwr_DSB.setValue(self.ui.sr_DSB.value())
```

4.4.3.6.2 dcdc_convUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'dcdc_convUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(486, 506)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("res/2W_Tr.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.bb_in_LBL.setFont(font)
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_out_LN.setObjectName("bb_out_LN")
        self.bb_out_LBL = QtWidgets.QLabel(self.widget)
        self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
self.bb_out_LBL.setFont(font)
self.bb_out_LBL.setStyleSheet("")
self.bb_out_LBL.setLineWidth(4)
self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.cancel_BTN = QtWidgets.QPushButton(self.widget)
self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.cancel_BTN.setFont(font)
self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.cancel_BTN.setObjectName("cancel_BTN")
self.store_BTN = QtWidgets.QPushButton(self.widget)
self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.store_BTN.setFont(font)
self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.store_BTN.setObjectName("store_BTN")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("background-color: rgb(0, 0,15);")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.sr_unit = QtWidgets.QLabel(self.Parameters)
self.sr_unit.setGeometry(QtCore.QRect(240, 10, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sr_unit.setFont(font)
self.sr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.sr_unit.setObjectName("sr_unit")
self.sr_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.sr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sr_DSB.setFont(font)
self.sr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.sr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.sr_DSB.setDecimals(3)
self.sr_DSB.setMaximum(999999.999)
self.sr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.sr_DSB.setProperty("value", 0.0)
self.sr_DSB.setObjectName("sr_DSB")
self.sr_LBL = QtWidgets.QLabel(self.Parameters)
self.sr_LBL.setGeometry(QtCore.QRect(60, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.sr_LBL.setFont(font)
self.sr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sr_LBL.setObjectName("sr_LBL")
self.tabWidget.addTab(self.Parameters, "")
self.LoadFlow = QtWidgets.QWidget()
self.LoadFlow.setObjectName("LoadFlow")
self.res_Q_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P2_DSB.setEnabled(False)
self.res_Q_P2_DSB.setGeometry(QtCore.QRect(220, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P2_DSB.setFont(font)
self.res_Q_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P2_DSB.setDecimals(3)
self.res_Q_P2_DSB.setMinimum(-999999.999)
self.res_Q_P2_DSB.setMaximum(999999.999)
self.res_Q_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P2_DSB.setProperty("value", 0.0)
self.res_Q_P2_DSB.setObjectName("res_Q_P2_DSB")
self.res_P_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P2_DSB.setEnabled(False)
self.res_P_P2_DSB.setGeometry(QtCore.QRect(220, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P2_DSB.setFont(font)
self.res_P_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P2_DSB.setDecimals(3)
self.res_P_P2_DSB.setMinimum(-999999.999)
self.res_P_P2_DSB.setMaximum(999999.999)
self.res_P_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P2_DSB.setProperty("value", 0.0)
self.res_P_P2_DSB.setObjectName("res_P_P2_DSB")
self.res_cosPhi_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P1_DSB.setEnabled(False)
self.res_cosPhi_P1_DSB.setGeometry(QtCore.QRect(90, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_DSB.setFont(font)
self.res_cosPhi_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P1_DSB.setDecimals(3)
self.res_cosPhi_P1_DSB.setMaximum(999999.999)
self.res_cosPhi_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P1_DSB.setProperty("value", 0.0)
```

```
self.res_cosPhi_P1_DSB.setObjectName("res_cosPhi_P1_DSB")
self.res_I_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P2_unit.setGeometry(QtCore.QRect(300, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P2_unit.setFont(font)
self.res_I_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P2_unit.setObjectName("res_I_P2_unit")
self.res_S_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P1_DSB.setEnabled(False)
self.res_S_P1_DSB.setGeometry(QtCore.QRect(90, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_DSB.setFont(font)
self.res_S_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P1_DSB.setDecimals(3)
self.res_S_P1_DSB.setMaximum(999999.999)
self.res_S_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P1_DSB.setProperty("value", 0.0)
self.res_S_P1_DSB.setObjectName("res_S_P1_DSB")
self.res_cosPhi_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_LBL.setGeometry(QtCore.QRect(0, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_cosPhi_LBL.setFont(font)
self.res_cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_LBL.setObjectName("res_cosPhi_LBL")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_U_LBL.setGeometry(QtCore.QRect(0, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.Port2_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Port2_LBL.setGeometry(QtCore.QRect(220, 10, 101, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Port2_LBL.setFont(font)
self.Port2_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Port2_LBL.setObjectName("Port2_LBL")
self.res_P_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P1_unit.setGeometry(QtCore.QRect(170, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_unit.setFont(font)
self.res_P_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P1_unit.setObjectName("res_P_P1_unit")
self.res_Qloss_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Qloss_unit.setGeometry(QtCore.QRect(170, 300, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qloss_unit.setFont(font)
self.res_Qloss_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Qloss_unit.setObjectName("res_Qloss_unit")
self.res_Ploss_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Ploss_LBL.setGeometry(QtCore.QRect(0, 270, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Ploss_LBL.setFont(font)
self.res_Ploss_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Ploss_LBL.setObjectName("res_Ploss_LBL")
self.res_Iangle_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Iangle_LBL.setFont(font)
self.res_Iangle_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_LBL.setObjectName("res_Iangle_LBL")
self.res_U_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P2_DSB.setEnabled(False)
self.res_U_P2_DSB.setGeometry(QtCore.QRect(220, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P2_DSB.setFont(font)
self.res_U_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P2_DSB.setDecimals(3)
self.res_U_P2_DSB.setMaximum(999999.999)
self.res_U_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P2_DSB.setProperty("value", 0.0)
self.res_U_P2_DSB.setObjectName("res_U_P2_DSB")
self.res_I_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_I_LBL.setGeometry(QtCore.QRect(0, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_I_LBL.setFont(font)
self.res_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_LBL.setObjectName("res_I_LBL")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFlow)
```

```
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(0, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_cosPhi_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P2_unit.setGeometry(QtCore.QRect(300, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P2_unit.setFont(font)
self.res_cosPhi_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P2_unit.setObjectName("res_cosPhi_P2_unit")
self.res_U_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P1_DSB.setEnabled(False)
self.res_U_P1_DSB.setGeometry(QtCore.QRect(90, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_DSB.setFont(font)
self.res_U_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P1_DSB.setDecimals(3)
self.res_U_P1_DSB.setMaximum(999999.999)
self.res_U_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P1_DSB.setProperty("value", 0.0)
self.res_U_P1_DSB.setObjectName("res_U_P1_DSB")
self.res_cosPhi_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P2_DSB.setEnabled(False)
self.res_cosPhi_P2_DSB.setGeometry(QtCore.QRect(220, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P2_DSB.setFont(font)
self.res_cosPhi_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P2_DSB.setDecimals(3)
self.res_cosPhi_P2_DSB.setMaximum(999999.999)
self.res_cosPhi_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P2_DSB.setProperty("value", 0.0)
self.res_cosPhi_P2_DSB.setObjectName("res_cosPhi_P2_DSB")
self.res_Iangle_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P1_DSB.setEnabled(False)
self.res_Iangle_P1_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_DSB.setFont(font)
self.res_Iangle_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P1_DSB.setDecimals(3)
self.res_Iangle_P1_DSB.setMinimum(-999999.999)
self.res_Iangle_P1_DSB.setMaximum(999999.999)
self.res_Iangle_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P1_DSB.setProperty("value", 0.0)
self.res_Iangle_P1_DSB.setObjectName("res_Iangle_P1_DSB")
self.res_S_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P1_unit.setGeometry(QtCore.QRect(170, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_unit.setFont(font)
self.res_S_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P1_unit.setObjectName("res_S_P1_unit")
self.res_Iangle_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P2_unit.setGeometry(QtCore.QRect(300, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P2_unit.setFont(font)
self.res_Iangle_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P2_unit.setObjectName("res_Iangle_P2_unit")
self.res_I_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P1_unit.setGeometry(QtCore.QRect(170, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_unit.setFont(font)
self.res_I_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P1_unit.setObjectName("res_I_P1_unit")
self.res_P_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P2_unit.setGeometry(QtCore.QRect(300, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P2_unit.setFont(font)
self.res_P_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P2_unit.setObjectName("res_P_P2_unit")
self.res_Q_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P1_unit.setGeometry(QtCore.QRect(170, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_unit.setFont(font)
self.res_Q_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P1_unit.setObjectName("res_Q_P1_unit")
self.res_Iangle_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P1_unit.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
```

```
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_unit.setFont(font)
self.res_Iangle_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_unit.setObjectName("res_Iangle_P1_unit")
self.res_Ploss_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Ploss_DSB.setEnabled(False)
self.res_Ploss_DSB.setGeometry(QtCore.QRect(90, 270, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Ploss_DSB.setFont(font)
self.res_Ploss_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Ploss_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Ploss_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Ploss_DSB.setDecimals(3)
self.res_Ploss_DSB.setMaximum(999999.999)
self.res_Ploss_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Ploss_DSB.setProperty("value", 0.0)
self.res_Ploss_DSB.setObjectName("res_Ploss_DSB")
self.res_Iangle_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P2_DSB.setEnabled(False)
self.res_Iangle_P2_DSB.setGeometry(QtCore.QRect(220, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P2_DSB.setFont(font)
self.res_Iangle_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P2_DSB.setDecimals(3)
self.res_Iangle_P2_DSB.setMinimum(-999999.999)
self.res_Iangle_P2_DSB.setMaximum(999999.999)
self.res_Iangle_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P2_DSB.setProperty("value", 0.0)
self.res_Iangle_P2_DSB.setObjectName("res_Iangle_P2_DSB")
self.res_Qloss_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Qloss_DSB.setEnabled(False)
self.res_Qloss_DSB.setGeometry(QtCore.QRect(90, 300, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qloss_DSB.setFont(font)
self.res_Qloss_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Qloss_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qloss_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Qloss_DSB.setDecimals(3)
self.res_Qloss_DSB.setMaximum(999999.999)
self.res_Qloss_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Qloss_DSB.setProperty("value", 0.0)
self.res_Qloss_DSB.setObjectName("res_Qloss_DSB")
self.res_I_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P1_DSB.setEnabled(False)
self.res_I_P1_DSB.setGeometry(QtCore.QRect(90, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_DSB.setFont(font)
self.res_I_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P1_DSB.setDecimals(3)
self.res_I_P1_DSB.setMaximum(999999.999)
self.res_I_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P1_DSB.setProperty("value", 0.0)
self.res_I_P1_DSB.setObjectName("res_I_P1_DSB")
self.res_S_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P2_unit.setGeometry(QtCore.QRect(300, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P2_unit.setFont(font)
self.res_S_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P2_unit.setObjectName("res_S_P2_unit")
self.res_Q_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_LBL.setGeometry(QtCore.QRect(0, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Q_LBL.setFont(font)
self.res_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_LBL.setObjectName("res_Q_LBL")
self.res_P_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_P_LBL.setGeometry(QtCore.QRect(0, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_P_LBL.setFont(font)
self.res_P_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_LBL.setObjectName("res_P_LBL")
self.res_S_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P2_DSB.setEnabled(False)
self.res_S_P2_DSB.setGeometry(QtCore.QRect(220, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P2_DSB.setFont(font)
self.res_S_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P2_DSB.setDecimals(3)
self.res_S_P2_DSB.setMaximum(999999.999)
self.res_S_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P2_DSB.setProperty("value", 0.0)
self.res_S_P2_DSB.setObjectName("res_S_P2_DSB")
self.res_P_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P1_DSB.setEnabled(False)
```

```
self.res_P_P1_DSB.setGeometry(QtCore.QRect(90, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_DSB.setFont(font)
self.res_P_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P1_DSB.setDecimals(3)
self.res_P_P1_DSB.setMinimum(-999999.999)
self.res_P_P1_DSB.setMaximum(999999.999)
self.res_P_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P1_DSB.setProperty("value", 0.0)
self.res_P_P1_DSB.setObjectName("res_P_P1_DSB")
self.res_Q_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P2_unit.setGeometry(QtCore.QRect(300, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P2_unit.setFont(font)
self.res_Q_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P2_unit.setObjectName("res_Q_P2_unit")
self.res_cosPhi_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P1_unit.setGeometry(QtCore.QRect(170, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_unit.setFont(font)
self.res_cosPhi_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_unit.setObjectName("res_cosPhi_P1_unit")
self.res_U_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P2_unit.setGeometry(QtCore.QRect(300, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P2_unit.setFont(font)
self.res_U_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P2_unit.setObjectName("res_U_P2_unit")
self.res_Q_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P1_DSB.setEnabled(False)
self.res_Q_P1_DSB.setGeometry(QtCore.QRect(90, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_DSB.setFont(font)
self.res_Q_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P1_DSB.setDecimals(3)
self.res_Q_P1_DSB.setMinimum(-999999.999)
self.res_Q_P1_DSB.setMaximum(999999.999)
self.res_Q_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P1_DSB.setProperty("value", 0.0)
self.res_Q_P1_DSB.setObjectName("res_Q_P1_DSB")
self.res_I_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P2_DSB.setEnabled(False)
self.res_I_P2_DSB.setGeometry(QtCore.QRect(220, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P2_DSB.setFont(font)
self.res_I_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P2_DSB.setDecimals(3)
self.res_I_P2_DSB.setMaximum(999999.999)
self.res_I_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P2_DSB.setProperty("value", 0.0)
self.res_I_P2_DSB.setObjectName("res_I_P2_DSB")
self.Por1_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Por1_LBL.setGeometry(QtCore.QRect(90, 10, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Por1_LBL.setFont(font)
self.Por1_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Por1_LBL.setObjectName("Por1_LBL")
self.res_S_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_S_LBL.setGeometry(QtCore.QRect(0, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_S_LBL.setFont(font)
self.res_S_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_LBL.setObjectName("res_S_LBL")
self.res_Qloss_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Qloss_LBL.setGeometry(QtCore.QRect(0, 300, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Qloss_LBL.setFont(font)
self.res_Qloss_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qloss_LBL.setObjectName("res_Qloss_LBL")
self.res_U_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P1_unit.setGeometry(QtCore.QRect(170, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_unit.setFont(font)
self.res_U_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P1_unit.setObjectName("res_U_P1_unit")
self.res_Ploss_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Ploss_unit.setGeometry(QtCore.QRect(170, 270, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
```

```
font.setBold(False)
font.setWeight(50)
self.res_Ploss_unit.setFont(font)
self.res_Ploss_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Ploss_unit.setObjectName("res_Ploss_unit")
self.tabWidget.addTab(self.LoadFlow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.cap_pwr_LBL = QtWidgets.QLabel(self.EMS)
self.cap_pwr_LBL.setGeometry(QtCore.QRect(10, 10, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.cap_pwr_LBL.setFont(font)
self.cap_pwr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_LBL.setObjectName("cap_pwr_LBL")
self.cap_pwr_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.cap_pwr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_DSB.setFont(font)
self.cap_pwr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cap_pwr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cap_pwr_DSB.setDecimals(3)
self.cap_pwr_DSB.setMaximum(999999.999)
self.cap_pwr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cap_pwr_DSB.setProperty("value", 0.0)
self.cap_pwr_DSB.setObjectName("cap_pwr_DSB")
self.cap_pwr_unit = QtWidgets.QLabel(self.EMS)
self.cap_pwr_unit.setGeometry(QtCore.QRect(240, 10, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_unit.setFont(font)
self.cap_pwr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cap_pwr_unit.setObjectName("cap_pwr_unit")
self.max_outin_unit = QtWidgets.QLabel(self.EMS)
self.max_outin_unit.setGeometry(QtCore.QRect(240, 40, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_outin_unit.setFont(font)
self.max_outin_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_outin_unit.setObjectName("max_outin_unit")
self.max_outin_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.max_outin_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_outin_DSB.setFont(font)
self.max_outin_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_outin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_outin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_outin_DSB.setDecimals(3)
self.max_outin_DSB.setMaximum(999999.999)
self.max_outin_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_outin_DSB.setProperty("value", 0.0)
self.max_outin_DSB.setObjectName("max_outin_DSB")
self.max_outin_LBL = QtWidgets.QLabel(self.EMS)
self.max_outin_LBL.setGeometry(QtCore.QRect(10, 40, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_outin_LBL.setFont(font)
self.max_outin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_outin_LBL.setObjectName("max_outin_LBL")
self.max_inout_unit = QtWidgets.QLabel(self.EMS)
self.max_inout_unit.setGeometry(QtCore.QRect(240, 70, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_inout_unit.setFont(font)
self.max_inout_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_inout_unit.setObjectName("max_inout_unit")
self.max_inout_LBL = QtWidgets.QLabel(self.EMS)
self.max_inout_LBL.setGeometry(QtCore.QRect(10, 70, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_inout_LBL.setFont(font)
self.max_inout_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_inout_LBL.setObjectName("max_inout_LBL")
self.max_inout_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.max_inout_DSB.setGeometry(QtCore.QRect(160, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_inout_DSB.setFont(font)
self.max_inout_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_inout_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_inout_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_inout_DSB.setDecimals(3)
self.max_inout_DSB.setMaximum(999999.999)
self.max_inout_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_inout_DSB.setProperty("value", 0.0)
self.max_inout_DSB.setObjectName("max_inout_DSB")
self.etainout_unit = QtWidgets.QLabel(self.EMS)
self.etainout_unit.setGeometry(QtCore.QRect(240, 130, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etainout_unit.setFont(font)
self.etainout_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
```

```
self.etainout_unit.setObjectName("etainout_unit")
self.etaoutin_LBL = QtWidgets.QLabel(self.EMS)
self.etaoutin_LBL.setGeometry(QtCore.QRect(10, 100, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.etaoutin_LBL.setFont(font)
self.etaoutin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etaoutin_LBL.setObjectName("etaoutin_LBL")
self.etainout_LBL = QtWidgets.QLabel(self.EMS)
self.etainout_LBL.setGeometry(QtCore.QRect(10, 130, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.etainout_LBL.setFont(font)
self.etainout_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etainout_LBL.setObjectName("etainout_LBL")
self.etaoutin_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.etaoutin_DSB.setGeometry(QtCore.QRect(160, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etaoutin_DSB.setFont(font)
self.etaoutin_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.etaoutin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etaoutin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.etaoutin_DSB.setDecimals(3)
self.etaoutin_DSB.setMaximum(999999.999)
self.etaoutin_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.etaoutin_DSB.setProperty("value", 0.0)
self.etaoutin_DSB.setObjectName("etaoutin_DSB")
self.etainout_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.etainout_DSB.setGeometry(QtCore.QRect(160, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etainout_DSB.setFont(font)
self.etainout_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.etainout_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etainout_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.etainout_DSB.setDecimals(3)
self.etainout_DSB.setMaximum(999999.999)
self.etainout_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.etainout_DSB.setProperty("value", 0.0)
self.etainout_DSB.setObjectName("etainout_DSB")
self.etaoutin_unit = QtWidgets.QLabel(self.EMS)
self.etaoutin_unit.setGeometry(QtCore.QRect(240, 100, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etaoutin_unit.setFont(font)
self.etaoutin_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.etaoutin_unit.setObjectName("etaoutin_unit")
self.tabWidget.addTab(self.EMS, "")
self.Relibility = QtWidgets.QWidget()
self.Relibility.setObjectName("Relibility")
self.rel_results_LBL = QtWidgets.QLabel(self.Relibility)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 70, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_alfa_unit = QtWidgets.QLabel(self.Relibility)
self.rel_alfa_unit.setGeometry(QtCore.QRect(240, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_R_LBL = QtWidgets.QLabel(self.Relibility)
self.rel_R_LBL.setGeometry(QtCore.QRect(160, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_lambda_unit = QtWidgets.QLabel(self.Relibility)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_beta_unit = QtWidgets.QLabel(self.Relibility)
self.rel_beta_unit.setGeometry(QtCore.QRect(240, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Relibility)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(0, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
```

```
self.rel_T0_LBL = QtWidgets.QLabel(self.Relibility)
self.rel_T0_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Relibility)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(160, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Relibility)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(160, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Relibility)
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.bottom_LN_4 = QtWidgets.QFrame(self.Relibility)
self.bottom_LN_4.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN_4.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_4.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_4.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_4.setObjectName("bottom_LN_4")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Relibility)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(160, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Relibility)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Relibility)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Relibility)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
```

```
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Relibility)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(240, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Relibility)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_beta_LBL = QtWidgets.QLabel(self.Relibility)
self.rel_beta_LBL.setGeometry(QtCore.QRect(0, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Relibility)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(240, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_T0_unit = QtWidgets.QLabel(self.Relibility)
self.rel_T0_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Relibility)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(0, 170, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Relibility)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(190, 350, 31, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Relibility)
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(160, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Relibility)
self.rel_MTBF_anni_DSB.setEnabled(False)
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Relibility)
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(0, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_R_unit = QtWidgets.QLabel(self.Relibility)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
```

```
self.rel_R_unit.setObjectName("rel_R_unit")
self.rel_R_LE = QtWidgets.QLineEdit(self.Relibility)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Relibility)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Relibility)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.rel_alfa_unit.raise_()
self.rel_R_LBL.raise_()
self.rel_lambda_unit.raise_()
self.rel_beta_unit.raise_()
self.rel_alfa_LBL.raise_()
self.rel_T0_LBL.raise_()
self.rel_beta_DSB.raise_()
self.rel_alfa_DSB.raise_()
self.rel_lambda_LBL.raise_()
self.bottom_LN_4.raise_()
self.rel_Pi_Q_DSB.raise_()
self.rel_T0_DSB.raise_()
self.rel_MTBF_ore_unit.raise_()
self.rel_MTBF_ore_DSB.raise_()
self.rel_Pi_Q_unit.raise_()
self.rel_MTBF_ore_LBL.raise_()
self.rel_beta_LBL.raise_()
self.rel_Pi_E_unit.raise_()
self.rel_T0_unit.raise_()
self.rel_Pi_Q_LBL.raise_()
self.rel_MTBF_anni_LBL.raise_()
self.rel_Pi_E_DSB.raise_()
self.rel_MTBF_anni_DSB.raise_()
self.rel_Pi_E_LBL.raise_()
self.rel_R_unit.raise_()
self.rel_R_LE.raise_()
self.rel_lambda_LE.raise_()
self.rel_results_LBL.raise_()
self.rel_MTBF_anni_unit.raise_()
self.tabWidget.addTab(self.Relibility, "")
self.Protections = QtWidgets.QWidget()
self.Protections.setObjectName("Protections")
self.prot_cal_I_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_pu_unit_LBL.setGeometry(QtCore.QRect(300, 150, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_pu_unit_LBL.setFont(font)
self.prot_cal_I_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_unit_LBL.setObjectName("prot_cal_I_pu_unit_LBL")
self.prot_cal_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_LBL.setGeometry(QtCore.QRect(0, 150, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_LBL.setFont(font)
self.prot_cal_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_LBL.setObjectName("prot_cal_I_LBL")
self.prot_charact_LBL = QtWidgets.QLabel(self.Protections)
self.prot_charact_LBL.setGeometry(QtCore.QRect(60, 10, 211, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_charact_LBL.setFont(font)
self.prot_charact_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_charact_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_charact_LBL.setObjectName("prot_charact_LBL")
self.prot_cal_I_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_sep_LBL.setGeometry(QtCore.QRect(190, 150, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_sep_LBL.setFont(font)
self.prot_cal_I_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_sep_LBL.setObjectName("prot_cal_I_sep_LBL")
self.prot_cal_I_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_val_unit_LBL.setGeometry(QtCore.QRect(170, 150, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_val_unit_LBL.setFont(font)
self.prot_cal_I_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_unit_LBL.setObjectName("prot_cal_I_val_unit_LBL")
self.prot_char_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_top_LN.setGeometry(QtCore.QRect(10, 20, 305, 1))
self.prot_char_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_top_LN.setObjectName("prot_char_top_LN")
self.prot_type_LE = QtWidgets.QLineEdit(self.Protections)
self.prot_type_LE.setGeometry(QtCore.QRect(90, 40, 201, 20))
self.prot_type_LE.setText("")
self.prot_type_LE.setObjectName("prot_type_LE")
self.prot_type_LBL = QtWidgets.QLabel(self.Protections)
self.prot_type_LBL.setGeometry(QtCore.QRect(0, 40, 81, 20))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
```

```
font.setWeight(75)
self.prot_type_LBL.setFont(font)
self.prot_type_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_type_LBL.setObjectName("prot_type_LBL")
self.prot_calib_LBL = QtWidgets.QLabel(self.Protections)
self.prot_calib_LBL.setGeometry(QtCore.QRect(110, 120, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_calib_LBL.setFont(font)
self.prot_calib_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_calib_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_calib_LBL.setObjectName("prot_calib_LBL")
self.prot_char_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_btm_LN.setGeometry(QtCore.QRect(10, 100, 305, 1))
self.prot_char_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_btm_LN.setObjectName("prot_char_btm_LN")
self.prot_calib_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_top_LN.setGeometry(QtCore.QRect(10, 130, 305, 1))
self.prot_calib_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_calib_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_top_LN.setObjectName("prot_calib_top_LN")
self.prot_calib_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_btm_LN.setGeometry(QtCore.QRect(10, 240, 305, 1))
self.prot_calib_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_calib_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_btm_LN.setObjectName("prot_calib_btm_LN")
self.prot_cal_I_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_val_DSB.setGeometry(QtCore.QRect(90, 150, 71, 21))
self.prot_cal_I_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_val_DSB.setDecimals(1)
self.prot_cal_I_val_DSB.setMaximum(99999999.0)
self.prot_cal_I_val_DSB.setObjectName("prot_cal_I_val_DSB")
self.prot_cal_Vmax_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_pu_unit_LBL.setGeometry(QtCore.QRect(300, 180, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmax_pu_unit_LBL.setFont(font)
self.prot_cal_Vmax_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_unit_LBL.setObjectName("prot_cal_Vmax_pu_unit_LBL")
self.prot_cal_Vmax_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_val_DSB.setGeometry(QtCore.QRect(90, 180, 71, 21))
self.prot_cal_Vmax_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_val_DSB.setDecimals(3)
self.prot_cal_Vmax_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_val_DSB.setObjectName("prot_cal_Vmax_val_DSB")
self.prot_cal_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_LBL.setGeometry(QtCore.QRect(0, 180, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_LBL.setFont(font)
self.prot_cal_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_LBL.setObjectName("prot_cal_Vmax_LBL")
self.prot_cal_Vmax_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_sep_LBL.setGeometry(QtCore.QRect(190, 180, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_sep_LBL.setFont(font)
self.prot_cal_Vmax_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_sep_LBL.setObjectName("prot_cal_Vmax_sep_LBL")
self.prot_delay_I_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_unit_LBL.setGeometry(QtCore.QRect(170, 290, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_I_unit_LBL.setFont(font)
self.prot_delay_I_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_I_unit_LBL.setObjectName("prot_delay_I_unit_LBL")
self.prot_cal_I_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_pu_DSB.setGeometry(QtCore.QRect(220, 150, 71, 21))
self.prot_cal_I_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_pu_DSB.setDecimals(1)
self.prot_cal_I_pu_DSB.setMaximum(99999999.0)
self.prot_cal_I_pu_DSB.setObjectName("prot_cal_I_pu_DSB")
self.prot_cal_Vmax_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_pu_DSB.setGeometry(QtCore.QRect(220, 180, 71, 21))
self.prot_cal_Vmax_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_pu_DSB.setDecimals(1)
self.prot_cal_Vmax_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_pu_DSB.setObjectName("prot_cal_Vmax_pu_DSB")
self.prot_cal_Vmin_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_pu_unit_LBL.setGeometry(QtCore.QRect(300, 210, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_pu_unit_LBL.setFont(font)
self.prot_cal_Vmin_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_unit_LBL.setObjectName("prot_cal_Vmin_pu_unit_LBL")
self.prot_cal_Vmin_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_LBL.setGeometry(QtCore.QRect(0, 210, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_LBL.setFont(font)
self.prot_cal_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_LBL.setObjectName("prot_cal_Vmin_LBL")
self.prot_cal_Vmin_sep_LBL = QtWidgets.QLabel(self.Protections)
```

```
self.prot_cal_Vmin_sep_LBL.setGeometry(QtCore.QRect(190, 210, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_sep_LBL.setFont(font)
self.prot_cal_Vmin_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_sep_LBL.setObjectName("prot_cal_Vmin_sep_LBL")
self.prot_cal_Vmin_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_val_DSB.setGeometry(QtCore.QRect(90, 210, 71, 21))
self.prot_cal_Vmin_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_val_DSB.setDecimals(3)
self.prot_cal_Vmin_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_val_DSB.setObjectName("prot_cal_Vmin_val_DSB")
self.prot_cal_Vmin_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_pu_DSB.setGeometry(QtCore.QRect(220, 210, 71, 21))
self.prot_cal_Vmin_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_pu_DSB.setDecimals(1)
self.prot_cal_Vmin_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_pu_DSB.setObjectName("prot_cal_Vmin_pu_DSB")
self.prot_cal_Vmin_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_val_unit_LBL.setGeometry(QtCore.QRect(170, 210, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_val_unit_LBL.setFont(font)
self.prot_cal_Vmin_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_unit_LBL.setObjectName("prot_cal_Vmin_val_unit_LBL")
self.prot_cdelay_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cdelay_LBL.setGeometry(QtCore.QRect(110, 260, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cdelay_LBL.setFont(font)
self.prot_cdelay_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_cdelay_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_cdelay_LBL.setObjectName("prot_cdelay_LBL")
self.prot_delay_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_top_LN.setGeometry(QtCore.QRect(10, 270, 305, 1))
self.prot_delay_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_top_LN.setObjectName("prot_delay_top_LN")
self.prot_delay_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_btm_LN.setGeometry(QtCore.QRect(10, 380, 305, 1))
self.prot_delay_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_btm_LN.setObjectName("prot_delay_btm_LN")
self.prot_delay_I_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_I_DSB.setGeometry(QtCore.QRect(90, 290, 71, 21))
self.prot_delay_I_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_I_DSB.setDecimals(3)
self.prot_delay_I_DSB.setMaximum(99999999.0)
self.prot_delay_I_DSB.setObjectName("prot_delay_I_DSB")
self.prot_delay_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_LBL.setGeometry(QtCore.QRect(0, 290, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_I_LBL.setFont(font)
self.prot_delay_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_LBL.setObjectName("prot_delay_I_LBL")
self.prot_delay_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_LBL.setGeometry(QtCore.QRect(0, 320, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_Vmax_LBL.setFont(font)
self.prot_delay_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_LBL.setObjectName("prot_delay_Vmax_LBL")
self.prot_delay_Vmax_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmax_DSB.setGeometry(QtCore.QRect(90, 320, 71, 21))
self.prot_delay_Vmax_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmax_DSB.setDecimals(3)
self.prot_delay_Vmax_DSB.setMaximum(99999999.0)
self.prot_delay_Vmax_DSB.setObjectName("prot_delay_Vmax_DSB")
self.prot_delay_Vmax_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_unit_LBL.setGeometry(QtCore.QRect(170, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmax_unit_LBL.setFont(font)
self.prot_delay_Vmax_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_unit_LBL.setObjectName("prot_delay_Vmax_unit_LBL")
self.prot_delay_Vmin_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_unit_LBL.setGeometry(QtCore.QRect(170, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmin_unit_LBL.setFont(font)
self.prot_delay_Vmin_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_unit_LBL.setObjectName("prot_delay_Vmin_unit_LBL")
self.prot_delay_Vmin_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmin_DSB.setGeometry(QtCore.QRect(90, 350, 71, 21))
self.prot_delay_Vmin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmin_DSB.setDecimals(3)
self.prot_delay_Vmin_DSB.setMaximum(99999999.0)
self.prot_delay_Vmin_DSB.setObjectName("prot_delay_Vmin_DSB")
self.prot_delay_Vmin_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_LBL.setGeometry(QtCore.QRect(0, 350, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
```

```
        font.setWeight(75)
        self.prot_delay_Vmin_LBL.setFont(font)
        self.prot_delay_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.prot_delay_Vmin_LBL.setObjectName("prot_delay_Vmin_LBL")
        self.prot_cost_unit_LBL = QtWidgets.QLabel(self.Protections)
        self.prot_cost_unit_LBL.setGeometry(QtCore.QRect(170, 70, 21, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setBold(False)
        font.setWeight(50)
        self.prot_cost_unit_LBL.setFont(font)
        self.prot_cost_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
        self.prot_cost_unit_LBL.setObjectName("prot_cost_unit_LBL")
        self.prot_cost_LBL = QtWidgets.QLabel(self.Protections)
        self.prot_cost_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setBold(True)
        font.setWeight(75)
        self.prot_cost_LBL.setFont(font)
        self.prot_cost_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.prot_cost_LBL.setObjectName("prot_cost_LBL")
        self.prot_cost_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
        self.prot_cost_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
        self.prot_cost_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.prot_cost_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
        self.prot_cost_DSB.setDecimals(2)
        self.prot_cost_DSB.setMaximum(99999999.0)
        self.prot_cost_DSB.setObjectName("prot_cost_DSB")
        self.prot_cal_Vmax_val_unit_LBL = QtWidgets.QLabel(self.Protections)
        self.prot_cal_Vmax_val_unit_LBL.setGeometry(QtCore.QRect(170, 180, 21, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setBold(False)
        font.setWeight(50)
        self.prot_cal_Vmax_val_unit_LBL.setFont(font)
        self.prot_cal_Vmax_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
        self.prot_cal_Vmax_val_unit_LBL.setObjectName("prot_cal_Vmax_val_unit_LBL")
        self.prot_calib_top_LN.raise_()
        self.prot_char_top_LN.raise_()
        self.prot_cal_I_pu_unit_LBL.raise_()
        self.prot_cal_I_LBL.raise_()
        self.prot_charact_LBL.raise_()
        self.prot_cal_I_sep_LBL.raise_()
        self.prot_cal_I_val_unit_LBL.raise_()
        self.prot_type_LE.raise_()
        self.prot_type_LBL.raise_()
        self.prot_calib_LBL.raise_()
        self.prot_char_btm_LN.raise_()
        self.prot_calib_btm_LN.raise_()
        self.prot_cal_I_val_DSB.raise_()
        self.prot_cal_Vmax_pu_unit_LBL.raise_()
        self.prot_cal_Vmax_val_DSB.raise_()
        self.prot_cal_Vmax_LBL.raise_()
        self.prot_cal_Vmax_sep_LBL.raise_()
        self.prot_delay_I_unit_LBL.raise_()
        self.prot_cal_I_pu_DSB.raise_()
        self.prot_cal_Vmax_pu_DSB.raise_()
        self.prot_cal_Vmin_pu_unit_LBL.raise_()
        self.prot_cal_Vmin_LBL.raise_()
        self.prot_cal_Vmin_sep_LBL.raise_()
        self.prot_cal_Vmin_val_DSB.raise_()
        self.prot_cal_Vmin_pu_DSB.raise_()
        self.prot_cal_Vmin_val_unit_LBL.raise_()
        self.prot_delay_top_LN.raise_()
        self.prot_cdelay_LBL.raise_()
        self.prot_delay_btm_LN.raise_()
        self.prot_delay_I_DSB.raise_()
        self.prot_delay_I_LBL.raise_()
        self.prot_delay_Vmax_LBL.raise_()
        self.prot_delay_Vmax_DSB.raise_()
        self.prot_delay_Vmax_unit_LBL.raise_()
        self.prot_delay_Vmin_unit_LBL.raise_()
        self.prot_delay_Vmin_DSB.raise_()
        self.prot_delay_Vmin_LBL.raise_()
        self.prot_cost_unit_LBL.raise_()
        self.prot_cost_LBL.raise_()
        self.prot_cost_DSB.raise_()
        self.prot_cal_Vmax_val_unit_LBL.raise_()
        self.tabWidget.addTab(self.Protections, "")
        self.bb_out_LBL.raise_()
        self.bb_in_LBL.raise_()
        self.cub_out_LBL.raise_()
        self.top_LN.raise_()
        self.elem_name_LN.raise_()
        self.elem_name_LBL.raise_()
        self.type_LBL.raise_()
        self.type_cap_LBL.raise_()
        self.cub_in_LBL.raise_()
        self.symbol_LBL.raise_()
        self.ot_Frame_LN.raise_()
        self.vdx_frame_LN.raise_()
        self.vsx_frame_LN.raise_()
        self.ob_frame_LN.raise_()
        self.bb_in_LN.raise_()
        self.bb_out_LN.raise_()
        self.cancel_BTN.raise_()
        self.store_BTN.raise_()
        self.tabWidget.raise_()

        self.retranslateUi(Form)
        self.tabWidget.setCurrentIndex(3)
        QtCore.QMetaObject.connectSlotsByName(Form)
        Form.setTabOrder(self.tabWidget, self.sr_DSB)
        Form.setTabOrder(self.sr_DSB, self.cap_pwr_DSB)
        Form.setTabOrder(self.cap_pwr_DSB, self.max_outin_DSB)
        Form.setTabOrder(self.max_outin_DSB, self.max_inout_DSB)
        Form.setTabOrder(self.max_inout_DSB, self.etaoutin_DSB)
        Form.setTabOrder(self.etaoutin_DSB, self.etainout_DSB)
        Form.setTabOrder(self.etainout_DSB, self.rel_TO_DSB)
        Form.setTabOrder(self.rel_TO_DSB, self.rel_alfa_DSB)
        Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
        Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
        Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
        Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
```

```
        Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
        Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_I_P1_DSB)
        Form.setTabOrder(self.res_I_P1_DSB, self.res_Iangle_P1_DSB)
        Form.setTabOrder(self.res_Iangle_P1_DSB, self.res_P_P1_DSB)
        Form.setTabOrder(self.res_P_P1_DSB, self.res_Q_P1_DSB)
        Form.setTabOrder(self.res_Q_P1_DSB, self.res_S_P1_DSB)
        Form.setTabOrder(self.res_S_P1_DSB, self.res_cosPhi_P1_DSB)
        Form.setTabOrder(self.res_cosPhi_P1_DSB, self.res_U_P1_DSB)
        Form.setTabOrder(self.res_U_P1_DSB, self.res_I_P2_DSB)
        Form.setTabOrder(self.res_I_P2_DSB, self.res_Iangle_P2_DSB)
        Form.setTabOrder(self.res_Iangle_P2_DSB, self.res_P_P2_DSB)
        Form.setTabOrder(self.res_P_P2_DSB, self.res_Q_P2_DSB)
        Form.setTabOrder(self.res_Q_P2_DSB, self.res_S_P2_DSB)
        Form.setTabOrder(self.res_S_P2_DSB, self.res_cosPhi_P2_DSB)
        Form.setTabOrder(self.res_cosPhi_P2_DSB, self.res_U_P2_DSB)
        Form.setTabOrder(self.res_U_P2_DSB, self.res_Ploss_DSB)
        Form.setTabOrder(self.res_Ploss_DSB, self.res_Qloss_DSB)
        Form.setTabOrder(self.res_Qloss_DSB, self.store_BTN)
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))
        self.store_BTN.setText(_translate("Form", "Salva"))
        self.sr_unit.setText(_translate("Form", "kVA"))
        self.sr_LBL.setText(_translate("Form", "Sr"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Parameters), _translate("Form", "Parametri"))
        self.res_I_P2_unit.setText(_translate("Form", "A"))
        self.res_cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.Port2_LBL.setText(_translate("Form", "Nodo 2"))
        self.res_P_P1_unit.setText(_translate("Form", "kW"))
        self.res_Qloss_unit.setText(_translate("Form", "kVA"))
        self.res_Ploss_LBL.setText(_translate("Form", "P loss"))
        self.res_Iangle_LBL.setText(_translate("Form", "I angle"))
        self.res_I_LBL.setText(_translate("Form", "  I"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMITI VIOLATI"))
        self.res_cosPhi_P2_unit.setText(_translate("Form", "-"))
        self.res_S_P1_unit.setText(_translate("Form", "kVA"))
        self.res_Iangle_P2_unit.setText(_translate("Form", "°"))
        self.res_I_P1_unit.setText(_translate("Form", "A"))
        self.res_P_P2_unit.setText(_translate("Form", "kW"))
        self.res_Q_P1_unit.setText(_translate("Form", "kVA"))
        self.res_Iangle_P1_unit.setText(_translate("Form", "°"))
        self.res_S_P2_unit.setText(_translate("Form", "kVA"))
        self.res_Q_LBL.setText(_translate("Form", "Q"))
        self.res_P_LBL.setText(_translate("Form", "P"))
        self.res_Q_P2_unit.setText(_translate("Form", "kVA"))
        self.res_cosPhi_P1_unit.setText(_translate("Form", "-"))
        self.res_U_P2_unit.setText(_translate("Form", "kV"))
        self.Port_LBL.setText(_translate("Form", "Nodo 1"))
        self.res_S_LBL.setText(_translate("Form", "S"))
        self.res_Qloss_LBL.setText(_translate("Form", "Q loss"))
        self.res_U_P1_unit.setText(_translate("Form", "kV"))
        self.res_Ploss_unit.setText(_translate("Form", "kW"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFlow), _translate("Form", "LoadFlow"))
        self.cap_pwr_LBL.setText(_translate("Form", "Potenza Nominale"))
        self.cap_pwr_unit.setText(_translate("Form", "kVA"))
        self.max_outin_unit.setText(_translate("Form", "p.u."))
        self.max_outin_LBL.setText(_translate("Form", "Potenza Massima IN"))
        self.max_inout_unit.setText(_translate("Form", "p.u."))
        self.max_inout_LBL.setText(_translate("Form", "Potenza Massima OUT"))
        self.etainout_unit.setText(_translate("Form", "p.u."))
        self.etaoutin_LBL.setText(_translate("Form", "Efficienza OUT-to-IN"))
        self.etainout_LBL.setText(_translate("Form", "Efficienza IN-to-OUT"))
        self.etaoutin_unit.setText(_translate("Form", "p.u."))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_T0_unit.setText(_translate("Form", "°C"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Relibility), _translate("Form", "Affidabilità"))
        self.prot_cal_I_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_I_LBL.setText(_translate("Form", "Corrente"))
        self.prot_charact_LBL.setText(_translate("Form", "Caratterisitiche dell\'interruttore"))
        self.prot_cal_I_sep_LBL.setText(_translate("Form", "-"))
```

```
        self.prot_cal_I_val_unit_LBL.setText(_translate("Form", "A"))
        self.prot_type_LBL.setText(_translate("Form", "Tipologia"))
        self.prot_calib_LBL.setText(_translate("Form", "Soglie di taratura"))
        self.prot_cal_Vmax_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_cal_Vmax_sep_LBL.setText(_translate("Form", "-"))
        self.prot_delay_I_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_cal_Vmin_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_cal_Vmin_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cal_Vmin_val_unit_LBL.setText(_translate("Form", "kV"))
        self.prot_cdelay_LBL.setText(_translate("Form", "Tempi di ritardo"))
        self.prot_delay_I_LBL.setText(_translate("Form", "Corrente"))
        self.prot_delay_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_delay_Vmax_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_delay_Vmin_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_delay_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_cost_unit_LBL.setText(_translate("Form", "Euro"))
        self.prot_cost_LBL.setText(_translate("Form", "Costo"))
        self.prot_cal_Vmax_val_unit_LBL.setText(_translate("Form", "kV"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Protections), _translate("Form", "Protezioni"))

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.3.7 DCLine

4.4.3.7.1 dcline.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .dclineUI import Ui_Form
from __shared__ import variables as v
import copy


class DCLine(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(DCLine, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")

        for box in ['cap_pwr', 'etainout', 'etaoutin']:
            self.ui.__getattribute__(box + '_DSB').setStyleSheet("color: rgb(127, 127, 127);")
            self.ui.__getattribute__(box + '_DSB').setEnabled(False)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])
        self.prot = copy.deepcopy(v.elements[element]['protections'])

        conn_list = list(v.elements[element]['conn'].keys())
        try:
            self.bb1 = v.elements[element]['conn']['h']
        except KeyError:
            self.bb1 = conn_list[0]
        conn_list.remove(self.bb1)
        conn_list.remove(('h'))
        self.bb2 = conn_list[0]

        self.cubicle1 = v.elements[self.element]['conn'][self.bb1]
        self.cubicle2 = v.elements[self.element]['conn'][self.bb2]

        self.u = v.elements[self.bb1]['parameters']['Ur']
        self.ui.bb_in_LBL.setText(self.bb1)
        self.ui.bb_out_LBL.setText(self.bb2)

        self.ui.symbol_LBL.setPixmap(QtGui.QPixmap("_images/Elements/ACLine/element.png"))
        self.switch_draw()
        self.fill()
        self.tab_activation()

        for attr in ['length', 'lines', 'R']:
            self.ui.__getattribute__(attr + '_DSB').valueChanged.connect(self.calculate)
        self.ui.cub_in_LBL.mouseDoubleClickEvent = self.cub1_switch
        self.ui.cub_out_LBL.mouseDoubleClickEvent = self.cub2_switch

    #
    def tab_activation(self):
        self.ui.tabWidget.setTabVisible(2, v.functionality == 'EMS')
        self.ui.tabWidget.setTabVisible(3, v.functionality == 'Reliability')
        self.ui.tabWidget.setTabVisible(4, v.protections and self.prot != {})

    #
    def store(self):
        self.par['length'] = self.ui.length_DSB.value()
        self.par['lines'] = int(self.ui.lines_DSB.value())
        self.par['R'] = self.ui.R_DSB.value()
        self.par['L'] = self.ui.L_DSB.value()
        self.par['C'] = self.ui.C_DSB.value()
        self.par['Irmax'] = self.ui.Imax_DSB.value()
        v.elements[self.element]['parameters'] = copy.deepcopy(self.par)

        v.elements[self.element]['conn'][self.bb1] = self.cubicle1
        v.elements[self.element]['conn'][self.bb2] = self.cubicle2

        self.ems['in'] = self.bb1
        self.ems['out'] = self.bb2
        self.ems['cap_pwr'] = self.ui.cap_pwr_DSB.value()
        self.ems['max_outin'] = self.ui.max_outin_DSB.value()
        self.ems['max_inout'] = self.ui.max_inout_DSB.value()
        self.ems['etaoutin'] = self.ui.etaoutin_DSB.value()
        self.ems['etainout'] = self.ui.etainout_DSB.value()
        v.elements[self.element]['ems'] = copy.deepcopy(self.ems)

        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)

        self.protections_par()

    #
    def protections_par(self):
        self.prot['Vn'] = self.u
        self.prot['In'] = self.par['Irmax']
        self.prot['Pn'] = self.prot['Vn'] * self.prot['In']
        v.elements[self.element]['protections'] = copy.deepcopy(self.prot)

    #
    def fill(self):
        self.ui.length_DSB.setValue(self.par['length'])
        self.ui.lines_DSB.setValue(self.par['lines'])
        self.ui.R_DSB.setValue(self.par['R'])
        self.ui.L_DSB.setValue(self.par['L'])
        self.ui.C_DSB.setValue(self.par['C'])
        self.ui.Imax_DSB.setValue(self.par['Irmax'])

        for par in ['max_outin', 'max_inout']:
            self.ui.__getattribute__(par + '_DSB').setValue(self.ems[par])
```

```
    self.calculate()

    if self.res != {}:
        self.fill_results()
    self.ui.tabWidget.setTabVisible(3, self.res != {})
    self.fill_reliability()
    if self.prot != {}:
        if self.prot['results'] != {} and v.protections:
            self.fill_protections()

#
def fill_reliability(self):
    for par in ['T0', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
        self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
    for par in self.rel['results']:
        try:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
        except:
            if self.rel['results'][par] == 0:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('0')
            elif self.rel['results'][par] < 0.01:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
            else:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])

#
def fill_protections(self):
    self.ui.prot_type_LE.setText(self.prot['results']['type'])
    self.ui.prot_cost_DSB.setValue(self.prot['results']['cost'])
    self.ui.prot_cal_I_val_DSB.setValue(self.prot['results']['soglia_I'])
    self.ui.prot_cal_I_pu_DSB.setValue(self.prot['results']['soglia_I']/self.prot['In'] * 100)
    self.ui.prot_cal_Vmax_val_DSB.setValue(self.prot['results']['soglia_Vmax'])
    self.ui.prot_cal_Vmax_pu_DSB.setValue(self.prot['results']['soglia_Vmax'] / self.prot['Vn'] * 100)
    self.ui.prot_cal_Vmin_val_DSB.setValue(self.prot['results']['soglia_Vmin'])
    self.ui.prot_cal_Vmin_pu_DSB.setValue(self.prot['results']['soglia_Vmin'] / self.prot['Vn'] * 100)
    self.ui.prot_delay_I_DSB.setValue(self.prot['results']['delay_I'])
    self.ui.prot_delay_Vmax_DSB.setValue(self.prot['results']['delay_Vmax'])
    self.ui.prot_delay_Vmin_DSB.setValue(self.prot['results']['delay_Vmin'])

#
def fill_results(self):
    results = ['I', 'Iangle', 'P', 'Q', 'S', 'cosPhi', 'U']
    ports = ['P1', 'P2']
    ports2 = ['Port1', 'Port2']
    for port in ports:
        p = ports2[ports.index(port)]
        for result in results:
            self.ui.__getattribute__('res_' + result + '_' + port + '_DSB').setValue(self.res[p][result])
    self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])
    self.ui.res_Ploss_DSB.setValue(self.res['Ploss'])
    self.ui.res_Qloss_DSB.setValue(self.res['Qloss'])

#
def switch_draw(self):
    if self.cubicle1:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
    else:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))
    if self.cubicle2:
        self.ui.cub_out_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
    else:
        self.ui.cub_out_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))

#
def cub1_switch(self, event):
    self.cubicle1 = not self.cubicle1
    self.switch_draw()

#
def cub2_switch(self, event):
    self.cubicle2 = not self.cubicle2
    self.switch_draw()

#
def calculate(self):
    i = self.ui.Imax_DSB.value()    # sarebbe dovuto essere la corrente reale, non la massima
    r = self.ui.R_DSB.value() * self.ui.length_DSB.value() / 1000
    u = self.u * 1000

    if u > 0:
        eta = 1 - (3**0.5 * r * i / u)
    else:
        eta = 1

    self.ui.cap_pwr_DSB.setValue(u / 1000 * self.ui.Imax_DSB.value() * self.ui.lines_DSB.value())
    for par in ['etainout', 'etaoutin']:
        self.ui.__getattribute__(par + '_DSB').setValue(eta)
```

4.4.3.7.2 dclineUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'dclineUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(491, 502)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        font = QtGui.QFont()
        font.setPointSize(9)
        self.widget.setFont(font)
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("res/2W_Tr.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.bb_in_LBL.setFont(font)
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
```

```
self.bb_out_LN.setObjectName("bb_out_LN")
self.bb_out_LBL = QtWidgets.QLabel(self.widget)
self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
self.bb_out_LBL.setFont(font)
self.bb_out_LBL.setStyleSheet("")
self.bb_out_LBL.setLineWidth(4)
self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.store_BTN = QtWidgets.QPushButton(self.widget)
self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.store_BTN.setFont(font)
self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.store_BTN.setObjectName("store_BTN")
self.cancel_BTN = QtWidgets.QPushButton(self.widget)
self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.cancel_BTN.setFont(font)
self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.cancel_BTN.setObjectName("cancel_BTN")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.R_LBL = QtWidgets.QLabel(self.Parameters)
self.R_LBL.setGeometry(QtCore.QRect(60, 70, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.R_LBL.setFont(font)
self.R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.R_LBL.setObjectName("R_LBL")
self.length_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.length_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.length_DSB.setFont(font)
self.length_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.length_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.length_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.length_DSB.setDecimals(1)
self.length_DSB.setMaximum(1000000.0)
self.length_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.length_DSB.setProperty("value", 0.0)
self.length_DSB.setObjectName("length_DSB")
self.lines_unit = QtWidgets.QLabel(self.Parameters)
self.lines_unit.setGeometry(QtCore.QRect(240, 40, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.lines_unit.setFont(font)
self.lines_unit.setText("")
self.lines_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.lines_unit.setObjectName("lines_unit")
self.length_unit = QtWidgets.QLabel(self.Parameters)
self.length_unit.setGeometry(QtCore.QRect(240, 10, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.length_unit.setFont(font)
self.length_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.length_unit.setObjectName("length_unit")
self.lines_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.lines_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.lines_DSB.setFont(font)
self.lines_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.lines_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.lines_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.lines_DSB.setDecimals(0)
self.lines_DSB.setMaximum(1000000.0)
self.lines_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.lines_DSB.setProperty("value", 0.0)
self.lines_DSB.setObjectName("lines_DSB")
self.L_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.L_DSB.setGeometry(QtCore.QRect(160, 100, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.L_DSB.setFont(font)
self.L_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.L_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.L_DSB.setDecimals(5)
self.L_DSB.setMinimum(0.0)
self.L_DSB.setMaximum(999.99999)
self.L_DSB.setSingleStep(0.1)
self.L_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.L_DSB.setProperty("value", 0.0)
self.L_DSB.setObjectName("L_DSB")
self.R_unit = QtWidgets.QLabel(self.Parameters)
self.R_unit.setGeometry(QtCore.QRect(240, 70, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.R_unit.setFont(font)
self.R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
```

```
self.R_unit.setObjectName("R_unit")
self.C_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.C_DSB.setGeometry(QtCore.QRect(160, 130, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.C_DSB.setFont(font)
self.C_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.C_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.C_DSB.setDecimals(5)
self.C_DSB.setMaximum(999.99999)
self.C_DSB.setSingleStep(0.1)
self.C_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.C_DSB.setProperty("value", 0.0)
self.C_DSB.setObjectName("C_DSB")
self.C_LBL = QtWidgets.QLabel(self.Parameters)
self.C_LBL.setGeometry(QtCore.QRect(60, 130, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.C_LBL.setFont(font)
self.C_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.C_LBL.setObjectName("C_LBL")
self.L_LBL = QtWidgets.QLabel(self.Parameters)
self.L_LBL.setGeometry(QtCore.QRect(60, 100, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.L_LBL.setFont(font)
self.L_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.L_LBL.setObjectName("L_LBL")
self.L_unit = QtWidgets.QLabel(self.Parameters)
self.L_unit.setGeometry(QtCore.QRect(240, 100, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.L_unit.setFont(font)
self.L_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.L_unit.setObjectName("L_unit")
self.C_unit = QtWidgets.QLabel(self.Parameters)
self.C_unit.setGeometry(QtCore.QRect(240, 130, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.C_unit.setFont(font)
self.C_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.C_unit.setObjectName("C_unit")
self.lines_LBL = QtWidgets.QLabel(self.Parameters)
self.lines_LBL.setGeometry(QtCore.QRect(60, 40, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.lines_LBL.setFont(font)
self.lines_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.lines_LBL.setObjectName("lines_LBL")
self.R_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.R_DSB.setGeometry(QtCore.QRect(160, 70, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.R_DSB.setFont(font)
self.R_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.R_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.R_DSB.setDecimals(5)
self.R_DSB.setMaximum(999999.999)
self.R_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.R_DSB.setProperty("value", 0.0)
self.R_DSB.setObjectName("R_DSB")
self.length_LBL = QtWidgets.QLabel(self.Parameters)
self.length_LBL.setGeometry(QtCore.QRect(60, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.length_LBL.setFont(font)
self.length_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.length_LBL.setObjectName("length_LBL")
self.Imax_unit = QtWidgets.QLabel(self.Parameters)
self.Imax_unit.setGeometry(QtCore.QRect(240, 160, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.Imax_unit.setFont(font)
self.Imax_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.Imax_unit.setObjectName("Imax_unit")
self.Imax_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.Imax_DSB.setGeometry(QtCore.QRect(160, 160, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.Imax_DSB.setFont(font)
self.Imax_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.Imax_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.Imax_DSB.setDecimals(3)
self.Imax_DSB.setMaximum(1000.0)
self.Imax_DSB.setSingleStep(0.1)
self.Imax_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.Imax_DSB.setProperty("value", 0.0)
self.Imax_DSB.setObjectName("Imax_DSB")
self.Imax_LBL = QtWidgets.QLabel(self.Parameters)
self.Imax_LBL.setGeometry(QtCore.QRect(60, 160, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Imax_LBL.setFont(font)
```

```
self.Imax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.Imax_LBL.setObjectName("Imax_LBL")
self.tabWidget.addTab(self.Parameters, "")
self.LoadFlow = QtWidgets.QWidget()
self.LoadFlow.setObjectName("LoadFlow")
self.res_P_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P2_unit.setGeometry(QtCore.QRect(300, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P2_unit.setFont(font)
self.res_P_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P2_unit.setObjectName("res_P_P2_unit")
self.res_cosPhi_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P2_DSB.setEnabled(False)
self.res_cosPhi_P2_DSB.setGeometry(QtCore.QRect(220, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P2_DSB.setFont(font)
self.res_cosPhi_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P2_DSB.setDecimals(3)
self.res_cosPhi_P2_DSB.setMaximum(999999.999)
self.res_cosPhi_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P2_DSB.setProperty("value", 0.0)
self.res_cosPhi_P2_DSB.setObjectName("res_cosPhi_P2_DSB")
self.Port2_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Port2_LBL.setGeometry(QtCore.QRect(220, 10, 101, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Port2_LBL.setFont(font)
self.Port2_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Port2_LBL.setObjectName("Port2_LBL")
self.res_U_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P2_unit.setGeometry(QtCore.QRect(300, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P2_unit.setFont(font)
self.res_U_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P2_unit.setObjectName("res_U_P2_unit")
self.res_S_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P1_DSB.setEnabled(False)
self.res_S_P1_DSB.setGeometry(QtCore.QRect(90, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_DSB.setFont(font)
self.res_S_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P1_DSB.setDecimals(3)
self.res_S_P1_DSB.setMaximum(999999.999)
self.res_S_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P1_DSB.setProperty("value", 0.0)
self.res_S_P1_DSB.setObjectName("res_S_P1_DSB")
self.res_Qloss_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Qloss_unit.setGeometry(QtCore.QRect(170, 300, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qloss_unit.setFont(font)
self.res_Qloss_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Qloss_unit.setObjectName("res_Qloss_unit")
self.res_cosPhi_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P2_unit.setGeometry(QtCore.QRect(300, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P2_unit.setFont(font)
self.res_cosPhi_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P2_unit.setObjectName("res_cosPhi_P2_unit")
self.res_P_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P1_unit.setGeometry(QtCore.QRect(170, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_unit.setFont(font)
self.res_P_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P1_unit.setObjectName("res_P_P1_unit")
self.res_Q_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P1_unit.setGeometry(QtCore.QRect(170, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_unit.setFont(font)
self.res_Q_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P1_unit.setObjectName("res_Q_P1_unit")
self.res_U_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P1_DSB.setEnabled(False)
self.res_U_P1_DSB.setGeometry(QtCore.QRect(90, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_DSB.setFont(font)
self.res_U_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P1_DSB.setDecimals(3)
self.res_U_P1_DSB.setMaximum(999999.999)
self.res_U_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P1_DSB.setProperty("value", 0.0)
```

