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

MINISTERO DELLO SVILUPPO ECONOMICO

Ricerca di Sistema elettrico

Studio degli scenari di flessibilità di un micro distretto orientato al Demand Response e rilevazione automatica di anomalie in un sistema di supervisione energetica in ambito Smart Buildings

S. Panzieri, D. Masucci, F. Lauro

ROMA TRE
UNIVERSITÀ DEGLI STUDI

RdS/PAR2016/088

STUDIO DEGLI SCENARI DI FLESSIBILITÀ DI UN MICRO DISTRETTO ORIENTATO AL DEMAND RESPONSE E RILEVAZIONE AUTOMATICA DI ANOMALIE IN UN SISTEMA DI SUPERVISIONE ENERGETICA IN AMBITO SMART BUILDINGS

S. Panzieri, D. Masucci, F. Lauro (Università degli Studi di Roma TRE)

Settembre 2017

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Annuale di Realizzazione 2016
Area: Efficienza energetica e risparmio di energia negli usi finali elettrici e interazione con altri vettori energetici
Progetto: Tecnologie per costruire gli edifici del futuro
Obiettivo: Gestione di edifici in contesto Smart District e scenari di Demand-Response
Responsabile del Progetto: Giovanni Puglisi, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione "Studio degli scenari di flessibilità di un micro distretto orientato al Demand Response e rilevazione automatica di anomalie in un sistema di supervisione energetica in ambito Smart Buildings"

Responsabile scientifico ENEA: Stefano Pizzuti

Responsabile scientifico Università Roma Tre: prof. Stefano Panzieri

Indice

1 Introduzione

L'attività di ricerca e sviluppo condotta nell'ambito del presente accordo di collaborazione tra ENEA e Università degli Studi Roma Tre ha proseguito alcuni studi iniziati nelle precedenti annualità (RdS/PAR2015/157) inerenti la gestione energetica efficiente di edifici terziari.
In particolare le linee di attività riprese dalle precedenti annualità e ulteriormente sviluppate sono state :

- Definizione e descrizione dei casi d'uso di gestione energetica efficiente degli edifici appartenenti allo "Smart Village" ENEA.
- Strategie di controllo predittivo per la regolazione di temperatura di edifici multi-zona sulla base del livello di occupazione e del prezzo dell'energia

Le nuove linee di attività hanno riguardato invece:

- Valutazione della flessibilità nell'ambito Demand-Response di un micro distretto costituito da edifici appartenenti allo "Smart Village" ENEA.
- Analisi dei principali open standard di comunicazione orientati al Demand-Response.

In questo ambito la rilevazione automatica dei guasti e delle anomalie di funzionamento assume un ruolo particolarmente significativo nei processi di monitoraggio e controllo dei dispositivi elettrici dislocati all'interno di uno smart building.
La realizzazione e l'integrazione di questa funzionalità all'interno di un sistema SCADA permette di migliorare la consistenza e la robustezza della rete di controllo, di riconoscere e segnalare in tempo reale i comportamenti anomali e i malfunzionamenti del sistema stesso.
La ricerca ha quindi riguardato anche la definizione e l'implementazione di una procedura di Fault Detection in grado di verificare l'avvenuta attuazione di comandi sequenziali impartiti da un sistema di controllo e supervisione, tale sistema gestisce l'accensione e lo spegnimento dei dispositivi elettrici dislocati all'interno di un edificio.
Le attività sono state delineate con le seguenti modalità:

- Identificare e definire una procedura di Falut Detection che accerti la corretta esecuzione di ogni comando impartito dal supervisore, che segnali eventuali anomalie di funzionamento;
- Implementare la strategia definita e le sue funzionalità nel sistema di supervisione e controllo;
- Verificare il corretto funzionamento dell'applicazione realizzata tramite prove sperimentali.

Le attività descritte di seguito sono volte alla realizzazione di una strategia per il riconoscimento e la segnalazione di comportamenti anomali nel processo di supervisione e controllo che interessa l'edificio F40, sito all'interno del Centro Ricerche ENEA "La Casaccia", e di valutare la flessibilità nell'ambito Demand-Response di un micro distretto di edifici del terziario.

2 Valutazione della flessibilità nell'ambito Demand-Response di un micro distretto costituito da edifici appartenenti allo "Smart Village" ENEA

2.1.1 Introduzione

L' attività di seguito descritta è il proseguimento di un'attiva di ricerca condotta nella precedente annualità. L'obiettivo è valutare la flessibilità nell'ambito Demand-Response di un micro distretto di edifici del terziario. Il micro distretto considerato consiste nel cluster di edifici ad uso ufficio e relativa Centrale Termica appartenenti allo "Smart Village" del Centro Ricerche ENEA Casaccia. Ai fini dell'analisi di flessibilità si è supposto che il micro distretto includa elementi di generazione di energia da fonti rinnovabili ed elementi di storage. Oggetto di analisi è stata la domanda elettrica del micro distretto in condizioni di esercizio estive, con particolare riferimento al profilo di carico di potenza attiva generale totale di tutti gli edifici, che comprende le utenze di illuminazione, fancoil (gruppo frigo nel caso della centrale termica) e forza elettromotrice.

L'attività è stata condotta presso lo Smart Lab del Centro Ricerche IREC (Catalonia Institute for Energy Research) di Barcellona, attraverso l'impiego di un emulatore di micro-distretto presente nel Centro (Fig. 1). Tale emulatore è in grado di replicare il comportamento energetico (elettrico) del micro-distretto, considerando elementi emulati (cabine elettriche per l'emulazione dei profili di carico complessivo e di energia generata da pannelli fotovoltaici) e reali (batterie fisiche utilizzate come elementi di storage).

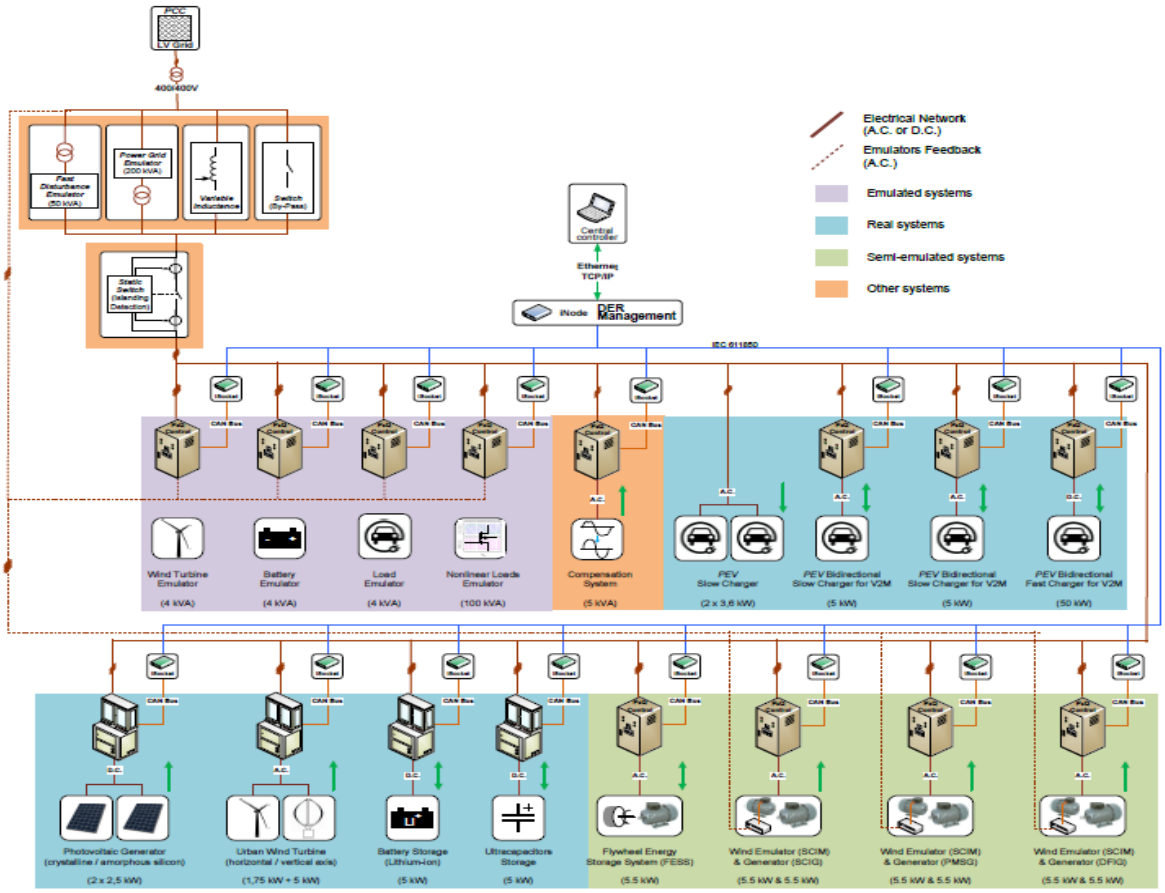

Figura 1 - Emulatore di microgrid IREC

Nella precedente annualità il primo step dell'attività è consistito nella calibrazione e messa a punto dei singoli elementi costituenti l'emulatore di micro-distretto. In particolare, nel caso della calibrazione delle cabine elettriche per l'emulazione dei profili di carico del micro distretto e dei profili di generazione di

energia da pannelli fotovoltaici, sono stati utilizzati i dati reali di consumo e meteo dello Smart Village ENEA. Di seguito sono riportati alcuni dati tecnici relativi alla parte di calibrazione dei singoli elementi dell'emulatore, di cui si è resa necessaria una "scalatura" per via del limite di 4 kW di potenza elettrica massima emulata dalle cabine:

- **Carico**: Potenza massima reale 111,6 kW; Potenza massima emulata 4 kW.
- **Pannelli solari**: Potenza di picco reale 102 kWp; Potenza massima solare reale 111,3 kW; Potenza massima solare emulata 3,93 kW.
- **Batterie**: Potenza massima reale 5 batterie * 10 kW ca per batteria = 50 kW ca; Potenza massima emulata 1,8 kW ca; Capacità reale 5 batterie * 23,33 kWh ca per batteria = 116,65 kWh ca; Capacità emulata = 0,175 kWh ca.

Il secondo step dell'attività è consistito nella sperimentazione di tre scenari: il primo ha previsto l'emulazione del comportamento energetico del micro-distretto in presenza degli elementi di generazione e di storage; il secondo ha previsto i soli elementi di storage; il terzo i soli elementi di generazione. Per ogni scenario l'obiettivo è stato generare per le 24 ore successive un profilo energetico ottimizzato di tutti i componenti del micro distretto che minimizzasse i costi giornalieri associati allo scambio di energia con il mercato dell'energia. L'algoritmo di ottimizzazione alla base di questo processo utilizza le previsioni di consumo, meteo e prezzo dell'energia per le 24 ore successive, cercando di ridurre l'acquisto di energia dal mercato elettrico nelle ore caratterizzate da prezzi più alti. Al fine della valutazione dei risultati relativi ai costi e ai risparmi, sono stati comparati i tre scenari sopra citati con lo scenario di benchmark che esclude la presenza di elementi di generazione e di storage e che corrisponde allo stato attuale del cluster di edifici ENEA. Tutti i risultati degli scenari emulati fanno riferimento alle reali condizioni di consumo e meteo registrate nello Smart Village ENEA in una giornata dell'estate 2014.

Scenario	Daily cost [€]	Saving compared to Scenario 0 [%]	Daily energy request to the grid [kWh]
0 (No Gen., No Stor.)	272,39	0	1626,48
1 (Gen., Stor.)	173,59	36,27	1027,38
2 (No Gen., Stor.)	294,61	-8,16	1763,58
3 (Gen., No Stor.)	155,89	42,77	901,18

Tabella 1 - Confronto dei risultati degli scenari emulati di micro distretto

Lo scenario che ha mostrato i risultati migliori dal punto di vista delle richiesta energetica alla grid e, quindi, dei costi da sostenere è stato quello in cui il micro distretto è dotato dei soli elementi di generazione in quanto l'utilizzo di batterie è risultato sconveniente.
Alla luce dei risultati della precedente annualità in merito all'utilizzo sconveniente delle batterie come elementi di storage, in questa annualità si è cercato di individuare il migliore dimensionamento dello storage per la micro-grid in esame. A tal proposito è stato utilizzato un modello di ottimizzazione GAMS (General Algebraic Modeling System) che ha permesso di individuare capacità e potenza ottime di storage in relazione alle caratteristiche di tutti gli elementi presenti nella micro-grid. Il dimensionamento ottimo della microgrid offre la possibilità di configurare ed emulare nuovi scenari che permettono di apprezzare maggiormente i vantaggi derivanti dagli elementi di generazione e storage.

2.1.2 Dimensionamento dello storage

Batterie presenti nell'emulatore di micro-grid IREC

Nel laboratorio dell'emulatore di microgrid IREC lo storage è costituito da batterie fisiche di tipo lithium – ion. In particolare nel corso degli esperimenti della precedente annualità è stata utilizzata una batteria second-life Renault, caratterizzata da 23,33 kWh di capacità e 40 kW di potenza nominale. Nella presente annualità si è proceduto all'acquisto di una batteria Saft più moderna e performante, caratterizzata da 20 kWh di capacità e 150 kW di potenza nominale, che ha sostituito la precedente batteria. Le caratteristiche tecniche complete di queste batterie sono riportate nelle Figure 2 e 3, rispettivamente.