```
self.res_U_P1_DSB.setObjectName("res_U_P1_DSB")
self.res_cosPhi_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P1_DSB.setEnabled(False)
self.res_cosPhi_P1_DSB.setGeometry(QtCore.QRect(90, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_DSB.setFont(font)
self.res_cosPhi_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P1_DSB.setDecimals(3)
self.res_cosPhi_P1_DSB.setMaximum(999999.999)
self.res_cosPhi_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P1_DSB.setProperty("value", 0.0)
self.res_cosPhi_P1_DSB.setObjectName("res_cosPhi_P1_DSB")
self.res_Q_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P2_DSB.setEnabled(False)
self.res_Q_P2_DSB.setGeometry(QtCore.QRect(220, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P2_DSB.setFont(font)
self.res_Q_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P2_DSB.setDecimals(3)
self.res_Q_P2_DSB.setMinimum(-999999.999)
self.res_Q_P2_DSB.setMaximum(999999.999)
self.res_Q_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P2_DSB.setProperty("value", 0.0)
self.res_Q_P2_DSB.setObjectName("res_Q_P2_DSB")
self.res_Ploss_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Ploss_LBL.setGeometry(QtCore.QRect(0, 270, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Ploss_LBL.setFont(font)
self.res_Ploss_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Ploss_LBL.setObjectName("res_Ploss_LBL")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(0, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_Q_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_LBL.setGeometry(QtCore.QRect(0, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Q_LBL.setFont(font)
self.res_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_LBL.setObjectName("res_Q_LBL")
self.res_Q_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P1_DSB.setEnabled(False)
self.res_Q_P1_DSB.setGeometry(QtCore.QRect(90, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_DSB.setFont(font)
self.res_Q_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P1_DSB.setDecimals(3)
self.res_Q_P1_DSB.setMinimum(-999999.999)
self.res_Q_P1_DSB.setMaximum(999999.999)
self.res_Q_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P1_DSB.setProperty("value", 0.0)
self.res_Q_P1_DSB.setObjectName("res_Q_P1_DSB")
self.res_Q_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P2_unit.setGeometry(QtCore.QRect(300, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P2_unit.setFont(font)
self.res_Q_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P2_unit.setObjectName("res_Q_P2_unit")
self.res_Qloss_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Qloss_LBL.setGeometry(QtCore.QRect(0, 300, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Qloss_LBL.setFont(font)
self.res_Qloss_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qloss_LBL.setObjectName("res_Qloss_LBL")
self.res_Iangle_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P2_unit.setGeometry(QtCore.QRect(300, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P2_unit.setFont(font)
self.res_Iangle_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P2_unit.setObjectName("res_Iangle_P2_unit")
self.res_Ploss_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Ploss_unit.setGeometry(QtCore.QRect(170, 270, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Ploss_unit.setFont(font)
self.res_Ploss_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Ploss_unit.setObjectName("res_Ploss_unit")
self.res_I_LBL = QtWidgets.QLabel(self.LoadFlow)
```

```
self.res_I_LBL.setGeometry(QtCore.QRect(0, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_I_LBL.setFont(font)
self.res_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_LBL.setObjectName("res_I_LBL")
self.res_cosPhi_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P1_unit.setGeometry(QtCore.QRect(170, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_unit.setFont(font)
self.res_cosPhi_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_unit.setObjectName("res_cosPhi_P1_unit")
self.res_Iangle_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P1_DSB.setEnabled(False)
self.res_Iangle_P1_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_DSB.setFont(font)
self.res_Iangle_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P1_DSB.setDecimals(3)
self.res_Iangle_P1_DSB.setMinimum(-999999.999)
self.res_Iangle_P1_DSB.setMaximum(999999.999)
self.res_Iangle_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P1_DSB.setProperty("value", 0.0)
self.res_Iangle_P1_DSB.setObjectName("res_Iangle_P1_DSB")
self.res_P_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P2_DSB.setEnabled(False)
self.res_P_P2_DSB.setGeometry(QtCore.QRect(220, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P2_DSB.setFont(font)
self.res_P_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P2_DSB.setDecimals(3)
self.res_P_P2_DSB.setMinimum(-999999.999)
self.res_P_P2_DSB.setMaximum(999999.999)
self.res_P_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P2_DSB.setProperty("value", 0.0)
self.res_P_P2_DSB.setObjectName("res_P_P2_DSB")
self.res_I_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P1_DSB.setEnabled(False)
self.res_I_P1_DSB.setGeometry(QtCore.QRect(90, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_DSB.setFont(font)
self.res_I_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P1_DSB.setDecimals(3)
self.res_I_P1_DSB.setMaximum(999999.999)
self.res_I_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P1_DSB.setProperty("value", 0.0)
self.res_I_P1_DSB.setObjectName("res_I_P1_DSB")
self.res_S_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P2_DSB.setEnabled(False)
self.res_S_P2_DSB.setGeometry(QtCore.QRect(220, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P2_DSB.setFont(font)
self.res_S_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P2_DSB.setDecimals(3)
self.res_S_P2_DSB.setMaximum(999999.999)
self.res_S_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P2_DSB.setProperty("value", 0.0)
self.res_S_P2_DSB.setObjectName("res_S_P2_DSB")
self.res_Iangle_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P1_unit.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_unit.setFont(font)
self.res_Iangle_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_unit.setObjectName("res_Iangle_P1_unit")
self.res_I_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P2_DSB.setEnabled(False)
self.res_I_P2_DSB.setGeometry(QtCore.QRect(220, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P2_DSB.setFont(font)
self.res_I_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P2_DSB.setDecimals(3)
self.res_I_P2_DSB.setMaximum(999999.999)
self.res_I_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P2_DSB.setProperty("value", 0.0)
self.res_I_P2_DSB.setObjectName("res_I_P2_DSB")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_U_LBL.setGeometry(QtCore.QRect(0, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
```

```
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.res_S_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_S_LBL.setGeometry(QtCore.QRect(0, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_S_LBL.setFont(font)
self.res_S_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_LBL.setObjectName("res_S_LBL")
self.res_cosPhi_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_LBL.setGeometry(QtCore.QRect(0, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_cosPhi_LBL.setFont(font)
self.res_cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_LBL.setObjectName("res_cosPhi_LBL")
self.res_I_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P2_unit.setGeometry(QtCore.QRect(300, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P2_unit.setFont(font)
self.res_I_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P2_unit.setObjectName("res_I_P2_unit")
self.res_Qloss_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Qloss_DSB.setEnabled(False)
self.res_Qloss_DSB.setGeometry(QtCore.QRect(90, 300, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qloss_DSB.setFont(font)
self.res_Qloss_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Qloss_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qloss_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Qloss_DSB.setDecimals(3)
self.res_Qloss_DSB.setMaximum(999999.999)
self.res_Qloss_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Qloss_DSB.setProperty("value", 0.0)
self.res_Qloss_DSB.setObjectName("res_Qloss_DSB")
self.res_I_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P1_unit.setGeometry(QtCore.QRect(170, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_unit.setFont(font)
self.res_I_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P1_unit.setObjectName("res_I_P1_unit")
self.res_U_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P2_DSB.setEnabled(False)
self.res_U_P2_DSB.setGeometry(QtCore.QRect(220, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P2_DSB.setFont(font)
self.res_U_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P2_DSB.setDecimals(3)
self.res_U_P2_DSB.setMaximum(999999.999)
self.res_U_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P2_DSB.setProperty("value", 0.0)
self.res_U_P2_DSB.setObjectName("res_U_P2_DSB")
self.res_Ploss_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Ploss_DSB.setEnabled(False)
self.res_Ploss_DSB.setGeometry(QtCore.QRect(90, 270, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Ploss_DSB.setFont(font)
self.res_Ploss_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Ploss_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Ploss_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Ploss_DSB.setDecimals(3)
self.res_Ploss_DSB.setMaximum(999999.999)
self.res_Ploss_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Ploss_DSB.setProperty("value", 0.0)
self.res_Ploss_DSB.setObjectName("res_Ploss_DSB")
self.res_P_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P1_DSB.setEnabled(False)
self.res_P_P1_DSB.setGeometry(QtCore.QRect(90, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_DSB.setFont(font)
self.res_P_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P1_DSB.setDecimals(3)
self.res_P_P1_DSB.setMinimum(-999999.999)
self.res_P_P1_DSB.setMaximum(999999.999)
self.res_P_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P1_DSB.setProperty("value", 0.0)
self.res_P_P1_DSB.setObjectName("res_P_P1_DSB")
self.res_Iangle_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Iangle_LBL.setFont(font)
self.res_Iangle_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_LBL.setObjectName("res_Iangle_LBL")
self.res_Iangle_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P2_DSB.setEnabled(False)
self.res_Iangle_P2_DSB.setGeometry(QtCore.QRect(220, 70, 71, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P2_DSB.setFont(font)
self.res_Iangle_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P2_DSB.setDecimals(3)
self.res_Iangle_P2_DSB.setMinimum(-999999.999)
self.res_Iangle_P2_DSB.setMaximum(999999.999)
self.res_Iangle_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P2_DSB.setProperty("value", 0.0)
self.res_Iangle_P2_DSB.setObjectName("res_Iangle_P2_DSB")
self.res_P_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_P_LBL.setGeometry(QtCore.QRect(0, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_P_LBL.setFont(font)
self.res_P_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_LBL.setObjectName("res_P_LBL")
self.res_U_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P1_unit.setGeometry(QtCore.QRect(170, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_unit.setFont(font)
self.res_U_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P1_unit.setObjectName("res_U_P1_unit")
self.Por1_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Por1_LBL.setGeometry(QtCore.QRect(90, 10, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Por1_LBL.setFont(font)
self.Por1_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Por1_LBL.setObjectName("Por1_LBL")
self.res_S_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P2_unit.setGeometry(QtCore.QRect(300, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P2_unit.setFont(font)
self.res_S_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P2_unit.setObjectName("res_S_P2_unit")
self.res_S_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P1_unit.setGeometry(QtCore.QRect(170, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_unit.setFont(font)
self.res_S_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P1_unit.setObjectName("res_S_P1_unit")
self.tabWidget.addTab(self.LoadFlow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.cap_pwr_LBL = QtWidgets.QLabel(self.EMS)
self.cap_pwr_LBL.setGeometry(QtCore.QRect(10, 10, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.cap_pwr_LBL.setFont(font)
self.cap_pwr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_LBL.setObjectName("cap_pwr_LBL")
self.cap_pwr_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.cap_pwr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_DSB.setFont(font)
self.cap_pwr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cap_pwr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cap_pwr_DSB.setDecimals(3)
self.cap_pwr_DSB.setMaximum(999999.999)
self.cap_pwr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cap_pwr_DSB.setProperty("value", 0.0)
self.cap_pwr_DSB.setObjectName("cap_pwr_DSB")
self.cap_pwr_unit = QtWidgets.QLabel(self.EMS)
self.cap_pwr_unit.setGeometry(QtCore.QRect(240, 10, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_unit.setFont(font)
self.cap_pwr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cap_pwr_unit.setObjectName("cap_pwr_unit")
self.max_outin_unit = QtWidgets.QLabel(self.EMS)
self.max_outin_unit.setGeometry(QtCore.QRect(240, 40, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_outin_unit.setFont(font)
self.max_outin_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_outin_unit.setObjectName("max_outin_unit")
self.max_outin_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.max_outin_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_outin_DSB.setFont(font)
self.max_outin_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_outin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_outin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_outin_DSB.setDecimals(3)
```

```
self.max_outin_DSB.setMaximum(999999.999)
self.max_outin_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_outin_DSB.setProperty("value", 0.0)
self.max_outin_DSB.setObjectName("max_outin_DSB")
self.max_outin_LBL = QtWidgets.QLabel(self.EMS)
self.max_outin_LBL.setGeometry(QtCore.QRect(10, 40, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_outin_LBL.setFont(font)
self.max_outin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_outin_LBL.setObjectName("max_outin_LBL")
self.max_inout_unit = QtWidgets.QLabel(self.EMS)
self.max_inout_unit.setGeometry(QtCore.QRect(240, 70, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_inout_unit.setFont(font)
self.max_inout_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_inout_unit.setObjectName("max_inout_unit")
self.max_inout_LBL = QtWidgets.QLabel(self.EMS)
self.max_inout_LBL.setGeometry(QtCore.QRect(10, 70, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_inout_LBL.setFont(font)
self.max_inout_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_inout_LBL.setObjectName("max_inout_LBL")
self.max_inout_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.max_inout_DSB.setGeometry(QtCore.QRect(160, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_inout_DSB.setFont(font)
self.max_inout_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_inout_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_inout_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_inout_DSB.setDecimals(3)
self.max_inout_DSB.setMaximum(999999.999)
self.max_inout_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_inout_DSB.setProperty("value", 0.0)
self.max_inout_DSB.setObjectName("max_inout_DSB")
self.etainout_unit = QtWidgets.QLabel(self.EMS)
self.etainout_unit.setGeometry(QtCore.QRect(240, 130, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etainout_unit.setFont(font)
self.etainout_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.etainout_unit.setObjectName("etainout_unit")
self.etaoutin_LBL = QtWidgets.QLabel(self.EMS)
self.etaoutin_LBL.setGeometry(QtCore.QRect(10, 100, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.etaoutin_LBL.setFont(font)
self.etaoutin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etaoutin_LBL.setObjectName("etaoutin_LBL")
self.etainout_LBL = QtWidgets.QLabel(self.EMS)
self.etainout_LBL.setGeometry(QtCore.QRect(10, 130, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.etainout_LBL.setFont(font)
self.etainout_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etainout_LBL.setObjectName("etainout_LBL")
self.etaoutin_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.etaoutin_DSB.setGeometry(QtCore.QRect(160, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etaoutin_DSB.setFont(font)
self.etaoutin_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.etaoutin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etaoutin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.etaoutin_DSB.setDecimals(3)
self.etaoutin_DSB.setMaximum(999999.999)
self.etaoutin_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.etaoutin_DSB.setProperty("value", 0.0)
self.etaoutin_DSB.setObjectName("etaoutin_DSB")
self.etainout_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.etainout_DSB.setGeometry(QtCore.QRect(160, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etainout_DSB.setFont(font)
self.etainout_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.etainout_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etainout_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.etainout_DSB.setDecimals(3)
self.etainout_DSB.setMaximum(999999.999)
self.etainout_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.etainout_DSB.setProperty("value", 0.0)
self.etainout_DSB.setObjectName("etainout_DSB")
self.etaoutin_unit = QtWidgets.QLabel(self.EMS)
self.etaoutin_unit.setGeometry(QtCore.QRect(240, 100, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etaoutin_unit.setFont(font)
self.etaoutin_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.etaoutin_unit.setObjectName("etaoutin_unit")
self.tabWidget.addTab(self.EMS, "")
self.Reliability = QtWidgets.QWidget()
self.Reliability.setObjectName("Reliability")
```

```
self.rel_results_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 65, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_alfa_unit = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_unit.setGeometry(QtCore.QRect(240, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_R_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_R_LBL.setGeometry(QtCore.QRect(160, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_lambda_unit = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_beta_unit = QtWidgets.QLabel(self.Reliability)
self.rel_beta_unit.setGeometry(QtCore.QRect(240, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(0, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
self.rel_T0_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_T0_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(160, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(160, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.bottom_LN_2 = QtWidgets.QFrame(self.Reliability)
self.bottom_LN_2.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN_2.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_2.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_2.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_2.setObjectName("bottom_LN_2")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(160, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
```

```
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(240, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_beta_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_beta_LBL.setGeometry(QtCore.QRect(0, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(240, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_T0_unit = QtWidgets.QLabel(self.Reliability)
self.rel_T0_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(0, 170, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(160, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
```

```
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(160, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_anni_DSB.setEnabled(False)
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(0, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_R_unit = QtWidgets.QLabel(self.Reliability)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_R_unit.setObjectName("rel_R_unit")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_R_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.bottom_LN_2.raise_()
self.rel_results_LBL.raise_()
self.rel_alfa_unit.raise_()
self.rel_R_LBL.raise_()
self.rel_lambda_unit.raise_()
self.rel_beta_unit.raise_()
self.rel_alfa_LBL.raise_()
self.rel_T0_LBL.raise_()
self.rel_beta_DSB.raise_()
self.rel_alfa_DSB.raise_()
self.rel_lambda_LBL.raise_()
self.rel_Pi_Q_DSB.raise_()
self.rel_T0_DSB.raise_()
self.rel_MTBF_ore_unit.raise_()
self.rel_MTBF_ore_DSB.raise_()
self.rel_Pi_Q_unit.raise_()
self.rel_MTBF_ore_LBL.raise_()
self.rel_beta_LBL.raise_()
self.rel_Pi_E_unit.raise_()
self.rel_T0_unit.raise_()
self.rel_Pi_Q_LBL.raise_()
self.rel_MTBF_anni_LBL.raise_()
self.rel_Pi_E_DSB.raise_()
self.rel_MTBF_anni_DSB.raise_()
self.rel_Pi_E_LBL.raise_()
self.rel_R_unit.raise_()
self.rel_lambda_LE.raise_()
self.rel_R_LE.raise_()
self.rel_MTBF_anni_unit.raise_()
self.tabWidget.addTab(self.Reliability, "")
self.Protections = QtWidgets.QWidget()
self.Protections.setObjectName("Protections")
self.prot_cal_Vmin_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_val_unit_LBL.setGeometry(QtCore.QRect(170, 210, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_val_unit_LBL.setFont(font)
```

```
self.prot_cal_Vmin_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_unit_LBL.setObjectName("prot_cal_Vmin_val_unit_LBL")
self.prot_cal_Vmin_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_LBL.setGeometry(QtCore.QRect(0, 210, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_LBL.setFont(font)
self.prot_cal_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_LBL.setObjectName("prot_cal_Vmin_LBL")
self.prot_calib_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_btm_LN.setGeometry(QtCore.QRect(10, 240, 305, 1))
self.prot_calib_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_calib_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_btm_LN.setObjectName("prot_calib_btm_LN")
self.prot_calib_LBL = QtWidgets.QLabel(self.Protections)
self.prot_calib_LBL.setGeometry(QtCore.QRect(110, 120, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_calib_LBL.setFont(font)
self.prot_calib_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_calib_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_calib_LBL.setObjectName("prot_calib_LBL")
self.prot_cal_Vmax_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_val_DSB.setGeometry(QtCore.QRect(90, 180, 71, 21))
self.prot_cal_Vmax_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_val_DSB.setDecimals(3)
self.prot_cal_Vmax_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_val_DSB.setObjectName("prot_cal_Vmax_val_DSB")
self.prot_cal_Vmin_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_pu_DSB.setGeometry(QtCore.QRect(220, 210, 71, 21))
self.prot_cal_Vmin_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_pu_DSB.setDecimals(1)
self.prot_cal_Vmin_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_pu_DSB.setObjectName("prot_cal_Vmin_pu_DSB")
self.prot_cal_Vmin_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_pu_unit_LBL.setGeometry(QtCore.QRect(300, 210, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_pu_unit_LBL.setFont(font)
self.prot_cal_Vmin_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_unit_LBL.setObjectName("prot_cal_Vmin_pu_unit_LBL")
self.prot_delay_Vmin_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_unit_LBL.setGeometry(QtCore.QRect(170, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmin_unit_LBL.setFont(font)
self.prot_delay_Vmin_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_unit_LBL.setObjectName("prot_delay_Vmin_unit_LBL")
self.prot_type_LBL = QtWidgets.QLabel(self.Protections)
self.prot_type_LBL.setGeometry(QtCore.QRect(0, 40, 81, 20))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_type_LBL.setFont(font)
self.prot_type_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_type_LBL.setObjectName("prot_type_LBL")
self.prot_cal_Vmax_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_pu_DSB.setGeometry(QtCore.QRect(220, 180, 71, 21))
self.prot_cal_Vmax_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_pu_DSB.setDecimals(1)
self.prot_cal_Vmax_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_pu_DSB.setObjectName("prot_cal_Vmax_pu_DSB")
self.prot_cal_I_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_sep_LBL.setGeometry(QtCore.QRect(190, 150, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_sep_LBL.setFont(font)
self.prot_cal_I_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_sep_LBL.setObjectName("prot_cal_I_sep_LBL")
self.prot_cost_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cost_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cost_LBL.setFont(font)
self.prot_cost_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cost_LBL.setObjectName("prot_cost_LBL")
self.prot_delay_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_btm_LN.setGeometry(QtCore.QRect(10, 380, 305, 1))
self.prot_delay_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_btm_LN.setObjectName("prot_delay_btm_LN")
self.prot_delay_Vmin_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_LBL.setGeometry(QtCore.QRect(0, 350, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_Vmin_LBL.setFont(font)
self.prot_delay_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_LBL.setObjectName("prot_delay_Vmin_LBL")
self.prot_delay_Vmin_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmin_DSB.setGeometry(QtCore.QRect(90, 350, 71, 21))
self.prot_delay_Vmin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmin_DSB.setDecimals(3)
self.prot_delay_Vmin_DSB.setMaximum(99999999.0)
self.prot_delay_Vmin_DSB.setObjectName("prot_delay_Vmin_DSB")
```

```
self.prot_cost_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cost_unit_LBL.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cost_unit_LBL.setFont(font)
self.prot_cost_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cost_unit_LBL.setObjectName("prot_cost_unit_LBL")
self.prot_calib_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_top_LN.setGeometry(QtCore.QRect(10, 130, 305, 1))
self.prot_calib_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_calib_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_top_LN.setObjectName("prot_calib_top_LN")
self.prot_delay_I_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_I_DSB.setGeometry(QtCore.QRect(90, 290, 71, 21))
self.prot_delay_I_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_I_DSB.setDecimals(3)
self.prot_delay_I_DSB.setMaximum(99999999.0)
self.prot_delay_I_DSB.setObjectName("prot_delay_I_DSB")
self.prot_cal_Vmax_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_sep_LBL.setGeometry(QtCore.QRect(190, 180, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_sep_LBL.setFont(font)
self.prot_cal_Vmax_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_sep_LBL.setObjectName("prot_cal_Vmax_sep_LBL")
self.prot_delay_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_top_LN.setGeometry(QtCore.QRect(10, 270, 305, 1))
self.prot_delay_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_top_LN.setObjectName("prot_delay_top_LN")
self.prot_cal_Vmin_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_sep_LBL.setGeometry(QtCore.QRect(190, 210, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_sep_LBL.setFont(font)
self.prot_cal_Vmin_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_sep_LBL.setObjectName("prot_cal_Vmin_sep_LBL")
self.prot_cdelay_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cdelay_LBL.setGeometry(QtCore.QRect(110, 260, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cdelay_LBL.setFont(font)
self.prot_cdelay_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_cdelay_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_cdelay_LBL.setObjectName("prot_cdelay_LBL")
self.prot_delay_I_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_unit_LBL.setGeometry(QtCore.QRect(170, 290, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_I_unit_LBL.setFont(font)
self.prot_delay_I_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_I_unit_LBL.setObjectName("prot_delay_I_unit_LBL")
self.prot_cal_Vmax_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_val_unit_LBL.setGeometry(QtCore.QRect(170, 180, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmax_val_unit_LBL.setFont(font)
self.prot_cal_Vmax_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_val_unit_LBL.setObjectName("prot_cal_Vmax_val_unit_LBL")
self.prot_char_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_top_LN.setGeometry(QtCore.QRect(10, 20, 305, 1))
self.prot_char_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_top_LN.setObjectName("prot_char_top_LN")
self.prot_delay_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_LBL.setGeometry(QtCore.QRect(0, 320, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_Vmax_LBL.setFont(font)
self.prot_delay_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_LBL.setObjectName("prot_delay_Vmax_LBL")
self.prot_delay_Vmax_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmax_DSB.setGeometry(QtCore.QRect(90, 320, 71, 21))
self.prot_delay_Vmax_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmax_DSB.setDecimals(3)
self.prot_delay_Vmax_DSB.setMaximum(99999999.0)
self.prot_delay_Vmax_DSB.setObjectName("prot_delay_Vmax_DSB")
self.prot_cal_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_LBL.setGeometry(QtCore.QRect(0, 180, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_LBL.setFont(font)
self.prot_cal_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_LBL.setObjectName("prot_cal_Vmax_LBL")
self.prot_charact_LBL = QtWidgets.QLabel(self.Protections)
self.prot_charact_LBL.setGeometry(QtCore.QRect(60, 10, 211, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_charact_LBL.setFont(font)
self.prot_charact_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_charact_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_charact_LBL.setObjectName("prot_charact_LBL")
```

```
self.prot_cal_I_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_pu_unit_LBL.setGeometry(QtCore.QRect(300, 150, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_pu_unit_LBL.setFont(font)
self.prot_cal_I_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_unit_LBL.setObjectName("prot_cal_I_pu_unit_LBL")
self.prot_cal_Vmin_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_val_DSB.setGeometry(QtCore.QRect(90, 210, 71, 21))
self.prot_cal_Vmin_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_val_DSB.setDecimals(3)
self.prot_cal_Vmin_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_val_DSB.setObjectName("prot_cal_Vmin_val_DSB")
self.prot_cal_Vmax_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_pu_unit_LBL.setGeometry(QtCore.QRect(300, 180, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmax_pu_unit_LBL.setFont(font)
self.prot_cal_Vmax_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_unit_LBL.setObjectName("prot_cal_Vmax_pu_unit_LBL")
self.prot_cal_I_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_val_unit_LBL.setGeometry(QtCore.QRect(170, 150, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_val_unit_LBL.setFont(font)
self.prot_cal_I_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_unit_LBL.setObjectName("prot_cal_I_val_unit_LBL")
self.prot_char_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_btm_LN.setGeometry(QtCore.QRect(10, 100, 305, 1))
self.prot_char_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_btm_LN.setObjectName("prot_char_btm_LN")
self.prot_type_LE = QtWidgets.QLineEdit(self.Protections)
self.prot_type_LE.setGeometry(QtCore.QRect(90, 40, 201, 20))
self.prot_type_LE.setText("")
self.prot_type_LE.setObjectName("prot_type_LE")
self.prot_cal_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_LBL.setGeometry(QtCore.QRect(0, 150, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_LBL.setFont(font)
self.prot_cal_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_LBL.setObjectName("prot_cal_I_LBL")
self.prot_delay_Vmax_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_unit_LBL.setGeometry(QtCore.QRect(170, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmax_unit_LBL.setFont(font)
self.prot_delay_Vmax_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_unit_LBL.setObjectName("prot_delay_Vmax_unit_LBL")
self.prot_cal_I_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_val_DSB.setGeometry(QtCore.QRect(90, 150, 71, 21))
self.prot_cal_I_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_val_DSB.setDecimals(1)
self.prot_cal_I_val_DSB.setMaximum(99999999.0)
self.prot_cal_I_val_DSB.setObjectName("prot_cal_I_val_DSB")
self.prot_cost_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cost_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
self.prot_cost_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cost_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cost_DSB.setDecimals(2)
self.prot_cost_DSB.setMaximum(99999999.0)
self.prot_cost_DSB.setObjectName("prot_cost_DSB")
self.prot_delay_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_LBL.setGeometry(QtCore.QRect(0, 290, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_I_LBL.setFont(font)
self.prot_delay_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_LBL.setObjectName("prot_delay_I_LBL")
self.prot_cal_I_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_pu_DSB.setGeometry(QtCore.QRect(220, 150, 71, 21))
self.prot_cal_I_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_pu_DSB.setDecimals(1)
self.prot_cal_I_pu_DSB.setMaximum(99999999.0)
self.prot_cal_I_pu_DSB.setObjectName("prot_cal_I_pu_DSB")
self.prot_cal_Vmin_val_unit_LBL.raise_()
self.prot_cal_Vmin_LBL.raise_()
self.prot_calib_btm_LN.raise_()
self.prot_cal_Vmax_val_DSB.raise_()
self.prot_cal_Vmin_pu_DSB.raise_()
self.prot_cal_Vmin_pu_unit_LBL.raise_()
self.prot_delay_Vmin_unit_LBL.raise_()
self.prot_type_LBL.raise_()
self.prot_cal_Vmax_pu_DSB.raise_()
self.prot_cal_I_sep_LBL.raise_()
self.prot_cost_LBL.raise_()
self.prot_delay_btm_LN.raise_()
self.prot_delay_Vmin_LBL.raise_()
self.prot_delay_Vmin_DSB.raise_()
self.prot_cost_unit_LBL.raise_()
self.prot_calib_top_LN.raise_()
self.prot_delay_I_DSB.raise_()
self.prot_cal_Vmax_sep_LBL.raise_()
self.prot_delay_top_LN.raise_()
self.prot_cal_Vmin_sep_LBL.raise_()
self.prot_cdelay_LBL.raise_()
self.prot_delay_I_unit_LBL.raise_()
self.prot_cal_Vmax_val_unit_LBL.raise_()
```

```
        self.prot_char_top_LN.raise_()
        self.prot_delay_Vmax_LBL.raise_()
        self.prot_delay_Vmax_DSB.raise_()
        self.prot_cal_Vmax_LBL.raise_()
        self.prot_charact_LBL.raise_()
        self.prot_cal_I_pu_unit_LBL.raise_()
        self.prot_cal_Vmin_val_DSB.raise_()
        self.prot_cal_Vmax_pu_unit_LBL.raise_()
        self.prot_cal_I_val_unit_LBL.raise_()
        self.prot_char_btm_LN.raise_()
        self.prot_type_LE.raise_()
        self.prot_cal_I_LBL.raise_()
        self.prot_delay_Vmax_unit_LBL.raise_()
        self.prot_cal_I_val_DSB.raise_()
        self.prot_cost_DSB.raise_()
        self.prot_delay_I_LBL.raise_()
        self.prot_cal_I_pu_DSB.raise_()
        self.prot_calib_LBL.raise_()
        self.tabWidget.addTab(self.Protections, "")
        self.bb_out_LBL.raise_()
        self.bb_in_LBL.raise_()
        self.cub_out_LBL.raise_()
        self.top_LN.raise_()
        self.elem_name_LN.raise_()
        self.elem_name_LBL.raise_()
        self.type_LBL.raise_()
        self.type_cap_LBL.raise_()
        self.cub_in_LBL.raise_()
        self.symbol_LBL.raise_()
        self.ot_Frame_LN.raise_()
        self.vdx_frame_LN.raise_()
        self.vsx_frame_LN.raise_()
        self.ob_frame_LN.raise_()
        self.bb_in_LN.raise_()
        self.bb_out_LN.raise_()
        self.store_BTN.raise_()
        self.cancel_BTN.raise_()
        self.tabWidget.raise_()

        self.retranslateUi(Form)
        self.tabWidget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(Form)
        Form.setTabOrder(self.tabWidget, self.length_DSB)
        Form.setTabOrder(self.length_DSB, self.lines_DSB)
        Form.setTabOrder(self.lines_DSB, self.R_DSB)
        Form.setTabOrder(self.R_DSB, self.L_DSB)
        Form.setTabOrder(self.L_DSB, self.C_DSB)
        Form.setTabOrder(self.C_DSB, self.Imax_DSB)
        Form.setTabOrder(self.Imax_DSB, self.cap_pwr_DSB)
        Form.setTabOrder(self.cap_pwr_DSB, self.max_outin_DSB)
        Form.setTabOrder(self.max_outin_DSB, self.max_inout_DSB)
        Form.setTabOrder(self.max_inout_DSB, self.etaoutin_DSB)
        Form.setTabOrder(self.etaoutin_DSB, self.etainout_DSB)
        Form.setTabOrder(self.etainout_DSB, self.rel_TO_DSB)
        Form.setTabOrder(self.rel_TO_DSB, self.rel_alfa_DSB)
        Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
        Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
        Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
        Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
        Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
        Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_I_P1_DSB)
        Form.setTabOrder(self.res_I_P1_DSB, self.res_Iangle_P1_DSB)
        Form.setTabOrder(self.res_Iangle_P1_DSB, self.res_P_P1_DSB)
        Form.setTabOrder(self.res_P_P1_DSB, self.res_Q_P1_DSB)
        Form.setTabOrder(self.res_Q_P1_DSB, self.res_S_P1_DSB)
        Form.setTabOrder(self.res_S_P1_DSB, self.res_cosPhi_P1_DSB)
        Form.setTabOrder(self.res_cosPhi_P1_DSB, self.res_U_P1_DSB)
        Form.setTabOrder(self.res_U_P1_DSB, self.res_I_P2_DSB)
        Form.setTabOrder(self.res_I_P2_DSB, self.res_Iangle_P2_DSB)
        Form.setTabOrder(self.res_Iangle_P2_DSB, self.res_P_P2_DSB)
        Form.setTabOrder(self.res_P_P2_DSB, self.res_Q_P2_DSB)
        Form.setTabOrder(self.res_Q_P2_DSB, self.res_S_P2_DSB)
        Form.setTabOrder(self.res_S_P2_DSB, self.res_cosPhi_P2_DSB)
        Form.setTabOrder(self.res_cosPhi_P2_DSB, self.res_U_P2_DSB)
        Form.setTabOrder(self.res_U_P2_DSB, self.res_Ploss_DSB)
        Form.setTabOrder(self.res_Ploss_DSB, self.res_Qloss_DSB)
        Form.setTabOrder(self.res_Qloss_DSB, self.store_BTN)
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria:"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.store_BTN.setText(_translate("Form", "Salva"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))
        self.R_LBL.setText(_translate("Form", "R"))
        self.length_unit.setText(_translate("Form", "m"))
        self.R_unit.setText(_translate("Form", "Ohm/km"))
        self.C_LBL.setText(_translate("Form", "C"))
        self.L_LBL.setText(_translate("Form", "L"))
        self.L_unit.setText(_translate("Form", "mH/km"))
        self.C_unit.setText(_translate("Form", "F/km"))
        self.lines_LBL.setText(_translate("Form", "N. linee"))
        self.length_LBL.setText(_translate("Form", "Lunghezza"))
        self.Imax_unit.setText(_translate("Form","A"))
        self.Imax_LBL.setText(_translate("Form", "I max"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Parameters), _translate("Form", "Parametri"))
        self.res_P_P2_unit.setText(_translate("Form", "kW"))
        self.Port2_LBL.setText(_translate("Form", "Nodo 2"))
        self.res_U_P2_unit.setText(_translate("Form", "kV"))
        self.res_Qloss_unit.setText(_translate("Form", "kVA"))
        self.res_cosPhi_P2_unit.setText(_translate("Form", "-"))
        self.res_P_P1_unit.setText(_translate("Form", "kW"))
        self.res_Q_P1_unit.setText(_translate("Form", "kVA"))
        self.res_Ploss_LBL.setText(_translate("Form", "P loss"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMITI VIOLATI"))
        self.res_Q_LBL.setText(_translate("Form", "Q"))
        self.res_Q_P2_unit.setText(_translate("Form", "kVA"))
        self.res_Qloss_LBL.setText(_translate("Form", "Q loss"))
        self.res_Iangle_P2_unit.setText(_translate("Form", "°"))
```

```
        self.res_Ploss_unit.setText(_translate("Form", "kW"))
        self.res_I_LBL.setText(_translate("Form", "_I"))
        self.res_cosPhi_P1_unit.setText(_translate("Form", "-"))
        self.res_Iangle_P1_unit.setText(_translate("Form", "°"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.res_S_LBL.setText(_translate("Form", "S"))
        self.res_cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.res_I_P2_unit.setText(_translate("Form", "A"))
        self.res_I_P1_unit.setText(_translate("Form", "A"))
        self.res_Iangle_LBL.setText(_translate("Form", "Angolo I"))
        self.res_P_LBL.setText(_translate("Form", "P"))
        self.res_U_P1_unit.setText(_translate("Form", "kV"))
        self.Por1_LBL.setText(_translate("Form", "Nodo 1"))
        self.res_S_P2_unit.setText(_translate("Form", "kVA"))
        self.res_S_P1_unit.setText(_translate("Form", "kVA"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFlow), _translate("Form", "LoadFlow"))
        self.cap_pwr_LBL.setText(_translate("Form", "Potenza Nominale"))
        self.cap_pwr_unit.setText(_translate("Form", "kVA"))
        self.max_outin_unit.setText(_translate("Form", "p.u."))
        self.max_outin_LBL.setText(_translate("Form", "Potenza Massima IN"))
        self.max_inout_unit.setText(_translate("Form", "p.u."))
        self.max_inout_LBL.setText(_translate("Form", "Potenza Massima OUT"))
        self.etainout_unit.setText(_translate("Form", "p.u."))
        self.etaoutin_LBL.setText(_translate("Form", "Efficienza OUT-to-IN"))
        self.etainout_LBL.setText(_translate("Form", "Efficienza IN-to-OUT"))
        self.etaoutin_unit.setText(_translate("Form", "p.u."))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_T0_unit.setText(_translate("Form", "°C"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Reliability), _translate("Form", "Affidabilità"))
        self.prot_cal_Vmin_val_unit_LBL.setText(_translate("Form", "kV"))
        self.prot_cal_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_calib_LBL.setText(_translate("Form", "Soglie di taratura"))
        self.prot_cal_Vmin_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_delay_Vmin_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_type_LBL.setText(_translate("Form", "Tipologia"))
        self.prot_cal_I_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cost_LBL.setText(_translate("Form", "Costo"))
        self.prot_delay_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_cost_unit_LBL.setText(_translate("Form", "Euro"))
        self.prot_cal_Vmax_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cal_Vmin_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cdelay_LBL.setText(_translate("Form", "Tempi di ritardo"))
        self.prot_delay_I_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_cal_Vmax_val_unit_LBL.setText(_translate("Form", "kV"))
        self.prot_delay_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_cal_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_charact_LBL.setText(_translate("Form", "Caratterisitiche dell\'interruttore"))
        self.prot_cal_I_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_Vmax_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_I_val_unit_LBL.setText(_translate("Form", "A"))
        self.prot_cal_I_LBL.setText(_translate("Form", "Corrente"))
        self.prot_delay_Vmax_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_delay_I_LBL.setText(_translate("Form", "Corrente"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Protections), _translate("Form", "Protezioni"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.3.8 DCLoads

4.4.3.8.1 dcload.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .dcloadUI import Ui_Form
from __shared__ import variables as v
import copy


class DCLoad(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(DCLoad, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.bb_out_LN.setVisible(False)
        self.ui.bb_out_LBL.setVisible(False)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")

        self.ui.tabWidget.setCurrentIndex(0)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])
        self.prot = copy.deepcopy(v.elements[element]['protections'])

        self.bb = v.elements[element]['conn']['h']
        self.cubicle = v.elements[self.element]['conn'][self.bb]
        self.u = v.elements[self.bb]['parameters']['Ur']
        self.ui.bb_in_LBL.setText(self.bb)

        self.ui.symbol_LBL.setPixmap(QtGui.QPixmap("_images/Elements/DCLoads/element.png"))
        self.switch_draw()
        self.fill()
        self.calculate()
        self.tab_activation()

        for attr in ['p_DSB', 'i_DSB', 'r_DSB']:
            self.ui.__getattribute__(attr).valueChanged.connect(self.calculate)
        self.ui.control_CB.currentIndexChanged.connect(self.calculate)
        self.ui.cub_in_LBL.mouseDoubleClickEvent = self.cub1_switch

    #
    def tab_activation(self):
        self.ui.tabWidget.setTabVisible(2, v.functionality == 'EMS')
        self.ui.tabWidget.setTabVisible(3, v.functionality == 'Reliability')
        self.ui.tabWidget.setTabVisible(4, v.protections and self.prot != {})

    #
    def calculate(self):
        u = self.u * 1000
        p = self.ui.p_DSB.value() * 1000
        r = self.ui.r_DSB.value()
        i = self.ui.i_DSB.value()

        if self.ui.control_CB.currentIndex() == 0 and p != 0 and u != 0:
            r = (u**2) / p
            i = p / u
            self.ui.i_DSB.setValue(i)
            self.ui.r_DSB.setValue(r)
        elif self.ui.control_CB.currentIndex() == 1 and i != 0:
            p = u * i
            r = u / i
            self.ui.p_DSB.setValue(p / 1000)
            self.ui.r_DSB.setValue(r)
        elif self.ui.control_CB.currentIndex() == 2 and r != 0:
            p = (u**2) / r
            i = u / r
            self.ui.p_DSB.setValue(p / 1000)
            self.ui.i_DSB.setValue(i)

    #
    def store(self):
        self.par['P'] = self.ui.p_DSB.value()
        self.par['I'] = self.ui.i_DSB.value()
        self.par['R'] = self.ui.r_DSB.value()
        self.par['control'] = self.ui.control_CB.currentIndex()
        v.elements[self.element]['parameters'] = copy.deepcopy(self.par)

        v.elements[self.element]['conn'][self.bb] = self.cubicle

        self.ems['unmet_cost'] = self.ui.unmet_cost_DSB.value()
        v.elements[self.element]['ems'] = copy.deepcopy(self.ems)

        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)

        self.protections_par()

    #
    def protections_par(self):
        self.prot['Vn'] = self.u
        self.prot['In'] = self.par['I']
        self.prot['Pn'] = self.prot['Vn'] * self.prot['In']
        v.elements[self.element]['protections'] = copy.deepcopy(self.prot)

    #
    def fill(self):
        self.ui.p_DSB.setValue(self.par['P'])
        self.ui.i_DSB.setValue(self.par['I'])
        self.ui.r_DSB.setValue(self.par['R'])
        self.ui.control_CB.setCurrentIndex(self.par['control'])
        self.calculate()
```

```
        self.control_mode(self.ui)

        self.ui.unmet_cost_DSB.setValue(self.ems['unmet_cost'])

        if self.res != {}:
            self.fill_results()
        self.ui.tabWidget.setTabVisible(3, self.res != {})
        self.fill_reliability()
        if self.prot != {}:
            if self.prot['results'] != {} and v.protections:
                self.fill_protections()

    #
    def fill_reliability(self):
        for par in ['T0', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
        for par in ['lambda', 'R', 'MTBF_ore', 'MTBF_anni']:
            try:
                self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
            except:
                if self.rel['results'][par] == 0:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('0')
                elif self.rel['results'][par] < 0.01:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
                else:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])

    #
    def fill_protections(self):
        self.ui.prot_type_LE.setText(self.prot['results']['type'])
        self.ui.prot_cost_DSB.setValue(self.prot['results']['cost'])
        self.ui.prot_cal_I_val_DSB.setValue(self.prot['results']['soglia_I'])
        self.ui.prot_cal_I_pu_DSB.setValue(self.prot['results']['soglia_I']/self.prot['In'] * 100)
        self.ui.prot_cal_Vmax_val_DSB.setValue(self.prot['results']['soglia_Vmax'])
        self.ui.prot_cal_Vmax_pu_DSB.setValue(self.prot['results']['soglia_Vmax'] / self.prot['Vn'] * 100)
        self.ui.prot_cal_Vmin_val_DSB.setValue(self.prot['results']['soglia_Vmin'])
        self.ui.prot_cal_Vmin_pu_DSB.setValue(self.prot['results']['soglia_Vmin'] / self.prot['Vn'] * 100)
        self.ui.prot_delay_I_DSB.setValue(self.prot['results']['delay_I'])
        self.ui.prot_delay_Vmax_DSB.setValue(self.prot['results']['delay_Vmax'])
        self.ui.prot_delay_Vmin_DSB.setValue(self.prot['results']['delay_Vmin'])

    #
    def fill_results(self):
        results = ['I', 'Iangle', 'P', 'Q', 'S', 'cosPhi', 'U']
        ports = ['P1']
        for port in ports:
            for result in results:
                self.ui.__getattribute__('res_' + result + '_' + port + '_DSB').setValue(self.res[result])
        self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])

    #
    def control_mode(self, ui):
        enabled = [True, False, False]
        if ui.control_CB.currentIndex() == 0:
            enabled = [True, False, False]
        elif ui.control_CB.currentIndex() == 1:
            enabled = [False, True, False]
        elif ui.control_CB.currentIndex() == 2:
            enabled = [False, False, True]

        i = 0
        for obj in [ui.p_DSB, ui.i_DSB, ui.r_DSB]:
            if enabled[i]:
                obj.setStyleSheet("color: rgb(255, 255, 255);")
            else:
                obj.setStyleSheet("color: rgb(127, 127, 127);")
            obj.setEnabled(enabled[i])
            i += 1

    #
    def switch_draw(self):
        if self.cubicle:
            self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
        else:
            self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))