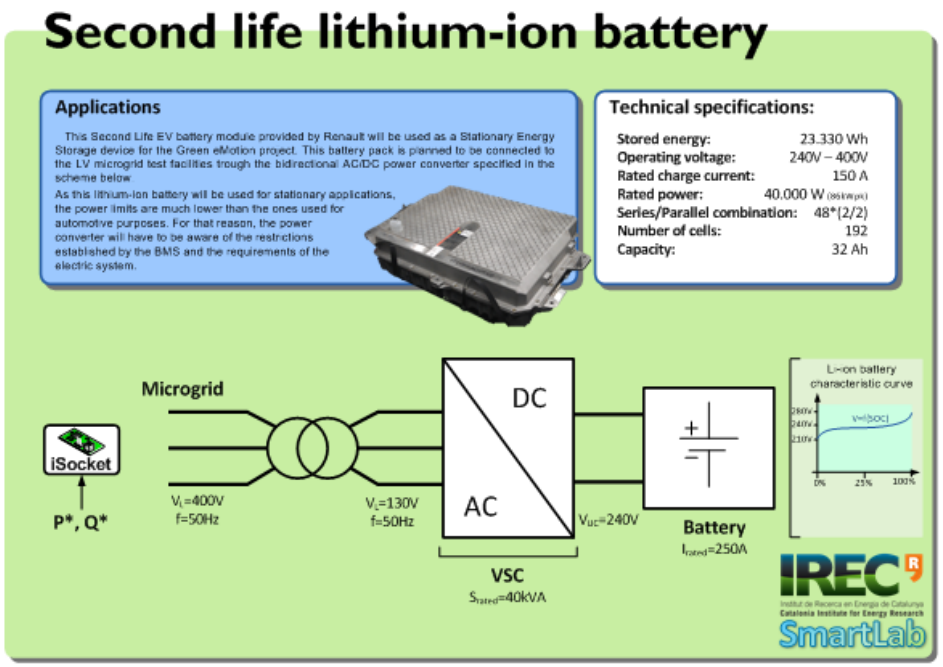

Figura 2 – Renault second life lithium-ion battery

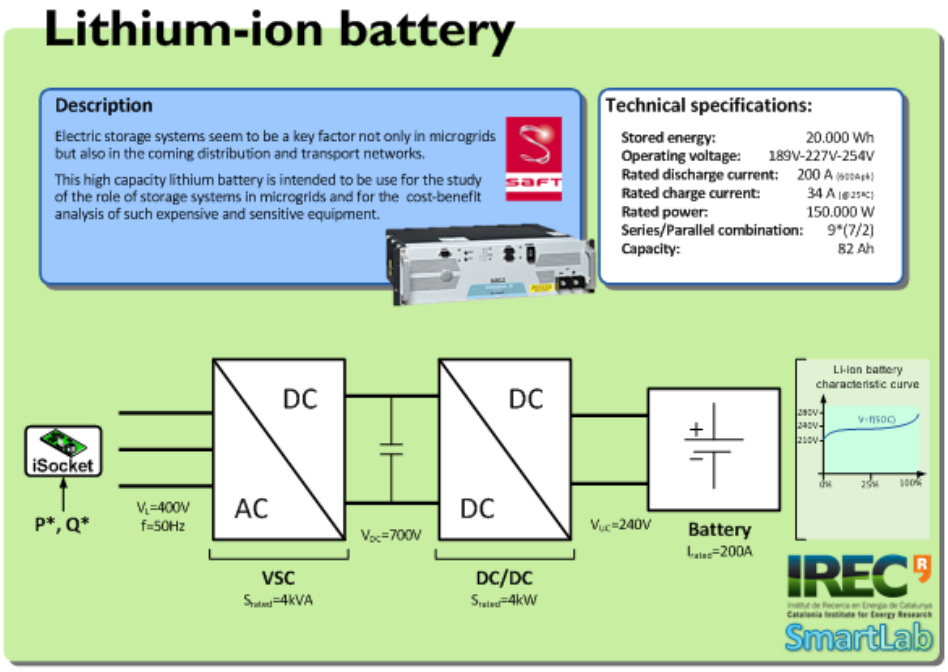

Figura 3 – Saft lithium-ion battery

Processo di ottimizzazione

Considerando i risultati della precedente annualità che mostravano un utilizzo sconveniente delle batterie come elementi di storage, in questa annualità si è cercato di individuare il migliore dimensionamento dello storage nell'ambito della micro-grid in esame. A tal proposito è stato utilizzato un modello di ottimizzazione GAMS che ha permesso di individuare capacità e potenza ottime di storage in relazione alle caratteristiche di tutti gli elementi presenti nella micro-grid. Di seguito tali caratteristiche sono descritte in dettaglio.

Condizioni al contorno del problema di ottimo

- *Profilo di carico totale (kW)*: profilo annuale della domanda elettrica del micro-distretto con timestamp di 15 minuti. Tale profilo è stato ricavato dai consumi registrati nel cluster di edifici ENEA nell'anno 2014.
- *Limite di interconnessione con la Grid (kW)*: profilo annuale del valore massimo di potenza richiedibile alla rete (grid) secondo contratto, con timestamp di 15 minuti. L'interconnessione rappresenta lo scambio di potenza elettrica tra grid e micro-grid. Nel caso di studio in esame si è supposto che l'interconnessione abbia queste caratteristiche: è bidirezionale (la microgrid può acquistare/vendere potenza elettrica dalla/alla grid) e ha un limite massimo costante di 200 kW (considerato il picco massimo di carico di 165 kW).
- *Costo dell'energia (€/kWh)*: profilo annuale di prezzo dell'energia sommato al costo di accesso alla grid, con timestamp di 15 minuti. I prezzi dell'energia utilizzati sono quelli del Centro-Nord Italia nell'anno 2015, consultabili sul sito internet del Gestore Mercati Energetici [1]. Il costo di accesso alla grid è stato supposto costante per tutto l'anno e pari a 0,12 €/kWh.
- *Profilo di generazione di potenza da pannelli fotovoltaici (kW)*: profilo annuale di potenza elettrica generata dai pannelli fotovoltaici presenti nella micro-grid, con timestamp di 15 minuti. Il profilo è stato calcolato a partire dai dati di radiazione solare e temperatura esterna registrate nel Centro Ricerche ENEA nell'anno 2014. Per quanto riguarda la tipologia di impianto fotovoltaico, sono state considerate le caratteristiche della precedente annualità: potenza di picco 102 kWp e potenza massima solare 111,3 kW, corrispondenti a una superficie del tetto impiegata pari al 25% della superficie totale (circa 3200 m^2) del tetto del cluster di edifici.
- *Periodo di ammortamento (anni)*: è il tempo di ritorno in anni dell'investimento dell'intero progetto della micro-grid. In [2] viene effettuata un'analisi sui tempi di ritorno di investimenti per progetti con batterie utilizzate come elementi di storage e sistemi PV impiegati come elementi di generazione. In generale risulta che il tempo di ritorno è più breve (5 – 10 anni circa) se non consideriamo un sistema PV nel progetto, e viceversa diventa più grande (10 – 25 anni circa) se nel progetto è presente un sistema PV. Nel nostro caso di studio si è pertanto ipotizzato un periodo di ammortamento di 20 anni.
- *Costo delle batterie (€/kWh)*: si è supposto un costo di 300 €/kWh. Il costo delle batterie Lithium-ion infatti è sceso da circa 1000 $/kWh nel 2007 ai 300 $/kWh al giorno d'oggi (costi non molto diversi in €).

Formulazione del problema di ottimo

Di seguito viene riportata la formulazione matematica del problema di ottimo appena introdotto. La funzione obiettivo di costo *z* che si intende minimizzare è costituita dai costi sostenuti dalla microgrid, in particolare costi legati all'acquisto della potenza dalla grid nel corso del periodo esaminato e costi della batteria considerando i tempi di ammortamento.

Grandezze scalari

Li	*lower bound of SOC (battery state of charge)*	0.2
Ls	*upper bound of SOC*	0.95
Rs	*Relationship between power and energy for the battery*	2.5
SOCi	*Initial SOC*	0.2
Bcost	*battery cost € per kWh*	300
a	*amortization period in years*	20
PPvn	*PV capacity*	102

Set

T *number of time steps (15 min in one year)* 1*34852

Parametri

d(T) *demand*
limit(T) *interconnection limit*
cost(T) *energy price vector*
Ta(t) *temperature*
Ir(t) *solar irradiation*
Tc(t) *cell temperature (of PV modules)* Tc(t) = 17.23292 + 0.451708*Ta(t) +0.022706*Ir(t);
PPVav(t) *solar generation power (of PV modules)*
PPVav(t) = max(0,((-0.062059*Ir(t)+42.77774)*Tc(t)+9.692792*Ir(t)-1885.868)/1000)*(PPvn/6.6);

Variabili

Cs *storage capacity*
Ps(t) *charging power into the battery*
Psd(t) *discharging power from the battery*
SOC(t) *state of charge*
z *objective function*
Pi_c(t) *power supplied by the interconnection*
Pi_v(t) *power delivered to the grid*

Equazioni

coste *objective function value*
demanda(t) *power balance equation*
Estado_C(t) *energy balance for battery*
Estado_i *initial SOC*
Estado_min(t) *lower bound of SOC*
Estado_max(t) *upper bound of SOC*
Cota_S_max(t) *upper bound for discharging power*
Cota_S_maxC(t) *upper bound for charging power*
UB_inter(t) *interconnection limit*

Funzione obiettivo: min z

coste = z = sum(t, cost(t)*0.25*(pi_c(t) - pi_v(t))) + Bcost*CS/a;

subject to (vincoli):

demanda(t) = - ps(t) + pi_c(t) - pi_v(t) + psd(t) + PPVav(t) = d(t);
Estado_C(t)$(ord(t)>1) = SOC(t) = SOC(t-1) + Ps(t)*0.25 - psd(t)*0.25;
Estado_i = SOC("1") = SOCi*Cs + Ps("1")*0.25 - psd("1")*0.25;
Estado_min(t) = Li*Cs = Soc(t);
Estado_max(t) = Soc(t) = Ls*Cs;
Cota_S_max(t) = psd(t) = Rs*Cs;
Cota_S_maxC(t) = Ps(t) = Rs*Cs;
UB_inter(t) = pi_c(t) + pi_v(t) = limit(t).

Risultati

Il risultato del processo di ottimizzazione illustrato ha prodotto le seguenti caratteristiche di storage per la microgrid in esame:

- Capacità della batteria: 65,6 kWh:
- Potenza della batteria: 164 kW.

Il dimensionamento dello storage della precedente annualità prevedeva invece una potenza molto inferiore (50 kW) e una capacità maggiore (117 kWh).

2.1.3 Conclusioni e sviluppi futuri

A valle del processo di ottimizzazione, gli scenari di test sopra citati verranno nuovamente emulati considerando il dimensionamento ottimo dello storage.
Sempre rispetto alla precedente annualità, saranno prese in esame curve giornaliere di prezzo dell'energia che presentano una maggiore variabilità nelle ore diurne in modo da poter apprezzare maggiormente i vantaggi derivanti dagli elementi di generazione e storage. In Figura 4 e 5 sono illustrate due curve gionaliere di prezzo caratterizzate da una maggiore variabilità rispetto a quelle considerate negli esperimenti precedenti: esse si riferiscono, rispettivamente, ai prezzi dell'energia del Centro-Nord Italia nelle giornate del 22 e 23 Luglio 2015, consultabili sul sito internet del Gestore Mercati Energetici [1].

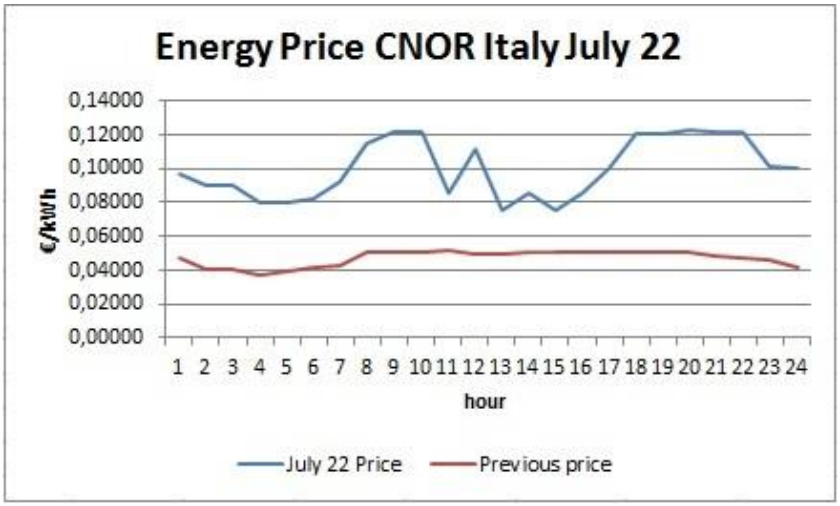

Figura 4 - Confronto tra curve di prezzo giornaliere (Giorno 1)

Scenari interessanti da valutare risultano inoltre quelli in cui il micro distretto disponga di una maggiore produzione di energia da pannelli fotovoltaici (la presente configurazione corrisponde a una superficie del tetto impiegata pari al 25% della superficie totale del tetto del cluster di edifici) da poter sfruttare maggiormente per la carica delle batterie.