    #
    def cub1_switch(self, event):
        self.cubicle = not self.cubicle
        self.switch_draw()

    #
    def calc_profile(self, pu_profile):
        profile = []
        for data in pu_profile:
            profile.append(data * self.par['P'])
        return profile
```

4.4.3.8.2 dcloadUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'dcloadUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(492, 503)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("res/Load.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_out_LN.setObjectName("bb_out_LN")
        self.bb_out_LBL = QtWidgets.QLabel(self.widget)
        self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
        self.bb_out_LBL.setStyleSheet("")
        self.bb_out_LBL.setLineWidth(4)
        self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
```

```
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.store_BTN = QtWidgets.QPushButton(self.widget)
self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.store_BTN.setFont(font)
self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.store_BTN.setObjectName("store_BTN")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.sf_profile_RB = QtWidgets.QRadioButton(self.Parameters)
self.sf_profile_RB.setGeometry(QtCore.QRect(240, 150, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
self.sf_profile_RB.setFont(font)
self.sf_profile_RB.setObjectName("sf_profile_RB")
self.i_LBL = QtWidgets.QLabel(self.Parameters)
self.i_LBL.setGeometry(QtCore.QRect(60, 70, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.i_LBL.setFont(font)
self.i_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.i_LBL.setObjectName("i_LBL")
self.p_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.p_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.p_DSB.setFont(font)
self.p_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.p_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.p_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.p_DSB.setDecimals(3)
self.p_DSB.setMaximum(999999.999)
self.p_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.p_DSB.setProperty("value", 0.0)
self.p_DSB.setObjectName("p_DSB")
self.p_unit = QtWidgets.QLabel(self.Parameters)
self.p_unit.setGeometry(QtCore.QRect(240, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.p_unit.setFont(font)
self.p_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.p_unit.setObjectName("p_unit")
self.sf_const_RB = QtWidgets.QRadioButton(self.Parameters)
self.sf_const_RB.setGeometry(QtCore.QRect(240, 130, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
self.sf_const_RB.setFont(font)
self.sf_const_RB.setObjectName("sf_const_RB")
self.r_LBL = QtWidgets.QLabel(self.Parameters)
self.r_LBL.setGeometry(QtCore.QRect(60, 100, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.r_LBL.setFont(font)
self.r_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.r_LBL.setObjectName("r_LBL")
self.i_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.i_DSB.setGeometry(QtCore.QRect(160, 70, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.i_DSB.setFont(font)
self.i_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.i_DSB.setReadOnly(False)
self.i_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.i_DSB.setDecimals(3)
self.i_DSB.setMaximum(999999.999)
self.i_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.i_DSB.setProperty("value", 0.0)
self.i_DSB.setObjectName("i_DSB")
self.control_CB = QtWidgets.QComboBox(self.Parameters)
self.control_CB.setGeometry(QtCore.QRect(160, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.control_CB.setFont(font)
self.control_CB.setObjectName("control_CB")
self.control_CB.addItem("")
self.control_CB.addItem("")
self.control_CB.addItem("")
self.lftype_LBL = QtWidgets.QLabel(self.Parameters)
self.lftype_LBL.setGeometry(QtCore.QRect(60, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.lftype_LBL.setFont(font)
self.lftype_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.lftype_LBL.setObjectName("lftype_LBL")
self.sf_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.sf_DSB.setGeometry(QtCore.QRect(160, 130, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sf_DSB.setFont(font)
self.sf_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sf_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
```

```
self.sf_DSB.setDecimals(5)
self.sf_DSB.setMaximum(1.0)
self.sf_DSB.setSingleStep(0.01)
self.sf_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.sf_DSB.setProperty("value", 0.0)
self.sf_DSB.setObjectName("sf_DSB")
self.r_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.r_DSB.setGeometry(QtCore.QRect(160, 100, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.r_DSB.setFont(font)
self.r_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.r_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.r_DSB.setDecimals(3)
self.r_DSB.setMaximum(999999.999)
self.r_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.r_DSB.setProperty("value", 0.0)
self.r_DSB.setObjectName("r_DSB")
self.p_LBL = QtWidgets.QLabel(self.Parameters)
self.p_LBL.setGeometry(QtCore.QRect(60, 40, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.p_LBL.setFont(font)
self.p_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.p_LBL.setObjectName("p_LBL")
self.sfi_LBL = QtWidgets.QLabel(self.Parameters)
self.sfi_LBL.setGeometry(QtCore.QRect(60, 130, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.sfi_LBL.setFont(font)
self.sfi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sfi_LBL.setObjectName("sfi_LBL")
self.s_unit = QtWidgets.QLabel(self.Parameters)
self.s_unit.setGeometry(QtCore.QRect(240, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.s_unit.setFont(font)
self.s_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.s_unit.setObjectName("s_unit")
self.i_unit = QtWidgets.QLabel(self.Parameters)
self.i_unit.setGeometry(QtCore.QRect(240, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.i_unit.setFont(font)
self.i_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.i_unit.setObjectName("i_unit")
self.tabWidget.addTab(self.Parameters, "")
self.LoadFlow = QtWidgets.QWidget()
self.LoadFlow.setObjectName("LoadFlow")
self.res_U_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P1_DSB.setEnabled(False)
self.res_U_P1_DSB.setGeometry(QtCore.QRect(90, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_DSB.setFont(font)
self.res_U_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P1_DSB.setDecimals(3)
self.res_U_P1_DSB.setMaximum(999999.999)
self.res_U_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P1_DSB.setProperty("value", 0.0)
self.res_U_P1_DSB.setObjectName("res_U_P1_DSB")
self.res_S_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P1_unit.setGeometry(QtCore.QRect(170, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_unit.setFont(font)
self.res_S_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P1_unit.setObjectName("res_S_P1_unit")
self.res_P_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_P_LBL.setGeometry(QtCore.QRect(0, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_P_LBL.setFont(font)
self.res_P_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_LBL.setObjectName("res_P_LBL")
self.res_S_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P1_DSB.setEnabled(False)
self.res_S_P1_DSB.setGeometry(QtCore.QRect(90, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_DSB.setFont(font)
self.res_S_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P1_DSB.setDecimals(3)
self.res_S_P1_DSB.setMaximum(999999.999)
self.res_S_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P1_DSB.setProperty("value", 0.0)
self.res_S_P1_DSB.setObjectName("res_S_P1_DSB")
self.res_Q_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_LBL.setGeometry(QtCore.QRect(0, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
```

```
self.res_Q_LBL.setFont(font)
self.res_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_LBL.setObjectName("res_Q_LBL")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(0, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_Iangle_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P1_unit.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_unit.setFont(font)
self.res_Iangle_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_unit.setObjectName("res_Iangle_P1_unit")
self.res_S_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_S_LBL.setGeometry(QtCore.QRect(0, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_S_LBL.setFont(font)
self.res_S_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_LBL.setObjectName("res_S_LBL")
self.res_Q_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P1_DSB.setEnabled(False)
self.res_Q_P1_DSB.setGeometry(QtCore.QRect(90, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_DSB.setFont(font)
self.res_Q_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P1_DSB.setDecimals(3)
self.res_Q_P1_DSB.setMinimum(-999999.999)
self.res_Q_P1_DSB.setMaximum(999999.999)
self.res_Q_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P1_DSB.setProperty("value", 0.0)
self.res_Q_P1_DSB.setObjectName("res_Q_P1_DSB")
self.res_cosPhi_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P1_unit.setGeometry(QtCore.QRect(170, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_unit.setFont(font)
self.res_cosPhi_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_unit.setObjectName("res_cosPhi_P1_unit")
self.res_cosPhi_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_LBL.setGeometry(QtCore.QRect(0, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_cosPhi_LBL.setFont(font)
self.res_cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_LBL.setObjectName("res_cosPhi_LBL")
self.res_I_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P1_DSB.setEnabled(False)
self.res_I_P1_DSB.setGeometry(QtCore.QRect(90, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_DSB.setFont(font)
self.res_I_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P1_DSB.setDecimals(3)
self.res_I_P1_DSB.setMaximum(999999.999)
self.res_I_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P1_DSB.setProperty("value", 0.0)
self.res_I_P1_DSB.setObjectName("res_I_P1_DSB")
self.res_Q_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P1_unit.setGeometry(QtCore.QRect(170, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_unit.setFont(font)
self.res_Q_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P1_unit.setObjectName("res_Q_P1_unit")
self.res_P_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P1_DSB.setEnabled(False)
self.res_P_P1_DSB.setGeometry(QtCore.QRect(90, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_DSB.setFont(font)
self.res_P_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P1_DSB.setDecimals(3)
self.res_P_P1_DSB.setMinimum(-999999.999)
self.res_P_P1_DSB.setMaximum(999999.999)
self.res_P_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P1_DSB.setProperty("value", 0.0)
self.res_P_P1_DSB.setObjectName("res_P_P1_DSB")
self.res_P_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P1_unit.setGeometry(QtCore.QRect(170, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_unit.setFont(font)
self.res_P_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
```

```
self.res_P_P1_unit.setObjectName("res_P_P1_unit")
self.res_I_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_I_LBL.setGeometry(QtCore.QRect(0, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_I_LBL.setFont(font)
self.res_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_LBL.setObjectName("res_I_LBL")
self.res_Iangle_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P1_DSB.setEnabled(False)
self.res_Iangle_P1_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_DSB.setFont(font)
self.res_Iangle_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P1_DSB.setDecimals(3)
self.res_Iangle_P1_DSB.setMinimum(-999999.999)
self.res_Iangle_P1_DSB.setMaximum(999999.999)
self.res_Iangle_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P1_DSB.setProperty("value", 0.0)
self.res_Iangle_P1_DSB.setObjectName("res_Iangle_P1_DSB")
self.res_cosPhi_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P1_DSB.setEnabled(False)
self.res_cosPhi_P1_DSB.setGeometry(QtCore.QRect(90, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_DSB.setFont(font)
self.res_cosPhi_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P1_DSB.setDecimals(3)
self.res_cosPhi_P1_DSB.setMaximum(999999.999)
self.res_cosPhi_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P1_DSB.setProperty("value", 0.0)
self.res_cosPhi_P1_DSB.setObjectName("res_cosPhi_P1_DSB")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_U_LBL.setGeometry(QtCore.QRect(0, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.res_Iangle_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Iangle_LBL.setFont(font)
self.res_Iangle_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_LBL.setObjectName("res_Iangle_LBL")
self.Por1_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Por1_LBL.setGeometry(QtCore.QRect(90, 10, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Por1_LBL.setFont(font)
self.Por1_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Por1_LBL.setObjectName("Por1_LBL")
self.res_I_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P1_unit.setGeometry(QtCore.QRect(170, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_unit.setFont(font)
self.res_I_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P1_unit.setObjectName("res_I_P1_unit")
self.res_U_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P1_unit.setGeometry(QtCore.QRect(170, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_unit.setFont(font)
self.res_U_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P1_unit.setObjectName("res_U_P1_unit")
self.tabWidget.addTab(self.LoadFlow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.unmet_cost_LBL = QtWidgets.QLabel(self.EMS)
self.unmet_cost_LBL.setGeometry(QtCore.QRect(10, 10, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.unmet_cost_LBL.setFont(font)
self.unmet_cost_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.unmet_cost_LBL.setObjectName("unmet_cost_LBL")
self.unmet_cost_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.unmet_cost_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.unmet_cost_DSB.setFont(font)
self.unmet_cost_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.unmet_cost_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.unmet_cost_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.unmet_cost_DSB.setDecimals(3)
self.unmet_cost_DSB.setMaximum(999999.999)
self.unmet_cost_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.unmet_cost_DSB.setProperty("value", 0.0)
self.unmet_cost_DSB.setObjectName("unmet_cost_DSB")
```

```
self.unmet_cost_unit = QtWidgets.QLabel(self.EMS)
self.unmet_cost_unit.setGeometry(QtCore.QRect(240, 10, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.unmet_cost_unit.setFont(font)
self.unmet_cost_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.unmet_cost_unit.setObjectName("unmet_cost_unit")
self.tabWidget.addTab(self.EMS, "")
self.Reliability = QtWidgets.QWidget()
self.Reliability.setObjectName("Reliability")
self.rel_results_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 65, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_alfa_unit = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_unit.setGeometry(QtCore.QRect(240, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_R_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_R_LBL.setGeometry(QtCore.QRect(160, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_lambda_unit = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_beta_unit = QtWidgets.QLabel(self.Reliability)
self.rel_beta_unit.setGeometry(QtCore.QRect(240, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(0, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
self.rel_T0_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_T0_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(160, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(160, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
```

```
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.bottom_LN_3 = QtWidgets.QFrame(self.Reliability)
self.bottom_LN_3.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN_3.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_3.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_3.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_3.setObjectName("bottom_LN_3")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(160, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(240, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_beta_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_beta_LBL.setGeometry(QtCore.QRect(0, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(240, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_T0_unit = QtWidgets.QLabel(self.Reliability)
self.rel_T0_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(0, 170, 151, 21))
font = QtGui.QFont()
```

```
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(160, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(160, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_anni_DSB.setEnabled(False)
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(0, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_R_unit = QtWidgets.QLabel(self.Reliability)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_R_unit.setObjectName("rel_R_unit")
self.rel_R_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.rel_alfa_unit.raise_()
self.rel_R_LBL.raise_()
self.rel_lambda_unit.raise_()
self.rel_beta_unit.raise_()
self.rel_alfa_LBL.raise_()
self.rel_T0_LBL.raise_()
self.rel_beta_DSB.raise_()
self.rel_alfa_DSB.raise_()
self.rel_lambda_LBL.raise_()
self.bottom_LN_3.raise_()
self.rel_Pi_Q_DSB.raise_()
self.rel_T0_DSB.raise_()
self.rel_MTBF_ore_unit.raise_()
self.rel_MTBF_ore_DSB.raise_()
self.rel_Pi_Q_unit.raise_()
self.rel_MTBF_ore_LBL.raise_()
self.rel_beta_LBL.raise_()
self.rel_Pi_E_unit.raise_()
self.rel_T0_unit.raise_()
self.rel_Pi_Q_LBL.raise_()
self.rel_MTBF_anni_LBL.raise_()
self.rel_Pi_E_DSB.raise_()
self.rel_MTBF_anni_DSB.raise_()
self.rel_Pi_E_LBL.raise_()
self.rel_R_unit.raise_()
self.rel_R_LE.raise_()
self.rel_lambda_LE.raise_()
```

```
self.rel_results_LBL.raise_()
self.rel_MTBF_anni_unit.raise_()
self.tabWidget.addTab(self.Reliability, "")
self.Protections = QtWidgets.QWidget()
self.Protections.setObjectName("Protections")
self.prot_cal_Vmin_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_val_unit_LBL.setGeometry(QtCore.QRect(170, 210, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_val_unit_LBL.setFont(font)
self.prot_cal_Vmin_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_unit_LBL.setObjectName("prot_cal_Vmin_val_unit_LBL")
self.prot_cal_Vmin_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_LBL.setGeometry(QtCore.QRect(0, 210, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_LBL.setFont(font)
self.prot_cal_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_LBL.setObjectName("prot_cal_Vmin_LBL")
self.prot_calib_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_btm_LN.setGeometry(QtCore.QRect(10, 240, 305, 1))
self.prot_calib_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_calib_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_btm_LN.setObjectName("prot_calib_btm_LN")
self.prot_calib_LBL = QtWidgets.QLabel(self.Protections)
self.prot_calib_LBL.setGeometry(QtCore.QRect(110, 120, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_calib_LBL.setFont(font)
self.prot_calib_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_calib_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_calib_LBL.setObjectName("prot_calib_LBL")
self.prot_cal_Vmax_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_val_DSB.setGeometry(QtCore.QRect(90, 180, 71, 21))
self.prot_cal_Vmax_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_val_DSB.setDecimals(3)
self.prot_cal_Vmax_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_val_DSB.setObjectName("prot_cal_Vmax_val_DSB")
self.prot_cal_Vmin_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_pu_DSB.setGeometry(QtCore.QRect(220, 210, 71, 21))
self.prot_cal_Vmin_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_pu_DSB.setDecimals(1)
self.prot_cal_Vmin_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_pu_DSB.setObjectName("prot_cal_Vmin_pu_DSB")
self.prot_cal_Vmin_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_pu_unit_LBL.setGeometry(QtCore.QRect(300, 210, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_pu_unit_LBL.setFont(font)
self.prot_cal_Vmin_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_unit_LBL.setObjectName("prot_cal_Vmin_pu_unit_LBL")
self.prot_delay_Vmin_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_unit_LBL.setGeometry(QtCore.QRect(170, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmin_unit_LBL.setFont(font)
self.prot_delay_Vmin_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_unit_LBL.setObjectName("prot_delay_Vmin_unit_LBL")
self.prot_type_LBL = QtWidgets.QLabel(self.Protections)
self.prot_type_LBL.setGeometry(QtCore.QRect(0, 40, 81, 20))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_type_LBL.setFont(font)
self.prot_type_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_type_LBL.setObjectName("prot_type_LBL")
self.prot_cal_Vmax_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_pu_DSB.setGeometry(QtCore.QRect(220, 180, 71, 21))
self.prot_cal_Vmax_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_pu_DSB.setDecimals(1)
self.prot_cal_Vmax_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_pu_DSB.setObjectName("prot_cal_Vmax_pu_DSB")
self.prot_cal_I_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_sep_LBL.setGeometry(QtCore.QRect(190, 150, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_sep_LBL.setFont(font)
self.prot_cal_I_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_sep_LBL.setObjectName("prot_cal_I_sep_LBL")
self.prot_cost_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cost_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cost_LBL.setFont(font)
self.prot_cost_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cost_LBL.setObjectName("prot_cost_LBL")
self.prot_delay_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_btm_LN.setGeometry(QtCore.QRect(10, 380, 305, 1))
self.prot_delay_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_btm_LN.setObjectName("prot_delay_btm_LN")
self.prot_delay_Vmin_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_LBL.setGeometry(QtCore.QRect(0, 350, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
```

```
font.setBold(True)
font.setWeight(75)
self.prot_delay_Vmin_LBL.setFont(font)
self.prot_delay_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_LBL.setObjectName("prot_delay_Vmin_LBL")
self.prot_delay_Vmin_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmin_DSB.setGeometry(QtCore.QRect(90, 350, 71, 21))
self.prot_delay_Vmin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmin_DSB.setDecimals(3)
self.prot_delay_Vmin_DSB.setMaximum(99999999.0)
self.prot_delay_Vmin_DSB.setObjectName("prot_delay_Vmin_DSB")
self.prot_cost_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cost_unit_LBL.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cost_unit_LBL.setFont(font)
self.prot_cost_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cost_unit_LBL.setObjectName("prot_cost_unit_LBL")
self.prot_calib_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_top_LN.setGeometry(QtCore.QRect(10, 130, 305, 1))
self.prot_calib_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_calib_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_top_LN.setObjectName("prot_calib_top_LN")
self.prot_delay_I_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_I_DSB.setGeometry(QtCore.QRect(90, 290, 71, 21))
self.prot_delay_I_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_I_DSB.setDecimals(3)
self.prot_delay_I_DSB.setMaximum(99999999.0)
self.prot_delay_I_DSB.setObjectName("prot_delay_I_DSB")
self.prot_cal_Vmax_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_sep_LBL.setGeometry(QtCore.QRect(190, 180, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_sep_LBL.setFont(font)
self.prot_cal_Vmax_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_sep_LBL.setObjectName("prot_cal_Vmax_sep_LBL")
self.prot_delay_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_top_LN.setGeometry(QtCore.QRect(10, 270, 305, 1))
self.prot_delay_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_top_LN.setObjectName("prot_delay_top_LN")
self.prot_cal_Vmin_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_sep_LBL.setGeometry(QtCore.QRect(190, 210, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_sep_LBL.setFont(font)
self.prot_cal_Vmin_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_sep_LBL.setObjectName("prot_cal_Vmin_sep_LBL")
self.prot_cdelay_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cdelay_LBL.setGeometry(QtCore.QRect(110, 260, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cdelay_LBL.setFont(font)
self.prot_cdelay_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_cdelay_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_cdelay_LBL.setObjectName("prot_cdelay_LBL")
self.prot_delay_I_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_unit_LBL.setGeometry(QtCore.QRect(170, 290, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_I_unit_LBL.setFont(font)
self.prot_delay_I_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_I_unit_LBL.setObjectName("prot_delay_I_unit_LBL")
self.prot_cal_Vmax_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_val_unit_LBL.setGeometry(QtCore.QRect(170, 180, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmax_val_unit_LBL.setFont(font)
self.prot_cal_Vmax_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_val_unit_LBL.setObjectName("prot_cal_Vmax_val_unit_LBL")
self.prot_char_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_top_LN.setGeometry(QtCore.QRect(10, 20, 305, 1))
self.prot_char_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_top_LN.setObjectName("prot_char_top_LN")
self.prot_delay_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_LBL.setGeometry(QtCore.QRect(0, 320, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_Vmax_LBL.setFont(font)
self.prot_delay_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_LBL.setObjectName("prot_delay_Vmax_LBL")
self.prot_delay_Vmax_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmax_DSB.setGeometry(QtCore.QRect(90, 320, 71, 21))
self.prot_delay_Vmax_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmax_DSB.setDecimals(3)
self.prot_delay_Vmax_DSB.setMaximum(99999999.0)
self.prot_delay_Vmax_DSB.setObjectName("prot_delay_Vmax_DSB")
self.prot_cal_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_LBL.setGeometry(QtCore.QRect(0, 180, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_LBL.setFont(font)
```

```
self.prot_cal_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_LBL.setObjectName("prot_cal_Vmax_LBL")
self.prot_charact_LBL = QtWidgets.QLabel(self.Protections)
self.prot_charact_LBL.setGeometry(QtCore.QRect(60, 10, 211, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_charact_LBL.setFont(font)
self.prot_charact_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_charact_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_charact_LBL.setObjectName("prot_charact_LBL")
self.prot_cal_I_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_pu_unit_LBL.setGeometry(QtCore.QRect(300, 150, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_pu_unit_LBL.setFont(font)
self.prot_cal_I_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_unit_LBL.setObjectName("prot_cal_I_pu_unit_LBL")
self.prot_cal_Vmin_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_val_DSB.setGeometry(QtCore.QRect(90, 210, 71, 21))
self.prot_cal_Vmin_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_val_DSB.setDecimals(3)
self.prot_cal_Vmin_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_val_DSB.setObjectName("prot_cal_Vmin_val_DSB")
self.prot_cal_Vmax_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_pu_unit_LBL.setGeometry(QtCore.QRect(300, 180, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmax_pu_unit_LBL.setFont(font)
self.prot_cal_Vmax_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_unit_LBL.setObjectName("prot_cal_Vmax_pu_unit_LBL")
self.prot_cal_I_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_val_unit_LBL.setGeometry(QtCore.QRect(170, 150, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_val_unit_LBL.setFont(font)
self.prot_cal_I_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_unit_LBL.setObjectName("prot_cal_I_val_unit_LBL")
self.prot_char_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_btm_LN.setGeometry(QtCore.QRect(10, 100, 305, 1))
self.prot_char_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_btm_LN.setObjectName("prot_char_btm_LN")
self.prot_type_LE = QtWidgets.QLineEdit(self.Protections)
self.prot_type_LE.setGeometry(QtCore.QRect(90, 40, 201, 20))
self.prot_type_LE.setText("")
self.prot_type_LE.setObjectName("prot_type_LE")
self.prot_cal_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_LBL.setGeometry(QtCore.QRect(0, 150, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_LBL.setFont(font)
self.prot_cal_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_LBL.setObjectName("prot_cal_I_LBL")
self.prot_delay_Vmax_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_unit_LBL.setGeometry(QtCore.QRect(170, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmax_unit_LBL.setFont(font)
self.prot_delay_Vmax_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_unit_LBL.setObjectName("prot_delay_Vmax_unit_LBL")
self.prot_cal_I_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_val_DSB.setGeometry(QtCore.QRect(90, 150, 71, 21))
self.prot_cal_I_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_val_DSB.setDecimals(1)
self.prot_cal_I_val_DSB.setMaximum(99999999.0)
self.prot_cal_I_val_DSB.setObjectName("prot_cal_I_val_DSB")
self.prot_cost_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cost_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
self.prot_cost_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cost_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cost_DSB.setDecimals(2)
self.prot_cost_DSB.setMaximum(99999999.0)
self.prot_cost_DSB.setObjectName("prot_cost_DSB")
self.prot_delay_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_LBL.setGeometry(QtCore.QRect(0, 290, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_I_LBL.setFont(font)
self.prot_delay_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_LBL.setObjectName("prot_delay_I_LBL")
self.prot_cal_I_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_pu_DSB.setGeometry(QtCore.QRect(220, 150, 71, 21))
self.prot_cal_I_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_pu_DSB.setDecimals(1)
self.prot_cal_I_pu_DSB.setMaximum(99999999.0)
self.prot_cal_I_pu_DSB.setObjectName("prot_cal_I_pu_DSB")
self.prot_calib_top_LN.raise_()
self.prot_cal_Vmin_val_unit_LBL.raise_()
self.prot_cal_Vmin_LBL.raise_()
self.prot_calib_btm_LN.raise_()
self.prot_calib_LBL.raise_()
self.prot_cal_Vmax_val_DSB.raise_()
self.prot_cal_Vmin_pu_DSB.raise_()
self.prot_cal_Vmin_pu_unit_LBL.raise_()
self.prot_delay_Vmin_unit_LBL.raise_()
self.prot_type_LBL.raise_()
self.prot_cal_Vmax_pu_DSB.raise_()
```

```
        self.prot_cal_I_sep_LBL.raise_()
        self.prot_cost_LBL.raise_()
        self.prot_delay_btm_LN.raise_()
        self.prot_delay_Vmin_LBL.raise_()
        self.prot_delay_Vmin_DSB.raise_()
        self.prot_cost_unit_LBL.raise_()
        self.prot_delay_I_DSB.raise_()
        self.prot_cal_Vmax_sep_LBL.raise_()
        self.prot_delay_top_LN.raise_()
        self.prot_cal_Vmin_sep_LBL.raise_()
        self.prot_cdelay_LBL.raise_()
        self.prot_delay_I_unit_LBL.raise_()
        self.prot_cal_Vmax_val_unit_LBL.raise_()
        self.prot_char_top_LN.raise_()
        self.prot_delay_Vmax_LBL.raise_()
        self.prot_delay_Vmax_DSB.raise_()
        self.prot_cal_Vmax_LBL.raise_()
        self.prot_charact_LBL.raise_()
        self.prot_cal_I_pu_unit_LBL.raise_()
        self.prot_cal_Vmin_val_DSB.raise_()
        self.prot_cal_Vmax_pu_unit_LBL.raise_()
        self.prot_cal_I_val_unit_LBL.raise_()
        self.prot_char_btm_LN.raise_()
        self.prot_type_LE.raise_()
        self.prot_cal_I_LBL.raise_()
        self.prot_delay_Vmax_unit_LBL.raise_()
        self.prot_cal_I_val_DSB.raise_()
        self.prot_cost_DSB.raise_()
        self.prot_delay_I_LBL.raise_()
        self.prot_cal_I_pu_DSB.raise_()
        self.tabWidget.addTab(self.Protections, "")
        self.cancel_BTN = QtWidgets.QPushButton(self.widget)
        self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.cancel_BTN.setFont(font)
        self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
        self.cancel_BTN.setObjectName("cancel_BTN")
        self.bb_out_LBL.raise_()
        self.bb_in_LBL.raise_()
        self.cub_out_LBL.raise_()
        self.top_LN.raise_()
        self.elem_name_LN.raise_()
        self.elem_name_LBL.raise_()
        self.type_LBL.raise_()
        self.type_cap_LBL.raise_()
        self.cub_in_LBL.raise_()
        self.symbol_LBL.raise_()
        self.ot_Frame_LN.raise_()
        self.vdx_frame_LN.raise_()
        self.vsx_frame_LN.raise_()
        self.ob_frame_LN.raise_()
        self.bb_in_LN.raise_()
        self.bb_out_LN.raise_()
        self.store_BTN.raise_()
        self.tabWidget.raise_()
        self.cancel_BTN.raise_()

        self.retranslateUi(Form)
        self.tabWidget.setCurrentIndex(3)
        QtCore.QMetaObject.connectSlotsByName(Form)
        Form.setTabOrder(self.tabWidget, self.control_CB)
        Form.setTabOrder(self.control_CB, self.p_DSB)
        Form.setTabOrder(self.p_DSB, self.i_DSB)
        Form.setTabOrder(self.i_DSB, self.r_DSB)
        Form.setTabOrder(self.r_DSB, self.sf_DSB)
        Form.setTabOrder(self.sf_DSB, self.sf_const_RB)
        Form.setTabOrder(self.sf_const_RB, self.sf_profile_RB)
        Form.setTabOrder(self.sf_profile_RB, self.unmet_cost_DSB)
        Form.setTabOrder(self.unmet_cost_DSB, self.rel_TO_DSB)
        Form.setTabOrder(self.rel_TO_DSB, self.rel_alfa_DSB)
        Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
        Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
        Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
        Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
        Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
        Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_I_P1_DSB)
        Form.setTabOrder(self.res_I_P1_DSB, self.res_Iangle_P1_DSB)
        Form.setTabOrder(self.res_Iangle_P1_DSB, self.res_P_P1_DSB)
        Form.setTabOrder(self.res_P_P1_DSB, self.res_Q_P1_DSB)
        Form.setTabOrder(self.res_Q_P1_DSB, self.res_S_P1_DSB)
        Form.setTabOrder(self.res_S_P1_DSB, self.res_cosPhi_P1_DSB)
        Form.setTabOrder(self.res_cosPhi_P1_DSB, self.res_U_P1_DSB)
        Form.setTabOrder(self.res_U_P1_DSB, self.store_BTN)
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria:"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.store_BTN.setText(_translate("Form", "Salva"))
        self.sf_profile_RB.setText(_translate("Form", "Profilo"))
        self.i_LBL.setText(_translate("Form", "I"))
        self.p_unit.setText(_translate("Form", "kW"))
        self.sf_const_RB.setText(_translate("Form", "Costante"))
        self.r_LBL.setText(_translate("Form", "R"))
        self.control_CB.setItemText(0, _translate("Form", "Power"))
        self.control_CB.setItemText(1, _translate("Form", "Current"))
        self.control_CB.setItemText(2, _translate("Form", "Impedance"))
        self.lftype_LBL.setText(_translate("Form", "Controllo"))
        self.p_LBL.setText(_translate("Form", "P"))
        self.sfi_LBL.setText(_translate("Form", "Scala"))
        self.s_unit.setText(_translate("Form", "Ohm"))
        self.i_unit.setText(_translate("Form", "A"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Parameters), _translate("Form", "Parametri"))
        self.res_S_P1_unit.setText(_translate("Form", "kVA"))
        self.res_P_LBL.setText(_translate("Form", "P"))
        self.res_Q_LBL.setText(_translate("Form", "Q"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMITI VIOLATI"))
        self.res_Iangle_P1_unit.setText(_translate("Form", "°"))
```

```
        self.res_S_LBL.setText(_translate("Form", "S"))
        self.res_cosPhi_P1_unit.setText(_translate("Form", "-"))
        self.res_cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.res_Q_P1_unit.setText(_translate("Form", "kVA"))
        self.res_P_P1_unit.setText(_translate("Form", "kW"))
        self.res_I_LBL.setText(_translate("Form", " I"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.res_Iangle_LBL.setText(_translate("Form", "Angolo I"))
        self.Por1_LBL.setText(_translate("Form", "Nodo 1"))
        self.res_I_P1_unit.setText(_translate("Form", "A"))
        self.res_U_P1_unit.setText(_translate("Form", "kV"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFlow), _translate("Form", "LoadFlow"))
        self.unmet_cost_LBL.setText(_translate("Form", "Costo Energia Non Fornita"))
        self.unmet_cost_unit.setText(_translate("Form", "€/kWh"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_T0_unit.setText(_translate("Form", "°C"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Reliability), _translate("Form", "Affidabilità"))
        self.prot_cal_Vmin_val_unit_LBL.setText(_translate("Form", "kV"))
        self.prot_cal_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_calib_LBL.setText(_translate("Form", "Soglie di taratura"))
        self.prot_cal_Vmin_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_delay_Vmin_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_type_LBL.setText(_translate("Form", "Tipologia"))
        self.prot_cal_I_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cost_LBL.setText(_translate("Form", "Costo"))
        self.prot_delay_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_cost_unit_LBL.setText(_translate("Form", "Euro"))
        self.prot_cal_Vmax_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cal_Vmin_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cdelay_LBL.setText(_translate("Form", "Tempi di ritardo"))
        self.prot_delay_I_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_cal_Vmax_val_unit_LBL.setText(_translate("Form", "kV"))
        self.prot_delay_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_cal_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_charact_LBL.setText(_translate("Form", "Caratterisitiche dell\'interruttore"))
        self.prot_cal_I_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_Vmax_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_I_val_unit_LBL.setText(_translate("Form", "A"))
        self.prot_cal_I_LBL.setText(_translate("Form", "Corrente"))
        self.prot_delay_Vmax_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_delay_I_LBL.setText(_translate("Form", "Corrente"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Protections), _translate("Form", "Protezioni"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.3.9 DCnode

4.4.3.9.1 dcnode.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .dcnodeUI import Ui_Form
from __shared__ import variables as v
import copy


class DCnode(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(DCnode, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.bb_in_LBL.setVisible(False)
        self.ui.bb_in_LN.setVisible(False)
        self.ui.bb_out_LN.setVisible(False)
        self.ui.bb_out_LBL.setVisible(False)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")
        self.ui.tabWidget.setCurrentIndex(0)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])

        self.fill()

    #
    def store(self):  # Inutile?? ------------------------------------------------------------------------------
        self.par['Ur'] = self.ui.u_DSB.value()
        v.elements[self.element]['parameters'] = copy.deepcopy(self.par)

        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)

    #
    def fill(self):
        self.ui.u_DSB.setValue(self.par['Ur'])

        if self.res != {}:
            self.fill_results()
        self.ui.tabWidget.setTabVisible(3, self.res != {})
        self.fill_reliability()

    #
    def fill_reliability(self):
        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
        for par in self.rel['results']:
            try:
                self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
            except:
                if self.rel['results'][par] == 0:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('0')
                elif self.rel['results'][par] < 0.01:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
                else:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])

    #
    def fill_results(self):
        results = ['Pgen', 'Pload', 'Qgen', 'Qload', 'U', 'Un', 'Up']
        for result in results:
            self.ui.__getattribute__('res_' + result + '_DSB').setValue(self.res[result])
        self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])
```

4.4.3.9.2 dcnodeUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'dcnodeUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(497, 507)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("res/Load.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.bb_in_LBL.setFont(font)
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_out_LN.setObjectName("bb_out_LN")
        self.bb_out_LBL = QtWidgets.QLabel(self.widget)
        self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
self.bb_out_LBL.setFont(font)
self.bb_out_LBL.setStyleSheet("")
self.bb_out_LBL.setLineWidth(4)
self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.store_BTN = QtWidgets.QPushButton(self.widget)
self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.store_BTN.setFont(font)
self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.store_BTN.setObjectName("store_BTN")
self.cancel_BTN = QtWidgets.QPushButton(self.widget)
self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.cancel_BTN.setFont(font)
self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.cancel_BTN.setObjectName("cancel_BTN")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.u_LBL = QtWidgets.QLabel(self.Parameters)
self.u_LBL.setGeometry(QtCore.QRect(60, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.u_LBL.setFont(font)
self.u_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.u_LBL.setObjectName("u_LBL")
self.u_unit = QtWidgets.QLabel(self.Parameters)
self.u_unit.setGeometry(QtCore.QRect(240, 10, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.u_unit.setFont(font)
self.u_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.u_unit.setObjectName("u_unit")
self.u_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.u_DSB.setGeometry(QtCore.QRect(160, 10, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.u_DSB.setFont(font)
self.u_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.u_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.u_DSB.setDecimals(3)
self.u_DSB.setMaximum(999999.999)
self.u_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.u_DSB.setProperty("value", 0.0)
self.u_DSB.setObjectName("u_DSB")
self.tabWidget.addTab(self.Parameters, "Parametri")
self.LoadFlow = QtWidgets.QWidget()
self.LoadFlow.setObjectName("LoadFlow")
self.res_Up_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Up_unit.setGeometry(QtCore.QRect(170, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Up_unit.setFont(font)
self.res_Up_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Up_unit.setObjectName("res_Up_unit")
self.res_Qgen_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Qgen_unit.setGeometry(QtCore.QRect(170, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qgen_unit.setFont(font)
self.res_Qgen_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Qgen_unit.setObjectName("res_Qgen_unit")
self.res_Pload_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Pload_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Pload_LBL.setFont(font)
self.res_Pload_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Pload_LBL.setObjectName("res_Pload_LBL")
self.res_Up_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Up_LBL.setGeometry(QtCore.QRect(0, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Up_LBL.setFont(font)
self.res_Up_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Up_LBL.setObjectName("res_Up_LBL")
self.res_Pgen_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Pgen_LBL.setGeometry(QtCore.QRect(0, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Pgen_LBL.setFont(font)
self.res_Pgen_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Pgen_LBL.setObjectName("res_Pgen_LBL")
self.res_Un_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Un_unit.setGeometry(QtCore.QRect(170, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
```

```
font.setWeight(50)
self.res_Un_unit.setFont(font)
self.res_Un_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Un_unit.setObjectName("res_Un_unit")
self.res_Qgen_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Qgen_LBL.setGeometry(QtCore.QRect(0, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Qgen_LBL.setFont(font)
self.res_Qgen_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qgen_LBL.setObjectName("res_Qgen_LBL")
self.res_Qload_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Qload_LBL.setGeometry(QtCore.QRect(0, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Qload_LBL.setFont(font)
self.res_Qload_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qload_LBL.setObjectName("res_Qload_LBL")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_U_LBL.setGeometry(QtCore.QRect(0, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.res_Qload_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Qload_unit.setGeometry(QtCore.QRect(170, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qload_unit.setFont(font)
self.res_Qload_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Qload_unit.setObjectName("res_Qload_unit")
self.sf_unit = QtWidgets.QLabel(self.LoadFlow)
self.sf_unit.setGeometry(QtCore.QRect(168, 215, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sf_unit.setFont(font)
self.sf_unit.setText("")
self.sf_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.sf_unit.setObjectName("sf_unit")
self.res_Un_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Un_LBL.setGeometry(QtCore.QRect(0, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Un_LBL.setFont(font)
self.res_Un_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Un_LBL.setObjectName("res_Un_LBL")
self.res_Pload_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Pload_DSB.setEnabled(False)
self.res_Pload_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Pload_DSB.setFont(font)
self.res_Pload_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Pload_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Pload_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Pload_DSB.setDecimals(3)
self.res_Pload_DSB.setMinimum(-999999.999)
self.res_Pload_DSB.setMaximum(999999.999)
self.res_Pload_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Pload_DSB.setProperty("value", 0.0)
self.res_Pload_DSB.setObjectName("res_Pload_DSB")
self.res_U_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_unit.setGeometry(QtCore.QRect(170, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_unit.setFont(font)
self.res_U_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_unit.setObjectName("res_U_unit")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(0, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_Pgen_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Pgen_unit.setGeometry(QtCore.QRect(170, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Pgen_unit.setFont(font)
self.res_Pgen_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Pgen_unit.setObjectName("res_Pgen_unit")
self.res_Up_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Up_DSB.setEnabled(False)
self.res_Up_DSB.setGeometry(QtCore.QRect(90, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Up_DSB.setFont(font)
self.res_Up_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Up_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Up_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Up_DSB.setDecimals(3)
```

```
self.res_Up_DSB.setMaximum(999999.999)
self.res_Up_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Up_DSB.setProperty("value", 0.0)
self.res_Up_DSB.setObjectName("res_Up_DSB")
self.res_Qgen_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Qgen_DSB.setEnabled(False)
self.res_Qgen_DSB.setGeometry(QtCore.QRect(90, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qgen_DSB.setFont(font)
self.res_Qgen_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Qgen_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qgen_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Qgen_DSB.setDecimals(3)
self.res_Qgen_DSB.setMinimum(-999999.999)
self.res_Qgen_DSB.setMaximum(999999.999)
self.res_Qgen_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Qgen_DSB.setProperty("value", 0.0)
self.res_Qgen_DSB.setObjectName("res_Qgen_DSB")
self.res_Un_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Un_DSB.setEnabled(False)
self.res_Un_DSB.setGeometry(QtCore.QRect(90, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Un_DSB.setFont(font)
self.res_Un_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Un_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Un_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Un_DSB.setDecimals(3)
self.res_Un_DSB.setMaximum(999999.999)
self.res_Un_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Un_DSB.setProperty("value", 0.0)
self.res_Un_DSB.setObjectName("res_Un_DSB")
self.res_U_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_DSB.setEnabled(False)
self.res_U_DSB.setGeometry(QtCore.QRect(90, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_DSB.setFont(font)
self.res_U_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_DSB.setDecimals(3)
self.res_U_DSB.setMaximum(999999.999)
self.res_U_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_DSB.setProperty("value", 0.0)
self.res_U_DSB.setObjectName("res_U_DSB")
self.res_Pload_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Pload_unit.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Pload_unit.setFont(font)
self.res_Pload_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Pload_unit.setObjectName("res_Pload_unit")
self.res_Pgen_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Pgen_DSB.setEnabled(False)
self.res_Pgen_DSB.setGeometry(QtCore.QRect(90, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Pgen_DSB.setFont(font)
self.res_Pgen_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Pgen_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Pgen_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Pgen_DSB.setDecimals(3)
self.res_Pgen_DSB.setMinimum(-999999.999)
self.res_Pgen_DSB.setMaximum(999999.999)
self.res_Pgen_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Pgen_DSB.setProperty("value", 0.0)
self.res_Pgen_DSB.setObjectName("res_Pgen_DSB")
self.res_Qload_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Qload_DSB.setEnabled(False)
self.res_Qload_DSB.setGeometry(QtCore.QRect(90, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qload_DSB.setFont(font)
self.res_Qload_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Qload_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qload_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Qload_DSB.setDecimals(3)
self.res_Qload_DSB.setMinimum(-999999.999)
self.res_Qload_DSB.setMaximum(999999.999)
self.res_Qload_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Qload_DSB.setProperty("value", 0.0)
self.res_Qload_DSB.setObjectName("res_Qload_DSB")
self.tabWidget.addTab(self.LoadFlow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.tabWidget.addTab(self.EMS, "")
self.Reliabiity = QtWidgets.QWidget()
self.Reliabiity.setObjectName("Reliabiity")
self.rel_results_LBL = QtWidgets.QLabel(self.Reliabiity)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 65, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_alfa_unit = QtWidgets.QLabel(self.Reliabiity)
self.rel_alfa_unit.setGeometry(QtCore.QRect(240, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
```

```
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_R_LBL = QtWidgets.QLabel(self.Reliabiity)
self.rel_R_LBL.setGeometry(QtCore.QRect(160, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_beta_unit = QtWidgets.QLabel(self.Reliabiity)
self.rel_beta_unit.setGeometry(QtCore.QRect(240, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Reliabiity)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(0, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
self.rel_T0_LBL = QtWidgets.QLabel(self.Reliabiity)
self.rel_T0_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Reliabiity)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(160, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Reliabiity)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(160, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Reliabiity)
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.bottom_LN_3 = QtWidgets.QFrame(self.Reliabiity)
self.bottom_LN_3.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN_3.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_3.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_3.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_3.setObjectName("bottom_LN_3")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Reliabiity)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(160, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Reliabiity)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
```

```
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Reliabiity)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Reliabiity)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Reliabiity)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(240, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Reliabiity)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_beta_LBL = QtWidgets.QLabel(self.Reliabiity)
self.rel_beta_LBL.setGeometry(QtCore.QRect(0, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Reliabiity)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(240, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_T0_unit = QtWidgets.QLabel(self.Reliabiity)
self.rel_T0_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Reliabiity)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(0, 170, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Reliabiity)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(160, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Reliabiity)
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(160, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Reliabiity)
self.rel_MTBF_anni_DSB.setEnabled(False)
```

```
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Reliabiity)
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(0, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_R_unit = QtWidgets.QLabel(self.Reliabiity)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_R_unit.setObjectName("rel_R_unit")
self.rel_R_LE = QtWidgets.QLineEdit(self.Reliabiity)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Reliabiity)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_lambda_unit = QtWidgets.QLabel(self.Reliabiity)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Reliabiity)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.rel_alfa_unit.raise_()
self.rel_R_LBL.raise_()
self.rel_beta_unit.raise_()
self.rel_alfa_LBL.raise_()
self.rel_T0_LBL.raise_()
self.rel_beta_DSB.raise_()
self.rel_alfa_DSB.raise_()
self.rel_lambda_LBL.raise_()
self.bottom_LN_3.raise_()
self.rel_Pi_Q_DSB.raise_()
self.rel_T0_DSB.raise_()
self.rel_MTBF_ore_unit.raise_()
self.rel_MTBF_ore_DSB.raise_()
self.rel_Pi_Q_unit.raise_()
self.rel_MTBF_ore_LBL.raise_()
self.rel_beta_LBL.raise_()
self.rel_Pi_E_unit.raise_()
self.rel_T0_unit.raise_()
self.rel_Pi_Q_LBL.raise_()
self.rel_MTBF_anni_LBL.raise_()
self.rel_Pi_E_DSB.raise_()
self.rel_MTBF_anni_DSB.raise_()
self.rel_Pi_E_LBL.raise_()
self.rel_R_unit.raise_()
self.rel_R_LE.raise_()
self.rel_lambda_LE.raise_()
self.rel_results_LBL.raise_()
self.rel_lambda_unit.raise_()
self.rel_MTBF_anni_unit.raise_()
self.tabWidget.addTab(self.Reliabiity, "")
self.bb_out_LBL.raise_()
self.bb_in_LBL.raise_()
self.cub_out_LBL.raise_()
self.top_LN.raise_()
self.elem_name_LN.raise_()
self.elem_name_LBL.raise_()
self.type_LBL.raise_()
self.type_cap_LBL.raise_()
self.cub_in_LBL.raise_()
self.symbol_LBL.raise_()
self.ot_Frame_LN.raise_()
self.vdx_frame_LN.raise_()
self.vsx_frame_LN.raise_()
self.ob_frame_LN.raise_()
self.bb_in_LN.raise_()
self.bb_out_LN.raise_()
self.store_BTN.raise_()
self.cancel_BTN.raise_()
self.tabWidget.raise_()

self.retranslateUi(Form)
self.tabWidget.setCurrentIndex(3)
```

```
        QtCore.QMetaObject.connectSlotsByName(Form)
        Form.setTabOrder(self.tabWidget, self.u_DSB)
        Form.setTabOrder(self.u_DSB, self.rel_T0_DSB)
        Form.setTabOrder(self.rel_T0_DSB, self.rel_alfa_DSB)
        Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
        Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
        Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
        Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
        Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
        Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_Pgen_DSB)
        Form.setTabOrder(self.res_Pgen_DSB, self.res_Pload_DSB)
        Form.setTabOrder(self.res_Pload_DSB, self.res_Qgen_DSB)
        Form.setTabOrder(self.res_Qgen_DSB, self.res_Qload_DSB)
        Form.setTabOrder(self.res_Qload_DSB, self.res_U_DSB)
        Form.setTabOrder(self.res_U_DSB, self.res_Un_DSB)
        Form.setTabOrder(self.res_Un_DSB, self.res_Up_DSB)
        Form.setTabOrder(self.res_Up_DSB, self.store_BTN)
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria:"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.store_BTN.setText(_translate("Form", "Salva"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))
        self.u_LBL.setText(_translate("Form", "Un"))
        self.u_unit.setText(_translate("Form", "kV"))
        self.res_Up_unit.setText(_translate("Form", "-"))
        self.res_Qgen_unit.setText(_translate("Form", "kVA"))
        self.res_Pload_LBL.setText(_translate("Form", "P load"))
        self.res_Up_LBL.setText(_translate("Form", "Up"))
        self.res_Pgen_LBL.setText(_translate("Form", "P gen"))
        self.res_Un_unit.setText(_translate("Form", "kV"))
        self.res_Qgen_LBL.setText(_translate("Form", "Q gen"))
        self.res_Qload_LBL.setText(_translate("Form", "Q load"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.res_Qload_unit.setText(_translate("Form", "kVA"))
        self.res_Un_LBL.setText(_translate("Form", "Un"))
        self.res_U_unit.setText(_translate("Form", "kV"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMITI VIOLATI"))
        self.res_Pgen_unit.setText(_translate("Form", "kW"))
        self.res_Pload_unit.setText(_translate("Form", "kW"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFlow), _translate("Form", "LoadFlow"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_T0_unit.setText(_translate("Form", "°C"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Reliabiity), _translate("Form", "Affidabilità"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.3.10 DCwind

4.4.3.10.1 dcwind.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .dcwindUI import Ui_Form
from __shared__ import variables as v
import copy


class DCwind(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(DCwind, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.bb_out_LN.setVisible(False)
        self.ui.bb_out_LBL.setVisible(False)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")

        for box in ['cap_pwr']:
            self.ui.__getattribute__(box + '_DSB').setStyleSheet("color: rgb(127, 127, 127);")
            self.ui.__getattribute__(box + '_DSB').setEnabled(False)

        self.ui.tabWidget.setCurrentIndex(0)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])
        self.prot = copy.deepcopy(v.elements[element]['protections'])

        self.bb = v.elements[element]['conn']['h']
        self.cubicle = v.elements[self.element]['conn'][self.bb]
        self.u = v.elements[self.bb]['parameters']['Ur']
        self.ui.bb_in_LBL.setText(self.bb)

        self.ui.symbol_LBL.setPixmap(QtGui.QPixmap("_images/Elements/DCwind/element.png"))
        self.switch_draw()
        self.fill()
        self.tab_activation()

        self.ui.pr_DSB.valueChanged.connect(self.calculate)
        self.ui.cub_in_LBL.mouseDoubleClickEvent = self.cub1_switch

    #
    def tab_activation(self):
        self.ui.tabWidget.setTabVisible(2, v.functionality == 'EMS')
        self.ui.tabWidget.setTabVisible(3, v.functionality == 'Reliability')
        self.ui.tabWidget.setTabVisible(4, v.protections and self.prot != {})

    #
    def store(self):
        self.par['Pr'] = self.ui.pr_DSB.value()
        self.par['P'] = self.ui.p_DSB.value()
        v.elements[self.element]['parameters'] = copy.deepcopy(self.par)

        v.elements[self.element]['conn'][self.bb] = self.cubicle

        self.ems['cap_pwr'] = self.ui.cap_pwr_DSB.value()
        v.elements[self.element]['ems'] = copy.deepcopy(self.ems)

        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)

        self.protections_par()

    #
    def protections_par(self):
        self.prot['Pn'] = self.par['Pr']
        self.prot['Vn'] = self.u
        self.prot['In'] = self.prot['Pn'] / self.prot['Vn']
        v.elements[self.element]['protections'] = copy.deepcopy(self.prot)

    #
    def fill(self):
        self.ui.pr_DSB.setValue(self.par['Pr'])
        self.ui.p_DSB.setValue(self.par['P'])
        self.control_mode(self.ui)

        self.calculate()

        if self.res != {}:
            self.fill_results()
        self.ui.tabWidget.setTabVisible(3, self.res != {})
        self.fill_reliability()
        if self.prot != {}:
            if self.prot['results'] != {} and v.protections:
                self.fill_protections()

    #
    def fill_reliability(self):
        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
        for par in self.rel['results']:
            try:
                self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
            except:
                if self.rel['results'][par] == 0:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('0')
                elif self.rel['results'][par] < 0.01:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
                else:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])
```

```
#
def fill_protections(self):
    self.ui.prot_type_LE.setText(self.prot['results']['type'])
    self.ui.prot_cost_DSB.setValue(self.prot['results']['cost'])
    self.ui.prot_cal_I_val_DSB.setValue(self.prot['results']['soglia_I'])
    self.ui.prot_cal_I_pu_DSB.setValue(self.prot['results']['soglia_I']/self.prot['In'] * 100)
    self.ui.prot_cal_Vmax_val_DSB.setValue(self.prot['results']['soglia_Vmax'])
    self.ui.prot_cal_Vmax_pu_DSB.setValue(self.prot['results']['soglia_Vmax'] / self.prot['Vn'] * 100)
    self.ui.prot_cal_Vmin_val_DSB.setValue(self.prot['results']['soglia_Vmin'])
    self.ui.prot_cal_Vmin_pu_DSB.setValue(self.prot['results']['soglia_Vmin'] / self.prot['Vn'] * 100)
    self.ui.prot_delay_I_DSB.setValue(self.prot['results']['delay_I'])
    self.ui.prot_delay_Vmax_DSB.setValue(self.prot['results']['delay_Vmax'])
    self.ui.prot_delay_Vmin_DSB.setValue(self.prot['results']['delay_Vmin'])

#
def fill_results(self):
    results = ['I', 'Iangle', 'P', 'Q', 'S', 'cosPhi', 'U']
    ports = ['P1']
    for port in ports:
        for result in results:
            self.ui.__getattribute__('res_' + result + '_' + port + '_DSB').setValue(self.res[result])
    self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])

#
def control_mode(self, ui):
    pass

#
def switch_draw(self):
    if self.cubicle:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
    else:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))

#
def cub1_switch(self, event):
    self.cubicle = not self.cubicle
    self.switch_draw()

#
def calc_profile(self, pu_profile):
    profile = []
    for data in pu_profile:
        profile.append(data * 1)
    return profile

#
def calculate(self):
    self.ui.cap_pwr_DSB.setValue(self.ui.pr_DSB.value())
```

4.4.3.10.2 dcwindUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'dcwindUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(493, 504)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("res/Load.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.bb_in_LBL.setFont(font)
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_out_LN.setObjectName("bb_out_LN")
        self.bb_out_LBL = QtWidgets.QLabel(self.widget)
        self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
self.bb_out_LBL.setFont(font)
self.bb_out_LBL.setStyleSheet("")
self.bb_out_LBL.setLineWidth(4)
self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.store_BTN = QtWidgets.QPushButton(self.widget)
self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.store_BTN.setFont(font)
self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.store_BTN.setObjectName("store_BTN")
self.cancel_BTN = QtWidgets.QPushButton(self.widget)
self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.cancel_BTN.setFont(font)
self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.cancel_BTN.setObjectName("cancel_BTN")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.sf_profile_RB = QtWidgets.QRadioButton(self.Parameters)
self.sf_profile_RB.setGeometry(QtCore.QRect(240, 90, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
self.sf_profile_RB.setFont(font)
self.sf_profile_RB.setObjectName("sf_profile_RB")
self.sf_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.sf_DSB.setGeometry(QtCore.QRect(160, 70, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sf_DSB.setFont(font)
self.sf_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sf_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.sf_DSB.setDecimals(5)
self.sf_DSB.setMaximum(1.0)
self.sf_DSB.setSingleStep(0.01)
self.sf_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.sf_DSB.setProperty("value", 0.0)
self.sf_DSB.setObjectName("sf_DSB")
self.p_LBL = QtWidgets.QLabel(self.Parameters)
self.p_LBL.setGeometry(QtCore.QRect(60, 40, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.p_LBL.setFont(font)
self.p_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.p_LBL.setObjectName("p_LBL")
self.p_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.p_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.p_DSB.setFont(font)
self.p_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.p_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.p_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.p_DSB.setDecimals(3)
self.p_DSB.setMaximum(999999.999)
self.p_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.p_DSB.setProperty("value", 0.0)
self.p_DSB.setObjectName("p_DSB")
self.pr_LBL = QtWidgets.QLabel(self.Parameters)
self.pr_LBL.setGeometry(QtCore.QRect(60, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.pr_LBL.setFont(font)
self.pr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pr_LBL.setObjectName("pr_LBL")
self.sf_const_RB = QtWidgets.QRadioButton(self.Parameters)
self.sf_const_RB.setGeometry(QtCore.QRect(240, 70, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
self.sf_const_RB.setFont(font)
self.sf_const_RB.setObjectName("sf_const_RB")
self.sfi_LBL = QtWidgets.QLabel(self.Parameters)
self.sfi_LBL.setGeometry(QtCore.QRect(60, 70, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.sfi_LBL.setFont(font)
self.sfi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sfi_LBL.setObjectName("sfi_LBL")
self.pr_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.pr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.pr_DSB.setFont(font)
self.pr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.pr_DSB.setReadOnly(False)
self.pr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.pr_DSB.setDecimals(3)
self.pr_DSB.setMaximum(999999.999)
self.pr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.pr_DSB.setProperty("value", 0.0)
self.pr_DSB.setObjectName("pr_DSB")
self.pri_unit = QtWidgets.QLabel(self.Parameters)
```

```
self.pri_unit.setGeometry(QtCore.QRect(240, 10, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.pri_unit.setFont(font)
self.pri_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.pri_unit.setObjectName("pri_unit")
self.p_unit = QtWidgets.QLabel(self.Parameters)
self.p_unit.setGeometry(QtCore.QRect(240, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.p_unit.setFont(font)
self.p_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.p_unit.setObjectName("p_unit")
self.tabWidget.addTab(self.Parameters, "")
self.LoadFlow = QtWidgets.QWidget()
self.LoadFlow.setObjectName("LoadFlow")
self.res_Iangle_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P1_DSB.setEnabled(False)
self.res_Iangle_P1_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_DSB.setFont(font)
self.res_Iangle_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P1_DSB.setDecimals(3)
self.res_Iangle_P1_DSB.setMinimum(-999999.999)
self.res_Iangle_P1_DSB.setMaximum(999999.999)
self.res_Iangle_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P1_DSB.setProperty("value", 0.0)
self.res_Iangle_P1_DSB.setObjectName("res_Iangle_P1_DSB")
self.res_S_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P1_unit.setGeometry(QtCore.QRect(170, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_unit.setFont(font)
self.res_S_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P1_unit.setObjectName("res_S_P1_unit")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_U_LBL.setGeometry(QtCore.QRect(0, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.res_I_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P1_DSB.setEnabled(False)
self.res_I_P1_DSB.setGeometry(QtCore.QRect(90, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_DSB.setFont(font)
self.res_I_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P1_DSB.setDecimals(3)
self.res_I_P1_DSB.setMaximum(999999.999)
self.res_I_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P1_DSB.setProperty("value", 0.0)
self.res_I_P1_DSB.setObjectName("res_I_P1_DSB")
self.res_S_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_S_LBL.setGeometry(QtCore.QRect(0, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_S_LBL.setFont(font)
self.res_S_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_LBL.setObjectName("res_S_LBL")
self.res_Q_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P1_unit.setGeometry(QtCore.QRect(170, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_unit.setFont(font)
self.res_Q_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P1_unit.setObjectName("res_Q_P1_unit")
self.res_U_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P1_unit.setGeometry(QtCore.QRect(170, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_unit.setFont(font)
self.res_U_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P1_unit.setObjectName("res_U_P1_unit")
self.res_P_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_P_LBL.setGeometry(QtCore.QRect(0, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_P_LBL.setFont(font)
self.res_P_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_LBL.setObjectName("res_P_LBL")
self.res_U_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P1_DSB.setEnabled(False)
self.res_U_P1_DSB.setGeometry(QtCore.QRect(90, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_DSB.setFont(font)
```

```
self.res_U_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P1_DSB.setDecimals(3)
self.res_U_P1_DSB.setMaximum(999999.999)
self.res_U_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P1_DSB.setProperty("value", 0.0)
self.res_U_P1_DSB.setObjectName("res_U_P1_DSB")
self.sf_unit = QtWidgets.QLabel(self.LoadFlow)
self.sf_unit.setGeometry(QtCore.QRect(168, 75, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sf_unit.setFont(font)
self.sf_unit.setText("")
self.sf_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.sf_unit.setObjectName("sf_unit")
self.Por1_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Por1_LBL.setGeometry(QtCore.QRect(90, 10, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Por1_LBL.setFont(font)
self.Por1_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Por1_LBL.setObjectName("Por1_LBL")
self.res_I_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_I_LBL.setGeometry(QtCore.QRect(0, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_I_LBL.setFont(font)
self.res_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_LBL.setObjectName("res_I_LBL")
self.res_S_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P1_DSB.setEnabled(False)
self.res_S_P1_DSB.setGeometry(QtCore.QRect(90, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_DSB.setFont(font)
self.res_S_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P1_DSB.setDecimals(3)
self.res_S_P1_DSB.setMaximum(999999.999)
self.res_S_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P1_DSB.setProperty("value", 0.0)
self.res_S_P1_DSB.setObjectName("res_S_P1_DSB")
self.res_I_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P1_unit.setGeometry(QtCore.QRect(170, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_unit.setFont(font)
self.res_I_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P1_unit.setObjectName("res_I_P1_unit")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(0, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_P_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P1_unit.setGeometry(QtCore.QRect(170, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_unit.setFont(font)
self.res_P_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P1_unit.setObjectName("res_P_P1_unit")
self.res_cosPhi_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_LBL.setGeometry(QtCore.QRect(0, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_cosPhi_LBL.setFont(font)
self.res_cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_LBL.setObjectName("res_cosPhi_LBL")
self.res_Iangle_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Iangle_LBL.setFont(font)
self.res_Iangle_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_LBL.setObjectName("res_Iangle_LBL")
self.res_Q_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_LBL.setGeometry(QtCore.QRect(0, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Q_LBL.setFont(font)
self.res_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_LBL.setObjectName("res_Q_LBL")
self.res_cosPhi_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P1_DSB.setEnabled(False)
self.res_cosPhi_P1_DSB.setGeometry(QtCore.QRect(90, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_DSB.setFont(font)
self.res_cosPhi_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
```

```
self.res_cosPhi_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P1_DSB.setDecimals(3)
self.res_cosPhi_P1_DSB.setMaximum(999999.999)
self.res_cosPhi_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P1_DSB.setProperty("value", 0.0)
self.res_cosPhi_P1_DSB.setObjectName("res_cosPhi_P1_DSB")
self.res_Iangle_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P1_unit.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_unit.setFont(font)
self.res_Iangle_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_unit.setObjectName("res_Iangle_P1_unit")
self.res_cosPhi_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P1_unit.setGeometry(QtCore.QRect(170, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_unit.setFont(font)
self.res_cosPhi_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_unit.setObjectName("res_cosPhi_P1_unit")
self.res_P_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P1_DSB.setEnabled(False)
self.res_P_P1_DSB.setGeometry(QtCore.QRect(90, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_DSB.setFont(font)
self.res_P_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P1_DSB.setDecimals(3)
self.res_P_P1_DSB.setMinimum(-999999.999)
self.res_P_P1_DSB.setMaximum(999999.999)
self.res_P_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P1_DSB.setProperty("value", 0.0)
self.res_P_P1_DSB.setObjectName("res_P_P1_DSB")
self.res_Q_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P1_DSB.setEnabled(False)
self.res_Q_P1_DSB.setGeometry(QtCore.QRect(90, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_DSB.setFont(font)
self.res_Q_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P1_DSB.setDecimals(3)
self.res_Q_P1_DSB.setMinimum(-999999.999)
self.res_Q_P1_DSB.setMaximum(999999.999)
self.res_Q_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P1_DSB.setProperty("value", 0.0)
self.res_Q_P1_DSB.setObjectName("res_Q_P1_DSB")
self.tabWidget.addTab(self.LoadFlow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.cap_pwr_LBL = QtWidgets.QLabel(self.EMS)
self.cap_pwr_LBL.setGeometry(QtCore.QRect(10, 10, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.cap_pwr_LBL.setFont(font)
self.cap_pwr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_LBL.setObjectName("cap_pwr_LBL")
self.cap_pwr_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.cap_pwr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_DSB.setFont(font)
self.cap_pwr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cap_pwr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cap_pwr_DSB.setDecimals(3)
self.cap_pwr_DSB.setMaximum(999999.999)
self.cap_pwr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cap_pwr_DSB.setProperty("value", 0.0)
self.cap_pwr_DSB.setObjectName("cap_pwr_DSB")
self.cap_pwr_unit = QtWidgets.QLabel(self.EMS)
self.cap_pwr_unit.setGeometry(QtCore.QRect(240, 10, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_unit.setFont(font)
self.cap_pwr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cap_pwr_unit.setObjectName("cap_pwr_unit")
self.tabWidget.addTab(self.EMS, "")
self.Reliability = QtWidgets.QWidget()
self.Reliability.setObjectName("Reliability")
self.rel_results_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 65, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_alfa_unit = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_unit.setGeometry(QtCore.QRect(250, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
```

```
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_R_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_R_LBL.setGeometry(QtCore.QRect(200, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_beta_unit = QtWidgets.QLabel(self.Reliability)
self.rel_beta_unit.setGeometry(QtCore.QRect(250, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(10, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
self.rel_T0_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_T0_LBL.setGeometry(QtCore.QRect(10, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(170, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(170, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.bottom_LN_3 = QtWidgets.QFrame(self.Reliability)
self.bottom_LN_3.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN_3.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_3.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_3.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_3.setObjectName("bottom_LN_3")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(170, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(170, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
```

```
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(250, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_beta_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_beta_LBL.setGeometry(QtCore.QRect(10, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(250, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_T0_unit = QtWidgets.QLabel(self.Reliability)
self.rel_T0_unit.setGeometry(QtCore.QRect(250, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(10, 170, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(160, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(170, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_anni_DSB.setEnabled(False)
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
```

```
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(10, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_R_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_R_unit = QtWidgets.QLabel(self.Reliability)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_R_unit.setObjectName("rel_R_unit")
self.bottom_LN = QtWidgets.QFrame(self.Reliability)
self.bottom_LN.setGeometry(QtCore.QRect(-22, 395, 490, 1))
self.bottom_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN.setObjectName("bottom_LN")
self.bottom_LN_2 = QtWidgets.QFrame(self.Reliability)
self.bottom_LN_2.setGeometry(QtCore.QRect(-22, 395, 490, 1))
self.bottom_LN_2.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_2.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_2.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_2.setObjectName("bottom_LN_2")
self.rel_lambda_unit = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.rel_alfa_unit.raise_()
self.rel_R_LBL.raise_()
self.rel_beta_unit.raise_()
self.rel_alfa_LBL.raise_()
self.rel_T0_LBL.raise_()
self.rel_beta_DSB.raise_()
self.rel_alfa_DSB.raise_()
self.rel_lambda_LBL.raise_()
self.bottom_LN_3.raise_()
self.rel_Pi_Q_DSB.raise_()
self.rel_T0_DSB.raise_()
self.rel_MTBF_ore_unit.raise_()
self.rel_MTBF_ore_DSB.raise_()
self.rel_Pi_Q_unit.raise_()
self.rel_MTBF_ore_LBL.raise_()
self.rel_beta_LBL.raise_()
self.rel_Pi_E_unit.raise_()
self.rel_T0_unit.raise_()
self.rel_Pi_Q_LBL.raise_()
self.rel_MTBF_anni_LBL.raise_()
self.rel_Pi_E_DSB.raise_()
self.rel_MTBF_anni_DSB.raise_()
self.rel_Pi_E_LBL.raise_()
self.rel_R_LE.raise_()
self.rel_lambda_LE.raise_()
self.rel_R_unit.raise_()
self.bottom_LN.raise_()
self.bottom_LN_2.raise_()
self.rel_results_LBL.raise_()
self.rel_lambda_unit.raise_()
self.rel_MTBF_anni_unit.raise_()
self.tabWidget.addTab(self.Reliability, "")
self.Protections = QtWidgets.QWidget()
self.Protections.setObjectName("Protections")
self.prot_cal_Vmin_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_val_unit_LBL.setGeometry(QtCore.QRect(170, 210, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_val_unit_LBL.setFont(font)
self.prot_cal_Vmin_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_unit_LBL.setObjectName("prot_cal_Vmin_val_unit_LBL")
self.prot_cal_Vmin_LBL = QtWidgets.QLabel(self.Protections)
```

```
self.prot_cal_Vmin_LBL.setGeometry(QtCore.QRect(0, 210, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_LBL.setFont(font)
self.prot_cal_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_LBL.setObjectName("prot_cal_Vmin_LBL")
self.prot_calib_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_btm_LN.setGeometry(QtCore.QRect(10, 240, 305, 1))
self.prot_calib_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_calib_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_btm_LN.setObjectName("prot_calib_btm_LN")
self.prot_calib_LBL = QtWidgets.QLabel(self.Protections)
self.prot_calib_LBL.setGeometry(QtCore.QRect(110, 120, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_calib_LBL.setFont(font)
self.prot_calib_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_calib_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_calib_LBL.setObjectName("prot_calib_LBL")
self.prot_cal_Vmax_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_val_DSB.setGeometry(QtCore.QRect(90, 180, 71, 21))
self.prot_cal_Vmax_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_val_DSB.setDecimals(3)
self.prot_cal_Vmax_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_val_DSB.setObjectName("prot_cal_Vmax_val_DSB")
self.prot_cal_Vmin_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_pu_DSB.setGeometry(QtCore.QRect(220, 210, 71, 21))
self.prot_cal_Vmin_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_pu_DSB.setDecimals(1)
self.prot_cal_Vmin_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_pu_DSB.setObjectName("prot_cal_Vmin_pu_DSB")
self.prot_cal_Vmin_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_pu_unit_LBL.setGeometry(QtCore.QRect(300, 210, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_pu_unit_LBL.setFont(font)
self.prot_cal_Vmin_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_unit_LBL.setObjectName("prot_cal_Vmin_pu_unit_LBL")
self.prot_delay_Vmin_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_unit_LBL.setGeometry(QtCore.QRect(170, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmin_unit_LBL.setFont(font)
self.prot_delay_Vmin_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_unit_LBL.setObjectName("prot_delay_Vmin_unit_LBL")
self.prot_type_LBL = QtWidgets.QLabel(self.Protections)
self.prot_type_LBL.setGeometry(QtCore.QRect(0, 40, 81, 20))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_type_LBL.setFont(font)
self.prot_type_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_type_LBL.setObjectName("prot_type_LBL")
self.prot_cal_Vmax_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_pu_DSB.setGeometry(QtCore.QRect(220, 180, 71, 21))
self.prot_cal_Vmax_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_pu_DSB.setDecimals(1)
self.prot_cal_Vmax_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_pu_DSB.setObjectName("prot_cal_Vmax_pu_DSB")
self.prot_cal_I_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_sep_LBL.setGeometry(QtCore.QRect(190, 150, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_sep_LBL.setFont(font)
self.prot_cal_I_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_sep_LBL.setObjectName("prot_cal_I_sep_LBL")
self.prot_cost_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cost_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cost_LBL.setFont(font)
self.prot_cost_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cost_LBL.setObjectName("prot_cost_LBL")
self.prot_delay_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_btm_LN.setGeometry(QtCore.QRect(10, 380, 305, 1))
self.prot_delay_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_btm_LN.setObjectName("prot_delay_btm_LN")
self.prot_delay_Vmin_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_LBL.setGeometry(QtCore.QRect(0, 350, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_Vmin_LBL.setFont(font)
self.prot_delay_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_LBL.setObjectName("prot_delay_Vmin_LBL")
self.prot_delay_Vmin_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmin_DSB.setGeometry(QtCore.QRect(90, 350, 71, 21))
self.prot_delay_Vmin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmin_DSB.setDecimals(3)
self.prot_delay_Vmin_DSB.setMaximum(99999999.0)
self.prot_delay_Vmin_DSB.setObjectName("prot_delay_Vmin_DSB")
self.prot_cost_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cost_unit_LBL.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
```

```
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cost_unit_LBL.setFont(font)
self.prot_cost_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cost_unit_LBL.setObjectName("prot_cost_unit_LBL")
self.prot_calib_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_top_LN.setGeometry(QtCore.QRect(10, 130, 305, 1))
self.prot_calib_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_calib_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_top_LN.setObjectName("prot_calib_top_LN")
self.prot_delay_I_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_I_DSB.setGeometry(QtCore.QRect(90, 290, 71, 21))
self.prot_delay_I_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_I_DSB.setDecimals(3)
self.prot_delay_I_DSB.setMaximum(99999999.0)
self.prot_delay_I_DSB.setObjectName("prot_delay_I_DSB")
self.prot_cal_Vmax_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_sep_LBL.setGeometry(QtCore.QRect(190, 180, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_sep_LBL.setFont(font)
self.prot_cal_Vmax_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_sep_LBL.setObjectName("prot_cal_Vmax_sep_LBL")
self.prot_delay_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_top_LN.setGeometry(QtCore.QRect(10, 270, 305, 1))
self.prot_delay_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_top_LN.setObjectName("prot_delay_top_LN")
self.prot_cal_Vmin_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_sep_LBL.setGeometry(QtCore.QRect(190, 210, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_sep_LBL.setFont(font)
self.prot_cal_Vmin_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_sep_LBL.setObjectName("prot_cal_Vmin_sep_LBL")
self.prot_cdelay_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cdelay_LBL.setGeometry(QtCore.QRect(110, 260, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cdelay_LBL.setFont(font)
self.prot_cdelay_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_cdelay_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_cdelay_LBL.setObjectName("prot_cdelay_LBL")
self.prot_delay_I_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_unit_LBL.setGeometry(QtCore.QRect(170, 290, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_I_unit_LBL.setFont(font)
self.prot_delay_I_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_I_unit_LBL.setObjectName("prot_delay_I_unit_LBL")
self.prot_cal_Vmax_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_val_unit_LBL.setGeometry(QtCore.QRect(170, 180, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmax_val_unit_LBL.setFont(font)
self.prot_cal_Vmax_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_val_unit_LBL.setObjectName("prot_cal_Vmax_val_unit_LBL")
self.prot_char_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_top_LN.setGeometry(QtCore.QRect(10, 20, 305, 1))
self.prot_char_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_top_LN.setObjectName("prot_char_top_LN")
self.prot_delay_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_LBL.setGeometry(QtCore.QRect(0, 320, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_Vmax_LBL.setFont(font)
self.prot_delay_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_LBL.setObjectName("prot_delay_Vmax_LBL")
self.prot_delay_Vmax_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmax_DSB.setGeometry(QtCore.QRect(90, 320, 71, 21))
self.prot_delay_Vmax_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmax_DSB.setDecimals(3)
self.prot_delay_Vmax_DSB.setMaximum(99999999.0)
self.prot_delay_Vmax_DSB.setObjectName("prot_delay_Vmax_DSB")
self.prot_cal_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_LBL.setGeometry(QtCore.QRect(0, 180, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_LBL.setFont(font)
self.prot_cal_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_LBL.setObjectName("prot_cal_Vmax_LBL")
self.prot_charact_LBL = QtWidgets.QLabel(self.Protections)
self.prot_charact_LBL.setGeometry(QtCore.QRect(60, 10, 211, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_charact_LBL.setFont(font)
self.prot_charact_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_charact_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_charact_LBL.setObjectName("prot_charact_LBL")
self.prot_cal_I_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_pu_unit_LBL.setGeometry(QtCore.QRect(300, 150, 21, 21))
font = QtGui.QFont()
```

```
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_pu_unit_LBL.setFont(font)
self.prot_cal_I_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_unit_LBL.setObjectName("prot_cal_I_pu_unit_LBL")
self.prot_cal_Vmin_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_val_DSB.setGeometry(QtCore.QRect(90, 210, 71, 21))
self.prot_cal_Vmin_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_val_DSB.setDecimals(3)
self.prot_cal_Vmin_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_val_DSB.setObjectName("prot_cal_Vmin_val_DSB")
self.prot_cal_Vmax_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_pu_unit_LBL.setGeometry(QtCore.QRect(300, 180, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmax_pu_unit_LBL.setFont(font)
self.prot_cal_Vmax_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_unit_LBL.setObjectName("prot_cal_Vmax_pu_unit_LBL")
self.prot_cal_I_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_val_unit_LBL.setGeometry(QtCore.QRect(170, 150, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_val_unit_LBL.setFont(font)
self.prot_cal_I_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_unit_LBL.setObjectName("prot_cal_I_val_unit_LBL")
self.prot_char_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_btm_LN.setGeometry(QtCore.QRect(10, 100, 305, 1))
self.prot_char_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_btm_LN.setObjectName("prot_char_btm_LN")
self.prot_type_LE = QtWidgets.QLineEdit(self.Protections)
self.prot_type_LE.setGeometry(QtCore.QRect(90, 40, 201, 20))
self.prot_type_LE.setText("")
self.prot_type_LE.setObjectName("prot_type_LE")
self.prot_cal_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_LBL.setGeometry(QtCore.QRect(0, 150, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_LBL.setFont(font)
self.prot_cal_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_LBL.setObjectName("prot_cal_I_LBL")
self.prot_delay_Vmax_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_unit_LBL.setGeometry(QtCore.QRect(170, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmax_unit_LBL.setFont(font)
self.prot_delay_Vmax_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_unit_LBL.setObjectName("prot_delay_Vmax_unit_LBL")
self.prot_cal_I_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_val_DSB.setGeometry(QtCore.QRect(90, 150, 71, 21))
self.prot_cal_I_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_val_DSB.setDecimals(1)
self.prot_cal_I_val_DSB.setMaximum(99999999.0)
self.prot_cal_I_val_DSB.setObjectName("prot_cal_I_val_DSB")
self.prot_cost_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cost_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
self.prot_cost_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cost_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cost_DSB.setDecimals(2)
self.prot_cost_DSB.setMaximum(99999999.0)
self.prot_cost_DSB.setObjectName("prot_cost_DSB")
self.prot_delay_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_LBL.setGeometry(QtCore.QRect(0, 290, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_I_LBL.setFont(font)
self.prot_delay_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_LBL.setObjectName("prot_delay_I_LBL")
self.prot_cal_I_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_pu_DSB.setGeometry(QtCore.QRect(220, 150, 71, 21))
self.prot_cal_I_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_pu_DSB.setDecimals(1)
self.prot_cal_I_pu_DSB.setMaximum(99999999.0)
self.prot_cal_I_pu_DSB.setObjectName("prot_cal_I_pu_DSB")
self.prot_cal_Vmin_val_unit_LBL.raise_()
self.prot_cal_Vmin_LBL.raise_()
self.prot_calib_btm_LN.raise_()
self.prot_cal_Vmax_val_DSB.raise_()
self.prot_cal_Vmin_pu_DSB.raise_()
self.prot_cal_Vmin_pu_unit_LBL.raise_()
self.prot_delay_Vmin_unit_LBL.raise_()
self.prot_type_LBL.raise_()
self.prot_cal_Vmax_pu_DSB.raise_()
self.prot_cal_I_sep_LBL.raise_()
self.prot_cost_LBL.raise_()
self.prot_delay_btm_LN.raise_()
self.prot_delay_Vmin_LBL.raise_()
self.prot_delay_Vmin_DSB.raise_()
self.prot_cost_unit_LBL.raise_()
self.prot_calib_top_LN.raise_()
self.prot_delay_I_DSB.raise_()
self.prot_cal_Vmax_sep_LBL.raise_()
self.prot_delay_top_LN.raise_()
self.prot_cal_Vmin_sep_LBL.raise_()
self.prot_cdelay_LBL.raise_()
self.prot_delay_I_unit_LBL.raise_()
self.prot_cal_Vmax_val_unit_LBL.raise_()
self.prot_char_top_LN.raise_()
self.prot_delay_Vmax_LBL.raise_()
self.prot_delay_Vmax_DSB.raise_()
```

```
        self.prot_cal_Vmax_LBL.raise_()
        self.prot_charact_LBL.raise_()
        self.prot_cal_I_pu_unit_LBL.raise_()
        self.prot_cal_Vmin_val_DSB.raise_()
        self.prot_cal_Vmax_pu_unit_LBL.raise_()
        self.prot_cal_I_val_unit_LBL.raise_()
        self.prot_char_btm_LN.raise_()
        self.prot_type_LE.raise_()
        self.prot_cal_I_LBL.raise_()
        self.prot_delay_Vmax_unit_LBL.raise_()
        self.prot_cal_I_val_DSB.raise_()
        self.prot_cost_DSB.raise_()
        self.prot_delay_I_LBL.raise_()
        self.prot_cal_I_pu_DSB.raise_()
        self.prot_calib_LBL.raise_()
        self.tabWidget.addTab(self.Protections, "")
        self.bb_out_LBL.raise_()
        self.bb_in_LBL.raise_()
        self.cub_out_LBL.raise_()
        self.top_LN.raise_()
        self.elem_name_LN.raise_()
        self.elem_name_LBL.raise_()
        self.type_LBL.raise_()
        self.type_cap_LBL.raise_()
        self.cub_in_LBL.raise_()
        self.symbol_LBL.raise_()
        self.ot_Frame_LN.raise_()
        self.vdx_frame_LN.raise_()
        self.vsx_frame_LN.raise_()
        self.ob_frame_LN.raise_()
        self.bb_in_LN.raise_()
        self.bb_out_LN.raise_()
        self.store_BTN.raise_()
        self.cancel_BTN.raise_()
        self.tabWidget.raise_()

        self.retranslateUi(Form)
        self.tabWidget.setCurrentIndex(3)
        QtCore.QMetaObject.connectSlotsByName(Form)
        Form.setTabOrder(self.tabWidget, self.pr_DSB)
        Form.setTabOrder(self.pr_DSB, self.p_DSB)
        Form.setTabOrder(self.p_DSB, self.sf_DSB)
        Form.setTabOrder(self.sf_DSB, self.sf_const_RB)
        Form.setTabOrder(self.sf_const_RB, self.sf_profile_RB)
        Form.setTabOrder(self.sf_profile_RB, self.cap_pwr_DSB)
        Form.setTabOrder(self.cap_pwr_DSB, self.rel_T0_DSB)
        Form.setTabOrder(self.rel_T0_DSB, self.rel_alfa_DSB)
        Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
        Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
        Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
        Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
        Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
        Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_I_P1_DSB)
        Form.setTabOrder(self.res_I_P1_DSB, self.res_Iangle_P1_DSB)
        Form.setTabOrder(self.res_Iangle_P1_DSB, self.res_P_P1_DSB)
        Form.setTabOrder(self.res_P_P1_DSB, self.res_Q_P1_DSB)
        Form.setTabOrder(self.res_Q_P1_DSB, self.res_S_P1_DSB)
        Form.setTabOrder(self.res_S_P1_DSB, self.res_cosPhi_P1_DSB)
        Form.setTabOrder(self.res_cosPhi_P1_DSB, self.res_U_P1_DSB)
        Form.setTabOrder(self.res_U_P1_DSB, self.store_BTN)
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria:"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.store_BTN.setText(_translate("Form", "Salva"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))
        self.sf_profile_RB.setText(_translate("Form", "Profilo"))
        self.p_LBL.setText(_translate("Form", "P"))
        self.pr_LBL.setText(_translate("Form", "Pr"))
        self.sf_const_RB.setText(_translate("Form", "Costante"))
        self.sfi_LBL.setText(_translate("Form", "scala"))
        self.pri_unit.setText(_translate("Form", "kW"))
        self.p_unit.setText(_translate("Form", "kW"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Parameters), _translate("Form", "Parametri"))
        self.res_S_P1_unit.setText(_translate("Form", "kVA"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.res_S_LBL.setText(_translate("Form", "S"))
        self.res_Q_P1_unit.setText(_translate("Form", "kVA"))
        self.res_U_P1_unit.setText(_translate("Form", "kV"))
        self.res_P_LBL.setText(_translate("Form", "P"))
        self.Por1_LBL.setText(_translate("Form", "Nodo 1"))
        self.res_I_LBL.setText(_translate("Form", " I"))
        self.res_I_P1_unit.setText(_translate("Form", "A"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMITI VIOLATI"))
        self.res_P_P1_unit.setText(_translate("Form", "kW"))
        self.res_cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.res_Iangle_LBL.setText(_translate("Form", "Angolo I"))
        self.res_Q_LBL.setText(_translate("Form", "Q"))
        self.res_Iangle_P1_unit.setText(_translate("Form", "°"))
        self.res_cosPhi_P1_unit.setText(_translate("Form", "-"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFlow), _translate("Form", "LoadFlow"))
        self.cap_pwr_LBL.setText(_translate("Form", "Potenza Nominale"))
        self.cap_pwr_unit.setText(_translate("Form", "kVA"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
```

```
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_T0_unit.setText(_translate("Form", "°C"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Reliability), _translate("Form", "Affidabilità"))
        self.prot_cal_Vmin_val_unit_LBL.setText(_translate("Form", "kV"))
        self.prot_cal_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_calib_LBL.setText(_translate("Form", "Soglie di taratura"))
        self.prot_cal_Vmin_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_delay_Vmin_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_type_LBL.setText(_translate("Form", "Tipologia"))
        self.prot_cal_I_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cost_LBL.setText(_translate("Form", "Costo"))
        self.prot_delay_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_cost_unit_LBL.setText(_translate("Form", "Euro"))
        self.prot_cal_Vmax_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cal_Vmin_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cdelay_LBL.setText(_translate("Form", "Tempi di ritardo"))
        self.prot_delay_I_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_cal_Vmax_val_unit_LBL.setText(_translate("Form", "kV"))
        self.prot_delay_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_cal_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_charact_LBL.setText(_translate("Form", "Caratterisitiche dell\'interruttore"))
        self.prot_cal_I_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_Vmax_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_I_val_unit_LBL.setText(_translate("Form", "A"))
        self.prot_cal_I_LBL.setText(_translate("Form", "Corrente"))
        self.prot_delay_Vmax_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_delay_I_LBL.setText(_translate("Form", "Corrente"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Protections), _translate("Form", "Protezioni"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.3.11 ExtGrid

4.4.3.11.1 extgrid.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .extgridUI import Ui_Form
from __shared__ import variables as v
import copy


class ExtGrid(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(ExtGrid, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.bb_out_LN.setVisible(False)
        self.ui.bb_out_LBL.setVisible(False)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")
        self.ui.tabWidget.setCurrentIndex(0)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])

        self.bb = v.elements[element]['conn']['h']
        self.cubicle = v.elements[self.element]['conn'][self.bb]
        self.u = v.elements[self.bb]['parameters']['Ur']
        self.ui.bb_in_LBL.setText(self.bb)

        self.ui.symbol_LBL.setPixmap(QtGui.QPixmap("_images/Elements/ExtGrid/element.png"))
        self.switch_draw()
        self.fill()

    #
    def calculate(self):
        pass

    #
    def store(self): # Inutile?? ------------------------------------------------------------------------------
        for e in ['cap_pwr', 'base_peak', 'peak_cost', 'max_load', 'max_supply']:
            self.ems[e] = self.ui.__getattribute__(e + '_DSB').value()
        price_buy = []
        price_sell = []
        for i in range(0, 96):
            price_buy.append(self.ui.price_buy_DSB.value())
            price_sell.append(self.ui.price_sell_DSB.value())
        self.ems['profile']['price_buy'] = price_buy
        self.ems['profile']['price_sell'] = price_sell

        v.elements[self.element]['ems'] = copy.deepcopy(self.ems)

        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)

    #
    def fill(self):
        for e in ['cap_pwr', 'base_peak', 'peak_cost', 'max_load', 'max_supply']:
            self.ui.__getattribute__(e + '_DSB').setValue(self.ems[e])
        self.ui.price_buy_DSB.setValue(self.ems['profile']['price_buy'][0])
        self.ui.price_sell_DSB.setValue(self.ems['profile']['price_sell'][0])

        if self.res != {}:
            self.fill_results()
        self.ui.tabWidget.setTabVisible(3, self.res != {})
        self.fill_reliability()

    #
    def fill_reliability(self):
        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
        for par in ['lambda', 'R', 'MTBF_ore', 'MTBF_anni']:
            try:
                self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
            except:
                if self.rel['results'][par] == 0:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('0')
                elif self.rel['results'][par] < 0.01:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
                else:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])

    #
    def fill_results(self):
        results = ['I', 'Iangle', 'P', 'Q', 'S', 'cosPhi', 'U']
        ports = ['P1']

        for port in ports:
            for result in results:
                self.ui.__getattribute__('res_' + result + '_' + port + '_DSB').setValue(self.res[result])
        self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])

    #
    def switch_draw(self):
        if self.cubicle:
            self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
        else:
            self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))

    def cub1_switch(self, event):
        self.cubicle = not self.cubicle
        self.switch_draw()

    #
```

```
def calc_profile(self, pu_profile):
    profile = []
    for data in pu_profile:
        profile.append(data * self.par['P'])
    return profile
```