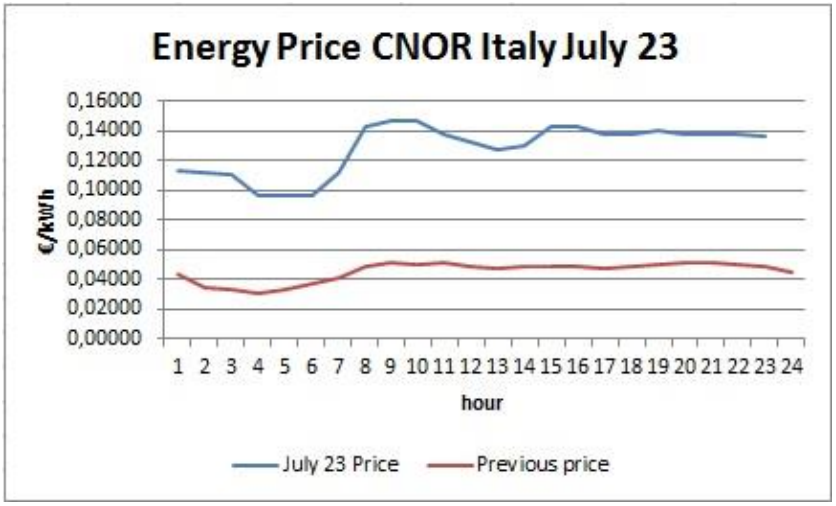

Figura 5 – Confronto tra curve di prezzo giornaliere (Giorno 2)

3 Realizzazione della procedura di Falut Detection.

Si vuole definire un approccio sperimentale che ha lo scopo di migliorare l'efficienza delle procedure di rilevazione e segnalazione automatica delle anomalie relative al funzionamento di un supervisore energetico in ambito Smart Building.

3.1.1 Identificazione e definizione della procedura di Falut Detection.

Durante questa attività è stato effettuato uno studio approfondito sul contesto di applicazione e sugli strumenti a disposizione. Sono stati analizzati i dati forniti dal sistema di supervisione e controllo, e sono state definite le informazioni necessarie alla realizzazione di una strategia di detection efficace ed efficiente.

Nel processo di realizzazione della strategia di controllo automatico si è proceduto considerando le seguenti fasi:

1. Individuazione e studio dei parametri fondamentali;
2. Definizione della regola di controllo;
3. Implementazione della procedura;
4. Sperimentazione sul caso reale.

Risulta quindi fondamentale:
avere a disposizione le seguenti informazioni riguardanti ogni comando:

1. Dispositivo elettrico associato – illuminazione, riscaldamento, raffrescamento;
2. Localizzazione del dispositivo – edificio, piano, stanza;
3. Tipologia di comando – accensione, spegnimento;

poter fare le seguenti assunzioni:

1. Ad ogni dispositivo è associata una grandezza di riferimento di cui è facilmente deducibile lo stato;
2. Allo stato della grandezza di riferimento è associato l'esito del comando.

La procedura risolutiva si basa sulla importante assunzione *-poter associare ad ogni comando impartito l'andamento di una data grandezza di riferimento-*.
Monitorando lo stato della grandezza e confrontandolo con valori soglia predefiniti è possibile stabilire se il comando sia stato eseguito o meno.
In pratica, l'accensione o lo spegnimento di un'apparecchiatura elettrica influisce, in relazione al proprio consumo, sulla richiesta totale di energia che risulta misurabile tramite il quadro elettrico a cui l'apparecchiatura è associata.
Ad esempio: le grandezze associabili ai comandi di accensione e spegnimento dell'illuminazione generale di un piano di un edificio coincidono con le misure registrate dal quadro elettrico generale che riporta il consumo relativo all'illuminazione di ogni piano.
Una volta inviato un comando dal supervisore, si prevede un tempo ragionevole di attesa perché la grandezza associata si stabilizzi mostrando il suo stato, determinando così l'esito del comando.
Si è sfruttata la struttura di archiviazione del sistema di supervisione, organizzata in diverse tabelle che compongono un database relazionale conservato all'interno di un server dedicato, per ottenere i dati necessari al corretto comportamento della procedura.

Sono dati disponibili real time:

- l'elenco dei comandi impartiti dal supervisore;
- l'elenco dei dispositivi installati e i relativi comandi o misure;
- l'elenco dei comandi con le rispettive grandezze associate.

E' possibile formalizzare come segue:

1. Si acquisiscono le informazioni riguardanti il comando impartito – tipologia di comando, grandezza di riferimento;

2. Si acquisiscono le informazioni riguardanti la grandezza di riferimento associata – tempo necessario alla stabilizzazione, valori soglia predefiniti;
3. Si attende il tempo necessario alla stabilizzazione della grandezza e se ne acquisisce lo stato;
4. Si confronta lo stato della grandezza con i valori soglia, e si definisce l'esito del comando;
5. Si segnala l'eventuale presenza di un'anomalia.

Per implementare questo tipo di controllo è stato sfruttato l'utilizzo di un componente Java molto importante, ossia il thread. Il linguaggio java infatti consente di creare applicazioni in grado di utilizzare la corretta logica dei processi, garantendo la condivisione in parallelo dello spazio di memoria tra i vari processi.
Dal punto di vista dell'applicazione i thread rappresentano una serie di processi logici che, da una parte condividono la stessa memoria della procedura che li ha creati, dall'altra concorrono con il processo principale al meccanismo di assegnazione della CPU.
Ad esempio, la gestione degli attuatori connessi alle luci dei corridoi dell'edificio F40 è affidata ad un thread chiamato "BuildingLightActuator, i cui dati vengono memorizzati nel database "smarttowndb", che può essere interrogato e gestito tramite il programma MySQL.
In particolare lo storico dei comandi impartiti è memorizzato nella tabella "historian_control_actions3".
MySQL consente di creare connessioni con il database tramite l'indirizzo IP ed eseguire i comandi per la manipolazione dei dati. Le query devono essere scritte seguendo regole precise di sintassi.
Esistono quindi delle librerie che consentono di far dialogare MySQL con Java in modo tale da poter eseguire le query tramite uno script Java fornendo la possibilità di agire direttamente sul database.
Ciò consente la lettura automatica delle tabelle in ambiente MySQL e l'esecuzione delle operazioni di CRUD senza alcuna interazione da parte dell'utente.
Di seguito si riporta la procedura per l'accesso ai dati utilizzata dall'applicazione.

Tale procedura è riassumibile nei seguenti step:

1. Attesa del tempo di stabilizzazione delle grandezze;
2. Connessione e autenticazione alla base dati;
3. Invio della query;
4. Ricezione del recordset risultato della query;
5. Valutazione dell'avvenuta attuazione del comando;
6. Aggiornamento ed eventuale inserimento dell'allarme;
7. Chiusura della connessione;

In questo modo l'applicazione tiene occupato il canale di connessione solo il tempo strettamente necessario utilizzando la risorsa database in modo efficiente.