4.4.3.11.2 extgridUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'extgridUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(497, 505)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.widget.setFont(font)
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("res/Load.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.bb_in_LBL.setFont(font)
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
```

```
self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bb_out_LN.setObjectName("bb_out_LN")
self.bb_out_LBL = QtWidgets.QLabel(self.widget)
self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
self.bb_out_LBL.setFont(font)
self.bb_out_LBL.setStyleSheet("")
self.bb_out_LBL.setLineWidth(4)
self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.sf_unit = QtWidgets.QLabel(self.widget)
self.sf_unit.setGeometry(QtCore.QRect(340, 300, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sf_unit.setFont(font)
self.sf_unit.setText("")
self.sf_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.sf_unit.setObjectName("sf_unit")
self.store_BTN = QtWidgets.QPushButton(self.widget)
self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.store_BTN.setFont(font)
self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.store_BTN.setObjectName("store_BTN")
self.cancel_BTN = QtWidgets.QPushButton(self.widget)
self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.cancel_BTN.setFont(font)
self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.cancel_BTN.setObjectName("cancel_BTN")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.tabWidget.addTab(self.Parameters, "")
self.LoadFLow = QtWidgets.QWidget()
self.LoadFLow.setObjectName("LoadFLow")
self.res_P_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_P_P1_DSB.setEnabled(False)
self.res_P_P1_DSB.setGeometry(QtCore.QRect(90, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_DSB.setFont(font)
self.res_P_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P1_DSB.setDecimals(3)
self.res_P_P1_DSB.setMinimum(-999999.999)
self.res_P_P1_DSB.setMaximum(999999.999)
self.res_P_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P1_DSB.setProperty("value", 0.0)
self.res_P_P1_DSB.setObjectName("res_P_P1_DSB")
self.res_Iangle_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_Iangle_P1_DSB.setEnabled(False)
self.res_Iangle_P1_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_DSB.setFont(font)
self.res_Iangle_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P1_DSB.setDecimals(3)
self.res_Iangle_P1_DSB.setMinimum(-999999.999)
self.res_Iangle_P1_DSB.setMaximum(999999.999)
self.res_Iangle_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P1_DSB.setProperty("value", 0.0)
self.res_Iangle_P1_DSB.setObjectName("res_Iangle_P1_DSB")
self.res_Iangle_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_Iangle_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Iangle_LBL.setFont(font)
self.res_Iangle_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_LBL.setObjectName("res_Iangle_LBL")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_U_LBL.setGeometry(QtCore.QRect(0, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.Por1_LBL = QtWidgets.QLabel(self.LoadFLow)
self.Por1_LBL.setGeometry(QtCore.QRect(90, 10, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Por1_LBL.setFont(font)
self.Por1_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Por1_LBL.setObjectName("Por1_LBL")
self.res_U_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_U_P1_DSB.setEnabled(False)
self.res_U_P1_DSB.setGeometry(QtCore.QRect(90, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
```

```
self.res_U_P1_DSB.setFont(font)
self.res_U_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P1_DSB.setDecimals(3)
self.res_U_P1_DSB.setMaximum(999999.999)
self.res_U_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P1_DSB.setProperty("value", 0.0)
self.res_U_P1_DSB.setObjectName("res_U_P1_DSB")
self.res_Iangle_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_Iangle_P1_unit.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_unit.setFont(font)
self.res_Iangle_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_unit.setObjectName("res_Iangle_P1_unit")
self.res_I_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_I_LBL.setGeometry(QtCore.QRect(0, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_I_LBL.setFont(font)
self.res_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_LBL.setObjectName("res_I_LBL")
self.res_S_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_S_P1_DSB.setEnabled(False)
self.res_S_P1_DSB.setGeometry(QtCore.QRect(90, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_DSB.setFont(font)
self.res_S_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P1_DSB.setDecimals(3)
self.res_S_P1_DSB.setMaximum(999999.999)
self.res_S_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P1_DSB.setProperty("value", 0.0)
self.res_S_P1_DSB.setObjectName("res_S_P1_DSB")
self.res_P_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_P_P1_unit.setGeometry(QtCore.QRect(170, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_unit.setFont(font)
self.res_P_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P1_unit.setObjectName("res_P_P1_unit")
self.res_S_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_S_LBL.setGeometry(QtCore.QRect(0, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_S_LBL.setFont(font)
self.res_S_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_LBL.setObjectName("res_S_LBL")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(0, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_I_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_I_P1_unit.setGeometry(QtCore.QRect(170, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_unit.setFont(font)
self.res_I_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P1_unit.setObjectName("res_I_P1_unit")
self.res_cosPhi_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_cosPhi_P1_unit.setGeometry(QtCore.QRect(170, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_unit.setFont(font)
self.res_cosPhi_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_unit.setObjectName("res_cosPhi_P1_unit")
self.res_Q_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_Q_P1_unit.setGeometry(QtCore.QRect(170, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_unit.setFont(font)
self.res_Q_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P1_unit.setObjectName("res_Q_P1_unit")
self.res_cosPhi_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_cosPhi_LBL.setGeometry(QtCore.QRect(0, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_cosPhi_LBL.setFont(font)
self.res_cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_LBL.setObjectName("res_cosPhi_LBL")
self.res_Q_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_Q_P1_DSB.setEnabled(False)
self.res_Q_P1_DSB.setGeometry(QtCore.QRect(90, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_DSB.setFont(font)
self.res_Q_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
```

```
self.res_Q_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P1_DSB.setDecimals(3)
self.res_Q_P1_DSB.setMinimum(-999999.999)
self.res_Q_P1_DSB.setMaximum(999999.999)
self.res_Q_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P1_DSB.setProperty("value", 0.0)
self.res_Q_P1_DSB.setObjectName("res_Q_P1_DSB")
self.res_P_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_P_LBL.setGeometry(QtCore.QRect(0, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_P_LBL.setFont(font)
self.res_P_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_LBL.setObjectName("res_P_LBL")
self.res_cosPhi_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_cosPhi_P1_DSB.setEnabled(False)
self.res_cosPhi_P1_DSB.setGeometry(QtCore.QRect(90, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_DSB.setFont(font)
self.res_cosPhi_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P1_DSB.setDecimals(3)
self.res_cosPhi_P1_DSB.setMaximum(999999.999)
self.res_cosPhi_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P1_DSB.setProperty("value", 0.0)
self.res_cosPhi_P1_DSB.setObjectName("res_cosPhi_P1_DSB")
self.res_S_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_S_P1_unit.setGeometry(QtCore.QRect(170, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_unit.setFont(font)
self.res_S_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P1_unit.setObjectName("res_S_P1_unit")
self.res_Q_LBL = QtWidgets.QLabel(self.LoadFLow)
self.res_Q_LBL.setGeometry(QtCore.QRect(0, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Q_LBL.setFont(font)
self.res_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_LBL.setObjectName("res_Q_LBL")
self.res_U_P1_unit = QtWidgets.QLabel(self.LoadFLow)
self.res_U_P1_unit.setGeometry(QtCore.QRect(170, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_unit.setFont(font)
self.res_U_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P1_unit.setObjectName("res_U_P1_unit")
self.res_I_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFLow)
self.res_I_P1_DSB.setEnabled(False)
self.res_I_P1_DSB.setGeometry(QtCore.QRect(90, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_DSB.setFont(font)
self.res_I_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P1_DSB.setDecimals(3)
self.res_I_P1_DSB.setMaximum(999999.999)
self.res_I_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P1_DSB.setProperty("value", 0.0)
self.res_I_P1_DSB.setObjectName("res_I_P1_DSB")
self.tabWidget.addTab(self.LoadFLow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.cap_pwr_LBL = QtWidgets.QLabel(self.EMS)
self.cap_pwr_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.cap_pwr_LBL.setFont(font)
self.cap_pwr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_LBL.setObjectName("cap_pwr_LBL")
self.cap_pwr_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.cap_pwr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_DSB.setFont(font)
self.cap_pwr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cap_pwr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cap_pwr_DSB.setDecimals(3)
self.cap_pwr_DSB.setMaximum(999999.999)
self.cap_pwr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cap_pwr_DSB.setProperty("value", 0.0)
self.cap_pwr_DSB.setObjectName("cap_pwr_DSB")
self.cap_pwr_unit = QtWidgets.QLabel(self.EMS)
self.cap_pwr_unit.setGeometry(QtCore.QRect(240, 10, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_unit.setFont(font)
self.cap_pwr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cap_pwr_unit.setObjectName("cap_pwr_unit")
self.base_peak_unit = QtWidgets.QLabel(self.EMS)
self.base_peak_unit.setGeometry(QtCore.QRect(240, 40, 61, 21))
font = QtGui.QFont()
```

```
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.base_peak_unit.setFont(font)
self.base_peak_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.base_peak_unit.setObjectName("base_peak_unit")
self.base_peak_LBL = QtWidgets.QLabel(self.EMS)
self.base_peak_LBL.setGeometry(QtCore.QRect(0, 40, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.base_peak_LBL.setFont(font)
self.base_peak_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.base_peak_LBL.setObjectName("base_peak_LBL")
self.base_peak_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.base_peak_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.base_peak_DSB.setFont(font)
self.base_peak_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.base_peak_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.base_peak_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.base_peak_DSB.setDecimals(3)
self.base_peak_DSB.setMaximum(999999.999)
self.base_peak_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.base_peak_DSB.setProperty("value", 0.0)
self.base_peak_DSB.setObjectName("base_peak_DSB")
self.peak_cost_unit = QtWidgets.QLabel(self.EMS)
self.peak_cost_unit.setGeometry(QtCore.QRect(240, 70, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.peak_cost_unit.setFont(font)
self.peak_cost_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.peak_cost_unit.setObjectName("peak_cost_unit")
self.peak_cost_LBL = QtWidgets.QLabel(self.EMS)
self.peak_cost_LBL.setGeometry(QtCore.QRect(0, 70, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.peak_cost_LBL.setFont(font)
self.peak_cost_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.peak_cost_LBL.setObjectName("peak_cost_LBL")
self.peak_cost_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.peak_cost_DSB.setGeometry(QtCore.QRect(160, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.peak_cost_DSB.setFont(font)
self.peak_cost_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.peak_cost_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.peak_cost_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.peak_cost_DSB.setDecimals(3)
self.peak_cost_DSB.setMaximum(999999.999)
self.peak_cost_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.peak_cost_DSB.setProperty("value", 0.0)
self.peak_cost_DSB.setObjectName("peak_cost_DSB")
self.max_supply_LBL = QtWidgets.QLabel(self.EMS)
self.max_supply_LBL.setGeometry(QtCore.QRect(10, 130, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_supply_LBL.setFont(font)
self.max_supply_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_supply_LBL.setObjectName("max_supply_LBL")
self.max_load_unit = QtWidgets.QLabel(self.EMS)
self.max_load_unit.setGeometry(QtCore.QRect(240, 100, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_load_unit.setFont(font)
self.max_load_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_load_unit.setObjectName("max_load_unit")
self.max_supply_unit = QtWidgets.QLabel(self.EMS)
self.max_supply_unit.setGeometry(QtCore.QRect(240, 130, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_supply_unit.setFont(font)
self.max_supply_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_supply_unit.setObjectName("max_supply_unit")
self.max_load_LBL = QtWidgets.QLabel(self.EMS)
self.max_load_LBL.setGeometry(QtCore.QRect(10, 100, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_load_LBL.setFont(font)
self.max_load_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_load_LBL.setObjectName("max_load_LBL")
self.max_supply_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.max_supply_DSB.setGeometry(QtCore.QRect(160, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_supply_DSB.setFont(font)
self.max_supply_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_supply_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_supply_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_supply_DSB.setDecimals(3)
self.max_supply_DSB.setMaximum(999999.999)
self.max_supply_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_supply_DSB.setProperty("value", 0.0)
self.max_supply_DSB.setObjectName("max_supply_DSB")
self.max_load_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
```

```
self.max_load_DSB.setGeometry(QtCore.QRect(160, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_load_DSB.setFont(font)
self.max_load_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_load_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_load_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_load_DSB.setDecimals(3)
self.max_load_DSB.setMaximum(999999.999)
self.max_load_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_load_DSB.setProperty("value", 0.0)
self.max_load_DSB.setObjectName("max_load_DSB")
self.price_buy_unit = QtWidgets.QLabel(self.EMS)
self.price_buy_unit.setGeometry(QtCore.QRect(240, 160, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.price_buy_unit.setFont(font)
self.price_buy_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.price_buy_unit.setObjectName("price_buy_unit")
self.price_buy_LBL = QtWidgets.QLabel(self.EMS)
self.price_buy_LBL.setGeometry(QtCore.QRect(0, 160, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.price_buy_LBL.setFont(font)
self.price_buy_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.price_buy_LBL.setObjectName("price_buy_LBL")
self.price_buy_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.price_buy_DSB.setGeometry(QtCore.QRect(160, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.price_buy_DSB.setFont(font)
self.price_buy_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.price_buy_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.price_buy_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.price_buy_DSB.setDecimals(3)
self.price_buy_DSB.setMaximum(999999.999)
self.price_buy_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.price_buy_DSB.setProperty("value", 0.0)
self.price_buy_DSB.setObjectName("price_buy_DSB")
self.price_sell_unit = QtWidgets.QLabel(self.EMS)
self.price_sell_unit.setGeometry(QtCore.QRect(240, 190, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.price_sell_unit.setFont(font)
self.price_sell_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.price_sell_unit.setObjectName("price_sell_unit")
self.price_sell_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.price_sell_DSB.setGeometry(QtCore.QRect(160, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.price_sell_DSB.setFont(font)
self.price_sell_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.price_sell_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.price_sell_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.price_sell_DSB.setDecimals(3)
self.price_sell_DSB.setMaximum(999999.999)
self.price_sell_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.price_sell_DSB.setProperty("value", 0.0)
self.price_sell_DSB.setObjectName("price_sell_DSB")
self.price_sell_LBL = QtWidgets.QLabel(self.EMS)
self.price_sell_LBL.setGeometry(QtCore.QRect(0, 190, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.price_sell_LBL.setFont(font)
self.price_sell_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.price_sell_LBL.setObjectName("price_sell_LBL")
self.tabWidget.addTab(self.EMS, "")
self.Reliability = QtWidgets.QWidget()
self.Reliability.setObjectName("Reliability")
self.rel_results_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 65, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_alfa_unit = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_unit.setGeometry(QtCore.QRect(240, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_R_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_R_LBL.setGeometry(QtCore.QRect(160, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_lambda_unit = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
```

```
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_beta_unit = QtWidgets.QLabel(self.Reliability)
self.rel_beta_unit.setGeometry(QtCore.QRect(240, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(0, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
self.rel_T0_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_T0_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(160, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(160, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.bottom_LN_2 = QtWidgets.QFrame(self.Reliability)
self.bottom_LN_2.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN_2.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_2.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_2.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_2.setObjectName("bottom_LN_2")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(160, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 21, 21))
font = QtGui.QFont()
```

```
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(240, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_beta_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_beta_LBL.setGeometry(QtCore.QRect(0, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(240, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_T0_unit = QtWidgets.QLabel(self.Reliability)
self.rel_T0_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(0, 170, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(160, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(160, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_anni_DSB.setEnabled(False)
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
```

```
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(0, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_R_unit = QtWidgets.QLabel(self.Reliability)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_R_unit.setObjectName("rel_R_unit")
self.rel_R_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
self.rel_R_LE.setFont(font)
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
self.rel_lambda_LE.setFont(font)
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.rel_alfa_unit.raise_()
self.rel_R_LBL.raise_()
self.rel_lambda_unit.raise_()
self.rel_beta_unit.raise_()
self.rel_alfa_LBL.raise_()
self.rel_T0_LBL.raise_()
self.rel_beta_DSB.raise_()
self.rel_alfa_DSB.raise_()
self.rel_lambda_LBL.raise_()
self.bottom_LN_2.raise_()
self.rel_Pi_Q_DSB.raise_()
self.rel_T0_DSB.raise_()
self.rel_MTBF_ore_unit.raise_()
self.rel_MTBF_ore_DSB.raise_()
self.rel_Pi_Q_unit.raise_()
self.rel_MTBF_ore_LBL.raise_()
self.rel_beta_LBL.raise_()
self.rel_Pi_E_unit.raise_()
self.rel_T0_unit.raise_()
self.rel_Pi_Q_LBL.raise_()
self.rel_MTBF_anni_LBL.raise_()
self.rel_Pi_E_DSB.raise_()
self.rel_MTBF_anni_DSB.raise_()
self.rel_Pi_E_LBL.raise_()
self.rel_R_unit.raise_()
self.rel_R_LE.raise_()
self.rel_lambda_LE.raise_()
self.rel_results_LBL.raise_()
self.rel_MTBF_anni_unit.raise_()
self.tabWidget.addTab(self.Reliability, "")
self.bb_out_LBL.raise_()
self.bb_in_LBL.raise_()
self.cub_out_LBL.raise_()
self.top_LN.raise_()
self.elem_name_LN.raise_()
self.elem_name_LBL.raise_()
self.type_LBL.raise_()
self.type_cap_LBL.raise_()
self.cub_in_LBL.raise_()
self.symbol_LBL.raise_()
self.ot_Frame_LN.raise_()
self.vdx_frame_LN.raise_()
self.vsx_frame_LN.raise_()
self.ob_frame_LN.raise_()
self.bb_in_LN.raise_()
self.bb_out_LN.raise_()
self.sf_unit.raise_()
self.store_BTN.raise_()
self.cancel_BTN.raise_()
self.tabWidget.raise_()

self.retranslateUi(Form)
self.tabWidget.setCurrentIndex(2)
QtCore.QMetaObject.connectSlotsByName(Form)
Form.setTabOrder(self.tabWidget, self.cap_pwr_DSB)
Form.setTabOrder(self.cap_pwr_DSB, self.rel_T0_DSB)
Form.setTabOrder(self.rel_T0_DSB, self.rel_alfa_DSB)
Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_I_P1_DSB)
Form.setTabOrder(self.res_I_P1_DSB, self.res_Iangle_P1_DSB)
Form.setTabOrder(self.res_Iangle_P1_DSB, self.res_P_P1_DSB)
```

```
        Form.setTabOrder(self.res_P_P1_DSB, self.res_Q_P1_DSB)
        Form.setTabOrder(self.res_Q_P1_DSB, self.res_S_P1_DSB)
        Form.setTabOrder(self.res_S_P1_DSB, self.res_cosPhi_P1_DSB)
        Form.setTabOrder(self.res_cosPhi_P1_DSB, self.res_U_P1_DSB)
        Form.setTabOrder(self.res_U_P1_DSB, self.store_BTN)
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria:"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.store_BTN.setText(_translate("Form", "Salva"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Parameters), _translate("Form", "Parametri"))
        self.res_Iangle_LBL.setText(_translate("Form", "I angle"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.Por1_LBL.setText(_translate("Form", "Port 1"))
        self.res_Iangle_P1_unit.setText(_translate("Form", "°"))
        self.res_I_LBL.setText(_translate("Form", "I"))
        self.res_P_P1_unit.setText(_translate("Form", "kW"))
        self.res_S_LBL.setText(_translate("Form", "S"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMIT VIOLATED"))
        self.res_I_P1_unit.setText(_translate("Form", "A"))
        self.res_cosPhi_P1_unit.setText(_translate("Form", "-"))
        self.res_Q_P1_unit.setText(_translate("Form", "kVA"))
        self.res_cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.res_P_LBL.setText(_translate("Form", "P"))
        self.res_S_P1_unit.setText(_translate("Form", "kVA"))
        self.res_Q_LBL.setText(_translate("Form", "Q"))
        self.res_U_P1_unit.setText(_translate("Form", "kV"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFLow), _translate("Form", "LoadFlow"))
        self.cap_pwr_LBL.setText(_translate("Form", "Potenza Nominale"))
        self.cap_pwr_unit.setText(_translate("Form", "kVA"))
        self.base_peak_unit.setText(_translate("Form", "kW"))
        self.base_peak_LBL.setText(_translate("Form", "Base Peak"))
        self.peak_cost_unit.setText(_translate("Form", "€/kWh"))
        self.peak_cost_LBL.setText(_translate("Form", "Peak Cost"))
        self.max_supply_LBL.setText(_translate("Form", "Potenza Massima OUT"))
        self.max_load_unit.setText(_translate("Form", "p.u."))
        self.max_supply_unit.setText(_translate("Form", "p.u."))
        self.max_load_LBL.setText(_translate("Form", "Potenza Massima IN"))
        self.price_buy_unit.setText(_translate("Form", "€/kWh"))
        self.price_buy_LBL.setText(_translate("Form", "Prezzo di acquisto"))
        self.price_sell_unit.setText(_translate("Form", "€/kWh"))
        self.price_sell_LBL.setText(_translate("Form", "Prezzo di vendita"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_T0_unit.setText(_translate("Form", "°C"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Reliability), _translate("Form", "Affidabilità"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.3.12 PV

4.4.3.12.1 pv.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .pvUI import Ui_Form
from __shared__ import variables as v
import copy


class PV(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(PV, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.bb_out_LN.setVisible(False)
        self.ui.bb_out_LBL.setVisible(False)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")

        for box in ['cap_pwr']:
            self.ui.__getattribute__(box + '_DSB').setStyleSheet("color: rgb(127, 127, 127);")
            self.ui.__getattribute__(box + '_DSB').setEnabled(False)

        self.ui.tabWidget.setCurrentIndex(0)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])
        self.prot = copy.deepcopy(v.elements[element]['protections'])

        self.bb = v.elements[element]['conn']['h']
        self.cubicle = v.elements[self.element]['conn'][self.bb]
        self.u = v.elements[self.bb]['parameters']['Ur']
        self.ui.bb_in_LBL.setText(self.bb)

        if v.software == 'neplan':
            bb_conns = list(v.elements[self.bb]['conn'].keys())
            bb_conns.remove(element)
            bb_conns.remove('h')
            pwm = bb_conns[0]
            self.par['Sr'] = v.elements[pwm]['parameters']['Sr']
            self.In = self.par['Sr'] / self.u
        else:
            self.In = self.par['In']    # In powerfactory questo parametro non cambia
            self.par['Sr'] = self.In * self.u

        self.ui.symbol_LBL.setPixmap(QtGui.QPixmap("_images/Elements/PV/element.png"))
        self.switch_draw()
        self.fill()
        self.tab_activation()

        self.ui.sr_DSB.valueChanged.connect(self.calculate)
        self.ui.cub_in_LBL.mouseDoubleClickEvent = self.cub1_switch

    #
    def tab_activation(self):
        self.ui.tabWidget.setTabVisible(2, v.functionality == 'EMS')
        self.ui.tabWidget.setTabVisible(3, v.functionality == 'Reliability')
        self.ui.tabWidget.setTabVisible(4, v.protections and self.prot != {})

    #
    def store(self):
        self.par['Sr'] = self.ui.sr_DSB.value()
        v.elements[self.element]['parameters'] = copy.deepcopy(self.par)

        v.elements[self.element]['conn'][self.bb] = self.cubicle

        self.ems['cap_pwr'] = self.ui.cap_pwr_DSB.value()
        v.elements[self.element]['ems'] = copy.deepcopy(self.ems)

        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)

        self.protections_par()

    #
    def protections_par(self):
        self.prot['Pn'] = self.par['Sr']
        self.prot['Vn'] = self.u
        self.prot['In'] = self.prot['Pn'] / self.prot['Vn']
        v.elements[self.element]['protections'] = copy.deepcopy(self.prot)

    #
    def fill(self):
        self.ui.sr_DSB.setValue(self.par['Sr'])
        self.calculate()
        if self.res != {}:
            self.fill_results()
        self.ui.tabWidget.setTabVisible(3, self.res != {})
        self.fill_reliability()
        if self.prot != {}:
            if self.prot['results'] != {} and v.protections:
                self.fill_protections()

    #
    def fill_reliability(self):
        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
        for par in self.rel['results']:
            try:
                self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
            except:
                if self.rel['results'][par] == 0:
```

```
                self.ui.__getattribute__('rel_' + par + '_LE').setText('0.0')
            elif self.rel['results'][par] < 0.01:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
            else:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])

#
def fill_protections(self):
    self.ui.prot_type_LE.setText(self.prot['results']['type'])
    self.ui.prot_cost_DSB.setValue(self.prot['results']['cost'])
    self.ui.prot_cal_I_val_DSB.setValue(self.prot['results']['soglia_I'])
    self.ui.prot_cal_I_pu_DSB.setValue(self.prot['results']['soglia_I']/self.prot['In'] * 100)
    self.ui.prot_cal_Vmax_val_DSB.setValue(self.prot['results']['soglia_Vmax'])
    self.ui.prot_cal_Vmax_pu_DSB.setValue(self.prot['results']['soglia_Vmax'] / self.prot['Vn'] * 100)
    self.ui.prot_cal_Vmin_val_DSB.setValue(self.prot['results']['soglia_Vmin'])
    self.ui.prot_cal_Vmin_pu_DSB.setValue(self.prot['results']['soglia_Vmin'] / self.prot['Vn'] * 100)
    self.ui.prot_delay_I_DSB.setValue(self.prot['results']['delay_I'])
    self.ui.prot_delay_Vmax_DSB.setValue(self.prot['results']['delay_Vmax'])
    self.ui.prot_delay_Vmin_DSB.setValue(self.prot['results']['delay_Vmin'])

#
def fill_results(self):
    results = ['I', 'Iangle', 'P', 'Q', 'S', 'cosPhi', 'U']
    ports = ['P1']
    for port in ports:
        for result in results:
            self.ui.__getattribute__('res_' + result + '_' + port + '_DSB').setValue(self.res[result])
    self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])

#
def switch_draw(self):
    if self.cubicle:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
    else:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))

#
def cub1_switch(self, event):
    self.cubicle = not self.cubicle
    self.switch_draw()

#
def calc_profile(self, pu_profile):
    profile = []
    for data in pu_profile:
        profile.append(data * 1)
    return profile

#
def calculate(self):
    self.ui.cap_pwr_DSB.setValue(self.ui.sr_DSB.value())
```

4.4.3.12.2 pvUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'pvUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(497, 509)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("../../../../../../.designer/backup/res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("../../../../../../.designer/backup/res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("../../../../../../.designer/backup/res/Load.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.bb_in_LBL.setFont(font)
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_out_LN.setObjectName("bb_out_LN")
        self.bb_out_LBL = QtWidgets.QLabel(self.widget)
        self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
self.bb_out_LBL.setFont(font)
self.bb_out_LBL.setStyleSheet("")
self.bb_out_LBL.setLineWidth(4)
self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.store_BTN = QtWidgets.QPushButton(self.widget)
self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.store_BTN.setFont(font)
self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.store_BTN.setObjectName("store_BTN")
self.cancel_BTN = QtWidgets.QPushButton(self.widget)
self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.cancel_BTN.setFont(font)
self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.cancel_BTN.setObjectName("cancel_BTN")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.sr_LBL = QtWidgets.QLabel(self.Parameters)
self.sr_LBL.setGeometry(QtCore.QRect(60, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.sr_LBL.setFont(font)
self.sr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sr_LBL.setObjectName("sr_LBL")
self.sr_unit = QtWidgets.QLabel(self.Parameters)
self.sr_unit.setGeometry(QtCore.QRect(240, 10, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sr_unit.setFont(font)
self.sr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.sr_unit.setObjectName("sr_unit")
self.sr_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.sr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sr_DSB.setFont(font)
self.sr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.sr_DSB.setDecimals(3)
self.sr_DSB.setMaximum(999999.999)
self.sr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.sr_DSB.setProperty("value", 0.0)
self.sr_DSB.setObjectName("sr_DSB")
self.sf_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.sf_DSB.setGeometry(QtCore.QRect(160, 40, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sf_DSB.setFont(font)
self.sf_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sf_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.sf_DSB.setDecimals(5)
self.sf_DSB.setMaximum(1.0)
self.sf_DSB.setSingleStep(0.01)
self.sf_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.sf_DSB.setProperty("value", 0.0)
self.sf_DSB.setObjectName("sf_DSB")
self.sf_const_RB = QtWidgets.QRadioButton(self.Parameters)
self.sf_const_RB.setGeometry(QtCore.QRect(240, 40, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
self.sf_const_RB.setFont(font)
self.sf_const_RB.setObjectName("sf_const_RB")
self.sf_profile_RB = QtWidgets.QRadioButton(self.Parameters)
self.sf_profile_RB.setGeometry(QtCore.QRect(240, 60, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
self.sf_profile_RB.setFont(font)
self.sf_profile_RB.setObjectName("sf_profile_RB")
self.sfi_LBL = QtWidgets.QLabel(self.Parameters)
self.sfi_LBL.setGeometry(QtCore.QRect(60, 40, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.sfi_LBL.setFont(font)
self.sfi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sfi_LBL.setObjectName("sfi_LBL")
self.tabWidget.addTab(self.Parameters, "")
self.LoadFlow = QtWidgets.QWidget()
self.LoadFlow.setObjectName("LoadFlow")
self.res_Q_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P1_unit.setGeometry(QtCore.QRect(170, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_unit.setFont(font)
self.res_Q_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P1_unit.setObjectName("res_Q_P1_unit")
self.res_S_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P1_DSB.setEnabled(False)
self.res_S_P1_DSB.setGeometry(QtCore.QRect(90, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
```

```
font.setBold(False)
font.setWeight(50)
self.res_S_P1_DSB.setFont(font)
self.res_S_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P1_DSB.setDecimals(3)
self.res_S_P1_DSB.setMaximum(999999.999)
self.res_S_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P1_DSB.setProperty("value", 0.0)
self.res_S_P1_DSB.setObjectName("res_S_P1_DSB")
self.res_S_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P1_unit.setGeometry(QtCore.QRect(170, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_unit.setFont(font)
self.res_S_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P1_unit.setObjectName("res_S_P1_unit")
self.res_P_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P1_DSB.setEnabled(False)
self.res_P_P1_DSB.setGeometry(QtCore.QRect(90, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_DSB.setFont(font)
self.res_P_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P1_DSB.setDecimals(3)
self.res_P_P1_DSB.setMinimum(-999999.999)
self.res_P_P1_DSB.setMaximum(999999.999)
self.res_P_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P1_DSB.setProperty("value", 0.0)
self.res_P_P1_DSB.setObjectName("res_P_P1_DSB")
self.res_I_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P1_unit.setGeometry(QtCore.QRect(170, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_unit.setFont(font)
self.res_I_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P1_unit.setObjectName("res_I_P1_unit")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(0, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_Q_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_LBL.setGeometry(QtCore.QRect(0, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Q_LBL.setFont(font)
self.res_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_LBL.setObjectName("res_Q_LBL")
self.res_Q_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P1_DSB.setEnabled(False)
self.res_Q_P1_DSB.setGeometry(QtCore.QRect(90, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_DSB.setFont(font)
self.res_Q_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P1_DSB.setDecimals(3)
self.res_Q_P1_DSB.setMinimum(-999999.999)
self.res_Q_P1_DSB.setMaximum(999999.999)
self.res_Q_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P1_DSB.setProperty("value", 0.0)
self.res_Q_P1_DSB.setObjectName("res_Q_P1_DSB")
self.res_Iangle_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Iangle_LBL.setFont(font)
self.res_Iangle_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_LBL.setObjectName("res_Iangle_LBL")
self.Por1_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Por1_LBL.setGeometry(QtCore.QRect(90, 10, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Por1_LBL.setFont(font)
self.Por1_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Por1_LBL.setObjectName("Por1_LBL")
self.res_S_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_S_LBL.setGeometry(QtCore.QRect(0, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_S_LBL.setFont(font)
self.res_S_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_LBL.setObjectName("res_S_LBL")
self.res_cosPhi_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P1_DSB.setEnabled(False)
self.res_cosPhi_P1_DSB.setGeometry(QtCore.QRect(90, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
```

```
self.res_cosPhi_P1_DSB.setFont(font)
self.res_cosPhi_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P1_DSB.setDecimals(3)
self.res_cosPhi_P1_DSB.setMaximum(999999.999)
self.res_cosPhi_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P1_DSB.setProperty("value", 0.0)
self.res_cosPhi_P1_DSB.setObjectName("res_cosPhi_P1_DSB")
self.res_U_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P1_unit.setGeometry(QtCore.QRect(170, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_unit.setFont(font)
self.res_U_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P1_unit.setObjectName("res_U_P1_unit")
self.res_cosPhi_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P1_unit.setGeometry(QtCore.QRect(170, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_unit.setFont(font)
self.res_cosPhi_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_unit.setObjectName("res_cosPhi_P1_unit")
self.res_P_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_P_LBL.setGeometry(QtCore.QRect(0, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_P_LBL.setFont(font)
self.res_P_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_LBL.setObjectName("res_P_LBL")
self.res_cosPhi_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_LBL.setGeometry(QtCore.QRect(0, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_cosPhi_LBL.setFont(font)
self.res_cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_LBL.setObjectName("res_cosPhi_LBL")
self.res_Iangle_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P1_DSB.setEnabled(False)
self.res_Iangle_P1_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_DSB.setFont(font)
self.res_Iangle_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P1_DSB.setDecimals(3)
self.res_Iangle_P1_DSB.setMinimum(-999999.999)
self.res_Iangle_P1_DSB.setMaximum(999999.999)
self.res_Iangle_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P1_DSB.setProperty("value", 0.0)
self.res_Iangle_P1_DSB.setObjectName("res_Iangle_P1_DSB")
self.res_U_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P1_DSB.setEnabled(False)
self.res_U_P1_DSB.setGeometry(QtCore.QRect(90, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_DSB.setFont(font)
self.res_U_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P1_DSB.setDecimals(3)
self.res_U_P1_DSB.setMaximum(999999.999)
self.res_U_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P1_DSB.setProperty("value", 0.0)
self.res_U_P1_DSB.setObjectName("res_U_P1_DSB")
self.res_Iangle_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P1_unit.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_unit.setFont(font)
self.res_Iangle_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_unit.setObjectName("res_Iangle_P1_unit")
self.res_P_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P1_unit.setGeometry(QtCore.QRect(170, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_unit.setFont(font)
self.res_P_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P1_unit.setObjectName("res_P_P1_unit")
self.res_I_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_I_LBL.setGeometry(QtCore.QRect(0, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_I_LBL.setFont(font)
self.res_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_LBL.setObjectName("res_I_LBL")
self.res_I_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P1_DSB.setEnabled(False)
self.res_I_P1_DSB.setGeometry(QtCore.QRect(90, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_DSB.setFont(font)
self.res_I_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
```

```
self.res_I_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P1_DSB.setDecimals(3)
self.res_I_P1_DSB.setMaximum(999999.999)
self.res_I_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P1_DSB.setProperty("value", 0.0)
self.res_I_P1_DSB.setObjectName("res_I_P1_DSB")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_U_LBL.setGeometry(QtCore.QRect(0, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.tabWidget.addTab(self.LoadFlow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.cap_pwr_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.cap_pwr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_DSB.setFont(font)
self.cap_pwr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cap_pwr_DSB.setDecimals(3)
self.cap_pwr_DSB.setMaximum(999999.999)
self.cap_pwr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cap_pwr_DSB.setProperty("value", 0.0)
self.cap_pwr_DSB.setObjectName("cap_pwr_DSB")
self.cap_pwr_unit = QtWidgets.QLabel(self.EMS)
self.cap_pwr_unit.setGeometry(QtCore.QRect(240, 10, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_unit.setFont(font)
self.cap_pwr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cap_pwr_unit.setObjectName("cap_pwr_unit")
self.cap_pwr_LBL = QtWidgets.QLabel(self.EMS)
self.cap_pwr_LBL.setGeometry(QtCore.QRect(10, 10, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.cap_pwr_LBL.setFont(font)
self.cap_pwr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_LBL.setObjectName("cap_pwr_LBL")
self.tabWidget.addTab(self.EMS, "")
self.Reliability = QtWidgets.QWidget()
self.Reliability.setObjectName("Reliability")
self.rel_results_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 65, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_alfa_unit = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_unit.setGeometry(QtCore.QRect(240, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_R_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_R_LBL.setGeometry(QtCore.QRect(160, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_beta_unit = QtWidgets.QLabel(self.Reliability)
self.rel_beta_unit.setGeometry(QtCore.QRect(240, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(0, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
self.rel_T0_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_T0_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(160, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
```

```
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(160, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.bottom_LN_3 = QtWidgets.QFrame(self.Reliability)
self.bottom_LN_3.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN_3.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_3.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_3.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_3.setObjectName("bottom_LN_3")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(160, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(240, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
```

```
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_beta_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_beta_LBL.setGeometry(QtCore.QRect(0, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(240, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_T0_unit = QtWidgets.QLabel(self.Reliability)
self.rel_T0_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(0, 170, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(160, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(160, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_anni_DSB.setEnabled(False)
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(0, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_R_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.rel_R_unit = QtWidgets.QLabel(self.Reliability)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_R_unit.setObjectName("rel_R_unit")
self.rel_lambda_unit = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
```

```
font.setBold(False)
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.rel_alfa_unit.raise_()
self.rel_R_LBL.raise_()
self.rel_beta_unit.raise_()
self.rel_alfa_LBL.raise_()
self.rel_T0_LBL.raise_()
self.rel_beta_DSB.raise_()
self.rel_alfa_DSB.raise_()
self.rel_lambda_LBL.raise_()
self.bottom_LN_3.raise_()
self.rel_Pi_Q_DSB.raise_()
self.rel_T0_DSB.raise_()
self.rel_MTBF_ore_unit.raise_()
self.rel_MTBF_ore_DSB.raise_()
self.rel_Pi_Q_unit.raise_()
self.rel_MTBF_ore_LBL.raise_()
self.rel_beta_LBL.raise_()
self.rel_Pi_E_unit.raise_()
self.rel_T0_unit.raise_()
self.rel_Pi_Q_LBL.raise_()
self.rel_MTBF_anni_LBL.raise_()
self.rel_Pi_E_DSB.raise_()
self.rel_MTBF_anni_DSB.raise_()
self.rel_Pi_E_LBL.raise_()
self.rel_lambda_LE.raise_()
self.rel_R_LE.raise_()
self.rel_R_unit.raise_()
self.rel_results_LBL.raise_()
self.rel_lambda_unit.raise_()
self.rel_MTBF_anni_unit.raise_()
self.tabWidget.addTab(self.Reliability, "")
self.Protections = QtWidgets.QWidget()
self.Protections.setObjectName("Protections")
self.prot_cal_Vmin_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_val_unit_LBL.setGeometry(QtCore.QRect(170, 210, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_val_unit_LBL.setFont(font)
self.prot_cal_Vmin_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_unit_LBL.setObjectName("prot_cal_Vmin_val_unit_LBL")
self.prot_cal_Vmin_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_LBL.setGeometry(QtCore.QRect(0, 210, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_LBL.setFont(font)
self.prot_cal_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_LBL.setObjectName("prot_cal_Vmin_LBL")
self.prot_calib_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_btm_LN.setGeometry(QtCore.QRect(10, 240, 305, 1))
self.prot_calib_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_calib_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_btm_LN.setObjectName("prot_calib_btm_LN")
self.prot_calib_LBL = QtWidgets.QLabel(self.Protections)
self.prot_calib_LBL.setGeometry(QtCore.QRect(110, 120, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_calib_LBL.setFont(font)
self.prot_calib_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_calib_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_calib_LBL.setObjectName("prot_calib_LBL")
self.prot_cal_Vmax_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_val_DSB.setGeometry(QtCore.QRect(90, 180, 71, 21))
self.prot_cal_Vmax_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_val_DSB.setDecimals(3)
self.prot_cal_Vmax_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_val_DSB.setObjectName("prot_cal_Vmax_val_DSB")
self.prot_cal_Vmin_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_pu_DSB.setGeometry(QtCore.QRect(220, 210, 71, 21))
self.prot_cal_Vmin_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_pu_DSB.setDecimals(1)
self.prot_cal_Vmin_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_pu_DSB.setObjectName("prot_cal_Vmin_pu_DSB")
self.prot_cal_Vmin_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_pu_unit_LBL.setGeometry(QtCore.QRect(300, 210, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_pu_unit_LBL.setFont(font)
self.prot_cal_Vmin_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_unit_LBL.setObjectName("prot_cal_Vmin_pu_unit_LBL")
self.prot_delay_Vmin_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_unit_LBL.setGeometry(QtCore.QRect(170, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmin_unit_LBL.setFont(font)
self.prot_delay_Vmin_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_unit_LBL.setObjectName("prot_delay_Vmin_unit_LBL")
self.prot_type_LBL = QtWidgets.QLabel(self.Protections)
self.prot_type_LBL.setGeometry(QtCore.QRect(0, 40, 81, 20))
font = QtGui.QFont()
```

```
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_type_LBL.setFont(font)
self.prot_type_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_type_LBL.setObjectName("prot_type_LBL")
self.prot_cal_Vmax_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_pu_DSB.setGeometry(QtCore.QRect(220, 180, 71, 21))
self.prot_cal_Vmax_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_pu_DSB.setDecimals(1)
self.prot_cal_Vmax_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_pu_DSB.setObjectName("prot_cal_Vmax_pu_DSB")
self.prot_cal_I_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_sep_LBL.setGeometry(QtCore.QRect(190, 150, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_sep_LBL.setFont(font)
self.prot_cal_I_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_sep_LBL.setObjectName("prot_cal_I_sep_LBL")
self.prot_cost_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cost_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cost_LBL.setFont(font)
self.prot_cost_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cost_LBL.setObjectName("prot_cost_LBL")
self.prot_delay_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_btm_LN.setGeometry(QtCore.QRect(10, 380, 305, 1))
self.prot_delay_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_btm_LN.setObjectName("prot_delay_btm_LN")
self.prot_delay_Vmin_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_LBL.setGeometry(QtCore.QRect(0, 350, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_Vmin_LBL.setFont(font)
self.prot_delay_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_LBL.setObjectName("prot_delay_Vmin_LBL")
self.prot_delay_Vmin_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmin_DSB.setGeometry(QtCore.QRect(90, 350, 71, 21))
self.prot_delay_Vmin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmin_DSB.setDecimals(3)
self.prot_delay_Vmin_DSB.setMaximum(99999999.0)
self.prot_delay_Vmin_DSB.setObjectName("prot_delay_Vmin_DSB")
self.prot_cost_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cost_unit_LBL.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cost_unit_LBL.setFont(font)
self.prot_cost_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cost_unit_LBL.setObjectName("prot_cost_unit_LBL")
self.prot_calib_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_top_LN.setGeometry(QtCore.QRect(10, 130, 305, 1))
self.prot_calib_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_calib_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_top_LN.setObjectName("prot_calib_top_LN")
self.prot_delay_I_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_I_DSB.setGeometry(QtCore.QRect(90, 290, 71, 21))
self.prot_delay_I_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_I_DSB.setDecimals(3)
self.prot_delay_I_DSB.setMaximum(99999999.0)
self.prot_delay_I_DSB.setObjectName("prot_delay_I_DSB")
self.prot_cal_Vmax_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_sep_LBL.setGeometry(QtCore.QRect(190, 180, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_sep_LBL.setFont(font)
self.prot_cal_Vmax_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_sep_LBL.setObjectName("prot_cal_Vmax_sep_LBL")
self.prot_delay_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_top_LN.setGeometry(QtCore.QRect(10, 270, 305, 1))
self.prot_delay_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_top_LN.setObjectName("prot_delay_top_LN")
self.prot_cal_Vmin_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_sep_LBL.setGeometry(QtCore.QRect(190, 210, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_sep_LBL.setFont(font)
self.prot_cal_Vmin_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_sep_LBL.setObjectName("prot_cal_Vmin_sep_LBL")
self.prot_cdelay_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cdelay_LBL.setGeometry(QtCore.QRect(110, 260, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cdelay_LBL.setFont(font)
self.prot_cdelay_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_cdelay_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_cdelay_LBL.setObjectName("prot_cdelay_LBL")
self.prot_delay_I_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_unit_LBL.setGeometry(QtCore.QRect(170, 290, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
```

```
self.prot_delay_I_unit_LBL.setFont(font)
self.prot_delay_I_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_I_unit_LBL.setObjectName("prot_delay_I_unit_LBL")
self.prot_cal_Vmax_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_val_unit_LBL.setGeometry(QtCore.QRect(170, 180, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmax_val_unit_LBL.setFont(font)
self.prot_cal_Vmax_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_val_unit_LBL.setObjectName("prot_cal_Vmax_val_unit_LBL")
self.prot_char_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_top_LN.setGeometry(QtCore.QRect(10, 20, 305, 1))
self.prot_char_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_top_LN.setObjectName("prot_char_top_LN")
self.prot_delay_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_LBL.setGeometry(QtCore.QRect(0, 320, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_Vmax_LBL.setFont(font)
self.prot_delay_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_LBL.setObjectName("prot_delay_Vmax_LBL")
self.prot_delay_Vmax_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmax_DSB.setGeometry(QtCore.QRect(90, 320, 71, 21))
self.prot_delay_Vmax_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmax_DSB.setDecimals(3)
self.prot_delay_Vmax_DSB.setMaximum(99999999.0)
self.prot_delay_Vmax_DSB.setObjectName("prot_delay_Vmax_DSB")
self.prot_cal_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_LBL.setGeometry(QtCore.QRect(0, 180, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_LBL.setFont(font)
self.prot_cal_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_LBL.setObjectName("prot_cal_Vmax_LBL")
self.prot_charact_LBL = QtWidgets.QLabel(self.Protections)
self.prot_charact_LBL.setGeometry(QtCore.QRect(60, 10, 211, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_charact_LBL.setFont(font)
self.prot_charact_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_charact_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_charact_LBL.setObjectName("prot_charact_LBL")
self.prot_cal_I_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_pu_unit_LBL.setGeometry(QtCore.QRect(300, 150, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_pu_unit_LBL.setFont(font)
self.prot_cal_I_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_unit_LBL.setObjectName("prot_cal_I_pu_unit_LBL")
self.prot_cal_Vmin_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_val_DSB.setGeometry(QtCore.QRect(90, 210, 71, 21))
self.prot_cal_Vmin_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_val_DSB.setDecimals(3)
self.prot_cal_Vmin_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_val_DSB.setObjectName("prot_cal_Vmin_val_DSB")
self.prot_cal_Vmax_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_pu_unit_LBL.setGeometry(QtCore.QRect(300, 180, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmax_pu_unit_LBL.setFont(font)
self.prot_cal_Vmax_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_unit_LBL.setObjectName("prot_cal_Vmax_pu_unit_LBL")
self.prot_cal_I_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_val_unit_LBL.setGeometry(QtCore.QRect(170, 150, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_val_unit_LBL.setFont(font)
self.prot_cal_I_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_unit_LBL.setObjectName("prot_cal_I_val_unit_LBL")
self.prot_char_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_btm_LN.setGeometry(QtCore.QRect(10, 100, 305, 1))
self.prot_char_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_btm_LN.setObjectName("prot_char_btm_LN")
self.prot_type_LE = QtWidgets.QLineEdit(self.Protections)
self.prot_type_LE.setGeometry(QtCore.QRect(90, 40, 201, 20))
self.prot_type_LE.setText("")
self.prot_type_LE.setObjectName("prot_type_LE")
self.prot_cal_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_LBL.setGeometry(QtCore.QRect(0, 150, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_LBL.setFont(font)
self.prot_cal_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_LBL.setObjectName("prot_cal_I_LBL")
self.prot_delay_Vmax_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_unit_LBL.setGeometry(QtCore.QRect(170, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmax_unit_LBL.setFont(font)
self.prot_delay_Vmax_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_unit_LBL.setObjectName("prot_delay_Vmax_unit_LBL")
```

```
self.prot_cal_I_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_val_DSB.setGeometry(QtCore.QRect(90, 150, 71, 21))
self.prot_cal_I_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_val_DSB.setDecimals(1)
self.prot_cal_I_val_DSB.setMaximum(99999999.0)
self.prot_cal_I_val_DSB.setObjectName("prot_cal_I_val_DSB")
self.prot_cost_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cost_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
self.prot_cost_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cost_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cost_DSB.setDecimals(2)
self.prot_cost_DSB.setMaximum(99999999.0)
self.prot_cost_DSB.setObjectName("prot_cost_DSB")
self.prot_delay_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_LBL.setGeometry(QtCore.QRect(0, 290, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_I_LBL.setFont(font)
self.prot_delay_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_LBL.setObjectName("prot_delay_I_LBL")
self.prot_cal_I_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_pu_DSB.setGeometry(QtCore.QRect(220, 150, 71, 21))
self.prot_cal_I_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_pu_DSB.setDecimals(1)
self.prot_cal_I_pu_DSB.setMaximum(99999999.0)
self.prot_cal_I_pu_DSB.setObjectName("prot_cal_I_pu_DSB")
self.prot_cal_Vmin_val_unit_LBL.raise_()
self.prot_cal_Vmin_LBL.raise_()
self.prot_calib_btm_LN.raise_()
self.prot_cal_Vmax_val_DSB.raise_()
self.prot_cal_Vmin_pu_DSB.raise_()
self.prot_cal_Vmin_pu_unit_LBL.raise_()
self.prot_delay_Vmin_unit_LBL.raise_()
self.prot_type_LBL.raise_()
self.prot_cal_Vmax_pu_DSB.raise_()
self.prot_cal_I_sep_LBL.raise_()
self.prot_cost_LBL.raise_()
self.prot_delay_btm_LN.raise_()
self.prot_delay_Vmin_LBL.raise_()
self.prot_delay_Vmin_DSB.raise_()
self.prot_cost_unit_LBL.raise_()
self.prot_calib_top_LN.raise_()
self.prot_delay_I_DSB.raise_()
self.prot_cal_Vmax_sep_LBL.raise_()
self.prot_delay_top_LN.raise_()
self.prot_cal_Vmin_sep_LBL.raise_()
self.prot_cdelay_LBL.raise_()
self.prot_delay_I_unit_LBL.raise_()
self.prot_cal_Vmax_val_unit_LBL.raise_()
self.prot_char_top_LN.raise_()
self.prot_delay_Vmax_LBL.raise_()
self.prot_delay_Vmax_DSB.raise_()
self.prot_cal_Vmax_LBL.raise_()
self.prot_charact_LBL.raise_()
self.prot_cal_I_pu_unit_LBL.raise_()
self.prot_cal_Vmin_val_DSB.raise_()
self.prot_cal_Vmax_pu_unit_LBL.raise_()
self.prot_cal_I_val_unit_LBL.raise_()
self.prot_char_btm_LN.raise_()
self.prot_type_LE.raise_()
self.prot_cal_I_LBL.raise_()
self.prot_delay_Vmax_unit_LBL.raise_()
self.prot_cal_I_val_DSB.raise_()
self.prot_cost_DSB.raise_()
self.prot_delay_I_LBL.raise_()
self.prot_cal_I_pu_DSB.raise_()
self.prot_calib_LBL.raise_()
self.tabWidget.addTab(self.Protections, "")
self.bb_out_LBL.raise_()
self.bb_in_LBL.raise_()
self.cub_out_LBL.raise_()
self.top_LN.raise_()
self.elem_name_LN.raise_()
self.elem_name_LBL.raise_()
self.type_LBL.raise_()
self.type_cap_LBL.raise_()
self.cub_in_LBL.raise_()
self.symbol_LBL.raise_()
self.ot_Frame_LN.raise_()
self.vdx_frame_LN.raise_()
self.vsx_frame_LN.raise_()
self.ob_frame_LN.raise_()
self.bb_in_LN.raise_()
self.bb_out_LN.raise_()
self.store_BTN.raise_()
self.cancel_BTN.raise_()
self.tabWidget.raise_()

self.retranslateUi(Form)
self.tabWidget.setCurrentIndex(4)
QtCore.QMetaObject.connectSlotsByName(Form)
Form.setTabOrder(self.tabWidget, self.sr_DSB)
Form.setTabOrder(self.sr_DSB, self.sf_DSB)
Form.setTabOrder(self.sf_DSB, self.sf_const_RB)
Form.setTabOrder(self.sf_const_RB, self.sf_profile_RB)
Form.setTabOrder(self.sf_profile_RB, self.cap_pwr_DSB)
Form.setTabOrder(self.cap_pwr_DSB, self.rel_TO_DSB)
Form.setTabOrder(self.rel_TO_DSB, self.rel_alfa_DSB)
Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_I_P1_DSB)
Form.setTabOrder(self.res_I_P1_DSB, self.res_Iangle_P1_DSB)
Form.setTabOrder(self.res_Iangle_P1_DSB, self.res_P_P1_DSB)
Form.setTabOrder(self.res_P_P1_DSB, self.res_Q_P1_DSB)
Form.setTabOrder(self.res_Q_P1_DSB, self.res_S_P1_DSB)
Form.setTabOrder(self.res_S_P1_DSB, self.res_cosPhi_P1_DSB)
Form.setTabOrder(self.res_cosPhi_P1_DSB, self.res_U_P1_DSB)
Form.setTabOrder(self.res_U_P1_DSB, self.store_BTN)
```

```
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria:"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.store_BTN.setText(_translate("Form", "Salva"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))
        self.sr_LBL.setText(_translate("Form", "Sr"))
        self.sr_unit.setText(_translate("Form", "kVA"))
        self.sf_const_RB.setText(_translate("Form", "Costante"))
        self.sf_profile_RB.setText(_translate("Form", "Profilo"))
        self.sfi_LBL.setText(_translate("Form", "scala"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Parameters), _translate("Form", "Parametri"))
        self.res_Q_P1_unit.setText(_translate("Form", "kVA"))
        self.res_S_P1_unit.setText(_translate("Form", "kVA"))
        self.res_I_P1_unit.setText(_translate("Form", "A"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMITI VIOLATI"))
        self.res_Q_LBL.setText(_translate("Form", "Q"))
        self.res_Iangle_LBL.setText(_translate("Form", "Angolo I"))
        self.Por1_LBL.setText(_translate("Form", "Nodo 1"))
        self.res_S_LBL.setText(_translate("Form", "S"))
        self.res_U_P1_unit.setText(_translate("Form", "kV"))
        self.res_cosPhi_P1_unit.setText(_translate("Form", "-"))
        self.res_P_LBL.setText(_translate("Form", "P"))
        self.res_cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.res_Iangle_P1_unit.setText(_translate("Form", "°"))
        self.res_P_P1_unit.setText(_translate("Form", "kW"))
        self.res_I_LBL.setText(_translate("Form", " I"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFlow), _translate("Form", "LoadFlow"))
        self.cap_pwr_unit.setText(_translate("Form", "kVA"))
        self.cap_pwr_LBL.setText(_translate("Form", "Potenza Nominale"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_T0_unit.setText(_translate("Form", "°C"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Reliability), _translate("Form", "Affidabilità"))
        self.prot_cal_Vmin_val_unit_LBL.setText(_translate("Form", "kV"))
        self.prot_cal_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_calib_LBL.setText(_translate("Form", "Soglie di taratura"))
        self.prot_cal_Vmin_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_delay_Vmin_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_type_LBL.setText(_translate("Form", "Tipologia"))
        self.prot_cal_I_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cost_LBL.setText(_translate("Form", "Costo"))
        self.prot_delay_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_cost_unit_LBL.setText(_translate("Form", "Euro"))
        self.prot_cal_Vmax_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cal_Vmin_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cdelay_LBL.setText(_translate("Form", "Tempi di ritardo"))
        self.prot_delay_I_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_cal_Vmax_val_unit_LBL.setText(_translate("Form", "kV"))
        self.prot_delay_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_cal_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_charact_LBL.setText(_translate("Form", "Caratterisitiche dell\'interruttore"))
        self.prot_cal_I_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_Vmax_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_I_val_unit_LBL.setText(_translate("Form", "A"))
        self.prot_cal_I_LBL.setText(_translate("Form", "Corrente"))
        self.prot_delay_Vmax_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_delay_I_LBL.setText(_translate("Form", "Corrente"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Protections), _translate("Form", "Protezioni"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.3.13 PWM

4.4.3.13.1 pwm.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .pwmUI import Ui_Form
from __shared__ import variables as v
import copy


class PWM(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(PWM, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")

        for box in ['cap_pwr', 'etainout', 'etaoutin']:
            self.ui.__getattribute__(box + '_DSB').setStyleSheet("color: rgb(127, 127, 127);")
            self.ui.__getattribute__(box + '_DSB').setEnabled(False)

        self.ui.tabWidget.setCurrentIndex(0)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])
        self.prot = copy.deepcopy(v.elements[element]['protections'])

        conn_list = list(v.elements[element]['conn'].keys())
        self.bb1 = v.elements[element]['conn']['h']
        conn_list.remove(self.bb1)
        conn_list.remove('h')
        self.bb2 = conn_list[0]

        self.cubicle1 = v.elements[self.element]['conn'][self.bb1]
        self.cubicle2 = v.elements[self.element]['conn'][self.bb2]

        self.ui.bb_in_LBL.setText(self.bb1)
        self.ui.bb_out_LBL.setText(self.bb2)

        self.ui.symbol_LBL.setPixmap(QtGui.QPixmap("_images/Elements/PWM/element.png"))
        self.switch_draw()
        self.fill()
        self.tab_activation()

        for attr in ['sr_DSB', 'Ur_DSB', 's_loss_IDLE_DSB', 'sw_loss_DSB', 'R_loss_DSB']:
            self.ui.__getattribute__(attr).valueChanged.connect(self.calculate)
        self.ui.cub_in_LBL.mouseDoubleClickEvent = self.cub1_switch
        self.ui.cub_out_LBL.mouseDoubleClickEvent = self.cub2_switch

    #
    def tab_activation(self):
        pass

    #
    def store(self):
        self.par['Sr'] = self.ui.sr_DSB.value()
        self.par['Ur'] = self.ui.Ur_DSB.value()
        self.par['S_loss_idle'] = self.ui.s_loss_IDLE_DSB.value()
        self.par['Sw_loss'] = self.ui.sw_loss_DSB.value()
        self.par['R_loss'] = self.ui.R_loss_DSB.value()
        v.elements[self.element]['parameters'] = copy.deepcopy(self.par)

        v.elements[self.element]['conn'][self.bb1] = self.cubicle1
        v.elements[self.element]['conn'][self.bb2] = self.cubicle2

        self.ems['in'] = self.bb1
        self.ems['out'] = self.bb2
        for par in ['cap_pwr', 'max_outin', 'max_inout', 'etaoutin', 'etainout']:
            self.ems[par] = self.ui.__getattribute__(par + '_DSB').value()
        v.elements[self.element]['ems'] = copy.deepcopy(self.ems)

        for par in ['TO', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)

        self.protections_par()

    #
    def protections_par(self):
        self.prot['Pn'] = self.par['Sr']
        self.prot['Vn'] = v.elements[self.bb2]['parameters']['Ur']
        self.prot['In'] = self.prot['Pn'] / self.prot['Vn']
        v.elements[self.element]['protections'] = copy.deepcopy(self.prot)

    #
    def fill(self):
        self.ui.sr_DSB.setValue(self.par['Sr'])
        self.ui.Ur_DSB.setValue(self.par['Ur'])
        self.ui.s_loss_IDLE_DSB.setValue(self.par['S_loss_idle'])
        self.ui.sw_loss_DSB.setValue(self.par['Sw_loss'])
        self.ui.R_loss_DSB.setValue(self.par['R_loss'])

        for par in ['max_outin', 'max_inout']:
            self.ui.__getattribute__(par + '_DSB').setValue(self.ems[par])

        self.calculate()

        if self.res != {}:
            self.fill_results()
        self.ui.tabWidget.setTabVisible(3, self.res != {})
        self.fill_reliability()
        if self.prot != {}:
            if self.prot['results'] != {} and v.protections:
                self.fill_protections()
```

```
#
def fill_reliability(self):
    for par in ['T0', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
        self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
    for par in self.rel['results']:
        try:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
        except:
            if self.rel['results'][par] == 0:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('0.0')
            elif self.rel['results'][par] < 0.01:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
            else:
                self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])

#
def fill_protections(self):
    self.ui.prot_type_LE.setText(self.prot['results']['type'])
    self.ui.prot_cost_DSB.setValue(self.prot['results']['cost'])
    self.ui.prot_cal_I_val_DSB.setValue(self.prot['results']['soglia_I'])
    self.ui.prot_cal_I_pu_DSB.setValue(self.prot['results']['soglia_I']/self.prot['In'] * 100)
    self.ui.prot_cal_Vmax_val_DSB.setValue(self.prot['results']['soglia_Vmax'])
    self.ui.prot_cal_Vmax_pu_DSB.setValue(self.prot['results']['soglia_Vmax'] / self.prot['Vn'] * 100)
    self.ui.prot_cal_Vmin_val_DSB.setValue(self.prot['results']['soglia_Vmin'])
    self.ui.prot_cal_Vmin_pu_DSB.setValue(self.prot['results']['soglia_Vmin'] / self.prot['Vn'] * 100)
    self.ui.prot_delay_I_DSB.setValue(self.prot['results']['delay_I'])
    self.ui.prot_delay_Vmax_DSB.setValue(self.prot['results']['delay_Vmax'])
    self.ui.prot_delay_Vmin_DSB.setValue(self.prot['results']['delay_Vmin'])

#
def fill_results(self):
    results = ['I', 'Iangle', 'P', 'Q', 'S', 'cosPhi', 'U']
    ports = ['P1', 'P2']
    ports2 = ['Port1', 'Port2']
    for port in ports:
        p = ports2[ports.index(port)]
        for result in results:
            self.ui.__getattribute__('res_' + result + '_' + port + '_DSB').setValue(self.res[p][result])
    self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])
    self.ui.res_Ploss_DSB.setValue(self.res['Ploss'])
    self.ui.res_Qloss_DSB.setValue(self.res['Qloss'])

#
def control_mode(self, ui):
    pass

#
def switch_draw(self):
    if self.cubicle1:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
    else:
        self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))
    if self.cubicle2:
        self.ui.cub_out_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
    else:
        self.ui.cub_out_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))

#
def cub1_switch(self, event):
    self.cubicle1 = not self.cubicle1
    self.switch_draw()

#
def cub2_switch(self, event):
    self.cubicle2 = not self.cubicle2
    self.switch_draw()

#
def calculate(self):
    p = self.ui.sr_DSB.value() * 1000 / 2
    u = self.ui.Ur_DSB.value() * 1000
    r_loss = self.ui.R_loss_DSB.value()
    sw_loss = self.ui.sw_loss_DSB.value() * 1000
    p_loss_idle = self.ui.s_loss_IDLE_DSB.value() * 1000
    if u > 0:
        i = p / u
    else:
        i = 0

    if p > 0:
        eta = 1 - (r_loss*i**2 + i*sw_loss + p_loss_idle) / p
    else:
        eta = 1

    self.ui.cap_pwr_DSB.setValue(self.ui.sr_DSB.value())
    for par in ['etainout', 'etaoutin']:
        self.ui.__getattribute__(par + '_DSB').setValue(eta)
```

4.4.3.13.2 pwmUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'pwmUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(491, 504)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("res/2W_Tr.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.bb_in_LBL.setFont(font)
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_out_LN.setObjectName("bb_out_LN")
        self.bb_out_LBL = QtWidgets.QLabel(self.widget)
        self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
self.bb_out_LBL.setFont(font)
self.bb_out_LBL.setStyleSheet("")
self.bb_out_LBL.setLineWidth(4)
self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.R_loss_LBL = QtWidgets.QLabel(self.Parameters)
self.R_loss_LBL.setGeometry(QtCore.QRect(60, 130, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.R_loss_LBL.setFont(font)
self.R_loss_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.R_loss_LBL.setObjectName("R_loss_LBL")
self.Ur_unit = QtWidgets.QLabel(self.Parameters)
self.Ur_unit.setGeometry(QtCore.QRect(240, 40, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.Ur_unit.setFont(font)
self.Ur_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.Ur_unit.setObjectName("Ur_unit")
self.sw_loss_unit = QtWidgets.QLabel(self.Parameters)
self.sw_loss_unit.setGeometry(QtCore.QRect(240, 100, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sw_loss_unit.setFont(font)
self.sw_loss_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.sw_loss_unit.setObjectName("sw_loss_unit")
self.s_loss_IDLE_LBL = QtWidgets.QLabel(self.Parameters)
self.s_loss_IDLE_LBL.setGeometry(QtCore.QRect(60, 70, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.s_loss_IDLE_LBL.setFont(font)
self.s_loss_IDLE_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.s_loss_IDLE_LBL.setObjectName("s_loss_IDLE_LBL")
self.Ur_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.Ur_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.Ur_DSB.setFont(font)
self.Ur_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.Ur_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.Ur_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.Ur_DSB.setDecimals(3)
self.Ur_DSB.setMaximum(999999.999)
self.Ur_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.Ur_DSB.setProperty("value", 0.0)
self.Ur_DSB.setObjectName("Ur_DSB")
self.s_loss_IDLE_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.s_loss_IDLE_DSB.setGeometry(QtCore.QRect(160, 70, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.s_loss_IDLE_DSB.setFont(font)
self.s_loss_IDLE_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.s_loss_IDLE_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.s_loss_IDLE_DSB.setDecimals(3)
self.s_loss_IDLE_DSB.setMaximum(99999.999)
self.s_loss_IDLE_DSB.setSingleStep(1.0)
self.s_loss_IDLE_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.s_loss_IDLE_DSB.setProperty("value", 0.0)
self.s_loss_IDLE_DSB.setObjectName("s_loss_IDLE_DSB")
self.sr_LBL = QtWidgets.QLabel(self.Parameters)
self.sr_LBL.setGeometry(QtCore.QRect(60, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.sr_LBL.setFont(font)
self.sr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sr_LBL.setObjectName("sr_LBL")
self.sr_unit = QtWidgets.QLabel(self.Parameters)
self.sr_unit.setGeometry(QtCore.QRect(240, 10, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sr_unit.setFont(font)
self.sr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.sr_unit.setObjectName("sr_unit")
self.sw_loss_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.sw_loss_DSB.setGeometry(QtCore.QRect(160, 100, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sw_loss_DSB.setFont(font)
self.sw_loss_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sw_loss_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.sw_loss_DSB.setDecimals(3)
self.sw_loss_DSB.setMinimum(0.0)
self.sw_loss_DSB.setMaximum(99999.999)
self.sw_loss_DSB.setSingleStep(1.0)
self.sw_loss_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.sw_loss_DSB.setProperty("value", 0.0)
```

```
self.sw_loss_DSB.setObjectName("sw_loss_DSB")
self.R_loss_unit = QtWidgets.QLabel(self.Parameters)
self.R_loss_unit.setGeometry(QtCore.QRect(240, 130, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.R_loss_unit.setFont(font)
self.R_loss_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.R_loss_unit.setObjectName("R_loss_unit")
self.R_loss_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.R_loss_DSB.setGeometry(QtCore.QRect(160, 130, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.R_loss_DSB.setFont(font)
self.R_loss_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.R_loss_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.R_loss_DSB.setDecimals(3)
self.R_loss_DSB.setMaximum(99999.999)
self.R_loss_DSB.setSingleStep(1.0)
self.R_loss_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.R_loss_DSB.setProperty("value", 0.0)
self.R_loss_DSB.setObjectName("R_loss_DSB")
self.sr_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.sr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sr_DSB.setFont(font)
self.sr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.sr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.sr_DSB.setDecimals(3)
self.sr_DSB.setMaximum(999999.999)
self.sr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.sr_DSB.setProperty("value", 0.0)
self.sr_DSB.setObjectName("sr_DSB")
self.UnLV_unit = QtWidgets.QLabel(self.Parameters)
self.UnLV_unit.setGeometry(QtCore.QRect(240, 70, 41, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.UnLV_unit.setFont(font)
self.UnLV_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.UnLV_unit.setObjectName("UnLV_unit")
self.Ur_LBL = QtWidgets.QLabel(self.Parameters)
self.Ur_LBL.setGeometry(QtCore.QRect(60, 40, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Ur_LBL.setFont(font)
self.Ur_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.Ur_LBL.setObjectName("Ur_LBL")
self.sw_loss_LBL = QtWidgets.QLabel(self.Parameters)
self.sw_loss_LBL.setGeometry(QtCore.QRect(60, 100, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.sw_loss_LBL.setFont(font)
self.sw_loss_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sw_loss_LBL.setObjectName("sw_loss_LBL")
self.tabWidget.addTab(self.Parameters, "")
self.LoadFlow = QtWidgets.QWidget()
self.LoadFlow.setObjectName("LoadFlow")
self.res_U_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P2_DSB.setEnabled(False)
self.res_U_P2_DSB.setGeometry(QtCore.QRect(220, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P2_DSB.setFont(font)
self.res_U_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P2_DSB.setDecimals(3)
self.res_U_P2_DSB.setMaximum(999999.999)
self.res_U_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P2_DSB.setProperty("value", 0.0)
self.res_U_P2_DSB.setObjectName("res_U_P2_DSB")
self.res_cosPhi_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P2_DSB.setEnabled(False)
self.res_cosPhi_P2_DSB.setGeometry(QtCore.QRect(220, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P2_DSB.setFont(font)
self.res_cosPhi_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P2_DSB.setDecimals(3)
self.res_cosPhi_P2_DSB.setMaximum(999999.999)
self.res_cosPhi_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P2_DSB.setProperty("value", 0.0)
self.res_cosPhi_P2_DSB.setObjectName("res_cosPhi_P2_DSB")
self.res_S_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P1_DSB.setEnabled(False)
self.res_S_P1_DSB.setGeometry(QtCore.QRect(90, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_DSB.setFont(font)
self.res_S_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P1_DSB.setDecimals(3)
self.res_S_P1_DSB.setMaximum(999999.999)
```

```
self.res_S_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P1_DSB.setProperty("value", 0.0)
self.res_S_P1_DSB.setObjectName("res_S_P1_DSB")
self.res_Q_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P2_DSB.setEnabled(False)
self.res_Q_P2_DSB.setGeometry(QtCore.QRect(220, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P2_DSB.setFont(font)
self.res_Q_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P2_DSB.setDecimals(3)
self.res_Q_P2_DSB.setMinimum(-999999.999)
self.res_Q_P2_DSB.setMaximum(999999.999)
self.res_Q_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P2_DSB.setProperty("value", 0.0)
self.res_Q_P2_DSB.setObjectName("res_Q_P2_DSB")
self.res_I_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P1_unit.setGeometry(QtCore.QRect(170, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_unit.setFont(font)
self.res_I_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P1_unit.setObjectName("res_I_P1_unit")
self.res_Iangle_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Iangle_LBL.setFont(font)
self.res_Iangle_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_LBL.setObjectName("res_Iangle_LBL")
self.res_U_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P2_unit.setGeometry(QtCore.QRect(300, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P2_unit.setFont(font)
self.res_U_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P2_unit.setObjectName("res_U_P2_unit")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_U_LBL.setGeometry(QtCore.QRect(0, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.res_cosPhi_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P2_unit.setGeometry(QtCore.QRect(300, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P2_unit.setFont(font)
self.res_cosPhi_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P2_unit.setObjectName("res_cosPhi_P2_unit")
self.res_Qloss_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Qloss_LBL.setGeometry(QtCore.QRect(0, 300, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Qloss_LBL.setFont(font)
self.res_Qloss_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qloss_LBL.setObjectName("res_Qloss_LBL")
self.res_Q_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P2_unit.setGeometry(QtCore.QRect(300, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P2_unit.setFont(font)
self.res_Q_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P2_unit.setObjectName("res_Q_P2_unit")
self.res_I_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_I_LBL.setGeometry(QtCore.QRect(0, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_I_LBL.setFont(font)
self.res_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_LBL.setObjectName("res_I_LBL")
self.res_P_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_P_LBL.setGeometry(QtCore.QRect(0, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_P_LBL.setFont(font)
self.res_P_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_LBL.setObjectName("res_P_LBL")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(0, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_P_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P1_unit.setGeometry(QtCore.QRect(170, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
```

```
font.setWeight(50)
self.res_P_P1_unit.setFont(font)
self.res_P_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P1_unit.setObjectName("res_P_P1_unit")
self.res_I_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P1_DSB.setEnabled(False)
self.res_I_P1_DSB.setGeometry(QtCore.QRect(90, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_DSB.setFont(font)
self.res_I_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P1_DSB.setDecimals(3)
self.res_I_P1_DSB.setMaximum(999999.999)
self.res_I_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P1_DSB.setProperty("value", 0.0)
self.res_I_P1_DSB.setObjectName("res_I_P1_DSB")
self.res_P_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P1_DSB.setEnabled(False)
self.res_P_P1_DSB.setGeometry(QtCore.QRect(90, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_DSB.setFont(font)
self.res_P_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P1_DSB.setDecimals(3)
self.res_P_P1_DSB.setMinimum(-999999.999)
self.res_P_P1_DSB.setMaximum(999999.999)
self.res_P_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P1_DSB.setProperty("value", 0.0)
self.res_P_P1_DSB.setObjectName("res_P_P1_DSB")
self.res_S_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_S_LBL.setGeometry(QtCore.QRect(0, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_S_LBL.setFont(font)
self.res_S_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_LBL.setObjectName("res_S_LBL")
self.res_Ploss_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Ploss_LBL.setGeometry(QtCore.QRect(0, 270, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Ploss_LBL.setFont(font)
self.res_Ploss_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Ploss_LBL.setObjectName("res_Ploss_LBL")
self.res_Qloss_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Qloss_DSB.setEnabled(False)
self.res_Qloss_DSB.setGeometry(QtCore.QRect(90, 300, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qloss_DSB.setFont(font)
self.res_Qloss_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Qloss_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qloss_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Qloss_DSB.setDecimals(3)
self.res_Qloss_DSB.setMaximum(999999.999)
self.res_Qloss_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Qloss_DSB.setProperty("value", 0.0)
self.res_Qloss_DSB.setObjectName("res_Qloss_DSB")
self.res_cosPhi_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_LBL.setGeometry(QtCore.QRect(0, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_cosPhi_LBL.setFont(font)
self.res_cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_LBL.setObjectName("res_cosPhi_LBL")
self.res_I_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P2_DSB.setEnabled(False)
self.res_I_P2_DSB.setGeometry(QtCore.QRect(220, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P2_DSB.setFont(font)
self.res_I_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P2_DSB.setDecimals(3)
self.res_I_P2_DSB.setMaximum(999999.999)
self.res_I_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P2_DSB.setProperty("value", 0.0)
self.res_I_P2_DSB.setObjectName("res_I_P2_DSB")
self.res_P_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P2_DSB.setEnabled(False)
self.res_P_P2_DSB.setGeometry(QtCore.QRect(220, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P2_DSB.setFont(font)
self.res_P_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P2_DSB.setDecimals(3)
self.res_P_P2_DSB.setMinimum(-999999.999)
self.res_P_P2_DSB.setMaximum(999999.999)
self.res_P_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P2_DSB.setProperty("value", 0.0)
self.res_P_P2_DSB.setObjectName("res_P_P2_DSB")
self.res_S_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P2_unit.setGeometry(QtCore.QRect(300, 160, 21, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P2_unit.setFont(font)
self.res_S_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P2_unit.setObjectName("res_S_P2_unit")
self.res_Ploss_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Ploss_DSB.setEnabled(False)
self.res_Ploss_DSB.setGeometry(QtCore.QRect(90, 270, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Ploss_DSB.setFont(font)
self.res_Ploss_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Ploss_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Ploss_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Ploss_DSB.setDecimals(3)
self.res_Ploss_DSB.setMaximum(999999.999)
self.res_Ploss_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Ploss_DSB.setProperty("value", 0.0)
self.res_Ploss_DSB.setObjectName("res_Ploss_DSB")
self.res_Iangle_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P1_unit.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_unit.setFont(font)
self.res_Iangle_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_unit.setObjectName("res_Iangle_P1_unit")
self.res_Iangle_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P2_DSB.setEnabled(False)
self.res_Iangle_P2_DSB.setGeometry(QtCore.QRect(220, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P2_DSB.setFont(font)
self.res_Iangle_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P2_DSB.setDecimals(3)
self.res_Iangle_P2_DSB.setMinimum(-999999.999)
self.res_Iangle_P2_DSB.setMaximum(999999.999)
self.res_Iangle_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P2_DSB.setProperty("value", 0.0)
self.res_Iangle_P2_DSB.setObjectName("res_Iangle_P2_DSB")
self.res_Q_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P1_unit.setGeometry(QtCore.QRect(170, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_unit.setFont(font)
self.res_Q_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P1_unit.setObjectName("res_Q_P1_unit")
self.res_Qloss_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Qloss_unit.setGeometry(QtCore.QRect(170, 300, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qloss_unit.setFont(font)
self.res_Qloss_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Qloss_unit.setObjectName("res_Qloss_unit")
self.res_Q_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P1_DSB.setEnabled(False)
self.res_Q_P1_DSB.setGeometry(QtCore.QRect(90, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_DSB.setFont(font)
self.res_Q_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P1_DSB.setDecimals(3)
self.res_Q_P1_DSB.setMinimum(-999999.999)
self.res_Q_P1_DSB.setMaximum(999999.999)
self.res_Q_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P1_DSB.setProperty("value", 0.0)
self.res_Q_P1_DSB.setObjectName("res_Q_P1_DSB")
self.res_U_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P1_DSB.setEnabled(False)
self.res_U_P1_DSB.setGeometry(QtCore.QRect(90, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_DSB.setFont(font)
self.res_U_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P1_DSB.setDecimals(3)
self.res_U_P1_DSB.setMaximum(999999.999)
self.res_U_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P1_DSB.setProperty("value", 0.0)
self.res_U_P1_DSB.setObjectName("res_U_P1_DSB")
self.res_Q_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_LBL.setGeometry(QtCore.QRect(0, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Q_LBL.setFont(font)
self.res_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_LBL.setObjectName("res_Q_LBL")
self.res_P_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P2_unit.setGeometry(QtCore.QRect(300, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
```

```
self.res_P_P2_unit.setFont(font)
self.res_P_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P2_unit.setObjectName("res_P_P2_unit")
self.res_cosPhi_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P1_unit.setGeometry(QtCore.QRect(170, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_unit.setFont(font)
self.res_cosPhi_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_unit.setObjectName("res_cosPhi_P1_unit")
self.res_Ploss_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Ploss_unit.setGeometry(QtCore.QRect(170, 270, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Ploss_unit.setFont(font)
self.res_Ploss_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Ploss_unit.setObjectName("res_Ploss_unit")
self.Por1_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Por1_LBL.setGeometry(QtCore.QRect(90, 10, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Por1_LBL.setFont(font)
self.Por1_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Por1_LBL.setObjectName("Por1_LBL")
self.Port2_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Port2_LBL.setGeometry(QtCore.QRect(220, 10, 101, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Port2_LBL.setFont(font)
self.Port2_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Port2_LBL.setObjectName("Port2_LBL")
self.res_Iangle_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P2_unit.setGeometry(QtCore.QRect(300, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P2_unit.setFont(font)
self.res_Iangle_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P2_unit.setObjectName("res_Iangle_P2_unit")
self.res_U_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P1_unit.setGeometry(QtCore.QRect(170, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_unit.setFont(font)
self.res_U_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P1_unit.setObjectName("res_U_P1_unit")
self.res_Iangle_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P1_DSB.setEnabled(False)
self.res_Iangle_P1_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_DSB.setFont(font)
self.res_Iangle_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P1_DSB.setDecimals(3)
self.res_Iangle_P1_DSB.setMinimum(-999999.999)
self.res_Iangle_P1_DSB.setMaximum(999999.999)
self.res_Iangle_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P1_DSB.setProperty("value", 0.0)
self.res_Iangle_P1_DSB.setObjectName("res_Iangle_P1_DSB")
self.res_cosPhi_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P1_DSB.setEnabled(False)
self.res_cosPhi_P1_DSB.setGeometry(QtCore.QRect(90, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_DSB.setFont(font)
self.res_cosPhi_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P1_DSB.setDecimals(3)
self.res_cosPhi_P1_DSB.setMaximum(999999.999)
self.res_cosPhi_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P1_DSB.setProperty("value", 0.0)
self.res_cosPhi_P1_DSB.setObjectName("res_cosPhi_P1_DSB")
self.res_I_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P2_unit.setGeometry(QtCore.QRect(300, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P2_unit.setFont(font)
self.res_I_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P2_unit.setObjectName("res_I_P2_unit")
self.res_S_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P2_DSB.setEnabled(False)
self.res_S_P2_DSB.setGeometry(QtCore.QRect(220, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P2_DSB.setFont(font)
self.res_S_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P2_DSB.setDecimals(3)
self.res_S_P2_DSB.setMaximum(999999.999)
self.res_S_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P2_DSB.setProperty("value", 0.0)
self.res_S_P2_DSB.setObjectName("res_S_P2_DSB")
```

```
self.res_S_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P1_unit.setGeometry(QtCore.QRect(170, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_unit.setFont(font)
self.res_S_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P1_unit.setObjectName("res_S_P1_unit")
self.tabWidget.addTab(self.LoadFlow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.cap_pwr_LBL = QtWidgets.QLabel(self.EMS)
self.cap_pwr_LBL.setGeometry(QtCore.QRect(10, 10, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.cap_pwr_LBL.setFont(font)
self.cap_pwr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_LBL.setObjectName("cap_pwr_LBL")
self.cap_pwr_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.cap_pwr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_DSB.setFont(font)
self.cap_pwr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cap_pwr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cap_pwr_DSB.setDecimals(3)
self.cap_pwr_DSB.setMaximum(999999.999)
self.cap_pwr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cap_pwr_DSB.setProperty("value", 0.0)
self.cap_pwr_DSB.setObjectName("cap_pwr_DSB")
self.cap_pwr_unit = QtWidgets.QLabel(self.EMS)
self.cap_pwr_unit.setGeometry(QtCore.QRect(240, 10, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_unit.setFont(font)
self.cap_pwr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cap_pwr_unit.setObjectName("cap_pwr_unit")
self.max_outin_unit = QtWidgets.QLabel(self.EMS)
self.max_outin_unit.setGeometry(QtCore.QRect(240, 40, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_outin_unit.setFont(font)
self.max_outin_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_outin_unit.setObjectName("max_outin_unit")
self.max_outin_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.max_outin_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_outin_DSB.setFont(font)
self.max_outin_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_outin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_outin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_outin_DSB.setDecimals(3)
self.max_outin_DSB.setMaximum(999999.999)
self.max_outin_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_outin_DSB.setProperty("value", 0.0)
self.max_outin_DSB.setObjectName("max_outin_DSB")
self.max_outin_LBL = QtWidgets.QLabel(self.EMS)
self.max_outin_LBL.setGeometry(QtCore.QRect(10, 40, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_outin_LBL.setFont(font)
self.max_outin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_outin_LBL.setObjectName("max_outin_LBL")
self.max_inout_unit = QtWidgets.QLabel(self.EMS)
self.max_inout_unit.setGeometry(QtCore.QRect(240, 70, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_inout_unit.setFont(font)
self.max_inout_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_inout_unit.setObjectName("max_inout_unit")
self.max_inout_LBL = QtWidgets.QLabel(self.EMS)
self.max_inout_LBL.setGeometry(QtCore.QRect(10, 70, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_inout_LBL.setFont(font)
self.max_inout_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_inout_LBL.setObjectName("max_inout_LBL")
self.max_inout_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.max_inout_DSB.setGeometry(QtCore.QRect(160, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_inout_DSB.setFont(font)
self.max_inout_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_inout_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_inout_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_inout_DSB.setDecimals(3)
self.max_inout_DSB.setMaximum(999999.999)
self.max_inout_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_inout_DSB.setProperty("value", 0.0)
self.max_inout_DSB.setObjectName("max_inout_DSB")
self.etainout_unit = QtWidgets.QLabel(self.EMS)
self.etainout_unit.setGeometry(QtCore.QRect(240, 130, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
```

```
font.setBold(False)
font.setWeight(50)
self.etainout_unit.setFont(font)
self.etainout_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.etainout_unit.setObjectName("etainout_unit")
self.etaoutin_LBL = QtWidgets.QLabel(self.EMS)
self.etaoutin_LBL.setGeometry(QtCore.QRect(10, 100, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.etaoutin_LBL.setFont(font)
self.etaoutin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etaoutin_LBL.setObjectName("etaoutin_LBL")
self.etainout_LBL = QtWidgets.QLabel(self.EMS)
self.etainout_LBL.setGeometry(QtCore.QRect(10, 130, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.etainout_LBL.setFont(font)
self.etainout_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etainout_LBL.setObjectName("etainout_LBL")
self.etaoutin_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.etaoutin_DSB.setGeometry(QtCore.QRect(160, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etaoutin_DSB.setFont(font)
self.etaoutin_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.etaoutin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etaoutin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.etaoutin_DSB.setDecimals(3)
self.etaoutin_DSB.setMaximum(999999.999)
self.etaoutin_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.etaoutin_DSB.setProperty("value", 0.0)
self.etaoutin_DSB.setObjectName("etaoutin_DSB")
self.etainout_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.etainout_DSB.setGeometry(QtCore.QRect(160, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etainout_DSB.setFont(font)
self.etainout_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.etainout_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etainout_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.etainout_DSB.setDecimals(3)
self.etainout_DSB.setMaximum(999999.999)
self.etainout_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.etainout_DSB.setProperty("value", 0.0)
self.etainout_DSB.setObjectName("etainout_DSB")
self.etaoutin_unit = QtWidgets.QLabel(self.EMS)
self.etaoutin_unit.setGeometry(QtCore.QRect(240, 100, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etaoutin_unit.setFont(font)
self.etaoutin_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.etaoutin_unit.setObjectName("etaoutin_unit")
self.tabWidget.addTab(self.EMS, "")
self.Reliability = QtWidgets.QWidget()
self.Reliability.setObjectName("Reliability")
self.rel_results_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 65, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_alfa_unit = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_unit.setGeometry(QtCore.QRect(240, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_R_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_R_LBL.setGeometry(QtCore.QRect(160, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_beta_unit = QtWidgets.QLabel(self.Reliability)
self.rel_beta_unit.setGeometry(QtCore.QRect(240, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(0, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
self.rel_T0_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_T0_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
```

```
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(160, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(160, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.bottom_LN_3 = QtWidgets.QFrame(self.Reliability)
self.bottom_LN_3.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN_3.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_3.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_3.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_3.setObjectName("bottom_LN_3")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(160, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(240, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
```

```
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_beta_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_beta_LBL.setGeometry(QtCore.QRect(0, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(240, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_T0_unit = QtWidgets.QLabel(self.Reliability)
self.rel_T0_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(0, 170, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(160, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(160, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_anni_DSB.setEnabled(False)
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(0, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_R_unit = QtWidgets.QLabel(self.Reliability)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_R_unit.setObjectName("rel_R_unit")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
```

```
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_R_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.rel_lambda_unit = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.rel_alfa_unit.raise_()
self.rel_R_LBL.raise_()
self.rel_beta_unit.raise_()
self.rel_alfa_LBL.raise_()
self.rel_T0_LBL.raise_()
self.rel_beta_DSB.raise_()
self.rel_alfa_DSB.raise_()
self.rel_lambda_LBL.raise_()
self.bottom_LN_3.raise_()
self.rel_Pi_Q_DSB.raise_()
self.rel_T0_DSB.raise_()
self.rel_MTBF_ore_unit.raise_()
self.rel_MTBF_ore_DSB.raise_()
self.rel_Pi_Q_unit.raise_()
self.rel_MTBF_ore_LBL.raise_()
self.rel_beta_LBL.raise_()
self.rel_Pi_E_unit.raise_()
self.rel_T0_unit.raise_()
self.rel_Pi_Q_LBL.raise_()
self.rel_MTBF_anni_LBL.raise_()
self.rel_Pi_E_DSB.raise_()
self.rel_MTBF_anni_DSB.raise_()
self.rel_Pi_E_LBL.raise_()
self.rel_R_unit.raise_()
self.rel_lambda_LE.raise_()
self.rel_R_LE.raise_()
self.rel_results_LBL.raise_()
self.rel_lambda_unit.raise_()
self.rel_MTBF_anni_unit.raise_()
self.tabWidget.addTab(self.Reliability, "")
self.Protections = QtWidgets.QWidget()
self.Protections.setObjectName("Protections")
self.bottom_LN_2 = QtWidgets.QFrame(self.Protections)
self.bottom_LN_2.setGeometry(QtCore.QRect(0, 500, 490, 1))
self.bottom_LN_2.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_2.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_2.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_2.setObjectName("bottom_LN_2")
self.prot_cal_Vmin_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_val_unit_LBL.setGeometry(QtCore.QRect(170, 210, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_val_unit_LBL.setFont(font)
self.prot_cal_Vmin_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_unit_LBL.setObjectName("prot_cal_Vmin_val_unit_LBL")
self.prot_cal_Vmin_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_LBL.setGeometry(QtCore.QRect(0, 210, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_LBL.setFont(font)
self.prot_cal_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_LBL.setObjectName("prot_cal_Vmin_LBL")
self.prot_calib_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_btm_LN.setGeometry(QtCore.QRect(10, 240, 305, 1))
self.prot_calib_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_calib_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_btm_LN.setObjectName("prot_calib_btm_LN")
self.prot_calib_LBL = QtWidgets.QLabel(self.Protections)
self.prot_calib_LBL.setGeometry(QtCore.QRect(110, 120, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_calib_LBL.setFont(font)
self.prot_calib_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_calib_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_calib_LBL.setObjectName("prot_calib_LBL")
self.prot_cal_Vmax_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_val_DSB.setGeometry(QtCore.QRect(90, 180, 71, 21))
self.prot_cal_Vmax_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_val_DSB.setDecimals(3)
self.prot_cal_Vmax_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_val_DSB.setObjectName("prot_cal_Vmax_val_DSB")
self.prot_cal_Vmin_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_pu_DSB.setGeometry(QtCore.QRect(220, 210, 71, 21))
self.prot_cal_Vmin_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_pu_DSB.setDecimals(1)
self.prot_cal_Vmin_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_pu_DSB.setObjectName("prot_cal_Vmin_pu_DSB")
self.prot_cal_Vmin_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_pu_unit_LBL.setGeometry(QtCore.QRect(300, 210, 21, 21))
font = QtGui.QFont()
```

```
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmin_pu_unit_LBL.setFont(font)
self.prot_cal_Vmin_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_pu_unit_LBL.setObjectName("prot_cal_Vmin_pu_unit_LBL")
self.prot_delay_Vmin_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_unit_LBL.setGeometry(QtCore.QRect(170, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmin_unit_LBL.setFont(font)
self.prot_delay_Vmin_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_unit_LBL.setObjectName("prot_delay_Vmin_unit_LBL")
self.prot_type_LBL = QtWidgets.QLabel(self.Protections)
self.prot_type_LBL.setGeometry(QtCore.QRect(0, 40, 81, 20))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_type_LBL.setFont(font)
self.prot_type_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_type_LBL.setObjectName("prot_type_LBL")
self.prot_cal_Vmax_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmax_pu_DSB.setGeometry(QtCore.QRect(220, 180, 71, 21))
self.prot_cal_Vmax_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmax_pu_DSB.setDecimals(1)
self.prot_cal_Vmax_pu_DSB.setMaximum(99999999.0)
self.prot_cal_Vmax_pu_DSB.setObjectName("prot_cal_Vmax_pu_DSB")
self.prot_cal_I_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_sep_LBL.setGeometry(QtCore.QRect(190, 150, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_sep_LBL.setFont(font)
self.prot_cal_I_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_sep_LBL.setObjectName("prot_cal_I_sep_LBL")
self.prot_cost_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cost_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cost_LBL.setFont(font)
self.prot_cost_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cost_LBL.setObjectName("prot_cost_LBL")
self.prot_delay_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_btm_LN.setGeometry(QtCore.QRect(10, 380, 305, 1))
self.prot_delay_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_btm_LN.setObjectName("prot_delay_btm_LN")
self.prot_delay_Vmin_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmin_LBL.setGeometry(QtCore.QRect(0, 350, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_Vmin_LBL.setFont(font)
self.prot_delay_Vmin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_LBL.setObjectName("prot_delay_Vmin_LBL")
self.prot_delay_Vmin_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmin_DSB.setGeometry(QtCore.QRect(90, 350, 71, 21))
self.prot_delay_Vmin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmin_DSB.setDecimals(3)
self.prot_delay_Vmin_DSB.setMaximum(99999999.0)
self.prot_delay_Vmin_DSB.setObjectName("prot_delay_Vmin_DSB")
self.prot_cost_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cost_unit_LBL.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cost_unit_LBL.setFont(font)
self.prot_cost_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cost_unit_LBL.setObjectName("prot_cost_unit_LBL")
self.prot_calib_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_calib_top_LN.setGeometry(QtCore.QRect(10, 130, 305, 1))
self.prot_calib_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_calib_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_calib_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_calib_top_LN.setObjectName("prot_calib_top_LN")
self.prot_delay_I_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_I_DSB.setGeometry(QtCore.QRect(90, 290, 71, 21))
self.prot_delay_I_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_I_DSB.setDecimals(3)
self.prot_delay_I_DSB.setMaximum(99999999.0)
self.prot_delay_I_DSB.setObjectName("prot_delay_I_DSB")
self.prot_cal_Vmax_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_sep_LBL.setGeometry(QtCore.QRect(190, 180, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_sep_LBL.setFont(font)
self.prot_cal_Vmax_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_sep_LBL.setObjectName("prot_cal_Vmax_sep_LBL")
self.prot_delay_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_delay_top_LN.setGeometry(QtCore.QRect(10, 270, 305, 1))
self.prot_delay_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_delay_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_delay_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_delay_top_LN.setObjectName("prot_delay_top_LN")
self.prot_cal_Vmin_sep_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmin_sep_LBL.setGeometry(QtCore.QRect(190, 210, 20, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmin_sep_LBL.setFont(font)
```

```
self.prot_cal_Vmin_sep_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_sep_LBL.setObjectName("prot_cal_Vmin_sep_LBL")
self.prot_cdelay_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cdelay_LBL.setGeometry(QtCore.QRect(110, 260, 111, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cdelay_LBL.setFont(font)
self.prot_cdelay_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_cdelay_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_cdelay_LBL.setObjectName("prot_cdelay_LBL")
self.prot_delay_I_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_unit_LBL.setGeometry(QtCore.QRect(170, 290, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_I_unit_LBL.setFont(font)
self.prot_delay_I_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_I_unit_LBL.setObjectName("prot_delay_I_unit_LBL")
self.prot_cal_Vmax_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_val_unit_LBL.setGeometry(QtCore.QRect(170, 180, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmax_val_unit_LBL.setFont(font)
self.prot_cal_Vmax_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_val_unit_LBL.setObjectName("prot_cal_Vmax_val_unit_LBL")
self.prot_char_top_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_top_LN.setGeometry(QtCore.QRect(10, 20, 305, 1))
self.prot_char_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_top_LN.setObjectName("prot_char_top_LN")
self.prot_delay_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_LBL.setGeometry(QtCore.QRect(0, 320, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_Vmax_LBL.setFont(font)
self.prot_delay_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_LBL.setObjectName("prot_delay_Vmax_LBL")
self.prot_delay_Vmax_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_delay_Vmax_DSB.setGeometry(QtCore.QRect(90, 320, 71, 21))
self.prot_delay_Vmax_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_delay_Vmax_DSB.setDecimals(3)
self.prot_delay_Vmax_DSB.setMaximum(99999999.0)
self.prot_delay_Vmax_DSB.setObjectName("prot_delay_Vmax_DSB")
self.prot_cal_Vmax_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_LBL.setGeometry(QtCore.QRect(0, 180, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_Vmax_LBL.setFont(font)
self.prot_cal_Vmax_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_LBL.setObjectName("prot_cal_Vmax_LBL")
self.prot_charact_LBL = QtWidgets.QLabel(self.Protections)
self.prot_charact_LBL.setGeometry(QtCore.QRect(60, 10, 211, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_charact_LBL.setFont(font)
self.prot_charact_LBL.setStyleSheet("background-color: rgb(0, 0,15);")
self.prot_charact_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.prot_charact_LBL.setObjectName("prot_charact_LBL")
self.prot_cal_I_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_pu_unit_LBL.setGeometry(QtCore.QRect(300, 150, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_pu_unit_LBL.setFont(font)
self.prot_cal_I_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_unit_LBL.setObjectName("prot_cal_I_pu_unit_LBL")
self.prot_cal_Vmin_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_Vmin_val_DSB.setGeometry(QtCore.QRect(90, 210, 71, 21))
self.prot_cal_Vmin_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmin_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_Vmin_val_DSB.setDecimals(3)
self.prot_cal_Vmin_val_DSB.setMaximum(99999999.0)
self.prot_cal_Vmin_val_DSB.setObjectName("prot_cal_Vmin_val_DSB")
self.prot_cal_Vmax_pu_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_Vmax_pu_unit_LBL.setGeometry(QtCore.QRect(300, 180, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_Vmax_pu_unit_LBL.setFont(font)
self.prot_cal_Vmax_pu_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_Vmax_pu_unit_LBL.setObjectName("prot_cal_Vmax_pu_unit_LBL")
self.prot_cal_I_val_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_val_unit_LBL.setGeometry(QtCore.QRect(170, 150, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_cal_I_val_unit_LBL.setFont(font)
self.prot_cal_I_val_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_unit_LBL.setObjectName("prot_cal_I_val_unit_LBL")
self.prot_char_btm_LN = QtWidgets.QFrame(self.Protections)
self.prot_char_btm_LN.setGeometry(QtCore.QRect(10, 100, 305, 1))
self.prot_char_btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
self.prot_char_btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
self.prot_char_btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
self.prot_char_btm_LN.setObjectName("prot_char_btm_LN")
self.prot_type_LE = QtWidgets.QLineEdit(self.Protections)
self.prot_type_LE.setGeometry(QtCore.QRect(90, 40, 201, 20))
self.prot_type_LE.setText("")
self.prot_type_LE.setObjectName("prot_type_LE")
```

```
self.prot_cal_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_cal_I_LBL.setGeometry(QtCore.QRect(0, 150, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_cal_I_LBL.setFont(font)
self.prot_cal_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_LBL.setObjectName("prot_cal_I_LBL")
self.prot_delay_Vmax_unit_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_Vmax_unit_LBL.setGeometry(QtCore.QRect(170, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.prot_delay_Vmax_unit_LBL.setFont(font)
self.prot_delay_Vmax_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.prot_delay_Vmax_unit_LBL.setObjectName("prot_delay_Vmax_unit_LBL")
self.prot_cal_I_val_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_val_DSB.setGeometry(QtCore.QRect(90, 150, 71, 21))
self.prot_cal_I_val_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_val_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_val_DSB.setDecimals(1)
self.prot_cal_I_val_DSB.setMaximum(99999999.0)
self.prot_cal_I_val_DSB.setObjectName("prot_cal_I_val_DSB")
self.prot_cost_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cost_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
self.prot_cost_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cost_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cost_DSB.setDecimals(2)
self.prot_cost_DSB.setMaximum(99999999.0)
self.prot_cost_DSB.setObjectName("prot_cost_DSB")
self.prot_delay_I_LBL = QtWidgets.QLabel(self.Protections)
self.prot_delay_I_LBL.setGeometry(QtCore.QRect(0, 290, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.prot_delay_I_LBL.setFont(font)
self.prot_delay_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_delay_I_LBL.setObjectName("prot_delay_I_LBL")
self.prot_cal_I_pu_DSB = QtWidgets.QDoubleSpinBox(self.Protections)
self.prot_cal_I_pu_DSB.setGeometry(QtCore.QRect(220, 150, 71, 21))
self.prot_cal_I_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.prot_cal_I_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.prot_cal_I_pu_DSB.setDecimals(1)
self.prot_cal_I_pu_DSB.setMaximum(99999999.0)
self.prot_cal_I_pu_DSB.setObjectName("prot_cal_I_pu_DSB")
self.bottom_LN_2.raise_()
self.prot_cal_Vmin_val_unit_LBL.raise_()
self.prot_cal_Vmin_LBL.raise_()
self.prot_calib_btm_LN.raise_()
self.prot_cal_Vmax_val_DSB.raise_()
self.prot_cal_Vmin_pu_DSB.raise_()
self.prot_cal_Vmin_pu_unit_LBL.raise_()
self.prot_delay_Vmin_unit_LBL.raise_()
self.prot_type_LBL.raise_()
self.prot_cal_Vmax_pu_DSB.raise_()
self.prot_cal_I_sep_LBL.raise_()
self.prot_cost_LBL.raise_()
self.prot_delay_btm_LN.raise_()
self.prot_delay_Vmin_LBL.raise_()
self.prot_delay_Vmin_DSB.raise_()
self.prot_cost_unit_LBL.raise_()
self.prot_calib_top_LN.raise_()
self.prot_delay_I_DSB.raise_()
self.prot_cal_Vmax_sep_LBL.raise_()
self.prot_delay_top_LN.raise_()
self.prot_cal_Vmin_sep_LBL.raise_()
self.prot_cdelay_LBL.raise_()
self.prot_delay_I_unit_LBL.raise_()
self.prot_cal_Vmax_val_unit_LBL.raise_()
self.prot_char_top_LN.raise_()
self.prot_delay_Vmax_LBL.raise_()
self.prot_delay_Vmax_DSB.raise_()
self.prot_cal_Vmax_LBL.raise_()
self.prot_charact_LBL.raise_()
self.prot_cal_I_pu_unit_LBL.raise_()
self.prot_cal_Vmin_val_DSB.raise_()
self.prot_cal_Vmax_pu_unit_LBL.raise_()
self.prot_cal_I_val_unit_LBL.raise_()
self.prot_char_btm_LN.raise_()
self.prot_type_LE.raise_()
self.prot_cal_I_LBL.raise_()
self.prot_delay_Vmax_unit_LBL.raise_()
self.prot_cal_I_val_DSB.raise_()
self.prot_cost_DSB.raise_()
self.prot_delay_I_LBL.raise_()
self.prot_cal_I_pu_DSB.raise_()
self.prot_calib_LBL.raise_()
self.tabWidget.addTab(self.Protections, "")
self.cancel_BTN = QtWidgets.QPushButton(self.widget)
self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.cancel_BTN.setFont(font)
self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.cancel_BTN.setObjectName("cancel_BTN")
self.store_BTN = QtWidgets.QPushButton(self.widget)
self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.store_BTN.setFont(font)
self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.store_BTN.setObjectName("store_BTN")
self.bb_out_LBL.raise_()
self.bb_in_LBL.raise_()
self.cub_out_LBL.raise_()
self.top_LN.raise_()
self.elem_name_LN.raise_()
self.elem_name_LBL.raise_()
self.type_LBL.raise_()
self.type_cap_LBL.raise_()
self.cub_in_LBL.raise_()
self.symbol_LBL.raise_()
```