Schematizzando si ha:

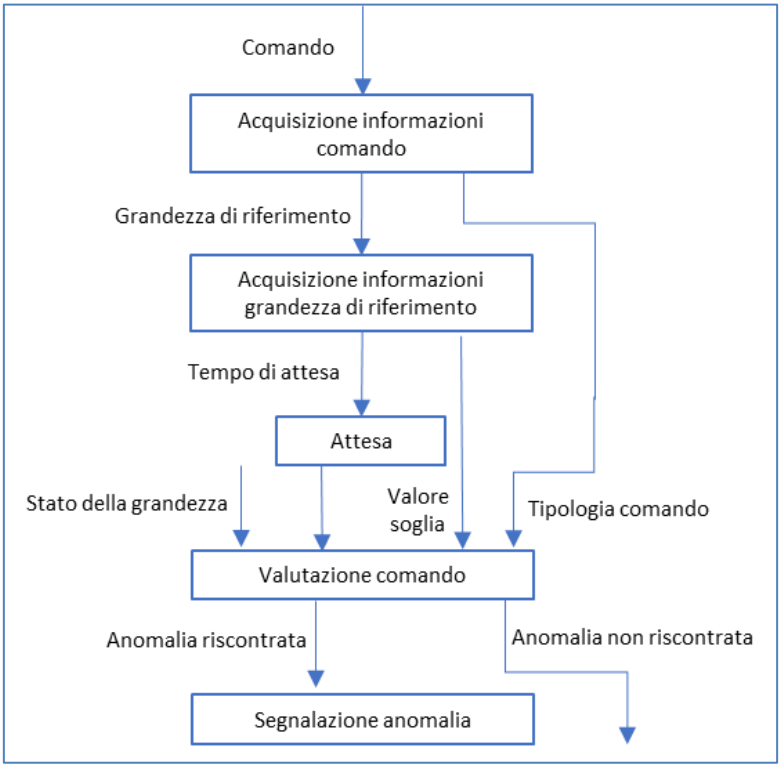

Figura 3 - Schema concettuale procedura di Fault Detection

3.1.2 Implementazione della procedura di Falut Detection

L'obiettivo di questa attività è stato quello di formalizzare, tramite un caso d'uso nell'ambito di uno Smart Building, la strategia di controllo definita nel precedente paragrafo.
Nel caso applicativo considerato si è deciso di prendere in esame i comandi di accensione e spegnimento dell'illuminazione generale di ognuno dei tre piani che compongono l'edificio F40. Le grandezze associabili ai comandi coincidono con le misure registrate dal quadro elettrico generale che riporta il consumo elettrico relativo all'illuminazione di ogni piano.

In particolare si fa riferimento alle seguenti corrispondenze tra comando impartito e misura associata:

ID Comando	Descrizione Comando		Misura Associata	ID Misura Associata
434	Comando_luci_Corr-PT_da_supervisore		Pot_Att_Tot_WM14-Linea_Luce_QPT	156
435	Comando_luci_Corr-1P_da_supervisore		Pot_Att_Tot_WM14-Linea_Luce_Q1P	169
436	Comando_luci_Corr-2P_da_supervisore		Pot_Att_Tot_WM14-Linea_Luce_Q2P	170

I valori identificativi dei comandi e delle misure associate sono assegnati univocamente dal supervisore.
Ogni informazione necessaria per il funzionamento del processo di detection è contenuta nel database, denominato *smarttowndb* in cui il sistema di supervisione memorizza i dati ottenuti dai sensori e attuatori dislocati all'interno dell'edificio.

Il termine database qualifica una struttura organizzata in cui memorizzare dati, in modo persistente, al fine di poterli leggere e analizzare anche successivamente e, eventualmente, per modificarli e cancellarli.
La gestione e l'interazione con la base di dati è affidata ad un particolare software detto DBMS (DataBase Management System) il quale interviene, in qualità di intermediario, in ogni operazione svolta da parte di altri software sul database.

Per realizzare questo compito è stato scelto uno tra i più diffusi software per la gestione di database, ossia MySQL che costituisce una soluzione flessibile e affidabile per la gestione degli accessi e eseguire le interrogazioni sul database SmartTown.
In particolare, MySQL appartiene alla classe di software RDBMS (Relational DataBase Managemet System) che opera in aderenza con la teoria relazionale definita da Codd[3] secondo cui il sistema deve operare su dati strutturati, all'interno del database, in differenti tabelle, interconnesse tra loro e organizzate secondo la relazione chiave-valore.
Nella figura 5 si riporta il diagramma EER in cui sono rappresentate le correlazioni tra le diverse tabelle interessate dall'attività di rilevamento delle anomalie.

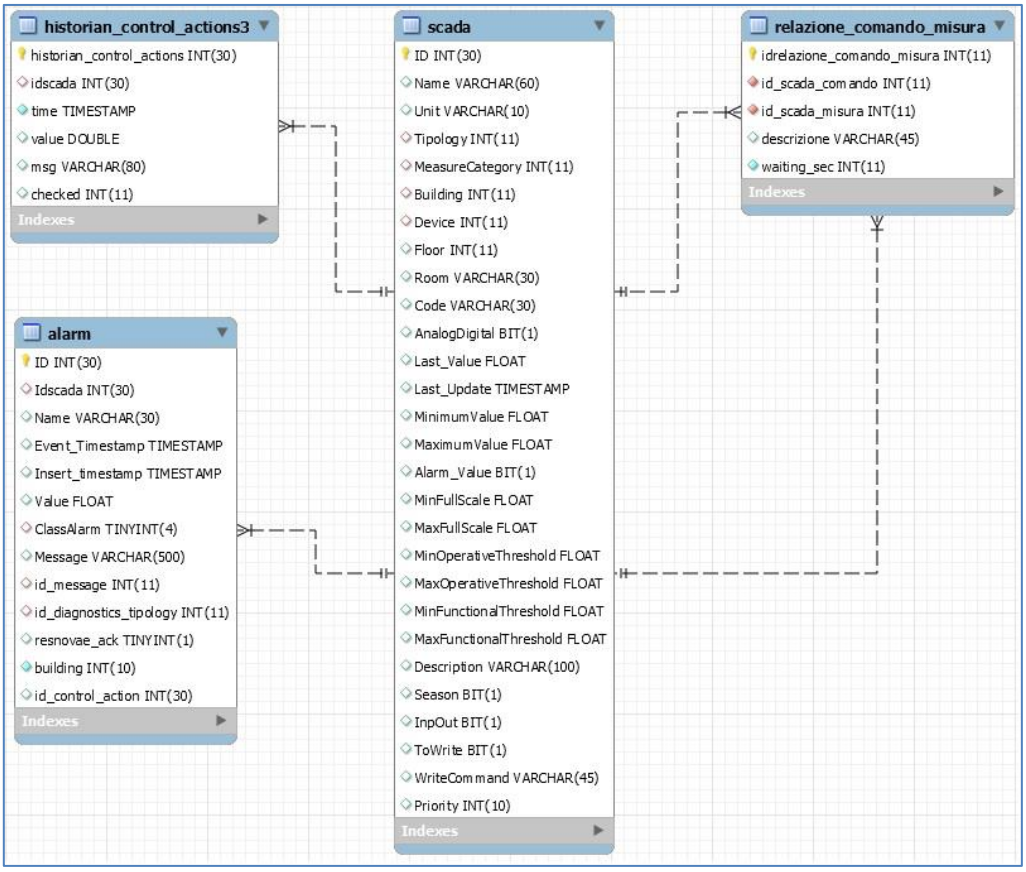

Figura 4 - Diagramma EER delle tabelle di interesse

Di seguito si fornisce una breve descrizione delle tabelle e dei campi utilizzati durante la procedura.

- *historian_control_actions3*: la tabella rappresenta lo storico dei comandi elettrici impartiti dal supervisore, dei quali deve essere verificata l'avvenuta attuazione.
  - *historian_control_actions*, codice univoco e identificativo di ogni comando;
  - *idscada*, codice identificativo del dispositivo scada coinvolto nel comando;
  - time, indica l'istante in cui il comando è stato impartito;
  - *value*, indica la tipologia di comando e può assumere due valori: 0 in caso di spegnimento, 100 in caso di accensione;
  - *checked*, segnala se l'attuazione del comando è stata verificata dalla procedura di rilevazione delle anomalie.
- *scada*: la tabella contiene le informazioni riguardanti i dispositivi di acquisizione e attuazione controllati dal supervisore.
  - *ID*, codice univoco e identificativo di ogni dispositivo scada;
  - *Building*, codice identificativo dell'edificio in cui è collocato il dispositivo scada;

  - *Last_value*, nel caso in cui il dispositivo sia un sensore in questo campo è riportato l'ultimo valore misurato;
  - *Last_update*, istante in cui è avvenuta l'ultima acquisizione o l'ultimo comando sul particolare dispositivo.
- *alarm*: la tabella rappresenta lo storico degli allarmi generati.
  - *ID*, codice univoco e identificativo di ogni allarme;
  - *Idscada*, codice del dispositivo scada su cui si è verificata l'anomalia;
  - *Insert_timestamp*, istante in cui è stato generato l'allarme;
  - *Message*, contiene il messaggio associato all'allarme;
  - *building*, contiene il riferimento all'edificio in cui opera il dispositivo scada che ha generato l'allarme;
  - *id_control_action*, codice identificativo del comando che ha generato l'allarme.
- *relazione_comando_misura*: contiene le relazioni tra comandi e misure, ossia quale grandezza da controllare è associata al particolare dispositivo interessato dal comando.
  - *idrelazione_comando_misura*, è il codice univoco e identificativo di ogni relazione;
  - *id_scada_comando*, codice identificativo del dispositivo scada su cui è stato impartito il comando;
  - *id_scada_misura*, codice identificativo del dispositivo scada da interrogare per ottenere la misura della grandezza associata.

Tramite le interrogazioni, o query, e i comandi, implementati utilizzando il linguaggio SQL[4] (Sctructure Query Language), avviene l'interazione con il database e la possibilità di compiere tutte le operazioni di lettura, scrittura e aggiornamento sulle tabelle.

In relazione agli obiettivi di questa ricerca si è implementato l'algoritmo tramite il linguaggio di programmazione orientato agli oggetti *Java*.
Ciò ha permesso, non solo una più agevole integrazione con il sistema di supervisione preesistente, ma anche una gestione immediata del database relazionale con cui il processo di controllo deve interagire.
Per rendere il codice scalabile e riutilizzabile è stato previsto l'uso di una comune libreria Java, *mySqlConnector*, messa a disposizione da *Oracle* che fornisce supporto alla procedura durante le operazioni di connessione e disconnessione con il database.

3.1.3 La Procedura di controllo

Ricordando che le informazioni necessarie per la corretta configurazione della procedura sono:

- Connessione al database:
  - *serverDBname*, nome del database in cui sono contenuti i dati di interesse;
  - *serverPath*, indirizzo del server su cui è implementato il database;
  - *serverUser*, username per l'accesso al database;
  - *serverPassword*, password per l'accesso al database.

- Parametrizzazione tabelle:
  - *commandTab*, tabella che contiene l'elenco dei comandi impartiti dal supervisore;
  - *scadaTab*, tabella contenente l'elenco dei dispositivi presenti nell'edificio F40 e i relativi comandi o misure associate;
  - *alarmTab*, tabella da aggiornare nel caso in cui si verifichi una mancata attuazione;
  - *relationTab*, tabella che contiene l'elenco dei comandi con le rispettive grandezze associate.

- Costanti del processo:
    - *ref_value*, valore di riferimento con cui confrontare le grandezze associate ad ogni comando per verificare l'attuazione;
    - *delay*, tempo di attesa necessario per permettere la stabilizzazione delle grandezze da misurare;
    - *tempo_ciclo*, tempo di attesa tra due attivazioni consecutive del processo di controllo.
- Creazione del file di log:
    - *logFilePath*, folder in cui salvare il file di log.

Si può riassumere la procedura come segue.

Input:

- database "*smarttowndb*";
- valore di riferimento per le grandezze *ref_value = 1W*;
- tempo di attesa *delay = 10sec*;
- tempo riattivazione del processo *tempoCiclo = 60sec*.

Procedura:

1. A seguito della creazione di un oggetto *CommandCheck,* si attende un tempo di default prestabilito (*delay*) che permetta la stabilizzazione delle grandezze elettriche di riferimento che si intende monitorare.
2. Si sfrutta la potenza del linguaggio SQL per definire un'interrogazione, *query*, in grado di raccogliere e rendere disponibili le seguenti informazioni:
    - Dalla tabella *historian_control_actions3* si ricavano gli ultimi *n* comandi che non sono stati ancora controllati, e si mantiene in memoria il loro *idscada*. Si memorizza anche il campo *value* che permette di distinguere un comando di accensione (*value = 100*) da un comando di spegnimento (*value = 0*).
    - Dalla tabella *relazione_comando_misura* si ricava la grandezza di riferimento corrispondente ad ogni comando identificato al passo precedente, tramite l'*idscada*, e si mantiene in memoria l'indice *id_scada_misura*.
    - Dalla tabella *scada* si ricava il valore assunto dalla grandezza di riferimento, per ogni comando, tramite l'indice *id_scada_misura*.
3. Tenendo in considerazione la natura del comando (accensione/spegnimento) osservata tramite il campo *value*, si confronta il valore assunto dalle grandezze di riferimento con il valore soglia prestabilito *ref_value*, per verificare l'avvenuta attuazione del comando stesso.

Output:

1. Si tiene traccia dell'avvenuto controllo sul comando, aggiornando il campo *checked* della tabella *historian_control_actions3* che corrisponde al record del comando.
2. Si tiene traccia di un eventuale anomalia riscontrata (attuazione non avvenuta) aggiungendo un record alla tabella *alarm* con le informazioni del comando esaminato e non avvenuto.

Di seguito si riporta una rappresentazione schematica delle operazioni logiche effettuate per svolgere l'operazione di *fault detection* descritta.