```
        self.ot_Frame_LN.raise_()
        self.vdx_frame_LN.raise_()
        self.vsx_frame_LN.raise_()
        self.ob_frame_LN.raise_()
        self.bb_in_LN.raise_()
        self.bb_out_LN.raise_()
        self.tabWidget.raise_()
        self.cancel_BTN.raise_()
        self.store_BTN.raise_()

        self.retranslateUi(Form)
        self.tabWidget.setCurrentIndex(4)
        QtCore.QMetaObject.connectSlotsByName(Form)
        Form.setTabOrder(self.tabWidget, self.sr_DSB)
        Form.setTabOrder(self.sr_DSB, self.Ur_DSB)
        Form.setTabOrder(self.Ur_DSB, self.s_loss_IDLE_DSB)
        Form.setTabOrder(self.s_loss_IDLE_DSB, self.sw_loss_DSB)
        Form.setTabOrder(self.sw_loss_DSB, self.R_loss_DSB)
        Form.setTabOrder(self.R_loss_DSB, self.cap_pwr_DSB)
        Form.setTabOrder(self.cap_pwr_DSB, self.max_outin_DSB)
        Form.setTabOrder(self.max_outin_DSB, self.max_inout_DSB)
        Form.setTabOrder(self.max_inout_DSB, self.etaoutin_DSB)
        Form.setTabOrder(self.etaoutin_DSB, self.etainout_DSB)
        Form.setTabOrder(self.etainout_DSB, self.rel_T0_DSB)
        Form.setTabOrder(self.rel_T0_DSB, self.rel_alfa_DSB)
        Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
        Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
        Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
        Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
        Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
        Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_I_P1_DSB)
        Form.setTabOrder(self.res_I_P1_DSB, self.res_Iangle_P1_DSB)
        Form.setTabOrder(self.res_Iangle_P1_DSB, self.res_P_P1_DSB)
        Form.setTabOrder(self.res_P_P1_DSB, self.res_Q_P1_DSB)
        Form.setTabOrder(self.res_Q_P1_DSB, self.res_S_P1_DSB)
        Form.setTabOrder(self.res_S_P1_DSB, self.res_cosPhi_P1_DSB)
        Form.setTabOrder(self.res_cosPhi_P1_DSB, self.res_U_P1_DSB)
        Form.setTabOrder(self.res_U_P1_DSB, self.res_I_P2_DSB)
        Form.setTabOrder(self.res_I_P2_DSB, self.res_Iangle_P2_DSB)
        Form.setTabOrder(self.res_Iangle_P2_DSB, self.res_P_P2_DSB)
        Form.setTabOrder(self.res_P_P2_DSB, self.res_Q_P2_DSB)
        Form.setTabOrder(self.res_Q_P2_DSB, self.res_S_P2_DSB)
        Form.setTabOrder(self.res_S_P2_DSB, self.res_cosPhi_P2_DSB)
        Form.setTabOrder(self.res_cosPhi_P2_DSB, self.res_U_P2_DSB)
        Form.setTabOrder(self.res_U_P2_DSB, self.res_Ploss_DSB)
        Form.setTabOrder(self.res_Ploss_DSB, self.res_Qloss_DSB)
        Form.setTabOrder(self.res_Qloss_DSB, self.store_BTN)
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria:"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.R_loss_LBL.setText(_translate("Form", "R Loss"))
        self.Ur_unit.setText(_translate("Form", "kV"))
        self.sw_loss_unit.setText(_translate("Form", "kW/A"))
        self.s_loss_IDLE_LBL.setText(_translate("Form", "S loss IDLE"))
        self.sr_LBL.setText(_translate("Form", "Sr"))
        self.sr_unit.setText(_translate("Form", "kVA"))
        self.R_loss_unit.setText(_translate("Form", "Ohm"))
        self.UnLV_unit.setText(_translate("Form", "kW"))
        self.Ur_LBL.setText(_translate("Form", "Ur"))
        self.sw_loss_LBL.setText(_translate("Form", "Sw Loss"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Parameters), _translate("Form", "Parametri"))
        self.res_I_P1_unit.setText(_translate("Form", "A"))
        self.res_Iangle_LBL.setText(_translate("Form", "Angolo I"))
        self.res_U_P2_unit.setText(_translate("Form", "kV"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.res_cosPhi_P2_unit.setText(_translate("Form", "-"))
        self.res_Qloss_LBL.setText(_translate("Form", "Q loss"))
        self.res_Q_P2_unit.setText(_translate("Form", "kVA"))
        self.res_I_LBL.setText(_translate("Form", " I"))
        self.res_P_LBL.setText(_translate("Form", "P"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMITI VIOLATI"))
        self.res_P_P1_unit.setText(_translate("Form", "kW"))
        self.res_S_LBL.setText(_translate("Form", "S"))
        self.res_Ploss_LBL.setText(_translate("Form", "P loss"))
        self.res_cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.res_S_P2_unit.setText(_translate("Form", "kVA"))
        self.res_Iangle_P1_unit.setText(_translate("Form", "°"))
        self.res_Q_P1_unit.setText(_translate("Form", "kVA"))
        self.res_Qloss_unit.setText(_translate("Form", "kVA"))
        self.res_Q_LBL.setText(_translate("Form", "Q"))
        self.res_P_P2_unit.setText(_translate("Form", "kW"))
        self.res_cosPhi_P1_unit.setText(_translate("Form", "-"))
        self.res_Ploss_unit.setText(_translate("Form", "kW"))
        self.Port1_LBL.setText(_translate("Form", "Port 1"))
        self.Port2_LBL.setText(_translate("Form", "Port 2"))
        self.res_Iangle_P2_unit.setText(_translate("Form", "°"))
        self.res_U_P1_unit.setText(_translate("Form", "kV"))
        self.res_I_P2_unit.setText(_translate("Form", "A"))
        self.res_S_P1_unit.setText(_translate("Form", "kVA"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFlow), _translate("Form", "LoadFlow"))
        self.cap_pwr_LBL.setText(_translate("Form", "Potenza Nominale"))
        self.cap_pwr_unit.setText(_translate("Form", "kVA"))
        self.max_outin_unit.setText(_translate("Form", "p.u."))
        self.max_outin_LBL.setText(_translate("Form", "Potenza Massima IN"))
        self.max_inout_unit.setText(_translate("Form", "p.u."))
        self.max_inout_LBL.setText(_translate("Form", "Potenza Massima OUT"))
        self.etainout_unit.setText(_translate("Form", "p.u."))
        self.etaoutin_LBL.setText(_translate("Form", "Efficienza OUT-to-IN"))
        self.etainout_LBL.setText(_translate("Form", "Efficienza IN-to-OUT "))
        self.etaoutin_unit.setText(_translate("Form", "p.u."))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
```

```
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_T0_unit.setText(_translate("Form", "°C"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Reliability), _translate("Form", "Affidabilità"))
        self.prot_cal_Vmin_val_unit_LBL.setText(_translate("Form", "kV"))
        self.prot_cal_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_calib_LBL.setText(_translate("Form", "Soglie di taratura"))
        self.prot_cal_Vmin_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_delay_Vmin_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_type_LBL.setText(_translate("Form", "Tipologia"))
        self.prot_cal_I_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cost_LBL.setText(_translate("Form", "Costo"))
        self.prot_delay_Vmin_LBL.setText(_translate("Form", "Tensione min"))
        self.prot_cost_unit_LBL.setText(_translate("Form", "Euro"))
        self.prot_cal_Vmax_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cal_Vmin_sep_LBL.setText(_translate("Form", "-"))
        self.prot_cdelay_LBL.setText(_translate("Form", "Tempi di ritardo"))
        self.prot_delay_I_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_cal_Vmax_val_unit_LBL.setText(_translate("Form", "kV"))
        self.prot_delay_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_cal_Vmax_LBL.setText(_translate("Form", "Tensione max"))
        self.prot_charact_LBL.setText(_translate("Form", "Caratterisitiche dell\'interruttore"))
        self.prot_cal_I_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_Vmax_pu_unit_LBL.setText(_translate("Form", "%"))
        self.prot_cal_I_val_unit_LBL.setText(_translate("Form", "A"))
        self.prot_cal_I_LBL.setText(_translate("Form", "Corrente"))
        self.prot_delay_Vmax_unit_LBL.setText(_translate("Form", "ms"))
        self.prot_delay_I_LBL.setText(_translate("Form", "Corrente"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Protections), _translate("Form", "Protezioni"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))
        self.store_BTN.setText(_translate("Form", "Salva"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.3.14 Transformers_TwoWings

4.4.3.14.1 transformer_TwoWings.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .transformer_TwoWingsUI import Ui_Form
from __shared__ import variables as v
import copy


class Tr2W(QtWidgets.QMainWindow):
    def __init__(self, element):
        super(Tr2W, self).__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.tabWidget.setStyleSheet("QTabBar::tab {background-color: rgb(0, 0, 15);} "
                                        "QTabBar::tab:selected {background-color: rgb(85, 85, 127);}")

        self.ui.cap_pwr_DSB.setStyleSheet("color: rgb(127, 127, 127);")
        self.ui.cap_pwr_DSB.setEnabled(False)

        for box in ['cap_pwr', 'etainout', 'etaoutin']:
            self.ui.__getattribute__(box + '_DSB').setStyleSheet("color: rgb(127, 127, 127);")
            self.ui.__getattribute__(box + '_DSB').setEnabled(False)

        self.ui.tabWidget.setCurrentIndex(0)

        self.element = element
        self.ui.elem_name_LBL.setText(element)
        self.ui.type_LBL.setText(v.elements[element]['category'])
        self.par = copy.deepcopy(v.elements[element]['parameters'])
        self.ems = copy.deepcopy(v.elements[element]['ems'])
        self.rel = copy.deepcopy(v.elements[element]['reliability'])
        self.res = copy.deepcopy(v.elements[element]['results'])

        conn_list = list(v.elements[element]['conn'].keys())
        self.bb1 = v.elements[element]['conn']['h']
        conn_list.remove(self.bb1)
        conn_list.remove('h')
        self.bb2 = conn_list[0]

        self.cubicle1 = v.elements[self.element]['conn'][self.bb1]
        self.cubicle2 = v.elements[self.element]['conn'][self.bb2]

        self.ui.bb_in_LBL.setText(self.bb1)
        self.ui.bb_out_LBL.setText(self.bb2)

        self.ui.symbol_LBL.setPixmap(QtGui.QPixmap("_images/Elements/Transformers_TwoWings/element.png"))
        self.switch_draw()
        self.fill()

        for attr in ['sr', 'URr1']:
            self.ui.__getattribute__(attr + '_DSB').valueChanged.connect(self.calculate)
        self.ui.cub_in_LBL.mouseDoubleClickEvent = self.cub1_switch
        self.ui.cub_out_LBL.mouseDoubleClickEvent = self.cub2_switch

    #
    def store(self):
        self.par['Sr'] = self.ui.sr_DSB.value()
        self.par['UnHV'] = self.ui.UnHV_DSB.value()
        self.par['UnLV'] = self.ui.UnLV_DSB.value()
        self.par['URr1'] = self.ui.URr1_DSB.value()
        self.par['Ukr1'] = self.ui.Ukr1_DSB.value()
        self.par['URr0'] = self.ui.URr0_DSB.value()
        self.par['Ukr0'] = self.ui.Ukr0_DSB.value()
        v.elements[self.element]['parameters'] = copy.deepcopy(self.par)

        v.elements[self.element]['conn'][self.bb1] = self.cubicle1
        v.elements[self.element]['conn'][self.bb2] = self.cubicle2

        self.ems['in'] = self.bb1
        self.ems['out'] = self.bb2
        for par in ['cap_pwr', 'max_outin', 'max_inout', 'etaoutin', 'etainout']:
            self.ems[par] = self.ui.__getattribute__(par + '_DSB').value()
        v.elements[self.element]['ems'] = copy.deepcopy(self.ems)

        for par in ['T0', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.rel[par] = self.ui.__getattribute__('rel_' + par + '_DSB').value()
        v.elements[self.element]['reliability'] = copy.deepcopy(self.rel)

    #
    def fill(self):
        self.ui.sr_DSB.setValue(self.par['Sr'])
        self.ui.UnHV_DSB.setValue(self.par['UnHV'])
        self.ui.UnLV_DSB.setValue(self.par['UnLV'])
        self.ui.URr1_DSB.setValue(self.par['URr1'])
        self.ui.Ukr1_DSB.setValue(self.par['Ukr1'])
        self.ui.URr0_DSB.setValue(self.par['URr0'])
        self.ui.Ukr0_DSB.setValue(self.par['Ukr0'])

        for par in ['max_outin', 'max_inout']:
            self.ui.__getattribute__(par + '_DSB').setValue(self.ems[par])

        self.calculate()

        if self.res != {}:
            self.fill_results()
        self.ui.tabWidget.setTabVisible(3, self.res != {})
        self.fill_reliability()

    #
    def fill_reliability(self):
        for par in ['T0', 'alfa', 'beta', 'Pi_E', 'Pi_Q']:
            self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel[par])
        for par in self.rel['results']:
            try:
                self.ui.__getattribute__('rel_' + par + '_DSB').setValue(self.rel['results'][par])
            except:
                if self.rel['results'][par] < 0.01:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('%.3E' % self.rel['results'][par])
```

```
                elif self.rel['results'][par] == 0:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('0')
                else:
                    self.ui.__getattribute__('rel_' + par + '_LE').setText('%.6f' % self.rel['results'][par])

    #
    def fill_results(self):
        results = ['I', 'Iangle', 'P', 'Q', 'S', 'cosPhi', 'U']
        ports = ['P1', 'P2']
        ports2 = ['Port1', 'Port2']
        for port in ports:
            p = ports2[ports.index(port)]
            for result in results:
                self.ui.__getattribute__('res_' + result + '_' + port + '_DSB').setValue(self.res[p][result])
        self.ui.res_LimViolated_LBL.setVisible(self.res['LimitViolated'])
        self.ui.res_Ploss_DSB.setValue(self.res['Ploss'])
        self.ui.res_Qloss_DSB.setValue(self.res['Qloss'])

    #
    def control_mode(self, ui):
        pass

#
    def switch_draw(self):
        if self.cubicle1:
            self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
        else:
            self.ui.cub_in_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))
        if self.cubicle2:
            self.ui.cub_out_LBL.setPixmap(QtGui.QPixmap("_images/Elements/closed.png"))
        else:
            self.ui.cub_out_LBL.setPixmap(QtGui.QPixmap("_images/Elements/opened.png"))

    #
    def cub1_switch(self, event):
        self.cubicle1 = not self.cubicle1
        self.switch_draw()

    #
    def cub2_switch(self, event):
        self.cubicle2 = not self.cubicle2
        self.switch_draw()

    #
    def calculate(self):
        urr = self.ui.URr1_DSB.value() / 100

        eta = 1 - (3**0.5) * urr

        self.ui.cap_pwr_DSB.setValue(self.ui.sr_DSB.value())
        for par in ['etainout', 'etaoutin']:
            self.ui.__getattribute__(par + '_DSB').setValue(eta)
```

4.4.3.14.2 transformer_TwoWingsUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'transformer_TwoWingsUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(495, 505)
        self.widget = QtWidgets.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(0, 0, 491, 501))
        self.widget.setStyleSheet("background-color: rgb(0, 0,15);\n"
"color: rgb(255, 255, 255);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 490, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.elem_name_LN = QtWidgets.QFrame(self.widget)
        self.elem_name_LN.setGeometry(QtCore.QRect(10, 30, 331, 1))
        self.elem_name_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.elem_name_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.elem_name_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.elem_name_LN.setObjectName("elem_name_LN")
        self.elem_name_LBL = QtWidgets.QLabel(self.widget)
        self.elem_name_LBL.setGeometry(QtCore.QRect(20, 10, 281, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.elem_name_LBL.setFont(font)
        self.elem_name_LBL.setObjectName("elem_name_LBL")
        self.type_LBL = QtWidgets.QLabel(self.widget)
        self.type_LBL.setGeometry(QtCore.QRect(100, 40, 201, 21))
        self.type_LBL.setObjectName("type_LBL")
        self.type_cap_LBL = QtWidgets.QLabel(self.widget)
        self.type_cap_LBL.setGeometry(QtCore.QRect(30, 40, 61, 21))
        font = QtGui.QFont()
        font.setItalic(True)
        self.type_cap_LBL.setFont(font)
        self.type_cap_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.type_cap_LBL.setObjectName("type_cap_LBL")
        self.cub_in_LBL = QtWidgets.QLabel(self.widget)
        self.cub_in_LBL.setGeometry(QtCore.QRect(360, 30, 121, 25))
        self.cub_in_LBL.setText("")
        self.cub_in_LBL.setPixmap(QtGui.QPixmap("res/opened.png"))
        self.cub_in_LBL.setObjectName("cub_in_LBL")
        self.cub_out_LBL = QtWidgets.QLabel(self.widget)
        self.cub_out_LBL.setGeometry(QtCore.QRect(360, 126, 121, 25))
        self.cub_out_LBL.setText("")
        self.cub_out_LBL.setPixmap(QtGui.QPixmap("res/closed.png"))
        self.cub_out_LBL.setObjectName("cub_out_LBL")
        self.symbol_LBL = QtWidgets.QLabel(self.widget)
        self.symbol_LBL.setGeometry(QtCore.QRect(360, 55, 121, 71))
        self.symbol_LBL.setText("")
        self.symbol_LBL.setPixmap(QtGui.QPixmap("res/2W_Tr.png"))
        self.symbol_LBL.setObjectName("symbol_LBL")
        self.ot_Frame_LN = QtWidgets.QFrame(self.widget)
        self.ot_Frame_LN.setGeometry(QtCore.QRect(360, 10, 121, 1))
        self.ot_Frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ot_Frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ot_Frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ot_Frame_LN.setObjectName("ot_Frame_LN")
        self.vdx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vdx_frame_LN.setGeometry(QtCore.QRect(480, 10, 1, 161))
        self.vdx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vdx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vdx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vdx_frame_LN.setObjectName("vdx_frame_LN")
        self.vsx_frame_LN = QtWidgets.QFrame(self.widget)
        self.vsx_frame_LN.setGeometry(QtCore.QRect(360, 10, 1, 161))
        self.vsx_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.vsx_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.vsx_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vsx_frame_LN.setObjectName("vsx_frame_LN")
        self.ob_frame_LN = QtWidgets.QFrame(self.widget)
        self.ob_frame_LN.setGeometry(QtCore.QRect(360, 171, 121, 1))
        self.ob_frame_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.ob_frame_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.ob_frame_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.ob_frame_LN.setObjectName("ob_frame_LN")
        self.bb_in_LBL = QtWidgets.QLabel(self.widget)
        self.bb_in_LBL.setGeometry(QtCore.QRect(365, 11, 111, 21))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.bb_in_LBL.setFont(font)
        self.bb_in_LBL.setStyleSheet("")
        self.bb_in_LBL.setLineWidth(4)
        self.bb_in_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.bb_in_LBL.setObjectName("bb_in_LBL")
        self.bb_in_LN = QtWidgets.QFrame(self.widget)
        self.bb_in_LN.setGeometry(QtCore.QRect(370, 30, 101, 3))
        self.bb_in_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_in_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_in_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_in_LN.setObjectName("bb_in_LN")
        self.bb_out_LN = QtWidgets.QFrame(self.widget)
        self.bb_out_LN.setGeometry(QtCore.QRect(370, 150, 101, 3))
        self.bb_out_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.bb_out_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.bb_out_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.bb_out_LN.setObjectName("bb_out_LN")
        self.bb_out_LBL = QtWidgets.QLabel(self.widget)
        self.bb_out_LBL.setGeometry(QtCore.QRect(365, 150, 111, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
self.bb_out_LBL.setFont(font)
self.bb_out_LBL.setStyleSheet("")
self.bb_out_LBL.setLineWidth(4)
self.bb_out_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.bb_out_LBL.setObjectName("bb_out_LBL")
self.cancel_BTN = QtWidgets.QPushButton(self.widget)
self.cancel_BTN.setGeometry(QtCore.QRect(360, 430, 121, 23))
font = QtGui.QFont()
font.setPointSize(8)
self.cancel_BTN.setFont(font)
self.cancel_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
self.cancel_BTN.setObjectName("cancel_BTN")
self.tabWidget = QtWidgets.QTabWidget(self.widget)
self.tabWidget.setGeometry(QtCore.QRect(10, 80, 331, 411))
font = QtGui.QFont()
font.setPointSize(10)
self.tabWidget.setFont(font)
self.tabWidget.setStyleSheet("")
self.tabWidget.setObjectName("tabWidget")
self.Parameters = QtWidgets.QWidget()
self.Parameters.setObjectName("Parameters")
self.URr0_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.URr0_DSB.setGeometry(QtCore.QRect(160, 160, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.URr0_DSB.setFont(font)
self.URr0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.URr0_DSB.setReadOnly(False)
self.URr0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.URr0_DSB.setDecimals(3)
self.URr0_DSB.setMaximum(100.0)
self.URr0_DSB.setSingleStep(0.1)
self.URr0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.URr0_DSB.setProperty("value", 0.0)
self.URr0_DSB.setObjectName("URr0_DSB")
self.URr1LBL = QtWidgets.QLabel(self.Parameters)
self.URr1LBL.setGeometry(QtCore.QRect(60, 100, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.URr1LBL.setFont(font)
self.URr1LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.URr1LBL.setObjectName("URr1LBL")
self.UnLV_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.UnLV_DSB.setGeometry(QtCore.QRect(160, 70, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.UnLV_DSB.setFont(font)
self.UnLV_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.UnLV_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.UnLV_DSB.setDecimals(3)
self.UnLV_DSB.setMaximum(999999.999)
self.UnLV_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.UnLV_DSB.setProperty("value", 0.0)
self.UnLV_DSB.setObjectName("UnLV_DSB")
self.Ukr1_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.Ukr1_DSB.setGeometry(QtCore.QRect(160, 130, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.Ukr1_DSB.setFont(font)
self.Ukr1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.Ukr1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.Ukr1_DSB.setDecimals(3)
self.Ukr1_DSB.setMaximum(100.0)
self.Ukr1_DSB.setSingleStep(0.1)
self.Ukr1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.Ukr1_DSB.setProperty("value", 0.0)
self.Ukr1_DSB.setObjectName("Ukr1_DSB")
self.Ukr1_LBL = QtWidgets.QLabel(self.Parameters)
self.Ukr1_LBL.setGeometry(QtCore.QRect(60, 130, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Ukr1_LBL.setFont(font)
self.Ukr1_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.Ukr1_LBL.setObjectName("Ukr1_LBL")
self.sr_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.sr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sr_DSB.setFont(font)
self.sr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.sr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.sr_DSB.setDecimals(3)
self.sr_DSB.setMaximum(999999.999)
self.sr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.sr_DSB.setProperty("value", 0.0)
self.sr_DSB.setObjectName("sr_DSB")
self.UnLV_LBL = QtWidgets.QLabel(self.Parameters)
self.UnLV_LBL.setGeometry(QtCore.QRect(60, 70, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.UnLV_LBL.setFont(font)
self.UnLV_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.UnLV_LBL.setObjectName("UnLV_LBL")
self.Ukr1_unit = QtWidgets.QLabel(self.Parameters)
self.Ukr1_unit.setGeometry(QtCore.QRect(240, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
```

```
font.setWeight(50)
self.Ukr1_unit.setFont(font)
self.Ukr1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.Ukr1_unit.setObjectName("Ukr1_unit")
self.URr0_LBL = QtWidgets.QLabel(self.Parameters)
self.URr0_LBL.setGeometry(QtCore.QRect(60, 160, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.URr0_LBL.setFont(font)
self.URr0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.URr0_LBL.setObjectName("URr0_LBL")
self.Ukr0_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.Ukr0_DSB.setGeometry(QtCore.QRect(160, 190, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.Ukr0_DSB.setFont(font)
self.Ukr0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.Ukr0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.Ukr0_DSB.setDecimals(3)
self.Ukr0_DSB.setMaximum(100.0)
self.Ukr0_DSB.setSingleStep(0.1)
self.Ukr0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.Ukr0_DSB.setProperty("value", 0.0)
self.Ukr0_DSB.setObjectName("Ukr0_DSB")
self.UnLV_unit = QtWidgets.QLabel(self.Parameters)
self.UnLV_unit.setGeometry(QtCore.QRect(240, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.UnLV_unit.setFont(font)
self.UnLV_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.UnLV_unit.setObjectName("UnLV_unit")
self.Ukr0_unit = QtWidgets.QLabel(self.Parameters)
self.Ukr0_unit.setGeometry(QtCore.QRect(240, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.Ukr0_unit.setFont(font)
self.Ukr0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.Ukr0_unit.setObjectName("Ukr0_unit")
self.UnHV_LBL = QtWidgets.QLabel(self.Parameters)
self.UnHV_LBL.setGeometry(QtCore.QRect(60, 40, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.UnHV_LBL.setFont(font)
self.UnHV_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.UnHV_LBL.setObjectName("UnHV_LBL")
self.UnHV_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.UnHV_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.UnHV_DSB.setFont(font)
self.UnHV_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.UnHV_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.UnHV_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.UnHV_DSB.setDecimals(3)
self.UnHV_DSB.setMaximum(999999.999)
self.UnHV_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.UnHV_DSB.setProperty("value", 0.0)
self.UnHV_DSB.setObjectName("UnHV_DSB")
self.sr_unit = QtWidgets.QLabel(self.Parameters)
self.sr_unit.setGeometry(QtCore.QRect(240, 10, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.sr_unit.setFont(font)
self.sr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.sr_unit.setObjectName("sr_unit")
self.URr0_unit = QtWidgets.QLabel(self.Parameters)
self.URr0_unit.setGeometry(QtCore.QRect(240, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.URr0_unit.setFont(font)
self.URr0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.URr0_unit.setObjectName("URr0_unit")
self.UnHV_unit = QtWidgets.QLabel(self.Parameters)
self.UnHV_unit.setGeometry(QtCore.QRect(240, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.UnHV_unit.setFont(font)
self.UnHV_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.UnHV_unit.setObjectName("UnHV_unit")
self.URr1_DSB = QtWidgets.QDoubleSpinBox(self.Parameters)
self.URr1_DSB.setGeometry(QtCore.QRect(160, 100, 71, 22))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.URr1_DSB.setFont(font)
self.URr1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.URr1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.URr1_DSB.setDecimals(3)
self.URr1_DSB.setMinimum(0.0)
self.URr1_DSB.setMaximum(100.0)
self.URr1_DSB.setSingleStep(0.1)
self.URr1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.URr1_DSB.setProperty("value", 0.0)
self.URr1_DSB.setObjectName("URr1_DSB")
self.Ukr0_LBL = QtWidgets.QLabel(self.Parameters)
self.Ukr0_LBL.setGeometry(QtCore.QRect(60, 190, 91, 21))
```

```
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.UkrO_LBL.setFont(font)
self.UkrO_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.UkrO_LBL.setObjectName("UkrO_LBL")
self.sr_LBL = QtWidgets.QLabel(self.Parameters)
self.sr_LBL.setGeometry(QtCore.QRect(60, 10, 91, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.sr_LBL.setFont(font)
self.sr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.sr_LBL.setObjectName("sr_LBL")
self.URr1_unit = QtWidgets.QLabel(self.Parameters)
self.URr1_unit.setGeometry(QtCore.QRect(240, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.URr1_unit.setFont(font)
self.URr1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.URr1_unit.setObjectName("URr1_unit")
self.tabWidget.addTab(self.Parameters, "")
self.LoadFlow = QtWidgets.QWidget()
self.LoadFlow.setObjectName("LoadFlow")
self.res_cosPhi_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P1_DSB.setEnabled(False)
self.res_cosPhi_P1_DSB.setGeometry(QtCore.QRect(90, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_DSB.setFont(font)
self.res_cosPhi_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P1_DSB.setDecimals(3)
self.res_cosPhi_P1_DSB.setMaximum(999999.999)
self.res_cosPhi_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P1_DSB.setProperty("value", 0.0)
self.res_cosPhi_P1_DSB.setObjectName("res_cosPhi_P1_DSB")
self.res_cosPhi_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_LBL.setGeometry(QtCore.QRect(0, 190, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_cosPhi_LBL.setFont(font)
self.res_cosPhi_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_LBL.setObjectName("res_cosPhi_LBL")
self.res_P_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P2_DSB.setEnabled(False)
self.res_P_P2_DSB.setGeometry(QtCore.QRect(220, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P2_DSB.setFont(font)
self.res_P_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P2_DSB.setDecimals(3)
self.res_P_P2_DSB.setMinimum(-999999.999)
self.res_P_P2_DSB.setMaximum(999999.999)
self.res_P_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P2_DSB.setProperty("value", 0.0)
self.res_P_P2_DSB.setObjectName("res_P_P2_DSB")
self.res_Qloss_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Qloss_LBL.setGeometry(QtCore.QRect(0, 300, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Qloss_LBL.setFont(font)
self.res_Qloss_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qloss_LBL.setObjectName("res_Qloss_LBL")
self.res_Iangle_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P1_DSB.setEnabled(False)
self.res_Iangle_P1_DSB.setGeometry(QtCore.QRect(90, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_DSB.setFont(font)
self.res_Iangle_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P1_DSB.setDecimals(3)
self.res_Iangle_P1_DSB.setMinimum(-999999.999)
self.res_Iangle_P1_DSB.setMaximum(999999.999)
self.res_Iangle_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P1_DSB.setProperty("value", 0.0)
self.res_Iangle_P1_DSB.setObjectName("res_Iangle_P1_DSB")
self.res_U_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P2_DSB.setEnabled(False)
self.res_U_P2_DSB.setGeometry(QtCore.QRect(220, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P2_DSB.setFont(font)
self.res_U_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P2_DSB.setDecimals(3)
self.res_U_P2_DSB.setMaximum(999999.999)
self.res_U_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P2_DSB.setProperty("value", 0.0)
self.res_U_P2_DSB.setObjectName("res_U_P2_DSB")
self.Por1_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Por1_LBL.setGeometry(QtCore.QRect(90, 10, 111, 21))
font = QtGui.QFont()
```

```
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Por1_LBL.setFont(font)
self.Por1_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Por1_LBL.setObjectName("Por1_LBL")
self.res_I_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P2_unit.setGeometry(QtCore.QRect(300, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P2_unit.setFont(font)
self.res_I_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P2_unit.setObjectName("res_I_P2_unit")
self.res_Ploss_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Ploss_LBL.setGeometry(QtCore.QRect(0, 270, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Ploss_LBL.setFont(font)
self.res_Ploss_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Ploss_LBL.setObjectName("res_Ploss_LBL")
self.res_S_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P2_DSB.setEnabled(False)
self.res_S_P2_DSB.setGeometry(QtCore.QRect(220, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P2_DSB.setFont(font)
self.res_S_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P2_DSB.setDecimals(3)
self.res_S_P2_DSB.setMaximum(999999.999)
self.res_S_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P2_DSB.setProperty("value", 0.0)
self.res_S_P2_DSB.setObjectName("res_S_P2_DSB")
self.res_P_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_P_P1_DSB.setEnabled(False)
self.res_P_P1_DSB.setGeometry(QtCore.QRect(90, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_DSB.setFont(font)
self.res_P_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_P_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_P_P1_DSB.setDecimals(3)
self.res_P_P1_DSB.setMinimum(-999999.999)
self.res_P_P1_DSB.setMaximum(999999.999)
self.res_P_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_P_P1_DSB.setProperty("value", 0.0)
self.res_P_P1_DSB.setObjectName("res_P_P1_DSB")
self.res_I_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_I_LBL.setGeometry(QtCore.QRect(0, 40, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_I_LBL.setFont(font)
self.res_I_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_LBL.setObjectName("res_I_LBL")
self.res_I_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_I_P1_unit.setGeometry(QtCore.QRect(170, 40, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_unit.setFont(font)
self.res_I_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_I_P1_unit.setObjectName("res_I_P1_unit")
self.res_Qloss_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Qloss_DSB.setEnabled(False)
self.res_Qloss_DSB.setGeometry(QtCore.QRect(90, 300, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qloss_DSB.setFont(font)
self.res_Qloss_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Qloss_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Qloss_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Qloss_DSB.setDecimals(3)
self.res_Qloss_DSB.setMaximum(999999.999)
self.res_Qloss_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Qloss_DSB.setProperty("value", 0.0)
self.res_Qloss_DSB.setObjectName("res_Qloss_DSB")
self.res_Q_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P2_unit.setGeometry(QtCore.QRect(300, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P2_unit.setFont(font)
self.res_Q_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P2_unit.setObjectName("res_Q_P2_unit")
self.res_Iangle_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P2_unit.setGeometry(QtCore.QRect(300, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P2_unit.setFont(font)
self.res_Iangle_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P2_unit.setObjectName("res_Iangle_P2_unit")
self.res_Q_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P1_DSB.setEnabled(False)
self.res_Q_P1_DSB.setGeometry(QtCore.QRect(90, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
```

```
font.setWeight(50)
self.res_Q_P1_DSB.setFont(font)
self.res_Q_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P1_DSB.setDecimals(3)
self.res_Q_P1_DSB.setMinimum(-999999.999)
self.res_Q_P1_DSB.setMaximum(999999.999)
self.res_Q_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P1_DSB.setProperty("value", 0.0)
self.res_Q_P1_DSB.setObjectName("res_Q_P1_DSB")
self.res_P_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_P_LBL.setGeometry(QtCore.QRect(0, 100, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_P_LBL.setFont(font)
self.res_P_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_P_LBL.setObjectName("res_P_LBL")
self.Port2_LBL = QtWidgets.QLabel(self.LoadFlow)
self.Port2_LBL.setGeometry(QtCore.QRect(220, 10, 101, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.Port2_LBL.setFont(font)
self.Port2_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.Port2_LBL.setObjectName("Port2_LBL")
self.res_I_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P2_DSB.setEnabled(False)
self.res_I_P2_DSB.setGeometry(QtCore.QRect(220, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P2_DSB.setFont(font)
self.res_I_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P2_DSB.setDecimals(3)
self.res_I_P2_DSB.setMaximum(999999.999)
self.res_I_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P2_DSB.setProperty("value", 0.0)
self.res_I_P2_DSB.setObjectName("res_I_P2_DSB")
self.res_cosPhi_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P1_unit.setGeometry(QtCore.QRect(170, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P1_unit.setFont(font)
self.res_cosPhi_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P1_unit.setObjectName("res_cosPhi_P1_unit")
self.res_S_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P1_unit.setGeometry(QtCore.QRect(170, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_unit.setFont(font)
self.res_S_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P1_unit.setObjectName("res_S_P1_unit")
self.res_Q_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Q_P2_DSB.setEnabled(False)
self.res_Q_P2_DSB.setGeometry(QtCore.QRect(220, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P2_DSB.setFont(font)
self.res_Q_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Q_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Q_P2_DSB.setDecimals(3)
self.res_Q_P2_DSB.setMinimum(-999999.999)
self.res_Q_P2_DSB.setMaximum(999999.999)
self.res_Q_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Q_P2_DSB.setProperty("value", 0.0)
self.res_Q_P2_DSB.setObjectName("res_Q_P2_DSB")
self.res_P_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P2_unit.setGeometry(QtCore.QRect(300, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P2_unit.setFont(font)
self.res_P_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P2_unit.setObjectName("res_P_P2_unit")
self.res_U_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P2_unit.setGeometry(QtCore.QRect(300, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P2_unit.setFont(font)
self.res_U_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P2_unit.setObjectName("res_U_P2_unit")
self.res_Iangle_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_P1_unit.setGeometry(QtCore.QRect(170, 70, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P1_unit.setFont(font)
self.res_Iangle_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Iangle_P1_unit.setObjectName("res_Iangle_P1_unit")
self.res_S_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_S_LBL.setGeometry(QtCore.QRect(0, 160, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_S_LBL.setFont(font)
self.res_S_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
```

```
self.res_S_LBL.setObjectName("res_S_LBL")
self.res_I_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_I_P1_DSB.setEnabled(False)
self.res_I_P1_DSB.setGeometry(QtCore.QRect(90, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_I_P1_DSB.setFont(font)
self.res_I_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_I_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_I_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_I_P1_DSB.setDecimals(3)
self.res_I_P1_DSB.setMaximum(999999.999)
self.res_I_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_I_P1_DSB.setProperty("value", 0.0)
self.res_I_P1_DSB.setObjectName("res_I_P1_DSB")
self.res_S_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_S_P2_unit.setGeometry(QtCore.QRect(300, 160, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P2_unit.setFont(font)
self.res_S_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_S_P2_unit.setObjectName("res_S_P2_unit")
self.res_Qloss_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Qloss_unit.setGeometry(QtCore.QRect(170, 300, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Qloss_unit.setFont(font)
self.res_Qloss_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Qloss_unit.setObjectName("res_Qloss_unit")
self.res_U_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_U_P1_DSB.setEnabled(False)
self.res_U_P1_DSB.setGeometry(QtCore.QRect(90, 220, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_DSB.setFont(font)
self.res_U_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_U_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_U_P1_DSB.setDecimals(3)
self.res_U_P1_DSB.setMaximum(999999.999)
self.res_U_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_U_P1_DSB.setProperty("value", 0.0)
self.res_U_P1_DSB.setObjectName("res_U_P1_DSB")
self.res_cosPhi_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_cosPhi_P2_DSB.setEnabled(False)
self.res_cosPhi_P2_DSB.setGeometry(QtCore.QRect(220, 190, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P2_DSB.setFont(font)
self.res_cosPhi_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_cosPhi_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_cosPhi_P2_DSB.setDecimals(3)
self.res_cosPhi_P2_DSB.setMaximum(999999.999)
self.res_cosPhi_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_cosPhi_P2_DSB.setProperty("value", 0.0)
self.res_cosPhi_P2_DSB.setObjectName("res_cosPhi_P2_DSB")
self.res_U_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_U_LBL.setGeometry(QtCore.QRect(0, 220, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_U_LBL.setFont(font)
self.res_U_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_U_LBL.setObjectName("res_U_LBL")
self.res_U_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_U_P1_unit.setGeometry(QtCore.QRect(170, 220, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_U_P1_unit.setFont(font)
self.res_U_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_U_P1_unit.setObjectName("res_U_P1_unit")
self.res_Iangle_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Iangle_LBL.setGeometry(QtCore.QRect(0, 70, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Iangle_LBL.setFont(font)
self.res_Iangle_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_LBL.setObjectName("res_Iangle_LBL")
self.res_cosPhi_P2_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_cosPhi_P2_unit.setGeometry(QtCore.QRect(300, 190, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_cosPhi_P2_unit.setFont(font)
self.res_cosPhi_P2_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_cosPhi_P2_unit.setObjectName("res_cosPhi_P2_unit")
self.res_Iangle_P2_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Iangle_P2_DSB.setEnabled(False)
self.res_Iangle_P2_DSB.setGeometry(QtCore.QRect(220, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Iangle_P2_DSB.setFont(font)
self.res_Iangle_P2_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Iangle_P2_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Iangle_P2_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Iangle_P2_DSB.setDecimals(3)
```

```
self.res_Iangle_P2_DSB.setMinimum(-999999.999)
self.res_Iangle_P2_DSB.setMaximum(999999.999)
self.res_Iangle_P2_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Iangle_P2_DSB.setProperty("value", 0.0)
self.res_Iangle_P2_DSB.setObjectName("res_Iangle_P2_DSB")
self.res_LimViolated_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_LimViolated_LBL.setGeometry(QtCore.QRect(0, 340, 321, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_LimViolated_LBL.setFont(font)
self.res_LimViolated_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.res_LimViolated_LBL.setObjectName("res_LimViolated_LBL")
self.res_P_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_P_P1_unit.setGeometry(QtCore.QRect(170, 100, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_P_P1_unit.setFont(font)
self.res_P_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_P_P1_unit.setObjectName("res_P_P1_unit")
self.res_Q_P1_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_P1_unit.setGeometry(QtCore.QRect(170, 130, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Q_P1_unit.setFont(font)
self.res_Q_P1_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Q_P1_unit.setObjectName("res_Q_P1_unit")
self.res_Ploss_unit = QtWidgets.QLabel(self.LoadFlow)
self.res_Ploss_unit.setGeometry(QtCore.QRect(170, 270, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Ploss_unit.setFont(font)
self.res_Ploss_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.res_Ploss_unit.setObjectName("res_Ploss_unit")
self.res_S_P1_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_S_P1_DSB.setEnabled(False)
self.res_S_P1_DSB.setGeometry(QtCore.QRect(90, 160, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_S_P1_DSB.setFont(font)
self.res_S_P1_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_S_P1_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_S_P1_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_S_P1_DSB.setDecimals(3)
self.res_S_P1_DSB.setMaximum(999999.999)
self.res_S_P1_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_S_P1_DSB.setProperty("value", 0.0)
self.res_S_P1_DSB.setObjectName("res_S_P1_DSB")
self.res_Q_LBL = QtWidgets.QLabel(self.LoadFlow)
self.res_Q_LBL.setGeometry(QtCore.QRect(0, 130, 81, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.res_Q_LBL.setFont(font)
self.res_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Q_LBL.setObjectName("res_Q_LBL")
self.res_Ploss_DSB = QtWidgets.QDoubleSpinBox(self.LoadFlow)
self.res_Ploss_DSB.setEnabled(False)
self.res_Ploss_DSB.setGeometry(QtCore.QRect(90, 270, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.res_Ploss_DSB.setFont(font)
self.res_Ploss_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.res_Ploss_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.res_Ploss_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.res_Ploss_DSB.setDecimals(3)
self.res_Ploss_DSB.setMaximum(999999.999)
self.res_Ploss_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.res_Ploss_DSB.setProperty("value", 0.0)
self.res_Ploss_DSB.setObjectName("res_Ploss_DSB")
self.tabWidget.addTab(self.LoadFlow, "")
self.EMS = QtWidgets.QWidget()
self.EMS.setObjectName("EMS")
self.cap_pwr_LBL = QtWidgets.QLabel(self.EMS)
self.cap_pwr_LBL.setGeometry(QtCore.QRect(10, 10, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.cap_pwr_LBL.setFont(font)
self.cap_pwr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_LBL.setObjectName("cap_pwr_LBL")
self.cap_pwr_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.cap_pwr_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.cap_pwr_DSB.setFont(font)
self.cap_pwr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.cap_pwr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.cap_pwr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.cap_pwr_DSB.setDecimals(3)
self.cap_pwr_DSB.setMaximum(999999.999)
self.cap_pwr_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.cap_pwr_DSB.setProperty("value", 0.0)
self.cap_pwr_DSB.setObjectName("cap_pwr_DSB")
self.cap_pwr_unit = QtWidgets.QLabel(self.EMS)
self.cap_pwr_unit.setGeometry(QtCore.QRect(240, 10, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
```

```
self.cap_pwr_unit.setFont(font)
self.cap_pwr_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.cap_pwr_unit.setObjectName("cap_pwr_unit")
self.max_outin_unit = QtWidgets.QLabel(self.EMS)
self.max_outin_unit.setGeometry(QtCore.QRect(240, 40, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_outin_unit.setFont(font)
self.max_outin_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_outin_unit.setObjectName("max_outin_unit")
self.max_outin_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.max_outin_DSB.setGeometry(QtCore.QRect(160, 40, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_outin_DSB.setFont(font)
self.max_outin_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_outin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_outin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_outin_DSB.setDecimals(3)
self.max_outin_DSB.setMaximum(999999.999)
self.max_outin_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_outin_DSB.setProperty("value", 0.0)
self.max_outin_DSB.setObjectName("max_outin_DSB")
self.max_outin_LBL = QtWidgets.QLabel(self.EMS)
self.max_outin_LBL.setGeometry(QtCore.QRect(10, 40, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_outin_LBL.setFont(font)
self.max_outin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_outin_LBL.setObjectName("max_outin_LBL")
self.max_inout_unit = QtWidgets.QLabel(self.EMS)
self.max_inout_unit.setGeometry(QtCore.QRect(240, 70, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_inout_unit.setFont(font)
self.max_inout_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.max_inout_unit.setObjectName("max_inout_unit")
self.max_inout_LBL = QtWidgets.QLabel(self.EMS)
self.max_inout_LBL.setGeometry(QtCore.QRect(10, 70, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.max_inout_LBL.setFont(font)
self.max_inout_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_inout_LBL.setObjectName("max_inout_LBL")
self.max_inout_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.max_inout_DSB.setGeometry(QtCore.QRect(160, 70, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.max_inout_DSB.setFont(font)
self.max_inout_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.max_inout_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.max_inout_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.max_inout_DSB.setDecimals(3)
self.max_inout_DSB.setMaximum(999999.999)
self.max_inout_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.max_inout_DSB.setProperty("value", 0.0)
self.max_inout_DSB.setObjectName("max_inout_DSB")
self.etainout_unit = QtWidgets.QLabel(self.EMS)
self.etainout_unit.setGeometry(QtCore.QRect(240, 130, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etainout_unit.setFont(font)
self.etainout_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.etainout_unit.setObjectName("etainout_unit")
self.etaoutin_LBL = QtWidgets.QLabel(self.EMS)
self.etaoutin_LBL.setGeometry(QtCore.QRect(10, 100, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.etaoutin_LBL.setFont(font)
self.etaoutin_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etaoutin_LBL.setObjectName("etaoutin_LBL")
self.etainout_LBL = QtWidgets.QLabel(self.EMS)
self.etainout_LBL.setGeometry(QtCore.QRect(10, 130, 141, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.etainout_LBL.setFont(font)
self.etainout_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etainout_LBL.setObjectName("etainout_LBL")
self.etaoutin_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.etaoutin_DSB.setGeometry(QtCore.QRect(160, 100, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etaoutin_DSB.setFont(font)
self.etaoutin_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.etaoutin_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etaoutin_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.etaoutin_DSB.setDecimals(3)
self.etaoutin_DSB.setMaximum(999999.999)
self.etaoutin_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.etaoutin_DSB.setProperty("value", 0.0)
self.etaoutin_DSB.setObjectName("etaoutin_DSB")
self.etainout_DSB = QtWidgets.QDoubleSpinBox(self.EMS)
self.etainout_DSB.setGeometry(QtCore.QRect(160, 130, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
```

```
font.setBold(False)
font.setWeight(50)
self.etainout_DSB.setFont(font)
self.etainout_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.etainout_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.etainout_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.etainout_DSB.setDecimals(3)
self.etainout_DSB.setMaximum(999999.999)
self.etainout_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.etainout_DSB.setProperty("value", 0.0)
self.etainout_DSB.setObjectName("etainout_DSB")
self.etaoutin_unit = QtWidgets.QLabel(self.EMS)
self.etaoutin_unit.setGeometry(QtCore.QRect(240, 100, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.etaoutin_unit.setFont(font)
self.etaoutin_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.etaoutin_unit.setObjectName("etaoutin_unit")
self.tabWidget.addTab(self.EMS, "")
self.Reliability = QtWidgets.QWidget()
self.Reliability.setObjectName("Reliability")
self.rel_results_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_results_LBL.setGeometry(QtCore.QRect(130, 290, 65, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_results_LBL.setFont(font)
self.rel_results_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.rel_results_LBL.setObjectName("rel_results_LBL")
self.rel_alfa_unit = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_unit.setGeometry(QtCore.QRect(240, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_unit.setFont(font)
self.rel_alfa_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_alfa_unit.setObjectName("rel_alfa_unit")
self.rel_R_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_R_LBL.setGeometry(QtCore.QRect(160, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_R_LBL.setFont(font)
self.rel_R_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LBL.setObjectName("rel_R_LBL")
self.rel_lambda_unit = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_unit.setGeometry(QtCore.QRect(140, 320, 51, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_lambda_unit.setFont(font)
self.rel_lambda_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_lambda_unit.setObjectName("rel_lambda_unit")
self.rel_beta_unit = QtWidgets.QLabel(self.Reliability)
self.rel_beta_unit.setGeometry(QtCore.QRect(240, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_unit.setFont(font)
self.rel_beta_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_beta_unit.setObjectName("rel_beta_unit")
self.rel_alfa_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_alfa_LBL.setGeometry(QtCore.QRect(0, 60, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_alfa_LBL.setFont(font)
self.rel_alfa_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_LBL.setObjectName("rel_alfa_LBL")
self.rel_T0_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_T0_LBL.setGeometry(QtCore.QRect(0, 10, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_T0_LBL.setFont(font)
self.rel_T0_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_LBL.setObjectName("rel_T0_LBL")
self.rel_beta_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_beta_DSB.setEnabled(True)
self.rel_beta_DSB.setGeometry(QtCore.QRect(160, 90, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_beta_DSB.setFont(font)
self.rel_beta_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_beta_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_beta_DSB.setDecimals(1)
self.rel_beta_DSB.setMaximum(1000000.0)
self.rel_beta_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_beta_DSB.setProperty("value", 1.0)
self.rel_beta_DSB.setObjectName("rel_beta_DSB")
self.rel_alfa_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_alfa_DSB.setEnabled(True)
self.rel_alfa_DSB.setGeometry(QtCore.QRect(160, 60, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_alfa_DSB.setFont(font)
self.rel_alfa_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_alfa_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_alfa_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_alfa_DSB.setDecimals(0)
self.rel_alfa_DSB.setMaximum(100000000.0)
```

```
self.rel_alfa_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_alfa_DSB.setProperty("value", 438000.0)
self.rel_alfa_DSB.setObjectName("rel_alfa_DSB")
self.rel_lambda_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_lambda_LBL.setGeometry(QtCore.QRect(0, 320, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_lambda_LBL.setFont(font)
self.rel_lambda_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LBL.setObjectName("rel_lambda_LBL")
self.bottom_LN_3 = QtWidgets.QFrame(self.Reliability)
self.bottom_LN_3.setGeometry(QtCore.QRect(10, 300, 305, 1))
self.bottom_LN_3.setStyleSheet("background-color: rgb(255, 255, 255);")
self.bottom_LN_3.setFrameShape(QtWidgets.QFrame.HLine)
self.bottom_LN_3.setFrameShadow(QtWidgets.QFrame.Sunken)
self.bottom_LN_3.setObjectName("bottom_LN_3")
self.rel_Pi_Q_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_Q_DSB.setEnabled(True)
self.rel_Pi_Q_DSB.setGeometry(QtCore.QRect(160, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_DSB.setFont(font)
self.rel_Pi_Q_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_Q_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_Q_DSB.setDecimals(1)
self.rel_Pi_Q_DSB.setMinimum(0.5)
self.rel_Pi_Q_DSB.setMaximum(8.0)
self.rel_Pi_Q_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_Q_DSB.setProperty("value", 5.5)
self.rel_Pi_Q_DSB.setObjectName("rel_Pi_Q_DSB")
self.rel_T0_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_T0_DSB.setEnabled(True)
self.rel_T0_DSB.setGeometry(QtCore.QRect(160, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_DSB.setFont(font)
self.rel_T0_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_T0_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_T0_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_T0_DSB.setDecimals(1)
self.rel_T0_DSB.setMinimum(-273.0)
self.rel_T0_DSB.setMaximum(999.0)
self.rel_T0_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_T0_DSB.setProperty("value", 30.0)
self.rel_T0_DSB.setObjectName("rel_T0_DSB")
self.rel_MTBF_ore_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_unit.setGeometry(QtCore.QRect(140, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_unit.setFont(font)
self.rel_MTBF_ore_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_unit.setObjectName("rel_MTBF_ore_unit")
self.rel_MTBF_ore_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_ore_DSB.setEnabled(False)
self.rel_MTBF_ore_DSB.setGeometry(QtCore.QRect(70, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_ore_DSB.setFont(font)
self.rel_MTBF_ore_DSB.setToolTip("")
self.rel_MTBF_ore_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_ore_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_ore_DSB.setDecimals(1)
self.rel_MTBF_ore_DSB.setMaximum(1000000.0)
self.rel_MTBF_ore_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_ore_DSB.setProperty("value", 0.0)
self.rel_MTBF_ore_DSB.setObjectName("rel_MTBF_ore_DSB")
self.rel_Pi_Q_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_unit.setGeometry(QtCore.QRect(240, 170, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_Q_unit.setFont(font)
self.rel_Pi_Q_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_unit.setObjectName("rel_Pi_Q_unit")
self.rel_MTBF_ore_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_ore_LBL.setGeometry(QtCore.QRect(0, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_ore_LBL.setFont(font)
self.rel_MTBF_ore_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_ore_LBL.setObjectName("rel_MTBF_ore_LBL")
self.rel_beta_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_beta_LBL.setGeometry(QtCore.QRect(0, 90, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_beta_LBL.setFont(font)
self.rel_beta_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_beta_LBL.setObjectName("rel_beta_LBL")
self.rel_Pi_E_unit = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_unit.setGeometry(QtCore.QRect(240, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_unit.setFont(font)
self.rel_Pi_E_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_unit.setObjectName("rel_Pi_E_unit")
self.rel_T0_unit = QtWidgets.QLabel(self.Reliability)
```

```
self.rel_T0_unit.setGeometry(QtCore.QRect(240, 10, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_T0_unit.setFont(font)
self.rel_T0_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_T0_unit.setObjectName("rel_T0_unit")
self.rel_Pi_Q_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_Q_LBL.setGeometry(QtCore.QRect(0, 170, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_Q_LBL.setFont(font)
self.rel_Pi_Q_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_Q_LBL.setObjectName("rel_Pi_Q_LBL")
self.rel_MTBF_anni_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_LBL.setGeometry(QtCore.QRect(160, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_MTBF_anni_LBL.setFont(font)
self.rel_MTBF_anni_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_LBL.setObjectName("rel_MTBF_anni_LBL")
self.rel_Pi_E_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_Pi_E_DSB.setEnabled(True)
self.rel_Pi_E_DSB.setGeometry(QtCore.QRect(160, 140, 71, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_Pi_E_DSB.setFont(font)
self.rel_Pi_E_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_Pi_E_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_Pi_E_DSB.setDecimals(1)
self.rel_Pi_E_DSB.setMinimum(1.0)
self.rel_Pi_E_DSB.setMaximum(12.0)
self.rel_Pi_E_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_Pi_E_DSB.setProperty("value", 1.0)
self.rel_Pi_E_DSB.setObjectName("rel_Pi_E_DSB")
self.rel_MTBF_anni_DSB = QtWidgets.QDoubleSpinBox(self.Reliability)
self.rel_MTBF_anni_DSB.setEnabled(False)
self.rel_MTBF_anni_DSB.setGeometry(QtCore.QRect(230, 350, 61, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_DSB.setFont(font)
self.rel_MTBF_anni_DSB.setToolTip("")
self.rel_MTBF_anni_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.rel_MTBF_anni_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.rel_MTBF_anni_DSB.setDecimals(1)
self.rel_MTBF_anni_DSB.setMaximum(1000000.0)
self.rel_MTBF_anni_DSB.setStepType(QtWidgets.QAbstractSpinBox.AdaptiveDecimalStepType)
self.rel_MTBF_anni_DSB.setProperty("value", 0.0)
self.rel_MTBF_anni_DSB.setObjectName("rel_MTBF_anni_DSB")
self.rel_Pi_E_LBL = QtWidgets.QLabel(self.Reliability)
self.rel_Pi_E_LBL.setGeometry(QtCore.QRect(0, 140, 151, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(True)
font.setWeight(75)
self.rel_Pi_E_LBL.setFont(font)
self.rel_Pi_E_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_Pi_E_LBL.setObjectName("rel_Pi_E_LBL")
self.rel_R_unit = QtWidgets.QLabel(self.Reliability)
self.rel_R_unit.setGeometry(QtCore.QRect(300, 320, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_R_unit.setFont(font)
self.rel_R_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_R_unit.setObjectName("rel_R_unit")
self.rel_lambda_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_lambda_LE.setGeometry(QtCore.QRect(70, 320, 61, 21))
self.rel_lambda_LE.setFrame(True)
self.rel_lambda_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_lambda_LE.setReadOnly(True)
self.rel_lambda_LE.setObjectName("rel_lambda_LE")
self.rel_R_LE = QtWidgets.QLineEdit(self.Reliability)
self.rel_R_LE.setGeometry(QtCore.QRect(230, 320, 61, 21))
self.rel_R_LE.setFrame(True)
self.rel_R_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.rel_R_LE.setReadOnly(True)
self.rel_R_LE.setObjectName("rel_R_LE")
self.rel_MTBF_anni_unit = QtWidgets.QLabel(self.Reliability)
self.rel_MTBF_anni_unit.setGeometry(QtCore.QRect(300, 350, 21, 21))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setWeight(50)
self.rel_MTBF_anni_unit.setFont(font)
self.rel_MTBF_anni_unit.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.rel_MTBF_anni_unit.setObjectName("rel_MTBF_anni_unit")
self.rel_alfa_unit.raise_()
self.rel_R_LBL.raise_()
self.rel_lambda_unit.raise_()
self.rel_beta_unit.raise_()
self.rel_alfa_LBL.raise_()
self.rel_T0_LBL.raise_()
self.rel_beta_DSB.raise_()
self.rel_alfa_DSB.raise_()
self.rel_lambda_LBL.raise_()
self.bottom_LN_3.raise_()
self.rel_Pi_Q_DSB.raise_()
self.rel_T0_DSB.raise_()
self.rel_MTBF_ore_unit.raise_()
self.rel_MTBF_ore_DSB.raise_()
self.rel_Pi_Q_unit.raise_()
self.rel_MTBF_ore_LBL.raise_()
```

```
        self.rel_beta_LBL.raise_()
        self.rel_Pi_E_unit.raise_()
        self.rel_TO_unit.raise_()
        self.rel_Pi_Q_LBL.raise_()
        self.rel_MTBF_anni_LBL.raise_()
        self.rel_Pi_E_DSB.raise_()
        self.rel_MTBF_anni_DSB.raise_()
        self.rel_Pi_E_LBL.raise_()
        self.rel_R_unit.raise_()
        self.rel_lambda_LE.raise_()
        self.rel_R_LE.raise_()
        self.rel_results_LBL.raise_()
        self.rel_MTBF_anni_unit.raise_()
        self.tabWidget.addTab(self.Reliability, "")
        self.store_BTN = QtWidgets.QPushButton(self.widget)
        self.store_BTN.setGeometry(QtCore.QRect(360, 470, 121, 23))
        font = QtGui.QFont()
        font.setPointSize(8)
        self.store_BTN.setFont(font)
        self.store_BTN.setStyleSheet("background-color: rgb(85, 85, 127);")
        self.store_BTN.setObjectName("store_BTN")
        self.bb_out_LBL.raise_()
        self.bb_in_LBL.raise_()
        self.cub_out_LBL.raise_()
        self.top_LN.raise_()
        self.elem_name_LN.raise_()
        self.elem_name_LBL.raise_()
        self.type_LBL.raise_()
        self.type_cap_LBL.raise_()
        self.cub_in_LBL.raise_()
        self.symbol_LBL.raise_()
        self.ot_Frame_LN.raise_()
        self.vdx_frame_LN.raise_()
        self.vsx_frame_LN.raise_()
        self.ob_frame_LN.raise_()
        self.bb_in_LN.raise_()
        self.bb_out_LN.raise_()
        self.cancel_BTN.raise_()
        self.tabWidget.raise_()
        self.store_BTN.raise_()

        self.retranslateUi(Form)
        self.tabWidget.setCurrentIndex(3)
        QtCore.QMetaObject.connectSlotsByName(Form)
        Form.setTabOrder(self.tabWidget, self.sr_DSB)
        Form.setTabOrder(self.sr_DSB, self.UnHV_DSB)
        Form.setTabOrder(self.UnHV_DSB, self.UnLV_DSB)
        Form.setTabOrder(self.UnLV_DSB, self.URr1_DSB)
        Form.setTabOrder(self.URr1_DSB, self.Ukr1_DSB)
        Form.setTabOrder(self.Ukr1_DSB, self.URr0_DSB)
        Form.setTabOrder(self.URr0_DSB, self.Ukr0_DSB)
        Form.setTabOrder(self.Ukr0_DSB, self.cap_pwr_DSB)
        Form.setTabOrder(self.cap_pwr_DSB, self.max_outin_DSB)
        Form.setTabOrder(self.max_outin_DSB, self.max_inout_DSB)
        Form.setTabOrder(self.max_inout_DSB, self.etaoutin_DSB)
        Form.setTabOrder(self.etaoutin_DSB, self.etainout_DSB)
        Form.setTabOrder(self.etainout_DSB, self.rel_TO_DSB)
        Form.setTabOrder(self.rel_TO_DSB, self.rel_alfa_DSB)
        Form.setTabOrder(self.rel_alfa_DSB, self.rel_beta_DSB)
        Form.setTabOrder(self.rel_beta_DSB, self.rel_Pi_E_DSB)
        Form.setTabOrder(self.rel_Pi_E_DSB, self.rel_Pi_Q_DSB)
        Form.setTabOrder(self.rel_Pi_Q_DSB, self.rel_MTBF_ore_DSB)
        Form.setTabOrder(self.rel_MTBF_ore_DSB, self.rel_MTBF_anni_DSB)
        Form.setTabOrder(self.rel_MTBF_anni_DSB, self.res_I_P1_DSB)
        Form.setTabOrder(self.res_I_P1_DSB, self.res_Iangle_P1_DSB)
        Form.setTabOrder(self.res_Iangle_P1_DSB, self.res_P_P1_DSB)
        Form.setTabOrder(self.res_P_P1_DSB, self.res_Q_P1_DSB)
        Form.setTabOrder(self.res_Q_P1_DSB, self.res_S_P1_DSB)
        Form.setTabOrder(self.res_S_P1_DSB, self.res_cosPhi_P1_DSB)
        Form.setTabOrder(self.res_cosPhi_P1_DSB, self.res_U_P1_DSB)
        Form.setTabOrder(self.res_U_P1_DSB, self.res_I_P2_DSB)
        Form.setTabOrder(self.res_I_P2_DSB, self.res_Iangle_P2_DSB)
        Form.setTabOrder(self.res_Iangle_P2_DSB, self.res_P_P2_DSB)
        Form.setTabOrder(self.res_P_P2_DSB, self.res_Q_P2_DSB)
        Form.setTabOrder(self.res_Q_P2_DSB, self.res_S_P2_DSB)
        Form.setTabOrder(self.res_S_P2_DSB, self.res_cosPhi_P2_DSB)
        Form.setTabOrder(self.res_cosPhi_P2_DSB, self.res_U_P2_DSB)
        Form.setTabOrder(self.res_U_P2_DSB, self.res_Ploss_DSB)
        Form.setTabOrder(self.res_Ploss_DSB, self.res_Qloss_DSB)
        Form.setTabOrder(self.res_Qloss_DSB, self.store_BTN)
        Form.setTabOrder(self.store_BTN, self.cancel_BTN)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.elem_name_LBL.setText(_translate("Form", "<html><head/><body><p><span style=\" font-size:10pt; font-weight:600;
color:#ffffff;\">Nome Elemento</span></p></body></html>"))
        self.type_LBL.setText(_translate("Form", "Type"))
        self.type_cap_LBL.setText(_translate("Form", "Categoria:"))
        self.bb_in_LBL.setText(_translate("Form", "Busbar IN"))
        self.bb_out_LBL.setText(_translate("Form", "Busbar OUT"))
        self.cancel_BTN.setText(_translate("Form", "Annulla"))
        self.URr1LBL.setText(_translate("Form", "URr(1)"))
        self.Ukr1_LBL.setText(_translate("Form", "Ukr(1)"))
        self.UnLV_LBL.setText(_translate("Form", "Un LV"))
        self.Ukr1_unit.setText(_translate("Form", "%"))
        self.URr0_LBL.setText(_translate("Form", "URr(0)"))
        self.UnLV_unit.setText(_translate("Form", "kv"))
        self.Ukr0_unit.setText(_translate("Form", "%"))
        self.UnHV_LBL.setText(_translate("Form", "Un HV"))
        self.sr_unit.setText(_translate("Form", "kWA"))
        self.URr0_unit.setText(_translate("Form", "%"))
        self.UnHV_unit.setText(_translate("Form", "kv"))
        self.Ukr0_LBL.setText(_translate("Form", "Ukr(0)"))
        self.sr_LBL.setText(_translate("Form", "Sr"))
        self.URr1_unit.setText(_translate("Form", "%"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Parameters), _translate("Form", "Parametri"))
        self.res_cosPhi_LBL.setText(_translate("Form", "cosPhi"))
        self.res_Qloss_LBL.setText(_translate("Form", "Q loss"))
        self.Por1_LBL.setText(_translate("Form", "Nodo 1"))
        self.res_I_P2_unit.setText(_translate("Form", "A"))
        self.res_Ploss_LBL.setText(_translate("Form", "P loss"))
        self.res_I_LBL.setText(_translate("Form", " I"))
        self.res_I_P1_unit.setText(_translate("Form", "A"))
        self.res_Q_P2_unit.setText(_translate("Form", "kVA"))
```

```
        self.res_Iangle_P2_unit.setText(_translate("Form", "°"))
        self.res_P_LBL.setText(_translate("Form", "P"))
        self.Port2_LBL.setText(_translate("Form", "Nodo 2"))
        self.res_cosPhi_P1_unit.setText(_translate("Form", "-"))
        self.res_S_P1_unit.setText(_translate("Form", "kVA"))
        self.res_P_P2_unit.setText(_translate("Form", "kW"))
        self.res_U_P2_unit.setText(_translate("Form", "kV"))
        self.res_Iangle_P1_unit.setText(_translate("Form", "°"))
        self.res_S_LBL.setText(_translate("Form", "S"))
        self.res_S_P2_unit.setText(_translate("Form", "kVA"))
        self.res_Qloss_unit.setText(_translate("Form", "kVA"))
        self.res_U_LBL.setText(_translate("Form", "U"))
        self.res_U_P1_unit.setText(_translate("Form", "kV"))
        self.res_Iangle_LBL.setText(_translate("Form", "Angolo I"))
        self.res_cosPhi_P2_unit.setText(_translate("Form", "-"))
        self.res_LimViolated_LBL.setText(_translate("Form", "LIMITI VIOLATI"))
        self.res_P_P1_unit.setText(_translate("Form", "kW"))
        self.res_Q_P1_unit.setText(_translate("Form", "kVA"))
        self.res_Ploss_unit.setText(_translate("Form", "kW"))
        self.res_Q_LBL.setText(_translate("Form", "Q"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.LoadFlow), _translate("Form", "LoadFlow"))
        self.cap_pwr_LBL.setText(_translate("Form", "Potenza Nominale"))
        self.cap_pwr_unit.setText(_translate("Form", "kVA"))
        self.max_outin_unit.setText(_translate("Form", "p.u."))
        self.max_outin_LBL.setText(_translate("Form", "Potenza Massima IN"))
        self.max_inout_unit.setText(_translate("Form", "p.u."))
        self.max_inout_LBL.setText(_translate("Form", "Potenzsa Massima OUT"))
        self.etainout_unit.setText(_translate("Form", "p.u."))
        self.etaoutin_LBL.setText(_translate("Form", "Efficienza OUT-to-IN"))
        self.etainout_LBL.setText(_translate("Form", "Efficienza IN-to-OUT"))
        self.etaoutin_unit.setText(_translate("Form", "p.u."))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.EMS), _translate("Form", "EMS"))
        self.rel_results_LBL.setText(_translate("Form", "Risultati"))
        self.rel_alfa_unit.setText(_translate("Form", "h"))
        self.rel_R_LBL.setText(_translate("Form", "R"))
        self.rel_lambda_unit.setText(_translate("Form", "fail/10^6"))
        self.rel_beta_unit.setText(_translate("Form", "-"))
        self.rel_alfa_LBL.setText(_translate("Form", "Alfa"))
        self.rel_T0_LBL.setText(_translate("Form", "T_0"))
        self.rel_beta_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di forma β
(Weibull)</span></p></body></html>"))
        self.rel_alfa_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di scala α
(Weibull)</span></p></body></html>"))
        self.rel_lambda_LBL.setText(_translate("Form", "lambda"))
        self.rel_Pi_Q_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di qualità del
componente</span></p></body></html>"))
        self.rel_T0_DSB.setToolTip(_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span
style=\" color:#00007f;\">Temperatura di riferimento</span></p></body></html>"))
        self.rel_MTBF_ore_unit.setText(_translate("Form", "ore"))
        self.rel_Pi_Q_unit.setText(_translate("Form", "-"))
        self.rel_MTBF_ore_LBL.setText(_translate("Form", "MTBF"))
        self.rel_beta_LBL.setText(_translate("Form", "Beta"))
        self.rel_Pi_E_unit.setText(_translate("Form", "-"))
        self.rel_T0_unit.setText(_translate("Form", "°C"))
        self.rel_Pi_Q_LBL.setText(_translate("Form", "Pi_Q"))
        self.rel_MTBF_anni_LBL.setText(_translate("Form", "MTBF"))
        self.rel_Pi_E_DSB.setToolTip(_translate("Form", "<html><head/><body><p><span style=\" color:#00007f;\">Fattore di stress
ambientale</span></p></body></html>"))
        self.rel_Pi_E_LBL.setText(_translate("Form", "Pi_E"))
        self.rel_R_unit.setText(_translate("Form", "-"))
        self.rel_lambda_LE.setText(_translate("Form", "0.0"))
        self.rel_R_LE.setText(_translate("Form", "0.0"))
        self.rel_MTBF_anni_unit.setText(_translate("Form", "anni"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.Reliability), _translate("Form", "Affidabilità"))
        self.store_BTN.setText(_translate("Form", "Salva"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.4.4 EMS

4.4.4.1 ems_results.py

```
from PyQt5 import QtWidgets
from .ems_resultsUI import Ui_MainWindow


class EmsResults(QtWidgets.QMainWindow):
    def __init__(self):
        super(EmsResults, self).__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
```

4.4.4.2 ems_resultsUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'ems_resultsUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1139, 671)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.widget = QtWidgets.QWidget(self.centralwidget)
        self.widget.setGeometry(QtCore.QRect(30, 20, 1101, 320))
        self.widget.setStyleSheet("background-color: rgb(0, 0, 17);")
        self.widget.setObjectName("widget")
        self.label = QtWidgets.QLabel(self.widget)
        self.label.setGeometry(QtCore.QRect(10, 35, 1080, 280))
        self.label.setStyleSheet("")
        self.label.setObjectName("label")
        self.comboBox = QtWidgets.QComboBox(self.widget)
        self.comboBox.setGeometry(QtCore.QRect(840, 4, 251, 22))
        self.comboBox.setStyleSheet("border-color: rgb(255, 255, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"background-color: rgb(0, 0, 17);\n"
"color: rgb(255, 255, 255);")
        self.comboBox.setObjectName("comboBox")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.top_LM = QtWidgets.QFrame(self.widget)
        self.top_LM.setGeometry(QtCore.QRect(0, 0, 1100, 1))
        self.top_LM.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LM.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LM.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LM.setObjectName("top_LM")
        self.mid_LN = QtWidgets.QFrame(self.widget)
        self.mid_LN.setGeometry(QtCore.QRect(10, 30, 1080, 1))
        self.mid_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.mid_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.mid_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.mid_LN.setObjectName("mid_LN")
        self.caption_LBL = QtWidgets.QLabel(self.widget)
        self.caption_LBL.setGeometry(QtCore.QRect(10, 0, 150, 30))
        font = QtGui.QFont()
        font.setPointSize(9)
        font.setBold(True)
        font.setItalic(False)
        font.setWeight(75)
        self.caption_LBL.setFont(font)
        self.caption_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.caption_LBL.setObjectName("caption_LBL")
        self.btm_LN = QtWidgets.QFrame(self.widget)
        self.btm_LN.setGeometry(QtCore.QRect(0, 320, 1100, 1))
        self.btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.btm_LN.setObjectName("btm_LN")
        self.label.raise_()
        self.mid_LN.raise_()
        self.btm_LN.raise_()
        self.caption_LBL.raise_()
        self.top_LM.raise_()
        self.comboBox.raise_()
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1139, 21))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.label.setText(_translate("MainWindow", "TextLabel"))
        self.comboBox.setItemText(0, _translate("MainWindow", "Power balance - DC-Node-363436330"))
        self.comboBox.setItemText(1, _translate("MainWindow", "Power Balance - Batt_BB"))
        self.caption_LBL.setText(_translate("MainWindow", "EMS: Risultati"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

4.4.5 EMS_stoch

4.4.5.1 EMS_stoch_results.py

```
from PyQt5 import QtWidgets, QtGui, QtCore , Qt
import yaml
import os
from __shared__ import variables as v

from .ems_stoch_resultsUI import Ui_MainWindow


class EmsReliabilityResults(QtWidgets.QMainWindow):
    def __init__(self):
        super(EmsReliabilityResults, self).__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        try:
            ems_res_path1 = os.path.join(v.project_folder, 'ems_fault_results.yml')
            ems_res_path2 = os.path.join(v.project_folder, 'ems_fault_results_ENEA.yml')
        except:
            ems_res_path1 = os.path.join('C:/Users/anton/PycharmProjects/RdS-TOOL-v.2/_data/Rete_ENEA_v4.1 - City',
                                         'ems_fault_results.yml')
            ems_res_path2 = os.path.join('C:/Users/anton/PycharmProjects/RdS-TOOL-v.2/_data/Rete_ENEA_v4.1 - City',
                                         'ems_fault_results_ENEA.yml')

        if os.path.isfile(ems_res_path1) and os.path.isfile(ems_res_path2):
            self.ems_rel_ind = yaml.safe_load(open(ems_res_path1))
            ems_rel_ind_enea = yaml.safe_load(open(ems_res_path2))

            self.ui.ens_BC_DSB.setValue(self.ems_rel_ind['Base case']['ALL_SYSTEM']['ENS'])
            self.ui.ens_BC_pu_DSB.setValue(self.ems_rel_ind['Base case']['ALL_SYSTEM']['ENS_pu'])
            self.ui.ens_EMS_DSB.setValue(self.ems_rel_ind['EMS model']['ALL_SYSTEM']['ENS'])
            self.ui.ens_EMS_pu_DSB.setValue(self.ems_rel_ind['EMS model']['ALL_SYSTEM']['ENS_pu'])

            self.ui.eic_BC_DSB.setValue(self.ems_rel_ind['Base case']['ALL_SYSTEM']['ENS_cost'])
            self.ui.eic_BC_pu_DSB.setValue(self.ems_rel_ind['Base case']['ALL_SYSTEM']['ENS_cost_pu'])
            self.ui.eic_EMS_DSB.setValue(self.ems_rel_ind['EMS model']['ALL_SYSTEM']['ENS_cost'])
            self.ui.eic_EMS_pu_DSB.setValue(self.ems_rel_ind['EMS model']['ALL_SYSTEM']['ENS_cost_pu'])

            self.ui.indices_TW.clear()

            self.ui.indices_TW.setRowCount(len(ems_rel_ind_enea['LPENS']['Base case']))
            i = 0
            for elem in ems_rel_ind_enea['LPENS']['Base case']:
                self.ui.indices_TW.setItem(i, 0, QtWidgets.QTableWidgetItem(elem))
                self.ui.indices_TW.setItem(i, 1, QtWidgets.QTableWidgetItem(
                    '%.4f' % ems_rel_ind_enea['LPENS']['Base case'][elem]))
                self.ui.indices_TW.setItem(i, 2, QtWidgets.QTableWidgetItem(
                    '%.4f' % ems_rel_ind_enea['LPENS']['EMS model'][elem]))
                self.ui.indices_TW.setItem(i, 3, QtWidgets.QTableWidgetItem(
                    '%.4f' % ems_rel_ind_enea['LPEIC']['Base case'][elem]))
                self.ui.indices_TW.setItem(i, 4, QtWidgets.QTableWidgetItem(
                    '%.4f' % ems_rel_ind_enea['LPEIC']['EMS model'][elem]))
                i += 1
            self.table_format()

            self.ui.indices_TW.setEnabled(True)

        self.ui.export_BTN.clicked.connect(self.results_save)

    #
    def table_format(self):
        self.ui.indices_TW.setShowGrid(False)
        self.ui.indices_TW.setStyleSheet('QTableView::item {border-top: 1px solid #333333;}')
        self.ui.indices_TW.verticalHeader().setVisible(False)

        stylesheet = "QHeaderView::section{color:rgb(251,251,251); Background-color:rgb(1,1,1); border - radius: 14 px;}"
        self.ui.indices_TW.horizontalHeader().setStyleSheet(stylesheet)

        for i in range(0, self.ui.indices_TW.rowCount()):
            for j in range(0, self.ui.indices_TW.columnCount()):
                self.ui.indices_TW.item(i, j).setForeground(QtGui.QColor(255, 255, 255))
                self.ui.indices_TW.item(i, j).setTextAlignment(QtCore.Qt.AlignCenter)

        self.ui.indices_TW.setColumnWidth(0, 200)
        for i in range(1, 5):
            self.ui.indices_TW.setColumnWidth(i, 120)
        self.ui.indices_TW.setHorizontalHeaderLabels(['Load Element', 'LPENS - Base Case', 'LPENS - EMS Model',
                                                      'LPEIC - Base Case', 'LPEIC - EMS Model'])

    #
    def results_save(self):
        import csv
        header = ['Load Element', 'LPENS - Base Case', 'LPENS - EMS Model', 'LPENS (p.u.) - Base Case',
                  'LPENS (p.u.)- EMS Model', 'LPEIC - Base Case', 'LPEIC - EMS Model', 'LPEIC (p.u.) - Base Case',
                  'LPEIC (p.u.)- EMS Model']

        bc = self. ems_rel_ind['Base case']
        emsc = self.ems_rel_ind['EMS model']

        data = []
        for elem in bc:
            line = [elem]
            for i in ['ENS', 'ENS_pu', 'ENS_cost', 'ENS_cost_pu']:
                line.append(bc[elem][i])
                line.append(emsc[elem][i])
            data.append(line)

        options = QtWidgets.QFileDialog.Options()
        options |= QtWidgets.QFileDialog.DontUseNativeDialog

        filename, _ = QtWidgets.QFileDialog.getSaveFileName(self, "Save File", '',
                                                            "CSV (*.csv)", options=options)

        if filename:
            if not filename.endswith('.csv'):
                filename = filename + '.csv'
            try:
                with open(filename, 'w', encoding='UTF8', newline='') as file:
```

```
            writer = csv.writer(file)
            writer.writerow(header)
            writer.writerows(data)
        file.close()
    except PermissionError:
        QtWidgets.QMessageBox.warning(self, 'File Access Denied', 'Impossibile accedere al file',
                                      QtWidgets.QMessageBox.Ok)
```

4.4.5.2 EMS_stoch_resultsUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'ems_stoch_resultsUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1139, 671)
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        MainWindow.setFont(font)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.widget = QtWidgets.QWidget(self.centralwidget)
        self.widget.setGeometry(QtCore.QRect(30, 20, 1101, 320))
        self.widget.setStyleSheet("background-color: rgb(0, 0, 17);")
        self.widget.setObjectName("widget")
        self.top_LM = QtWidgets.QFrame(self.widget)
        self.top_LM.setGeometry(QtCore.QRect(0, 0, 1100, 1))
        self.top_LM.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LM.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LM.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LM.setObjectName("top_LM")
        self.mid_LN = QtWidgets.QFrame(self.widget)
        self.mid_LN.setGeometry(QtCore.QRect(10, 30, 1080, 1))
        self.mid_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.mid_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.mid_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.mid_LN.setObjectName("mid_LN")
        self.caption_LBL = QtWidgets.QLabel(self.widget)
        self.caption_LBL.setGeometry(QtCore.QRect(10, 0, 450, 30))
        font = QtGui.QFont()
        font.setPointSize(9)
        font.setBold(True)
        font.setItalic(False)
        font.setWeight(75)
        self.caption_LBL.setFont(font)
        self.caption_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.caption_LBL.setObjectName("caption_LBL")
        self.btm_LN = QtWidgets.QFrame(self.widget)
        self.btm_LN.setGeometry(QtCore.QRect(20, 330, 1100, 1))
        self.btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.btm_LN.setObjectName("btm_LN")
        self.indices_TW = QtWidgets.QTableWidget(self.widget)
        self.indices_TW.setEnabled(True)
        self.indices_TW.setGeometry(QtCore.QRect(390, 50, 701, 251))
        self.indices_TW.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
        self.indices_TW.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)
        self.indices_TW.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)
        self.indices_TW.setObjectName("indices_TW")
        self.indices_TW.setColumnCount(5)
        self.indices_TW.setRowCount(0)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setHorizontalHeaderItem(2, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setHorizontalHeaderItem(3, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setHorizontalHeaderItem(4, item)
        self.ens_LBL = QtWidgets.QLabel(self.widget)
        self.ens_LBL.setGeometry(QtCore.QRect(50, 80, 60, 30))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.ens_LBL.setFont(font)
        self.ens_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.ens_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.ens_LBL.setObjectName("ens_LBL")
        self.eic_LBL = QtWidgets.QLabel(self.widget)
        self.eic_LBL.setGeometry(QtCore.QRect(50, 190, 60, 30))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.eic_LBL.setFont(font)
        self.eic_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.eic_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.eic_LBL.setObjectName("eic_LBL")
        self.ens_BC_DSB = QtWidgets.QDoubleSpinBox(self.widget)
        self.ens_BC_DSB.setEnabled(False)
        self.ens_BC_DSB.setGeometry(QtCore.QRect(120, 80, 121, 30))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.ens_BC_DSB.setFont(font)
        self.ens_BC_DSB.setStyleSheet("color: rgb(255, 255, 255);")
        self.ens_BC_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.ens_BC_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
        self.ens_BC_DSB.setDecimals(4)
        self.ens_BC_DSB.setMaximum(1000000.0)
        self.ens_BC_DSB.setProperty("value", 0.0)
        self.ens_BC_DSB.setObjectName("ens_BC_DSB")
        self.ens_EMS_DSB = QtWidgets.QDoubleSpinBox(self.widget)
        self.ens_EMS_DSB.setEnabled(False)
        self.ens_EMS_DSB.setGeometry(QtCore.QRect(260, 80, 121, 30))
```

```
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.ens_EMS_DSB.setFont(font)
self.ens_EMS_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.ens_EMS_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ens_EMS_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.ens_EMS_DSB.setDecimals(4)
self.ens_EMS_DSB.setMaximum(1000000.0)
self.ens_EMS_DSB.setProperty("value", 0.0)
self.ens_EMS_DSB.setObjectName("ens_EMS_DSB")
self.ens_EMS_pu_DSB = QtWidgets.QDoubleSpinBox(self.widget)
self.ens_EMS_pu_DSB.setEnabled(False)
self.ens_EMS_pu_DSB.setGeometry(QtCore.QRect(260, 120, 121, 30))
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.ens_EMS_pu_DSB.setFont(font)
self.ens_EMS_pu_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.ens_EMS_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ens_EMS_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.ens_EMS_pu_DSB.setDecimals(4)
self.ens_EMS_pu_DSB.setMaximum(1000000.0)
self.ens_EMS_pu_DSB.setProperty("value", 0.0)
self.ens_EMS_pu_DSB.setObjectName("ens_EMS_pu_DSB")
self.ens_BC_pu_DSB = QtWidgets.QDoubleSpinBox(self.widget)
self.ens_BC_pu_DSB.setEnabled(False)
self.ens_BC_pu_DSB.setGeometry(QtCore.QRect(120, 120, 121, 30))
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.ens_BC_pu_DSB.setFont(font)
self.ens_BC_pu_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.ens_BC_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.ens_BC_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.ens_BC_pu_DSB.setDecimals(4)
self.ens_BC_pu_DSB.setMaximum(1000000.0)
self.ens_BC_pu_DSB.setProperty("value", 0.0)
self.ens_BC_pu_DSB.setObjectName("ens_BC_pu_DSB")
self.eic_EMS_pu_DSB = QtWidgets.QDoubleSpinBox(self.widget)
self.eic_EMS_pu_DSB.setEnabled(False)
self.eic_EMS_pu_DSB.setGeometry(QtCore.QRect(260, 230, 121, 30))
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.eic_EMS_pu_DSB.setFont(font)
self.eic_EMS_pu_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.eic_EMS_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.eic_EMS_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.eic_EMS_pu_DSB.setDecimals(4)
self.eic_EMS_pu_DSB.setMaximum(1000000.0)
self.eic_EMS_pu_DSB.setProperty("value", 0.0)
self.eic_EMS_pu_DSB.setObjectName("eic_EMS_pu_DSB")
self.eic_BC_DSB = QtWidgets.QDoubleSpinBox(self.widget)
self.eic_BC_DSB.setEnabled(False)
self.eic_BC_DSB.setGeometry(QtCore.QRect(120, 190, 121, 30))
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.eic_BC_DSB.setFont(font)
self.eic_BC_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.eic_BC_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.eic_BC_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.eic_BC_DSB.setDecimals(4)
self.eic_BC_DSB.setMaximum(1000000.0)
self.eic_BC_DSB.setProperty("value", 0.0)
self.eic_BC_DSB.setObjectName("eic_BC_DSB")
self.eic_BC_pu_DSB = QtWidgets.QDoubleSpinBox(self.widget)
self.eic_BC_pu_DSB.setEnabled(False)
self.eic_BC_pu_DSB.setGeometry(QtCore.QRect(120, 230, 121, 30))
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.eic_BC_pu_DSB.setFont(font)
self.eic_BC_pu_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.eic_BC_pu_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.eic_BC_pu_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.eic_BC_pu_DSB.setDecimals(4)
self.eic_BC_pu_DSB.setMaximum(1000000.0)
self.eic_BC_pu_DSB.setProperty("value", 0.0)
self.eic_BC_pu_DSB.setObjectName("eic_BC_pu_DSB")
self.eic_EMS_DSB = QtWidgets.QDoubleSpinBox(self.widget)
self.eic_EMS_DSB.setEnabled(False)
self.eic_EMS_DSB.setGeometry(QtCore.QRect(260, 190, 121, 30))
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.eic_EMS_DSB.setFont(font)
self.eic_EMS_DSB.setStyleSheet("color: rgb(255, 255, 255);")
self.eic_EMS_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.eic_EMS_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
self.eic_EMS_DSB.setDecimals(4)
self.eic_EMS_DSB.setMaximum(1000000.0)
self.eic_EMS_DSB.setProperty("value", 0.0)
self.eic_EMS_DSB.setObjectName("eic_EMS_DSB")
self.basecase_LBL = QtWidgets.QLabel(self.widget)
self.basecase_LBL.setGeometry(QtCore.QRect(120, 50, 121, 30))
font = QtGui.QFont()
font.setPointSize(8)
font.setBold(False)
font.setItalic(True)
font.setWeight(50)
self.basecase_LBL.setFont(font)
self.basecase_LBL.setStyleSheet("color: rgb(255, 255, 255);")
self.basecase_LBL.setAlignment(QtCore.Qt.AlignCenter)
self.basecase_LBL.setObjectName("basecase_LBL")
self.EMSmodel_LBL = QtWidgets.QLabel(self.widget)
self.EMSmodel_LBL.setGeometry(QtCore.QRect(260, 50, 121, 30))
```

```
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setBold(False)
        font.setItalic(True)
        font.setWeight(50)
        self.EMSmodel_LBL.setFont(font)
        self.EMSmodel_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.EMSmodel_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.EMSmodel_LBL.setObjectName("EMSmodel_LBL")
        self.ens_pu_LBL = QtWidgets.QLabel(self.widget)
        self.ens_pu_LBL.setGeometry(QtCore.QRect(50, 120, 60, 30))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setBold(False)
        font.setItalic(True)
        font.setWeight(50)
        self.ens_pu_LBL.setFont(font)
        self.ens_pu_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.ens_pu_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.ens_pu_LBL.setObjectName("ens_pu_LBL")
        self.eic_pu_LBL = QtWidgets.QLabel(self.widget)
        self.eic_pu_LBL.setGeometry(QtCore.QRect(50, 230, 60, 30))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setBold(False)
        font.setItalic(True)
        font.setWeight(50)
        self.eic_pu_LBL.setFont(font)
        self.eic_pu_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.eic_pu_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.eic_pu_LBL.setObjectName("eic_pu_LBL")
        self.export_BTN = QtWidgets.QPushButton(self.widget)
        self.export_BTN.setGeometry(QtCore.QRect(260, 280, 121, 23))
        self.export_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.export_BTN.setObjectName("export_BTN")
        self.mid_LN.raise_()
        self.btm_LN.raise_()
        self.caption_LBL.raise_()
        self.top_LM.raise_()
        self.indices_TW.raise_()
        self.ens_LBL.raise_()
        self.eic_LBL.raise_()
        self.ens_BC_DSB.raise_()
        self.ens_EMS_DSB.raise_()
        self.ens_EMS_pu_DSB.raise_()
        self.ens_BC_pu_DSB.raise_()
        self.eic_EMS_pu_DSB.raise_()
        self.eic_BC_DSB.raise_()
        self.eic_BC_pu_DSB.raise_()
        self.eic_EMS_DSB.raise_()
        self.basecase_LBL.raise_()
        self.EMSmodel_LBL.raise_()
        self.ens_pu_LBL.raise_()
        self.eic_pu_LBL.raise_()
        self.export_BTN.raise_()
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1139, 21))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.caption_LBL.setText(_translate("MainWindow", "Indici affidabilistici: Guasti stocastici e controllo EMS"))
        self.indices_TW.setSortingEnabled(True)
        item = self.indices_TW.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Load Element"))
        item = self.indices_TW.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "LPENS - Base Case"))
        item = self.indices_TW.horizontalHeaderItem(2)
        item.setText(_translate("MainWindow", "LPENS - EMS Model"))
        item = self.indices_TW.horizontalHeaderItem(3)
        item.setText(_translate("MainWindow", "LPEIC - Base Case"))
        item = self.indices_TW.horizontalHeaderItem(4)
        item.setText(_translate("MainWindow", "LPEIC - EMS Model"))
        self.ens_LBL.setText(_translate("MainWindow", "ENS"))
        self.eic_LBL.setText(_translate("MainWindow", "EIC"))
        self.basecase_LBL.setText(_translate("MainWindow", "Caso Base"))
        self.EMSmodel_LBL.setText(_translate("MainWindow", "Modello EMS"))
        self.ens_pu_LBL.setText(_translate("MainWindow", "p.u."))
        self.eic_pu_LBL.setText(_translate("MainWindow", "p.u."))
        self.export_BTN.setText(_translate("MainWindow", "Esporta dati"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

4.4.6 Protections

4.4.6.1 protections.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
import yaml
import os
from __shared__ import variables as v

from .protectionsUI import Ui_MainWindow


class Protections(QtWidgets.QMainWindow):
    def __init__(self):
        super(Protections, self).__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        res_path = os.path.join(v.project_folder, 'protections.yml')
        self.prot_res = yaml.safe_load((open(res_path)))

        cost_min = 0
        cost_max = 0
        cost_curr = 0
        i = 0

        self.ui.indices_TW.clear()
        self.ui.indices_TW.setRowCount(len(self.prot_res))

        for elem in self.prot_res:
            self.ui.indices_TW.setItem(i, 0, QtWidgets.QTableWidgetItem(elem))
            self.ui.indices_TW.setItem(i, 1, QtWidgets.QTableWidgetItem(
                '%.1f A' % self.prot_res[elem]['results']['comparison']['em']['overcurrent']))
            self.ui.indices_TW.setItem(i, 2, QtWidgets.QTableWidgetItem(
                '%.4f kV' % self.prot_res[elem]['results']['comparison']['em']['overvoltage']))
            self.ui.indices_TW.setItem(i, 3, QtWidgets.QTableWidgetItem(
                '€ %.2f' % self.prot_res[elem]['results']['comparison']['em']['cost']))
            self.ui.indices_TW.setItem(i, 4, QtWidgets.QTableWidgetItem(
                '%.1f A' % self.prot_res[elem]['results']['comparison']['el']['overcurrent']))
            self.ui.indices_TW.setItem(i, 5, QtWidgets.QTableWidgetItem(
                '%.4f kV' % self.prot_res[elem]['results']['comparison']['el']['overvoltage']))
            self.ui.indices_TW.setItem(i, 6, QtWidgets.QTableWidgetItem(
                '€ %.2f' % self.prot_res[elem]['results']['comparison']['el']['cost']))
            i += 1
            cost_curr = cost_curr + self.prot_res[elem]['results']['cost']
            cost_min = cost_min + self.prot_res[elem]['results']['comparison']['em']['cost']
            cost_max = cost_max + self.prot_res[elem]['results']['comparison']['el']['cost']

        self.table_format()
        self.ui.indices_TW.setEnabled(True)

        self.ui.cost_max_DSB.setValue(cost_max)
        self.ui.cost_min_DSB.setValue(cost_min)
        self.ui.cost_curr_DSB.setValue(cost_curr)

        self.ui.export_BTN.clicked.connect(self.results_save)

    #
    def table_format(self):
        self.ui.indices_TW.setShowGrid(False)
        self.ui.indices_TW.setStyleSheet('QTableView::item {border-top: 1px solid #333333;}')
        self.ui.indices_TW.verticalHeader().setVisible(False)

        stylesheet = \
            "QHeaderView::section{color:rgb(251,251,251); Background-color:rgb(1,1,1); border - radius: 14 px;}"
        self.ui.indices_TW.horizontalHeader().setStyleSheet(stylesheet)

        for i in range(0, self.ui.indices_TW.rowCount()):
            for j in range(0, self.ui.indices_TW.columnCount()):
                self.ui.indices_TW.item(i, j).setForeground(QtGui.QColor(255, 255, 255))
                self.ui.indices_TW.item(i, j).setTextAlignment(QtCore.Qt.AlignCenter)
                element = self.ui.indices_TW.item(i, 0).text()
                if self.prot_res[element]['results']['type'] == "Interruttore elettronico":
                    columns = [1, 2, 3]
                else:
                    columns = [4, 5, 6]
                for k in columns:
                    self.ui.indices_TW.item(i, k).setForeground((QtGui.QColor(96, 96, 96)))

        self.ui.indices_TW.setColumnWidth(0, 200)
        for i in range(1, 7):
            self.ui.indices_TW.setColumnWidth(i, 100)
        self.ui.indices_TW.setHorizontalHeaderLabels(['Elemento', 'Sovracorrente', 'Sovratensione', ' Costo',
                                                      'Sovracorrente', 'Sovratensione', ' Costo'])

    #
    def results_save(self):
        import csv
        header = ['Load Element', 'LPENS - Base Case', 'LPENS - EMS Model', 'LPENS (p.u.) - Base Case',
                  'LPENS (p.u.)- EMS Model', 'LPEIC - Base Case', 'LPEIC - EMS Model', 'LPEIC (p.u.) - Base Case',
                  'LPEIC (p.u.)- EMS Model']

        bc = self. ems_rel_ind['Base case']
        emsc = self.ems_rel_ind['EMS model']

        data = []
        for elem in bc:
            line = [elem]
            for i in ['ENS', 'ENS_pu', 'ENS_cost', 'ENS_cost_pu']:
                line.append(bc[elem][i])
                line.append(emsc[elem][i])
            data.append(line)

        options = QtWidgets.QFileDialog.Options()
        options |= QtWidgets.QFileDialog.DontUseNativeDialog

        filename = None

        filename, _ = QtWidgets.QFileDialog.getSaveFileName(self, "Save File", '',
                                                            "CSV (*.csv)", options=options)
```

```
if filename:
    if not filename.endswith('.csv'):
        filename = filename + '.csv'
    try:
        with open(filename, 'w', encoding='UTF8', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(header)
            writer.writerows(data)
        file.close()
    except PermissionError:
        QtWidgets.QMessageBox.warning(self, 'File Access Denied', 'Impossibile accedere al file',
                                      QtWidgets.QMessageBox.Ok)
```

4.4.6.2 protectionsUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'protectionsUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1139, 671)
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        MainWindow.setFont(font)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.widget = QtWidgets.QWidget(self.centralwidget)
        self.widget.setGeometry(QtCore.QRect(30, 20, 1101, 320))
        self.widget.setStyleSheet("background-color: rgb(0, 0, 17);")
        self.widget.setObjectName("widget")
        self.top_LM = QtWidgets.QFrame(self.widget)
        self.top_LM.setGeometry(QtCore.QRect(0, 0, 1100, 1))
        self.top_LM.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LM.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LM.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LM.setObjectName("top_LM")
        self.mid_LN = QtWidgets.QFrame(self.widget)
        self.mid_LN.setGeometry(QtCore.QRect(10, 30, 1080, 1))
        self.mid_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.mid_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.mid_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.mid_LN.setObjectName("mid_LN")
        self.caption_LBL = QtWidgets.QLabel(self.widget)
        self.caption_LBL.setGeometry(QtCore.QRect(10, 0, 450, 30))
        font = QtGui.QFont()
        font.setPointSize(9)
        font.setBold(True)
        font.setItalic(False)
        font.setWeight(75)
        self.caption_LBL.setFont(font)
        self.caption_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.caption_LBL.setObjectName("caption_LBL")
        self.btm_LN = QtWidgets.QFrame(self.widget)
        self.btm_LN.setGeometry(QtCore.QRect(20, 330, 1100, 1))
        self.btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.btm_LN.setObjectName("btm_LN")
        self.indices_TW = QtWidgets.QTableWidget(self.widget)
        self.indices_TW.setEnabled(True)
        self.indices_TW.setGeometry(QtCore.QRect(270, 70, 821, 231))
        self.indices_TW.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
        self.indices_TW.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)
        self.indices_TW.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)
        self.indices_TW.setObjectName("indices_TW")
        self.indices_TW.setColumnCount(7)
        self.indices_TW.setRowCount(3)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setVerticalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setVerticalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setVerticalHeaderItem(2, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setHorizontalHeaderItem(2, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setHorizontalHeaderItem(3, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setHorizontalHeaderItem(4, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setHorizontalHeaderItem(5, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setHorizontalHeaderItem(6, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(0, 0, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(0, 1, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(0, 2, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(0, 3, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(0, 4, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(0, 5, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(0, 6, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(1, 0, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(1, 1, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(1, 2, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(1, 3, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(1, 4, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(1, 5, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(1, 6, item)
```

```
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(2, 0, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(2, 1, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(2, 2, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(2, 3, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(2, 4, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(2, 5, item)
        item = QtWidgets.QTableWidgetItem()
        self.indices_TW.setItem(2, 6, item)
        self.cost_min_LBL = QtWidgets.QLabel(self.widget)
        self.cost_min_LBL.setGeometry(QtCore.QRect(19, 80, 91, 30))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.cost_min_LBL.setFont(font)
        self.cost_min_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.cost_min_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.cost_min_LBL.setObjectName("cost_min_LBL")
        self.cost_curr_LBL = QtWidgets.QLabel(self.widget)
        self.cost_curr_LBL.setGeometry(QtCore.QRect(19, 190, 91, 30))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.cost_curr_LBL.setFont(font)
        self.cost_curr_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.cost_curr_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.cost_curr_LBL.setObjectName("cost_curr_LBL")
        self.cost_min_DSB = QtWidgets.QDoubleSpinBox(self.widget)
        self.cost_min_DSB.setEnabled(False)
        self.cost_min_DSB.setGeometry(QtCore.QRect(120, 80, 121, 30))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.cost_min_DSB.setFont(font)
        self.cost_min_DSB.setStyleSheet("color: rgb(255, 255, 255);")
        self.cost_min_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.cost_min_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
        self.cost_min_DSB.setDecimals(2)
        self.cost_min_DSB.setMaximum(99999999.99)
        self.cost_min_DSB.setProperty("value", 0.0)
        self.cost_min_DSB.setObjectName("cost_min_DSB")
        self.cost_max_DSB = QtWidgets.QDoubleSpinBox(self.widget)
        self.cost_max_DSB.setEnabled(False)
        self.cost_max_DSB.setGeometry(QtCore.QRect(120, 120, 121, 30))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.cost_max_DSB.setFont(font)
        self.cost_max_DSB.setStyleSheet("color: rgb(255, 255, 255);")
        self.cost_max_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.cost_max_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
        self.cost_max_DSB.setDecimals(2)
        self.cost_max_DSB.setMaximum(99999999.99)
        self.cost_max_DSB.setProperty("value", 0.0)
        self.cost_max_DSB.setObjectName("cost_max_DSB")
        self.cost_curr_DSB = QtWidgets.QDoubleSpinBox(self.widget)
        self.cost_curr_DSB.setEnabled(False)
        self.cost_curr_DSB.setGeometry(QtCore.QRect(120, 190, 121, 30))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.cost_curr_DSB.setFont(font)
        self.cost_curr_DSB.setStyleSheet("color: rgb(255, 255, 255);")
        self.cost_curr_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.cost_curr_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
        self.cost_curr_DSB.setDecimals(2)
        self.cost_curr_DSB.setMaximum(99999999.99)
        self.cost_curr_DSB.setProperty("value", 0.0)
        self.cost_curr_DSB.setObjectName("cost_curr_DSB")
        self.el_LBL = QtWidgets.QLabel(self.widget)
        self.el_LBL.setGeometry(QtCore.QRect(770, 40, 300, 30))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setBold(False)
        font.setItalic(True)
        font.setWeight(50)
        self.el_LBL.setFont(font)
        self.el_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.el_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.el_LBL.setObjectName("el_LBL")
        self.export_BTN = QtWidgets.QPushButton(self.widget)
        self.export_BTN.setGeometry(QtCore.QRect(120, 280, 121, 23))
        self.export_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.export_BTN.setObjectName("export_BTN")
        self.cost_max_LBL = QtWidgets.QLabel(self.widget)
        self.cost_max_LBL.setGeometry(QtCore.QRect(20, 120, 91, 30))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.cost_max_LBL.setFont(font)
        self.cost_max_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.cost_max_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.cost_max_LBL.setObjectName("cost_max_LBL")
        self.em_LBL = QtWidgets.QLabel(self.widget)
        self.em_LBL.setGeometry(QtCore.QRect(470, 40, 300, 30))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setBold(False)
        font.setItalic(True)
        font.setWeight(50)
        self.em_LBL.setFont(font)
        self.em_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.em_LBL.setAlignment(QtCore.Qt.AlignCenter)
```

```
        self.em_LBL.setObjectName("em_LBL")
        self.vline_2 = QtWidgets.QFrame(self.widget)
        self.vline_2.setGeometry(QtCore.QRect(770, 40, 1, 260))
        self.vline_2.setStyleSheet("color: rgb(255, 255, 255);\n"
"background-color: rgb(255, 255, 255);")
        self.vline_2.setFrameShape(QtWidgets.QFrame.VLine)
        self.vline_2.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vline_2.setObjectName("vline_2")
        self.vline_1 = QtWidgets.QFrame(self.widget)
        self.vline_1.setGeometry(QtCore.QRect(470, 40, 1, 260))
        self.vline_1.setStyleSheet("color: rgb(255, 255, 255);\n"
"background-color: rgb(255, 255, 255);")
        self.vline_1.setFrameShape(QtWidgets.QFrame.VLine)
        self.vline_1.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vline_1.setObjectName("vline_1")
        self.vline_4 = QtWidgets.QFrame(self.widget)
        self.vline_4.setGeometry(QtCore.QRect(1070, 40, 1, 260))
        self.vline_4.setStyleSheet("color: rgb(255, 255, 255);\n"
"background-color: rgb(255, 255, 255);")
        self.vline_4.setFrameShape(QtWidgets.QFrame.VLine)
        self.vline_4.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.vline_4.setObjectName("vline_4")
        self.mid_LN.raise_()
        self.btm_LN.raise_()
        self.caption_LBL.raise_()
        self.top_LM.raise_()
        self.indices_TW.raise_()
        self.cost_min_LBL.raise_()
        self.cost_curr_LBL.raise_()
        self.cost_min_DSB.raise_()
        self.cost_max_DSB.raise_()
        self.cost_curr_DSB.raise_()
        self.el_LBL.raise_()
        self.export_BTN.raise_()
        self.cost_max_LBL.raise_()
        self.em_LBL.raise_()
        self.vline_2.raise_()
        self.vline_1.raise_()
        self.vline_4.raise_()
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1139, 22))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.caption_LBL.setText(_translate("MainWindow", "Indici affidabilistici: Guasti stocastici e controllo EMS"))
        self.indices_TW.setSortingEnabled(True)
        item = self.indices_TW.verticalHeaderItem(0)
        item.setText(_translate("MainWindow", "1"))
        item = self.indices_TW.verticalHeaderItem(1)
        item.setText(_translate("MainWindow", "2"))
        item = self.indices_TW.verticalHeaderItem(2)
        item.setText(_translate("MainWindow", "3"))
        item = self.indices_TW.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Elemento"))
        item = self.indices_TW.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Sovracorrente"))
        item = self.indices_TW.horizontalHeaderItem(2)
        item.setText(_translate("MainWindow", "Sovratensione"))
        item = self.indices_TW.horizontalHeaderItem(3)
        item.setText(_translate("MainWindow", "Costo"))
        item = self.indices_TW.horizontalHeaderItem(4)
        item.setText(_translate("MainWindow", "Sovracorrente"))
        item = self.indices_TW.horizontalHeaderItem(5)
        item.setText(_translate("MainWindow", "Sovratensione"))
        item = self.indices_TW.horizontalHeaderItem(6)
        item.setText(_translate("MainWindow", "Costo"))
        __sortingEnabled = self.indices_TW.isSortingEnabled()
        self.indices_TW.setSortingEnabled(False)
        item = self.indices_TW.item(0, 0)
        item.setText(_translate("MainWindow", "nome"))
        item = self.indices_TW.item(0, 1)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(0, 2)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(0, 3)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(0, 4)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(0, 5)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(0, 6)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(1, 0)
        item.setText(_translate("MainWindow", "nome"))
        item = self.indices_TW.item(1, 1)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(1, 2)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(1, 3)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(1, 4)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(1, 5)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(1, 6)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(2, 0)
        item.setText(_translate("MainWindow", "nome"))
        item = self.indices_TW.item(2, 1)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(2, 2)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(2, 3)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(2, 4)
```

```
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(2, 5)
        item.setText(_translate("MainWindow", "0"))
        item = self.indices_TW.item(2, 6)
        item.setText(_translate("MainWindow", "0"))
        self.indices_TW.setSortingEnabled(__sortingEnabled)
        self.cost_min_LBL.setText(_translate("MainWindow", "Costo Min"))
        self.cost_curr_LBL.setText(_translate("MainWindow", "Costo Sol."))
        self.cost_min_DSB.setPrefix(_translate("MainWindow", "€ "))
        self.cost_max_DSB.setPrefix(_translate("MainWindow", "€ "))
        self.cost_curr_DSB.setPrefix(_translate("MainWindow", "€ "))
        self.el_LBL.setText(_translate("MainWindow", "Protezioni elettroniche"))
        self.export_BTN.setText(_translate("MainWindow", "Esporta dati"))
        self.cost_max_LBL.setText(_translate("MainWindow", "Costo Max"))
        self.em_LBL.setText(_translate("MainWindow", "Protezioni elettromeccaniche"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

4.4.7 Reliability

4.4.7.1 reliability_results.py

```
from PyQt5 import QtWidgets
from .reliability_resultsUI import Ui_MainWindow
from __shared__ import variables as v


class ReliabilityResults(QtWidgets.QMainWindow):
    def __init__(self):
        super(ReliabilityResults, self).__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        self.ui.comp_export_BTN.clicked.connect(self.save_comp)
        self.ui.loads_export_BTN.clicked.connect(self.save_loads)

    #
    def save_comp(self):
        import csv
        header = ['Elementi', 'lambda', 'R', 'MTBF_ore', 'MTBF_anni']

        data = []
        for element in v.elements:
            if v.elements[element]['category'] not in ['AC-Node', 'DC-Node', 'Extc-Grid']:
                line = [element]
                for i in range(1, 5):
                    line.append(v.elements[element]['reliability']['results'][header[i]])
                data.append(line)

        options = QtWidgets.QFileDialog.Options()
        filename, _ = QtWidgets.QFileDialog.getSaveFileName(self, "Save File", '',
                                                            "CSV (*.csv)", options=options)
        if filename:
            if not filename.endswith('.csv'):
                filename = filename + '.csv'
            try:
                with open(filename, 'w', encoding='UTF8', newline='') as file:
                    writer = csv.writer(file)
                    writer.writerow(header)
                    writer.writerows(data)
                file.close()
            except PermissionError:
                QtWidgets.QMessageBox.warning(self, 'File Access Denied', 'Impossibile accedere al file',
                                              QtWidgets.QMessageBox.Ok)

    #
    def save_loads(self):
        import csv
        header = ['Elementi', 't', 'R(t)']

        data = []
        for element in v.elements:
            if v.elements[element]['category'] in ['AC-Load', 'DC-Load']:
                line = [element, self.ui.load_time_DSB.value(),
                        v.elements[element]['reliability']['results']['load_rel']]
                data.append(line)

        options = QtWidgets.QFileDialog.Options()

        filename, _ = QtWidgets.QFileDialog.getSaveFileName(self, "Save File", '',
                                                            "CSV (*.csv)", options=options)
        if filename:
            if not filename.endswith('.csv'):
                filename = filename + '.csv'
            try:
                with open(filename, 'w', encoding='UTF8', newline='') as file:
                    writer = csv.writer(file)
                    writer.writerow(header)
                    writer.writerows(data)
                file.close()
            except PermissionError:
                QtWidgets.QMessageBox.warning(self, 'File Access Denied', 'Impossibile accedere al file',
                                              QtWidgets.QMessageBox.Ok)
```

4.4.7.2 reliability_resultsUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'reliability_resultsUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1105, 365)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.widget = QtWidgets.QWidget(self.centralwidget)
        self.widget.setGeometry(QtCore.QRect(0, 0, 1101, 320))
        self.widget.setStyleSheet("background-color: rgb(0, 0, 17);")
        self.widget.setObjectName("widget")
        self.top_LN = QtWidgets.QFrame(self.widget)
        self.top_LN.setGeometry(QtCore.QRect(0, 0, 1100, 1))
        self.top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.top_LN.setObjectName("top_LN")
        self.mid_sx_LN = QtWidgets.QFrame(self.widget)
        self.mid_sx_LN.setGeometry(QtCore.QRect(10, 30, 600, 1))
        self.mid_sx_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.mid_sx_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.mid_sx_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.mid_sx_LN.setObjectName("mid_sx_LN")
        self.comp_rel_LBL = QtWidgets.QLabel(self.widget)
        self.comp_rel_LBL.setGeometry(QtCore.QRect(10, 0, 181, 30))
        font = QtGui.QFont()
        font.setPointSize(9)
        font.setBold(True)
        font.setItalic(False)
        font.setWeight(75)
        self.comp_rel_LBL.setFont(font)
        self.comp_rel_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.comp_rel_LBL.setObjectName("comp_rel_LBL")
        self.btm_LN = QtWidgets.QFrame(self.widget)
        self.btm_LN.setGeometry(QtCore.QRect(0, 320, 1100, 1))
        self.btm_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btm_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.btm_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.btm_LN.setObjectName("btm_LN")
        self.loads_WGT = QtWidgets.QWidget(self.widget)
        self.loads_WGT.setGeometry(QtCore.QRect(640, 40, 441, 281))
        self.loads_WGT.setObjectName("loads_WGT")
        self.loads_rel_TW = QtWidgets.QTableWidget(self.loads_WGT)
        self.loads_rel_TW.setGeometry(QtCore.QRect(0, 0, 301, 271))
        self.loads_rel_TW.setStyleSheet("color: rgb(255, 255, 255);")
        self.loads_rel_TW.setObjectName("loads_rel_TW")
        self.loads_rel_TW.setColumnCount(2)
        self.loads_rel_TW.setRowCount(0)
        item = QtWidgets.QTableWidgetItem()
        self.loads_rel_TW.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.loads_rel_TW.setHorizontalHeaderItem(1, item)
        self.loads_rel_TW.verticalHeader().setVisible(False)
        self.load_time_DSB = QtWidgets.QDoubleSpinBox(self.loads_WGT)
        self.load_time_DSB.setEnabled(False)
        self.load_time_DSB.setGeometry(QtCore.QRect(340, 20, 61, 21))
        self.load_time_DSB.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.load_time_DSB.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButtons)
        self.load_time_DSB.setDecimals(0)
        self.load_time_DSB.setMaximum(10000000.0)
        self.load_time_DSB.setObjectName("load_time_DSB")
        self.loads_time_LBL = QtWidgets.QLabel(self.loads_WGT)
        self.loads_time_LBL.setGeometry(QtCore.QRect(310, 20, 21, 21))
        self.loads_time_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.loads_time_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.loads_time_LBL.setObjectName("loads_time_LBL")
        self.loads_time_unit_LBL = QtWidgets.QLabel(self.loads_WGT)
        self.loads_time_unit_LBL.setGeometry(QtCore.QRect(410, 20, 21, 21))
        self.loads_time_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.loads_time_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
        self.loads_time_unit_LBL.setObjectName("loads_time_unit_LBL")
        self.loads_res_WGT = QtWidgets.QWidget(self.loads_WGT)
        self.loads_res_WGT.setGeometry(QtCore.QRect(310, 180, 131, 91))
        self.loads_res_WGT.setObjectName("loads_res_WGT")
        self.loads_name_LBL = QtWidgets.QLabel(self.loads_res_WGT)
        self.loads_name_LBL.setGeometry(QtCore.QRect(0, 0, 131, 21))
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        self.loads_name_LBL.setFont(font)
        self.loads_name_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.loads_name_LBL.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.loads_name_LBL.setFrameShadow(QtWidgets.QFrame.Plain)
        self.loads_name_LBL.setMidLineWidth(0)
        self.loads_name_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.loads_name_LBL.setObjectName("loads_name_LBL")
        self.loads_rel_unit_LBL = QtWidgets.QLabel(self.loads_res_WGT)
        self.loads_rel_unit_LBL.setGeometry(QtCore.QRect(111, 30, 20, 21))
        self.loads_rel_unit_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.loads_rel_unit_LBL.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
        self.loads_rel_unit_LBL.setObjectName("loads_rel_unit_LBL")
        self.loads_rel_LE = QtWidgets.QLineEdit(self.loads_res_WGT)
        self.loads_rel_LE.setEnabled(False)
        self.loads_rel_LE.setGeometry(QtCore.QRect(30, 30, 71, 21))
        self.loads_rel_LE.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.loads_rel_LE.setObjectName("loads_rel_LE")
        self.loads_rel_LBL = QtWidgets.QLabel(self.loads_res_WGT)
        self.loads_rel_LBL.setGeometry(QtCore.QRect(0, 30, 21, 21))
        self.loads_rel_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.loads_rel_LBL.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
```

```
        self.loads_rel_LBL.setObjectName("loads_rel_LBL")
        self.loads_grafo_BTN = QtWidgets.QPushButton(self.loads_res_WGT)
        self.loads_grafo_BTN.setGeometry(QtCore.QRect(0, 70, 131, 23))
        self.loads_grafo_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.loads_grafo_BTN.setObjectName("loads_grafo_BTN")
        self.comp_rel_TW = QtWidgets.QTableWidget(self.widget)
        self.comp_rel_TW.setGeometry(QtCore.QRect(20, 40, 581, 271))
        self.comp_rel_TW.setStyleSheet("color: rgb(255, 255, 255);")
        self.comp_rel_TW.setObjectName("comp_rel_TW")
        self.comp_rel_TW.setColumnCount(5)
        self.comp_rel_TW.setRowCount(0)
        item = QtWidgets.QTableWidgetItem()
        self.comp_rel_TW.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.comp_rel_TW.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        self.comp_rel_TW.setHorizontalHeaderItem(2, item)
        item = QtWidgets.QTableWidgetItem()
        self.comp_rel_TW.setHorizontalHeaderItem(3, item)
        item = QtWidgets.QTableWidgetItem()
        self.comp_rel_TW.setHorizontalHeaderItem(4, item)
        self.comp_rel_TW.verticalHeader().setVisible(False)
        self.loads_rel_cap_LBL = QtWidgets.QLabel(self.widget)
        self.loads_rel_cap_LBL.setGeometry(QtCore.QRect(650, 0, 181, 30))
        font = QtGui.QFont()
        font.setPointSize(9)
        font.setBold(True)
        font.setItalic(False)
        font.setWeight(75)
        self.loads_rel_cap_LBL.setFont(font)
        self.loads_rel_cap_LBL.setStyleSheet("color: rgb(255, 255, 255);")
        self.loads_rel_cap_LBL.setObjectName("loads_rel_cap_LBL")
        self.mid_dx_LN = QtWidgets.QFrame(self.widget)
        self.mid_dx_LN.setGeometry(QtCore.QRect(630, 30, 461, 1))
        self.mid_dx_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.mid_dx_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.mid_dx_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.mid_dx_LN.setObjectName("mid_dx_LN")
        self.comp_export_BTN = QtWidgets.QPushButton(self.widget)
        self.comp_export_BTN.setGeometry(QtCore.QRect(520, 5, 80, 23))
        self.comp_export_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.comp_export_BTN.setObjectName("comp_export_BTN")
        self.loads_export_BTN = QtWidgets.QPushButton(self.widget)
        self.loads_export_BTN.setGeometry(QtCore.QRect(1010, 5, 80, 23))
        self.loads_export_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.loads_export_BTN.setObjectName("loads_export_BTN")
        self.loads_rel_cap_LBL.raise_()
        self.mid_sx_LN.raise_()
        self.comp_rel_LBL.raise_()
        self.top_LM.raise_()
        self.loads_WGT.raise_()
        self.comp_rel_TW.raise_()
        self.btm_LN.raise_()
        self.mid_dx_LN.raise_()
        self.comp_export_BTN.raise_()
        self.loads_export_BTN.raise_()
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1105, 22))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.comp_rel_LBL.setText(_translate("MainWindow", "Affidabilità dei componenti"))
        item = self.loads_rel_TW.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Elemento"))
        item = self.loads_rel_TW.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Affidabilità"))
        self.loads_time_LBL.setText(_translate("MainWindow", "t"))
        self.loads_time_unit_LBL.setText(_translate("MainWindow", "h"))
        self.loads_name_LBL.setText(_translate("MainWindow", "item"))
        self.loads_rel_unit_LBL.setText(_translate("MainWindow", "-"))
        self.loads_rel_LBL.setText(_translate("MainWindow", "R(t)"))
        self.loads_grafo_BTN.setText(_translate("MainWindow", "Visualizza schema"))
        item = self.comp_rel_TW.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Elemento"))
        item = self.comp_rel_TW.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Lambda"))
        item = self.comp_rel_TW.horizontalHeaderItem(2)
        item.setText(_translate("MainWindow", "t"))
        item = self.comp_rel_TW.horizontalHeaderItem(3)
        item.setText(_translate("MainWindow", "MTBT (ore)"))
        item = self.comp_rel_TW.horizontalHeaderItem(4)
        item.setText(_translate("MainWindow", "MTBF (anni)"))
        self.loads_rel_cap_LBL.setText(_translate("MainWindow", "Affidabilità delle forniture"))
        self.comp_export_BTN.setText(_translate("MainWindow", "Esporta dati"))
        self.loads_export_BTN.setText(_translate("MainWindow", "Esporta dati"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

4.5 SpalshScreen

4.5.1.1 splash.py

```
import os
import sys

from PyQt5 import QtWidgets, QtGui, QtCore
from .splashScreenUI import Ui_SplashScreen
from .widgets.functionalities import Functionalities
from .widgets.open import Open
from .widgets.logics import Logics
from .widgets.specificlogics import SpecificLogics
from .widgets.scenarios import Scenario
from UI._general.Popup.Configurations.configurations import Configurations

from functools import partial
import yaml
from __shared__ import variables as v


class Splash(QtWidgets.QMainWindow):
    def __init__(self):
        super(Splash, self).__init__()
        self.ui = Ui_SplashScreen()
        self.ui.setupUi(self)

        # inizializzazione delle variabili di classe
        self.protections_projects = []
        self.reliability_projects = []
        self.ctrl_proj = []
        self.ctrl_proj_names = []
        self.myProjects = None
        self.list_VB = None
        self.list_WG = None
        self.project = None
        self.prj_list = None
        self.bench_dict = dict()

        # self.ws = Webservices()

        func_class = Functionalities()
        self.func_ui = func_class.ui
        self.func_WG = self.func_ui.functionalities_WG

        open_class = Open()
        self.open_ui = open_class.ui
        self.open_WG = self.open_ui.widget

        scenario_class = Scenario()
        self.scenario_ui = scenario_class.ui
        self.scenario_WG = self.scenario_ui.widget

        logics_class = Logics()
        self.logics_ui = logics_class.ui
        self.logics_WG = self.logics_ui.logics_WG

        specific_logics_class = SpecificLogics()
        self.specific_logics_ui = specific_logics_class.ui
        self.specific_logics_WG = self.specific_logics_ui.specificlogics_WG

        self.func_ui.protectionsLogo_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/Pr_80x80.png"))
        self.func_ui.reliabilityLogo_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/Re_80x80.png"))
        self.func_ui.logicsLogo_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/Lc_80x80.png"))
        # self.func_ui.emsLogo_LBL.setPixmap(QtGui.QPixmap("UI/SplashScreen/res/EMS_80x80.png"))
        self.logics_ui.logSpecLogo_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/LogicheSpecifiche_300x141.png"))
        self.logics_ui.logMultLogo_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/LogicheMultiple_300x141.png"))
        self.specific_logics_ui.controlsLogo_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/Co_80x80.png"))
        self.specific_logics_ui.emsLogo_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/EMS_80x80.png"))
        self.specific_logics_ui.energyindexesLogo_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/Ei_80x80.png"))

        self.ui.neplan_img_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/Neplan_141x61.png"))
        self.ui.pf_img_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/PowerFactory_141x61.png"))
        self.ui.logo_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/ORAT.png"))
        self.ui.bg_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/RAT_sfondo.jpg"))
        self.ui.home_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/Home_31x31.png"))
        self.ui.back_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/back_31x31.png"))
        self.ui.back_LBL.setVisible(False)

        self.ui.widgets_SWG.addWidget(self.open_WG)
        self.ui.widgets_SWG.addWidget(self.func_WG)
        # self.ui.widgets_SWG.addWidget(self.scenario_WG)
        # self.ui.widgets_SWG.addWidget(self.logics_WG)
        self.ui.widgets_SWG.addWidget(self.specific_logics_WG)

        # self.show()

        self.scenario_reset()

        self.ui.widgets_SWG.setCurrentIndex(0)

        self.check_software()

        print(v.config)

        self.create_list()

        self.ui.home_LBL.mousePressEvent = self.home
        self.ui.back_LBL.mousePressEvent = self.back
        self.ui.config_BTN.clicked.connect(self.configuration)

        self.func_ui.logicsLogo_LBL.mousePressEvent = self.logics_selected
        self.func_ui.logics_LBL.mousePressEvent = self.logics_selected
        self.func_ui.protectionsLogo_LBL.mousePressEvent = self.protections_selected
        self.func_ui.protections_LBL.mousePressEvent = self.protections_selected
        self.func_ui.reliabilityLogo_LBL.mousePressEvent = self.reliability_selected
        self.func_ui.reliability_LBL.mousePressEvent = self.reliability_selected

        self.logics_ui.logSpecLogo_LBL.mousePressEvent = self.specific_logics_selected
        self.logics_ui.logMultLogo_LBL.mousePressEvent = self.multiple_logics_selected
```

```
        self.specific_logics_ui.controlsLogo_LBL.mousePressEvent = self.controls_selected
        self.specific_logics_ui.controls_LBL.mousePressEvent = self.controls_selected
        self.specific_logics_ui.emsLogo_LBL.mousePressEvent = self.ems_selected
        self.specific_logics_ui.ems_LBL.mousePressEvent = self.ems_selected
        self.specific_logics_ui.energyindexesLogo_LBL.mousePressEvent = self.energy_indexes_selected
        self.specific_logics_ui.energyindexes_LBL.mousePressEvent = self.energy_indexes_selected

        self.open_ui.neplan_BTN.clicked.connect(self.open_Neplan)
        self.open_ui.digsilent_BTN.clicked.connect(self.open_PowerFactory)
        self.scenario_ui.select_BTN.clicked.connect(self.scenario_confirm)
        for elem in ['underground', 'roadservices', 'residential', 'y2020', 'y2030', 'y2040', 'bc', 'dec']:
            # self.scenario_ui.__getattribute__(elem + '_RB').setChecked(False)
            self.scenario_ui.__getattribute__(elem + '_RB').clicked.connect(self.scenarios)

    # Nasconde i pulsanti delle funzionalità (FORSE NON PIù USATA)
    def hide_buttons(self):
        pass

    # Inizializzazione della lista delle reti benchmark
    def create_list(self):
        self.list_WG = QtWidgets.QWidget()
        self.list_WG.setGeometry(QtCore.QRect(0, 0, 360, 500))
        self.list_VB = QtWidgets.QVBoxLayout(self.list_WG)
        self.list_VB.setSpacing(12)

        self.bench_dict = yaml.safe_load(open('__benchmark__/bench_grids.yml'))

        self.prj_list = []
        i = 0
        for project in self.bench_dict:
            if self.bench_dict[project]['pf']['exists'] or self.bench_dict[project]['neplan']['exists'] or \
                    self.bench_dict[project]['controls']['exists']:
                self.prj_list.append(FileList(project))
                self.prj_list[i].bench_name.setText(project)
                self.prj_list[i].bench_description.setText(self.bench_dict[project]['scenario'])

                # self.prj_list[i].widget.mousePressEvent = self.azione
                self.prj_list[i].button.clicked.connect(partial(self.selected_benchmark, project))

                self.list_VB.addWidget(self.prj_list[i].widget)
                # self.prj_list[i].widget.parentWidget()
                i += 1

        self.list_WG.setGeometry(QtCore.QRect(0, 0, 351, len(self.prj_list)*75))
        self.open_ui.bench_SA.setWidget(self.list_WG)

    #
    def selected_benchmark(self, name):
        v.features['name'] = name
        print(name)
        # v.source = 'benchmark'
        v.features['source'] = 'benchmark'
        v.features['neplan'] = self.bench_dict[name]['neplan']
        v.features['pf'] = self.bench_dict[name]['pf']
        # v.neplan_source = self.bench_dict[name]['neplan']['name']
        # v.digsilent_source = self.bench_dict[name]['digsilent']

        self.ui.widgets_SWG.setCurrentIndex(1)
        self.ui.back_LBL.setVisible(True)
        if self.bench_dict[name]['pf']['exists']:
            v.software = 'pf'
        else:
            v.software = 'neplan'
        v.project_folder = os.path.join(os.getcwd(), '_data', name)
        self.functionalities_activation()

    def functionalities_activation(self):
        self.func_ui.reliabilityShadow_LBL.setVisible(not self.bench_dict[v.features['name']]['pf']['exists'])
        self.specific_logics_ui.controlsShadow_LBL.setVisible(
            not self.bench_dict[v.features['name']]['controls']['exists'])
        no_one = not (self.bench_dict[v.features['name']]['pf']['exists'] or
                      self.bench_dict[v.features['name']]['neplan']['exists'])
        self.func_ui.protectionsShadow_LBL.setVisible(no_one)
        self.specific_logics_ui.emsShadow_LBL.setVisible(no_one)
        self.specific_logics_ui.energyindexesShadow_LBL.setVisible(no_one)

    #
    def file_import(self):
        pass

    def logics_selected(self, void):
        self.ui.widgets_SWG.setCurrentIndex(2)

    def protections_selected(self, void):
        v.functionality = "Protections"
        v.lm = False
        self.close()

    def reliability_selected(self, void):
        v.functionality = "Reliability"
        v.lm = False
        self.close()

    def specific_logics_selected(self, void):
        self.ui.widgets_SWG.setCurrentIndex(2)

    def multiple_logics_selected(self, void):
        pass

    def controls_selected(self, command):
        v.functionality = "Controls"
        v.lm = True
        self.close()

    def ems_selected(self, void):
        v.functionality = "EMS"
        v.lm = True
        self.close()

    def energy_indexes_selected(self, void):
        v.functionality = "Energy Indexes"
        v.lm = True
        self.close()

    def scenarios(self):
```

```
        y = None
        conf = ''

        if self.scenario_ui.underground_RB.isChecked():
            v.area = 'Underground'
        elif self.scenario_ui.roadservices_RB.isChecked():
            v.aera = 'Road Services'
        elif self.scenario_ui.residential_RB.isChecked():
            v.area = 'Residential'

        self.scenario_ui.year_WGT.setVisible(v.area is not None)
        # self.scenario_ui.cus.cust_WGT.setVisible(v.area is not None)

        if self.scenario_ui.y2020_RB.isChecked():
            y = 2020
        elif self.scenario_ui.y2030_RB.isChecked():
            y = 2030
        elif self.scenario_ui.y2040_RB.isChecked():
            y = 2040

        self.scenario_ui.config_WGT.setVisible(self.scenario_ui.y2040_RB.isChecked() or
                                               self.scenario_ui.y2030_RB.isChecked())

        if self.scenario_ui.bc_RB.isChecked():
            conf = 'BC'
        elif self.scenario_ui.dec_RB.isChecked():
            conf = 'DEC'

        if y and v.area and (conf or y == 2020):
            self.scenario_ui.select_BTN.setVisible(True)
            v.scenario = str(y) + str(conf)

    def scenario_reset(self):
        v.scenario = None
        for elem in ['underground', 'roadservices', 'residential', 'y2020', 'y2030', 'y2040', 'bc', 'dec']:
            self.scenario_ui.__getattribute__(elem + '_RB').setChecked(False)
        self.scenario_ui.config_WGT.setVisible(False)
        self.scenario_ui.select_BTN.setVisible(False)
        self.scenario_ui.year_WGT.setVisible(False)
        v.area = None

    def scenario_confirm(self):
        print(v.scenario)
        self.close()

    #
    def home(self, event):
        self.ui.widgets_SWG.setCurrentIndex(0)
        self.ui.back_LBL.setVisible(False)

    def back(self, event):
        self.ui.widgets_SWG.setCurrentIndex(self.ui.widgets_SWG.currentIndex()-1)
        if self.ui.widgets_SWG.currentIndex() == 0:
            self.ui.back_LBL.setVisible(False)

    def benchmarks(self):
        names = []
        scenarios = []
        self.bench_dict = yaml.safe_load(open('__benchmark__/bench_grids.yml'))
        for name in self.bench_dict:
            names.append(name)
            scenarios.append(self.bench_dict[name]['scenario'])
        return names, scenarios

    # def chk_bench_soft(self):
    #     for name in self.bench_dict:

    def open_PowerFactory(self):
        path = v.config['pf']['path'].replace('/PowerFactory.exe', '')
        sys.path.append(path + '/Python/3.7')
        import powerfactory as pfactory
        app = pfactory.GetApplication()

        # Da elimiare ----------------------------------
        user = app.GetCurrentUser()
        print(user)
        # projects = user.GetContents('*.IntPrj', 0)
        # print(projects)
        projects = []
        for project in user.GetContents('*.IntPrj', 0):
            print(project.loc_name)
            projects.append(project.loc_name)
        print('end')
        # QtWidgets.QInputDialog.Da

        options = QtWidgets.QFileDialog.Options()
        options |= QtWidgets.QFileDialog.DontUseNativeDialog

        sel_proj, ok_pressed = QtWidgets.QInputDialog.getItem(self, 'Titolo', 'label', projects, 0, False)
        # items = (['Ryan', '24', 'm'], ['Lisa', '22', 'f'], ['Joe', '30', 'm'])
        # items = [" ".join(item) for item in items]  # <<<-----<
        # item, okPressed = QtWidgets.QInputDialog.getItem(self, "Get item", "Color:", projects, 0, False)
        if ok_pressed:
            print(sel_proj)
            v.software = 'pf'
            v.features = dict()
            v.features['name'] = sel_proj
            # print(name)
            # v.source = 'benchmark'
            v.features['source'] = 'custom'
            v.features['pf'] = dict()
            for soft in ['neplan', 'controls']:
                v.features[soft] = dict()
                v.features[soft]['exists'] = False
                v.features[soft]['project_name'] = ''
            v.features['pf']['exists'] = True
            v.features['pf']['folder_name'] = ''
            v.features['pf']['project_name'] = sel_proj
            v.features['pf']['study_case_name'] = ''

            self.bench_dict[sel_proj] = dict()
            self.bench_dict[sel_proj]['neplan'] = v.features['neplan']
            self.bench_dict[sel_proj]['pf'] = v.features['pf']
            self.bench_dict[sel_proj]['controls'] = v.features['controls']
```

```
            self.ui.widgets_SWG.setCurrentIndex(1)
            self.ui.back_LBL.setVisible(True)
            v.project_folder = os.path.join(os.getcwd(), '_data', sel_proj)
            self.functionalities_activation()
        pass

    def open_Neplan(self):
        file, void = QtWidgets.QFileDialog.getOpenFileUrl(self, 'TITOLO', QtCore.QUrl(), '*.nep360')
        pass

    def configuration(self):
        conf = Configurations()
        if conf.exec_():
            print('fatto')
        self.software_images()
        self.create_list()

    def check_software(self):
        conf = Configurations()
        with open(os.getcwd() + '/__shared__/config.uti', 'w') as file:
            documents = yaml.dump(v.config, file)
        with open(os.getcwd() + '/__benchmark__/bench_grids.yml', 'w') as file:
            documents = yaml.dump(conf.bench_grids, file)
        # conf.echo = False
        # v.config = yaml.safe_load(open(os.getcwd() + '/__shared__/config.uti'))
        # if v.config['neplan']['activated']:
        #     conf.neplan_verify()
        #     print('Neplan: ' + str(v.config['neplan']['working']))
        # if v.config['pf']['activated']:
        #     conf.pf_verify()
        self.software_images()

    def software_images(self):
        self.ui.neplan_img_LBL.setVisible(v.config['neplan']['working'] and v.config['neplan']['activated'])
        self.open_ui.neplan_BTN.setVisible(v.config['neplan']['working'] and v.config['neplan']['activated'])
        self.ui.pf_img_LBL.setVisible(v.config['pf']['working'] and v.config['pf']['activated'])
        self.open_ui.digsilent_BTN.setVisible(v.config['pf']['working'] and v.config['pf']['activated'])


class FileList:
    def __init__(self, name):
        font1 = QtGui.QFont()
        font1.setFamily("Calibri")
        font1.setPointSize(16)
        font1.setBold(True)
        font1.setWeight(75)

        font2 = QtGui.QFont()
        font2.setFamily("Calibri")
        font2.setPointSize(12)
        font2.setBold(True)
        font2.setWeight(75)

        self.bench_name = QtWidgets.QLabel('Nome del progetto')
        self.bench_name.setGeometry(0, 0, 20, 51)
        self.bench_name.setFont(font1)
        self.bench_name.setStyleSheet("color: rgb(173, 185, 202);")

        self.bench_description = QtWidgets.QLabel('descrizione')
        self.bench_description.setGeometry(0, 0, 20, 21)
        self.bench_description.setFont(font2)
        self.bench_description.setStyleSheet("color: rgb(173, 185, 202);")

        self.icon_LBL = QtWidgets.QLabel()
        self.icon_LBL.setText('')
        self.icon_LBL.setGeometry(0, 0, 51, 31)
        self.icon_LBL.setPixmap(QtGui.QPixmap("_images/SplashScreen/open_61x61.png"))

        self.widget = QtWidgets.QWidget()
        self.widget.setGeometry(0, 0, 100, 41)

        self.h_layout = QtWidgets.QHBoxLayout(self.widget)
        self.h_layout.addWidget(self.icon_LBL, 1)

        self.v_layout = QtWidgets.QVBoxLayout()
        self.v_layout.setSpacing(0)
        self.v_layout.addWidget(self.bench_name)
        self.v_layout.addWidget(self.bench_description)

        self.h_layout.addLayout(self.v_layout, 5)

        self.button = QtWidgets.QPushButton(self.widget)
        self.button.setGeometry(0, 10, 351, 51)
        self.button.setStyleSheet("background-color: rgb(255, 255, 255, 0);")

    #     self.icon_LBL.mousePressEvent = self.action1
    #
    # def action1(self, event):
    #     # self.exp = self.bench_name
    #     # print('Esportato dalla classe: ' + str(self.exp))
    #     print('1')
    #     v.filename = self.exp
```

4.5.1.2 spalshScreenUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'splashScreenUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_SplashScreen(object):
    def setupUi(self, SplashScreen):
        SplashScreen.setObjectName("SplashScreen")
        SplashScreen.resize(843, 441)
        SplashScreen.setMinimumSize(QtCore.QSize(843, 441))
        SplashScreen.setMaximumSize(QtCore.QSize(843, 441))
        self.centralwidget = QtWidgets.QWidget(SplashScreen)
        self.centralwidget.setEnabled(True)
        self.centralwidget.setObjectName("centralwidget")
        self.buttonsFrame = QtWidgets.QFrame(self.centralwidget)
        self.buttonsFrame.setGeometry(QtCore.QRect(410, 0, 431, 441))
        self.buttonsFrame.setStyleSheet("background-image: url(:/newPrefix/RAT_sfondo.jpg);")
        self.buttonsFrame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.buttonsFrame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.buttonsFrame.setObjectName("buttonsFrame")
        self.home_LBL = QtWidgets.QLabel(self.buttonsFrame)
        self.home_LBL.setGeometry(QtCore.QRect(390, 10, 31, 31))
        self.home_LBL.setText("")
        self.home_LBL.setPixmap(QtGui.QPixmap("res/Home_31x31.png"))
        self.home_LBL.setObjectName("home_LBL")
        self.widgets_SWG = QtWidgets.QStackedWidget(self.buttonsFrame)
        self.widgets_SWG.setGeometry(QtCore.QRect(20, 10, 351, 341))
        self.widgets_SWG.setStyleSheet("selection-color: rgb(255, 255, 255);")
        self.widgets_SWG.setObjectName("widgets_SWG")
        self.back_LBL = QtWidgets.QLabel(self.buttonsFrame)
        self.back_LBL.setGeometry(QtCore.QRect(390, 60, 31, 31))
        self.back_LBL.setText("")
        self.back_LBL.setPixmap(QtGui.QPixmap("res/back_31x31.png"))
        self.back_LBL.setObjectName("back_LBL")
        self.config_BTN = QtWidgets.QPushButton(self.buttonsFrame)
        self.config_BTN.setGeometry(QtCore.QRect(10, 370, 111, 61))
        self.config_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.config_BTN.setObjectName("config_BTN")
        self.neplan_img_LBL = QtWidgets.QLabel(self.buttonsFrame)
        self.neplan_img_LBL.setGeometry(QtCore.QRect(130, 370, 141, 61))
        self.neplan_img_LBL.setText("")
        self.neplan_img_LBL.setPixmap(QtGui.QPixmap("res/Neplan_141x61.png"))
        self.neplan_img_LBL.setObjectName("neplan_img_LBL")
        self.pf_img_LBL = QtWidgets.QLabel(self.buttonsFrame)
        self.pf_img_LBL.setGeometry(QtCore.QRect(280, 370, 141, 61))
        self.pf_img_LBL.setText("")
        self.pf_img_LBL.setPixmap(QtGui.QPixmap("res/PowerFactory_141x61.png"))
        self.pf_img_LBL.setObjectName("pf_img_LBL")
        self.logoFrame = QtWidgets.QFrame(self.centralwidget)
        self.logoFrame.setGeometry(QtCore.QRect(0, 0, 401, 471))
        self.logoFrame.setStyleSheet("QFrame{\n"
"    background-color: rgb(25, 10, 60);\n"
"}")
        self.logoFrame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.logoFrame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.logoFrame.setObjectName("logoFrame")
        self.logoText_LBL = QtWidgets.QLabel(self.logoFrame)
        self.logoText_LBL.setGeometry(QtCore.QRect(10, 270, 381, 51))
        font = QtGui.QFont()
        font.setFamily("Segoe UI")
        font.setPointSize(20)
        self.logoText_LBL.setFont(font)
        self.logoText_LBL.setObjectName("logoText_LBL")
        self.logo_LBL = QtWidgets.QLabel(self.logoFrame)
        self.logo_LBL.setGeometry(QtCore.QRect(50, 10, 291, 254))
        self.logo_LBL.setText("")
        self.logo_LBL.setPixmap(QtGui.QPixmap("res/RAT.png"))
        self.logo_LBL.setObjectName("logo_LBL")
        self.log_LBL = QtWidgets.QLabel(self.logoFrame)
        self.log_LBL.setGeometry(QtCore.QRect(0, 330, 401, 21))
        self.log_LBL.setStyleSheet("background-color: rgb(103, 75, 113);\n"
"color: rgb(236, 221, 241);")
        self.log_LBL.setText("")
        self.log_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.log_LBL.setObjectName("log_LBL")
        self.lineFrame = QtWidgets.QFrame(self.centralwidget)
        self.lineFrame.setGeometry(QtCore.QRect(401, 0, 5, 471))
        self.lineFrame.setStyleSheet("QFrame{\n"
"    background-color: rgb(103, 75, 113);\n"
"}")
        self.lineFrame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.lineFrame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.lineFrame.setObjectName("lineFrame")
        self.backgroundFrame = QtWidgets.QFrame(self.centralwidget)
        self.backgroundFrame.setGeometry(QtCore.QRect(0, 0, 941, 481))
        self.backgroundFrame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.backgroundFrame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.backgroundFrame.setObjectName("backgroundFrame")
        self.bg_LBL = QtWidgets.QLabel(self.backgroundFrame)
        self.bg_LBL.setGeometry(QtCore.QRect(0, 0, 1081, 471))
        self.bg_LBL.setText("")
        self.bg_LBL.setPixmap(QtGui.QPixmap("res/RAT_sfondo.jpg"))
        self.bg_LBL.setObjectName("bg_LBL")
        self.backgroundFrame.raise_()
        self.logoFrame.raise_()
        self.lineFrame.raise_()
        self.buttonsFrame.raise_()
        SplashScreen.setCentralWidget(self.centralwidget)

        self.retranslateUi(SplashScreen)
        self.widgets_SWG.setCurrentIndex(-1)
        QtCore.QMetaObject.connectSlotsByName(SplashScreen)
```

```
    def retranslateUi(self, SplashScreen):
        _translate = QtCore.QCoreApplication.translate
        SplashScreen.setWindowTitle(_translate("SplashScreen", "ORAtool - Optimization and Reliability Assessment Tool"))
        self.config_BTN.setText(_translate("SplashScreen", "Configura\n"
"Software\n"
"Esterni"))
        self.logoText_LBL.setToolTip(_translate("SplashScreen", "<html><head/><body><p>utyfvoiygupioughpiouòklnm</p></body></html>"))
        self.logoText_LBL.setText(_translate("SplashScreen", "<html><head/><body><p align=\"center\"><span style=\" font-size:14pt;
font-weight:600; color:#afabab;\">Optimization</span><span style=\" font-size:14pt; color:#afabab;\"> and <br/></span><span style=\"
font-family:\'Gadugi\'; font-size:14pt; font-weight:600; color:#8e7f91;\">Reliability</span><span style=\" font-family:\'Gadugi\';
font-size:14pt; font-weight:600; color:#7f8c9b;\">Assessment</span><span style=\" font-family:\'Gadugi\'; font-size:14pt; font-
weight:600; color:#867e99;\">Tool</span></p></body></html>"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    SplashScreen = QtWidgets.QMainWindow()
    ui = Ui_SplashScreen()
    ui.setupUi(SplashScreen)
    SplashScreen.show()
    sys.exit(app.exec_())
```

4.5.1.3 widgets

4.5.1.4 functionalities.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .functionalitiesUI import Ui_main


class Functionalities(QtWidgets.QMainWindow):
    def __init__(self):
        super(Functionalities, self).__init__()
        self.ui = Ui_main()
        self.ui.setupUi(self)
```

4.5.1.5 functionalitiesUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'functionalitiesUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_main(object):
    def setupUi(self, main):
        main.setObjectName("main")
        main.resize(387, 421)
        self.functionalities_WG = QtWidgets.QWidget(main)
        self.functionalities_WG.setGeometry(QtCore.QRect(0, 0, 351, 341))
        self.functionalities_WG.setObjectName("functionalities_WG")
        self.logicsLogo_LBL = QtWidgets.QLabel(self.functionalities_WG)
        self.logicsLogo_LBL.setGeometry(QtCore.QRect(0, 0, 81, 81))
        self.logicsLogo_LBL.setText("")
        self.logicsLogo_LBL.setPixmap(QtGui.QPixmap("../res/Lc_80x80.png"))
        self.logicsLogo_LBL.setObjectName("logicsLogo_LBL")
        self.logics_LBL = QtWidgets.QLabel(self.functionalities_WG)
        self.logics_LBL.setGeometry(QtCore.QRect(90, 0, 261, 81))
        self.logics_LBL.setStyleSheet("")
        self.logics_LBL.setObjectName("logics_LBL")
        self.reliability_LBL = QtWidgets.QLabel(self.functionalities_WG)
        self.reliability_LBL.setGeometry(QtCore.QRect(90, 100, 261, 81))
        self.reliability_LBL.setObjectName("reliability_LBL")
        self.reliabilityLogo_LBL = QtWidgets.QLabel(self.functionalities_WG)
        self.reliabilityLogo_LBL.setGeometry(QtCore.QRect(0, 100, 81, 81))
        self.reliabilityLogo_LBL.setText("")
        self.reliabilityLogo_LBL.setPixmap(QtGui.QPixmap("../res/Af_80x80.png"))
        self.reliabilityLogo_LBL.setObjectName("reliabilityLogo_LBL")
        self.protections_LBL = QtWidgets.QLabel(self.functionalities_WG)
        self.protections_LBL.setGeometry(QtCore.QRect(90, 200, 261, 81))
        self.protections_LBL.setObjectName("protections_LBL")
        self.protectionsLogo_LBL = QtWidgets.QLabel(self.functionalities_WG)
        self.protectionsLogo_LBL.setGeometry(QtCore.QRect(0, 200, 81, 81))
        self.protectionsLogo_LBL.setText("")
        self.protectionsLogo_LBL.setPixmap(QtGui.QPixmap("../res/Pr_80x80.png"))
        self.protectionsLogo_LBL.setObjectName("protectionsLogo_LBL")
        self.protectionsShadow_LBL = QtWidgets.QLabel(self.functionalities_WG)
        self.protectionsShadow_LBL.setGeometry(QtCore.QRect(0, 200, 351, 81))
        self.protectionsShadow_LBL.setStyleSheet("QFrame{\n"
"    background-color: rgba(25, 10, 60, 195);\n"
"}")
        self.protectionsShadow_LBL.setText("")
        self.protectionsShadow_LBL.setObjectName("protectionsShadow_LBL")
        self.reliabilityShadow_LBL = QtWidgets.QLabel(self.functionalities_WG)
        self.reliabilityShadow_LBL.setGeometry(QtCore.QRect(0, 100, 351, 81))
        self.reliabilityShadow_LBL.setStyleSheet("QFrame{\n"
"    background-color: rgba(25, 10, 60, 195);\n"
"}")
        self.reliabilityShadow_LBL.setText("")
        self.reliabilityShadow_LBL.setObjectName("reliabilityShadow_LBL")

        self.retranslateUi(main)
        QtCore.QMetaObject.connectSlotsByName(main)

    def retranslateUi(self, main):
        _translate = QtCore.QCoreApplication.translate
        main.setWindowTitle(_translate("main", "Form"))
        self.logics_LBL.setText(_translate("main", "<html><head/><body><p><span style=\" font-family:\'Calibri\'; font-size:18pt;
font-weight:600; color:#adb9ca;\">Logiche di Controllo</span><br/><span style=\" font-family:\'Calibri\'; font-size:12pt;
color:#adb9ca;\">Logiche Specifiche e Logiche Multiple</span></p></body></html>"))
        self.reliability_LBL.setText(_translate("main", "<html><head/><body><p><span style=\" font-family:\'Calibri\'; font-
size:18pt; font-weight:600; color:#adb9ca;\">Calcolo dell\'Affidabilità</span><span style=\" font-family:\'Calibri\'; font-size:18pt;
color:#adb9ca;\"><br/></span><span style=\" font-family:\'Calibri\'; font-size:12pt; color:#adb9ca;\">Affidabilità della rete e dei
componenti</span></p></body></html>"))
        self.protections_LBL.setText(_translate("main", "<html><head/><body><p><span style=\" font-family:\'Calibri\'; font-
size:18pt; font-weight:600; color:#adb9ca;\">Protezioni</span><span style=\" font-family:\'Calibri\'; font-size:18pt;
color:#adb9ca;\"><br/></span><span style=\" font-family:\'Calibri\'; font-size:12pt; color:#adb9ca;\">Valutazioni tecniche ed
economiche</span></p></body></html>"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    main = QtWidgets.QWidget()
    ui = Ui_main()
    ui.setupUi(main)
    main.show()
    sys.exit(app.exec_())
```

4.5.1.6 logics.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .logicsUI import Ui_main


class Logics(QtWidgets.QMainWindow):
    def __init__(self):
        super(Logics, self).__init__()
        self.ui = Ui_main()
        self.ui.setupUi(self)
```

4.5.1.7 logicsUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'logicsUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_main(object):
    def setupUi(self, main):
        main.setObjectName("main")
        main.resize(351, 421)
        self.logics_WG = QtWidgets.QWidget(main)
        self.logics_WG.setGeometry(QtCore.QRect(0, 0, 351, 341))
        self.logics_WG.setObjectName("logics_WG")
        self.emsLogo_LBL = QtWidgets.QLabel(self.logics_WG)
        self.emsLogo_LBL.setGeometry(QtCore.QRect(0, 300, 81, 81))
        self.emsLogo_LBL.setText("")
        self.emsLogo_LBL.setPixmap(QtGui.QPixmap("../res/EMS.png"))
        self.emsLogo_LBL.setObjectName("emsLogo_LBL")
        self.logSpecLogo_LBL = QtWidgets.QLabel(self.logics_WG)
        self.logSpecLogo_LBL.setGeometry(QtCore.QRect(20, 40, 300, 141))
        self.logSpecLogo_LBL.setText("")
        self.logSpecLogo_LBL.setPixmap(QtGui.QPixmap("../res/logicheSpecifiche_300x141.png"))
        self.logSpecLogo_LBL.setObjectName("logSpecLogo_LBL")
        self.logMultLogo_LBL = QtWidgets.QLabel(self.logics_WG)
        self.logMultLogo_LBL.setGeometry(QtCore.QRect(20, 190, 300, 141))
        self.logMultLogo_LBL.setText("")
        self.logMultLogo_LBL.setPixmap(QtGui.QPixmap("../res/logicheMultiple_300x141.png"))
        self.logMultLogo_LBL.setObjectName("logMultLogo_LBL")
        self.logContr_LBL = QtWidgets.QLabel(self.logics_WG)
        self.logContr_LBL.setGeometry(QtCore.QRect(20, 0, 301, 31))
        self.logContr_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.logContr_LBL.setObjectName("logContr_LBL")
        self.logMultiShadow_LBL = QtWidgets.QLabel(self.logics_WG)
        self.logMultiShadow_LBL.setGeometry(QtCore.QRect(20, 190, 301, 141))
        self.logMultiShadow_LBL.setStyleSheet("QFrame{\n"
"    background-color: rgba(25, 10, 60, 195);\n"
"}")
        self.logMultiShadow_LBL.setText("")
        self.logMultiShadow_LBL.setObjectName("logMultiShadow_LBL")

        self.retranslateUi(main)
        QtCore.QMetaObject.connectSlotsByName(main)

    def retranslateUi(self, main):
        _translate = QtCore.QCoreApplication.translate
        main.setWindowTitle(_translate("main", "Form"))
        self.logContr_LBL.setText(_translate("main", "<html><head/><body><p><span style=\" font-family:\'Calibri\'; font-size:18pt;
font-weight:600; color:#adb9ca;\">Logiche di Controllo</span></p></body></html>"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    main = QtWidgets.QWidget()
    ui = Ui_main()
    ui.setupUi(main)
    main.show()
    sys.exit(app.exec_())
```

4.5.1.8 open.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .openUI import Ui_main


class Open(QtWidgets.QMainWindow):
    def __init__(self):
        super(Open, self).__init__()
        self.ui = Ui_main()
        self.ui.setupUi(self)
```

4.5.1.9 openUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'openUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_main(object):
    def setupUi(self, main):
        main.setObjectName("main")
        main.resize(351, 535)
        self.widget = QtWidgets.QWidget(main)
        self.widget.setGeometry(QtCore.QRect(0, 0, 351, 341))
        self.widget.setObjectName("widget")
        self.projectsLoad_LBL = QtWidgets.QLabel(self.widget)
        self.projectsLoad_LBL.setGeometry(QtCore.QRect(0, 280, 351, 20))
        self.projectsLoad_LBL.setObjectName("projectsLoad_LBL")
        self.digsilent_BTN = QtWidgets.QPushButton(self.widget)
        self.digsilent_BTN.setGeometry(QtCore.QRect(0, 310, 161, 23))
        self.digsilent_BTN.setStyleSheet("QPushButton{\n"
"    border-radius: 7px;\n"
"    background-color: rgb(194, 214, 236);\n"
"    selection-background-color: rgb(0, 0, 127);\n"
"}")
        self.digsilent_BTN.setObjectName("digsilent_BTN")
        self.neplan_BTN = QtWidgets.QPushButton(self.widget)
        self.neplan_BTN.setGeometry(QtCore.QRect(190, 310, 161, 23))
        self.neplan_BTN.setStyleSheet("QPushButton{\n"
"    border-radius: 7px;\n"
"    background-color: rgb(194, 214, 236);\n"
"    selection-background-color: rgb(0, 0, 127);\n"
"}")
        self.neplan_BTN.setObjectName("neplan_BTN")
        self.projectsLoad_LBL_2 = QtWidgets.QLabel(self.widget)
        self.projectsLoad_LBL_2.setGeometry(QtCore.QRect(0, 0, 351, 20))
        self.projectsLoad_LBL_2.setObjectName("projectsLoad_LBL_2")
        self.bench_SA = QtWidgets.QScrollArea(self.widget)
        self.bench_SA.setGeometry(QtCore.QRect(0, 30, 351, 241))
        self.bench_SA.setStyleSheet("background-color: rgb(25, 10, 60, 127);")
        self.bench_SA.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)
        self.bench_SA.setWidgetResizable(False)
        self.bench_SA.setObjectName("bench_SA")
        self.scrollAreaWidgetContents = QtWidgets.QWidget()
        self.scrollAreaWidgetContents.setGeometry(QtCore.QRect(0, 0, 349, 299))
        self.scrollAreaWidgetContents.setObjectName("scrollAreaWidgetContents")
        self.bench_SA.setWidget(self.scrollAreaWidgetContents)

        self.retranslateUi(main)
        QtCore.QMetaObject.connectSlotsByName(main)

    def retranslateUi(self, main):
        _translate = QtCore.QCoreApplication.translate
        main.setWindowTitle(_translate("main", "Form"))
        self.projectsLoad_LBL.setText(_translate("main", "<html><head/><body><p align=\"center\"><span style=\" font-weight:600;
font-style:italic; color:#c2d6ec;\">oppure carica un modello di rete</span></p></body></html>"))
        self.digsilent_BTN.setText(_translate("main", "DIgSilent PowerFactory"))
        self.neplan_BTN.setText(_translate("main", "Neplan"))
        self.projectsLoad_LBL_2.setText(_translate("main", "<html><head/><body><p align=\"center\"><span style=\" font-weight:600;
font-style:italic; color:#c2d6ec;\">Seleziona una rete benchmark</span></p></body></html>"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    main = QtWidgets.QWidget()
    ui = Ui_main()
    ui.setupUi(main)
    main.show()
    sys.exit(app.exec_())
```

4.5.1.10 scenarios.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .scenariosUI import Ui_main


class Scenario(QtWidgets.QMainWindow):
    def __init__(self):
        super(Scenario, self).__init__()
        self.ui = Ui_main()
        self.ui.setupUi(self)
```

4.5.1.11 scenariosUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'scenariosUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.4
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_main(object):
    def setupUi(self, main):
        main.setObjectName("main")
        main.resize(589, 629)
        self.widget = QtWidgets.QWidget(main)
        self.widget.setGeometry(QtCore.QRect(0, 0, 351, 391))
        self.widget.setStyleSheet("background-color: rgb(25, 10, 60, 255);")
        self.widget.setObjectName("widget")
        self.scenariosSelect_LBL = QtWidgets.QLabel(self.widget)
        self.scenariosSelect_LBL.setGeometry(QtCore.QRect(10, 0, 331, 20))
        self.scenariosSelect_LBL.setObjectName("scenariosSelect_LBL")
        self.select_BTN = QtWidgets.QPushButton(self.widget)
        self.select_BTN.setGeometry(QtCore.QRect(20, 340, 81, 23))
        self.select_BTN.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.select_BTN.setObjectName("select_BTN")
        self.area_WGT = QtWidgets.QWidget(self.widget)
        self.area_WGT.setGeometry(QtCore.QRect(10, 50, 151, 161))
        self.area_WGT.setStyleSheet("")
        self.area_WGT.setObjectName("area_WGT")
        self.area_top_LN = QtWidgets.QFrame(self.area_WGT)
        self.area_top_LN.setGeometry(QtCore.QRect(0, 0, 311, 3))
        self.area_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.area_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.area_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.area_top_LN.setObjectName("area_top_LN")
        self.area_mid_LN = QtWidgets.QFrame(self.area_WGT)
        self.area_mid_LN.setGeometry(QtCore.QRect(0, 20, 311, 1))
        self.area_mid_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.area_mid_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.area_mid_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.area_mid_LN.setObjectName("area_mid_LN")
        self.area_bottom_LN = QtWidgets.QFrame(self.area_WGT)
        self.area_bottom_LN.setGeometry(QtCore.QRect(0, 150, 311, 3))
        self.area_bottom_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.area_bottom_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.area_bottom_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.area_bottom_LN.setObjectName("area_bottom_LN")
        self.area_LBL = QtWidgets.QLabel(self.area_WGT)
        self.area_LBL.setGeometry(QtCore.QRect(0, 0, 151, 20))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setItalic(True)
        self.area_LBL.setFont(font)
        self.area_LBL.setStyleSheet("color: rgb(220, 233, 255);")
        self.area_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.area_LBL.setObjectName("area_LBL")
        self.underground_RB = QtWidgets.QRadioButton(self.area_WGT)
        self.underground_RB.setGeometry(QtCore.QRect(10, 30, 131, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.underground_RB.setFont(font)
        self.underground_RB.setStyleSheet("color: rgb(220, 233, 255);")
        self.underground_RB.setObjectName("underground_RB")
        self.roadservices_RB = QtWidgets.QRadioButton(self.area_WGT)
        self.roadservices_RB.setGeometry(QtCore.QRect(10, 70, 131, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.roadservices_RB.setFont(font)
        self.roadservices_RB.setStyleSheet("color: rgb(220, 233, 255);")
        self.roadservices_RB.setObjectName("roadservices_RB")
        self.residential_RB = QtWidgets.QRadioButton(self.area_WGT)
        self.residential_RB.setGeometry(QtCore.QRect(10, 110, 131, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.residential_RB.setFont(font)
        self.residential_RB.setStyleSheet("color: rgb(220, 233, 255);")
        self.residential_RB.setObjectName("residential_RB")
        self.area_LBL.raise_()
        self.area_top_LN.raise_()
        self.area_mid_LN.raise_()
        self.area_bottom_LN.raise_()
        self.underground_RB.raise_()
        self.roadservices_RB.raise_()
        self.residential_RB.raise_()
        self.config_WGT = QtWidgets.QWidget(self.widget)
        self.config_WGT.setGeometry(QtCore.QRect(10, 230, 331, 75))
        self.config_WGT.setStyleSheet("")
        self.config_WGT.setObjectName("config_WGT")
        self.config_top_LN = QtWidgets.QFrame(self.config_WGT)
        self.config_top_LN.setGeometry(QtCore.QRect(0, 0, 331, 3))
        self.config_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.config_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.config_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.config_top_LN.setObjectName("config_top_LN")
        self.config_mid_LN = QtWidgets.QFrame(self.config_WGT)
        self.config_mid_LN.setGeometry(QtCore.QRect(0, 20, 331, 1))
        self.config_mid_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.config_mid_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.config_mid_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.config_mid_LN.setObjectName("config_mid_LN")
        self.config_bottom_LN = QtWidgets.QFrame(self.config_WGT)
        self.config_bottom_LN.setGeometry(QtCore.QRect(0, 70, 331, 3))
```

```
        self.config_bottom_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.config_bottom_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.config_bottom_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.config_bottom_LN.setObjectName("config_bottom_LN")
        self.config_LBL = QtWidgets.QLabel(self.config_WGT)
        self.config_LBL.setGeometry(QtCore.QRect(0, 0, 331, 20))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setItalic(True)
        self.config_LBL.setFont(font)
        self.config_LBL.setStyleSheet("color: rgb(220, 233, 255);")
        self.config_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.config_LBL.setObjectName("config_LBL")
        self.bc_RB = QtWidgets.QRadioButton(self.config_WGT)
        self.bc_RB.setGeometry(QtCore.QRect(10, 30, 151, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.bc_RB.setFont(font)
        self.bc_RB.setStyleSheet("color: rgb(220, 233, 255);")
        self.bc_RB.setObjectName("bc_RB")
        self.dec_RB = QtWidgets.QRadioButton(self.config_WGT)
        self.dec_RB.setGeometry(QtCore.QRect(180, 30, 151, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.dec_RB.setFont(font)
        self.dec_RB.setStyleSheet("color: rgb(220, 233, 255);")
        self.dec_RB.setObjectName("dec_RB")
        self.config_LBL.raise_()
        self.config_top_LN.raise_()
        self.config_mid_LN.raise_()
        self.config_bottom_LN.raise_()
        self.bc_RB.raise_()
        self.dec_RB.raise_()
        self.year_WGT = QtWidgets.QWidget(self.widget)
        self.year_WGT.setGeometry(QtCore.QRect(190, 50, 151, 161))
        self.year_WGT.setStyleSheet("")
        self.year_WGT.setObjectName("year_WGT")
        self.year_top_LN = QtWidgets.QFrame(self.year_WGT)
        self.year_top_LN.setGeometry(QtCore.QRect(0, 0, 150, 3))
        self.year_top_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.year_top_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.year_top_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.year_top_LN.setObjectName("year_top_LN")
        self.year_mid_LN = QtWidgets.QFrame(self.year_WGT)
        self.year_mid_LN.setGeometry(QtCore.QRect(0, 20, 150, 1))
        self.year_mid_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.year_mid_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.year_mid_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.year_mid_LN.setObjectName("year_mid_LN")
        self.year_bottom_LN = QtWidgets.QFrame(self.year_WGT)
        self.year_bottom_LN.setGeometry(QtCore.QRect(0, 150, 150, 3))
        self.year_bottom_LN.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.year_bottom_LN.setFrameShape(QtWidgets.QFrame.HLine)
        self.year_bottom_LN.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.year_bottom_LN.setObjectName("year_bottom_LN")
        self.year_LBL = QtWidgets.QLabel(self.year_WGT)
        self.year_LBL.setGeometry(QtCore.QRect(0, 0, 151, 20))
        font = QtGui.QFont()
        font.setPointSize(8)
        font.setItalic(True)
        self.year_LBL.setFont(font)
        self.year_LBL.setStyleSheet("color: rgb(220, 233, 255);")
        self.year_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.year_LBL.setObjectName("year_LBL")
        self.y2020_RB = QtWidgets.QRadioButton(self.year_WGT)
        self.y2020_RB.setGeometry(QtCore.QRect(10, 30, 80, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.y2020_RB.setFont(font)
        self.y2020_RB.setStyleSheet("color: rgb(220, 233, 255);")
        self.y2020_RB.setObjectName("y2020_RB")
        self.y2030_RB = QtWidgets.QRadioButton(self.year_WGT)
        self.y2030_RB.setGeometry(QtCore.QRect(10, 70, 80, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.y2030_RB.setFont(font)
        self.y2030_RB.setStyleSheet("color: rgb(220, 233, 255);")
        self.y2030_RB.setObjectName("y2030_RB")
        self.y2040_RB = QtWidgets.QRadioButton(self.year_WGT)
        self.y2040_RB.setGeometry(QtCore.QRect(10, 110, 80, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.y2040_RB.setFont(font)
        self.y2040_RB.setStyleSheet("color: rgb(220, 233, 255);")
        self.y2040_RB.setObjectName("y2040_RB")
        self.year_LBL.raise_()
        self.year_top_LN.raise_()
        self.year_mid_LN.raise_()
        self.year_bottom_LN.raise_()
        self.y2020_RB.raise_()
        self.y2030_RB.raise_()
        self.y2040_RB.raise_()
        self.select_BTN_2 = QtWidgets.QPushButton(self.widget)
        self.select_BTN_2.setGeometry(QtCore.QRect(250, 340, 81, 23))
        self.select_BTN_2.setStyleSheet("background-color: rgb(85, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.select_BTN_2.setObjectName("select_BTN_2")

        self.retranslateUi(main)
        QtCore.QMetaObject.connectSlotsByName(main)

    def retranslateUi(self, main):
        _translate = QtCore.QCoreApplication.translate
        main.setWindowTitle(_translate("main", "Form"))
```

```
        self.scenariosSelect_LBL.setText(_translate("main", "<html><head/><body><p align=\"center\"><span style=\" font-weight:600;
font-style:italic; color:#c2d6ec;\">Seleziona un\'area e uno scenario</span></p></body></html>"))
        self.select_BTN.setText(_translate("main", "Custom..."))
        self.area_LBL.setText(_translate("main", "Area"))
        self.underground_RB.setText(_translate("main", "Underground"))
        self.roadservices_RB.setText(_translate("main", "Road Services"))
        self.residential_RB.setText(_translate("main", "Residential"))
        self.config_LBL.setText(_translate("main", "Configurazione"))
        self.bc_RB.setText(_translate("main", "Base / Centralized"))
        self.dec_RB.setText(_translate("main", "Decentralized"))
        self.year_LBL.setText(_translate("main", "Anno"))
        self.y2020_RB.setText(_translate("main", "2020"))
        self.y2030_RB.setText(_translate("main", "2030"))
        self.y2040_RB.setText(_translate("main", "2040"))
        self.select_BTN_2.setText(_translate("main", "Select"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    main = QtWidgets.QWidget()
    ui = Ui_main()
    ui.setupUi(main)
    main.show()
    sys.exit(app.exec_())
```

4.5.1.12 specificlogics.py

```
from PyQt5 import QtWidgets, QtGui, QtCore
from .specificlogicsUI import Ui_main


class SpecificLogics(QtWidgets.QMainWindow):
    def __init__(self):
        super(SpecificLogics, self).__init__()
        self.ui = Ui_main()
        self.ui.setupUi(self)
```

4.5.1.13 specificlogicsUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'specificlogicsUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_main(object):
    def setupUi(self, main):
        main.setObjectName("main")
        main.resize(351, 421)
        self.specificlogics_WG = QtWidgets.QWidget(main)
        self.specificlogics_WG.setGeometry(QtCore.QRect(0, 0, 351, 341))
        self.specificlogics_WG.setObjectName("specificlogics_WG")
        self.controlsLogo_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.controlsLogo_LBL.setGeometry(QtCore.QRect(0, 50, 81, 81))
        self.controlsLogo_LBL.setText("")
        self.controlsLogo_LBL.setPixmap(QtGui.QPixmap("../res/Co_80x80.png"))
        self.controlsLogo_LBL.setObjectName("controlsLogo_LBL")
        self.controls_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.controls_LBL.setGeometry(QtCore.QRect(90, 50, 261, 81))
        self.controls_LBL.setStyleSheet("")
        self.controls_LBL.setObjectName("controls_LBL")
        self.ems_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.ems_LBL.setGeometry(QtCore.QRect(90, 150, 261, 81))
        self.ems_LBL.setObjectName("ems_LBL")
        self.emsLogo_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.emsLogo_LBL.setGeometry(QtCore.QRect(0, 150, 81, 81))
        self.emsLogo_LBL.setText("")
        self.emsLogo_LBL.setPixmap(QtGui.QPixmap("../res/EMS_80x80.png"))
        self.emsLogo_LBL.setObjectName("emsLogo_LBL")
        self.energyindexes_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.energyindexes_LBL.setGeometry(QtCore.QRect(90, 260, 261, 81))
        self.energyindexes_LBL.setObjectName("energyindexes_LBL")
        self.energyindexesLogo_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.energyindexesLogo_LBL.setGeometry(QtCore.QRect(0, 260, 81, 81))
        self.energyindexesLogo_LBL.setText("")
        self.energyindexesLogo_LBL.setPixmap(QtGui.QPixmap("../res/Ei_80x80.png"))
        self.energyindexesLogo_LBL.setObjectName("energyindexesLogo_LBL")
        self.logContr_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.logContr_LBL.setGeometry(QtCore.QRect(10, 0, 301, 31))
        self.logContr_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.logContr_LBL.setObjectName("logContr_LBL")
        self.energyindexesShadow_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.energyindexesShadow_LBL.setGeometry(QtCore.QRect(0, 260, 351, 81))
        self.energyindexesShadow_LBL.setStyleSheet("QFrame{\n"
"    background-color: rgba(25, 10, 60, 195);\n"
"}")
        self.energyindexesShadow_LBL.setText("")
        self.energyindexesShadow_LBL.setObjectName("energyindexesShadow_LBL")
        self.emsShadow_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.emsShadow_LBL.setGeometry(QtCore.QRect(0, 150, 351, 81))
        self.emsShadow_LBL.setStyleSheet("QFrame{\n"
"    background-color: rgba(25, 10, 60, 195);\n"
"}")
        self.emsShadow_LBL.setText("")
        self.emsShadow_LBL.setObjectName("emsShadow_LBL")
        self.controlsShadow_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.controlsShadow_LBL.setGeometry(QtCore.QRect(0, 50, 351, 81))
        self.controlsShadow_LBL.setStyleSheet("QFrame{\n"
"    background-color: rgba(25, 10, 60, 195);\n"
"}")
        self.controlsShadow_LBL.setText("")
        self.controlsShadow_LBL.setObjectName("controlsShadow_LBL")

        self.retranslateUi(main)
        QtCore.QMetaObject.connectSlotsByName(main)

    def retranslateUi(self, main):
        _translate = QtCore.QCoreApplication.translate
        main.setWindowTitle(_translate("main", "Form"))
        self.controls_LBL.setText(_translate("main", "<html><head/><body><p><span style=\" font-family:\'Calibri\'; font-size:18pt;
font-weight:600; color:#adb9ca;\">Controlli</span><br/><span style=\" font-family:\'Calibri\'; font-size:12pt;
color:#adb9ca;\">Azioni di controllo su guasti specifici</span></p></body></html>"))
        self.ems_LBL.setText(_translate("main", "<html><head/><body><p><span style=\" font-family:\'Calibri\'; font-size:18pt; font-
weight:600; color:#adb9ca;\">Energy Management</span><span style=\" font-family:\'Calibri\'; font-size:18pt;
color:#adb9ca;\"><br/></span><span style=\" font-family:\'Calibri\'; font-size:12pt; color:#adb9ca;\">Ottimizzazione economica dello
storage</span></p></body></html>"))
        self.energyindexes_LBL.setText(_translate("main", "<html><head/><body><p><span style=\" font-family:\'Calibri\'; font-
size:18pt; font-weight:600; color:#adb9ca;\">Energy Indexes</span><span style=\" font-family:\'Calibri\'; font-size:18pt;
color:#adb9ca;\"><br/></span><span style=\" font-family:\'Calibri\'; font-size:12pt; color:#adb9ca;\">Indici di affidabilità sui
punti di carico</span></p></body></html>"))
        self.logContr_LBL.setText(_translate("main", "<html><head/><body><p><span style=\" font-family:\'Calibri\'; font-size:18pt;
font-weight:600; color:#adb9ca;\">Logiche di Controllo</span></p></body></html>"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    main = QtWidgets.QWidget()
    ui = Ui_main()
    ui.setupUi(main)
    main.show()
    sys.exit(app.exec_())
```

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'specificlogicsUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets
```

```
class Ui_main(object):
    def setupUi(self, main):
        main.setObjectName("main")
        main.resize(351, 421)
        self.specificlogics_WG = QtWidgets.QWidget(main)
        self.specificlogics_WG.setGeometry(QtCore.QRect(0, 0, 351, 341))
        self.specificlogics_WG.setObjectName("specificlogics_WG")
        self.controlsLogo_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.controlsLogo_LBL.setGeometry(QtCore.QRect(0, 50, 81, 81))
        self.controlsLogo_LBL.setText("")
        self.controlsLogo_LBL.setPixmap(QtGui.QPixmap("../res/Co_80x80.png"))
        self.controlsLogo_LBL.setObjectName("controlsLogo_LBL")
        self.controls_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.controls_LBL.setGeometry(QtCore.QRect(90, 50, 261, 81))
        self.controls_LBL.setStyleSheet("")
        self.controls_LBL.setObjectName("controls_LBL")
        self.ems_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.ems_LBL.setGeometry(QtCore.QRect(90, 150, 261, 81))
        self.ems_LBL.setObjectName("ems_LBL")
        self.emsLogo_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.emsLogo_LBL.setGeometry(QtCore.QRect(0, 150, 81, 81))
        self.emsLogo_LBL.setText("")
        self.emsLogo_LBL.setPixmap(QtGui.QPixmap("../res/EMS_80x80.png"))
        self.emsLogo_LBL.setObjectName("emsLogo_LBL")
        self.energyindexes_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.energyindexes_LBL.setGeometry(QtCore.QRect(90, 260, 261, 81))
        self.energyindexes_LBL.setObjectName("energyindexes_LBL")
        self.energyindexesLogo_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.energyindexesLogo_LBL.setGeometry(QtCore.QRect(0, 260, 81, 81))
        self.energyindexesLogo_LBL.setText("")
        self.energyindexesLogo_LBL.setPixmap(QtGui.QPixmap("../res/Ei_80x80.png"))
        self.energyindexesLogo_LBL.setObjectName("energyindexesLogo_LBL")
        self.logContr_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.logContr_LBL.setGeometry(QtCore.QRect(10, 0, 301, 31))
        self.logContr_LBL.setAlignment(QtCore.Qt.AlignCenter)
        self.logContr_LBL.setObjectName("logContr_LBL")
        self.energyindexesShadow_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.energyindexesShadow_LBL.setGeometry(QtCore.QRect(0, 260, 351, 81))
        self.energyindexesShadow_LBL.setStyleSheet("QFrame{\n"
"    background-color: rgba(25, 10, 60, 195);\n"
"}")
        self.energyindexesShadow_LBL.setText("")
        self.energyindexesShadow_LBL.setObjectName("energyindexesShadow_LBL")
        self.emsShadow_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.emsShadow_LBL.setGeometry(QtCore.QRect(0, 150, 351, 81))
        self.emsShadow_LBL.setStyleSheet("QFrame{\n"
"    background-color: rgba(25, 10, 60, 195);\n"
"}")
        self.emsShadow_LBL.setText("")
        self.emsShadow_LBL.setObjectName("emsShadow_LBL")
        self.controlsShadow_LBL = QtWidgets.QLabel(self.specificlogics_WG)
        self.controlsShadow_LBL.setGeometry(QtCore.QRect(0, 50, 351, 81))
        self.controlsShadow_LBL.setStyleSheet("QFrame{\n"
"    background-color: rgba(25, 10, 60, 195);\n"
"}")
        self.controlsShadow_LBL.setText("")
        self.controlsShadow_LBL.setObjectName("controlsShadow_LBL")

        self.retranslateUi(main)
        QtCore.QMetaObject.connectSlotsByName(main)

    def retranslateUi(self, main):
        _translate = QtCore.QCoreApplication.translate
        main.setWindowTitle(_translate("main", "Form"))
        self.controls_LBL.setText(_translate("main", "<html><head/><body><p><span style=\" font-family:\'Calibri\'; font-size:18pt;
font-weight:600; color:#adb9ca;\">Controlli</span><br/><span style=\" font-family:\'Calibri\'; font-size:12pt;
color:#adb9ca;\">Azioni di controllo su guasti specifici</span></p></body></html>"))
        self.ems_LBL.setText(_translate("main", "<html><head/><body><p><span style=\" font-family:\'Calibri\'; font-size:18pt; font-
weight:600; color:#adb9ca;\">Energy Management</span><span style=\" font-family:\'Calibri\'; font-size:18pt;
color:#adb9ca;\"><br/></span><span style=\" font-family:\'Calibri\'; font-size:12pt; color:#adb9ca;\">Ottimizzazione economica dello
storage</span></p></body></html>"))
        self.energyindexes_LBL.setText(_translate("main", "<html><head/><body><p><span style=\" font-family:\'Calibri\'; font-
size:18pt; font-weight:600; color:#adb9ca;\">Energy Indexes</span><span style=\" font-family:\'Calibri\'; font-size:18pt;
color:#adb9ca;\"><br/></span><span style=\" font-family:\'Calibri\'; font-size:12pt; color:#adb9ca;\">Indici di affidabilità sui
punti di carico</span></p></body></html>"))
        self.logContr_LBL.setText(_translate("main", "<html><head/><body><p><span style=\" font-family:\'Calibri\'; font-size:18pt;
font-weight:600; color:#adb9ca;\">Logiche di Controllo</span></p></body></html>"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    main = QtWidgets.QWidget()
    ui = Ui_main()
    ui.setupUi(main)
    main.show()
    sys.exit(app.exec_())
```