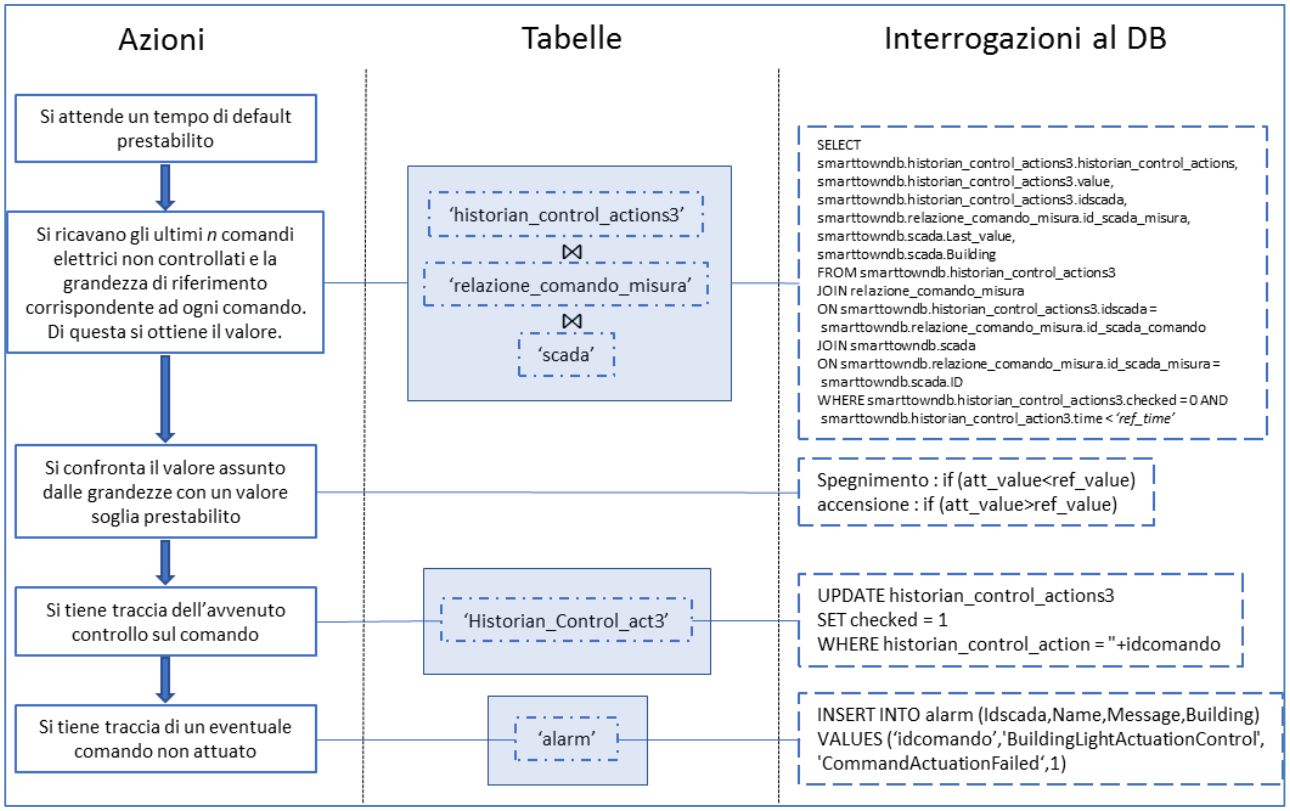

Figura 5 - Schema logico della procedura di Fault Detection

3.1.4 Dettagli di implementazione – La classe *CommandCheck*

Il corretto comportamento del processo di controllo è direttamente connesso con un'implementazione curata nel dettaglio e di cui si riportano gli aspetti salienti.

Con l'obiettivo di rendere la procedura di fault detection portabile e riusabile è stato sfruttato un importante strumento messo a disposizione dal linguaggio java.
E'stato, infatti, creato un file config.properties, riportato nella figura 7, contenente i parametri di connessione al database, e altre costanti utilizzate nel processo, per evitare la loro configurazione all'interno del codice principale. Per realizzare ciò si è utilizzata la classe java.Properties grazie alla quale è possibile leggere dal file di config una serie di righe strutturate in formato chiave:valore da utilizzare nella classe principale.

Le informazioni ricavate dal file.properties sono:

- Connessione al database
  - Nome e path del database;
  - Credenziali di accesso al database, ossia username e password.
- Parametrizzazione delle tabelle del database
  - commandTab, tabella contenete lo storico dei comandi;
  - scadaTab, tabella dei dispositivi scada;
  - relationTab, tabella in cui sono riportate le relazioni tra comando e misura associata;
  - alarmTab, tabella contenente lo storico degli allarmi.
- Costanti di riferimento del processo
  - ref_value, valore soglia di riferimento per le grandezze elettriche;
  - delay, tempo di attesa per la stabilizzazione delle grandezze elettriche da misurare;

  - tempoCiclo, tempo che intercorre tra due attivazioni consecutive della procedura.
- Informazioni per la creazione del file di log.

```
# File properties associated with the CommandCheck project

# References to connect database
serverDBname = smarttowndb
serverPath = jdbc:mysql://192.107.82.244:3306/
serverUser = smartvillage
serverPassword = fox24

# References for parameterizing the db's tables
commandTab = historian_control_actions3
scadaTab = scada
alarmTab = alarm
relationTab = relazione_comando_misura

# Reference for creating the log file
fileLogPath = ./CommandCheckLog.txt

# Process constants references
ref_value = 1
delay = 10000
tempoCiclo = 5000
```

Figura 6 - Snapshot file.properties

Parallelamente allo sviluppo dell'applicazione, è stato creato un processo di logging in grado di tenere traccia di tutti gli eventi di interesse che si riscontrano durante l'esecuzione della procedura di fault detection, portando enormi benefici per l'analisi dell'intero processo.
In particolare, le informazioni che si è ritenuto opportuno mantenere nel file.log, riportato nella figura 8, sono:

- Ciclo di attivazione, numero progressivo di attivazione della procedura;
- Comandi controllati, numero di comandi controllati durante l'attivazione corrente;
- Allarmi generati, numero di allarmi generati dalla procedura durante l'attivazione corrente;
- Tempo computazionale, tempo in millisecondi impiegato per eseguire le valutazioni;
- Timestamp, istante in cui si è avviata l'attivazione corrente della procedura.

```
32, 12, 12, 89, 2017-07-20 19:18:53
33, 12, 12, 77, 2017-07-20 19:20:03
34, 12, 12, 107, 2017-07-20 19:21:13
35, 12, 12, 146, 2017-07-20 19:22:23
36, 12, 12, 148, 2017-07-20 19:23:33
37, 24, 24, 204, 2017-07-20 19:24:43
38, 24, 24, 123, 2017-07-20 19:25:54
39, 24, 24, 158, 2017-07-20 19:27:04
40, 24, 24, 168, 2017-07-20 19:28:14
41, 24, 24, 195, 2017-07-20 19:29:24
42, 24, 24, 165, 2017-07-20 19:30:34
43, 3, 2, 50, 2017-07-20 19:31:45
44, 3, 2, 56, 2017-07-20 19:32:55
45, 3, 2, 67, 2017-07-20 19:34:05
46, 3, 2, 22, 2017-07-20 19:35:15
47, 3, 2, 84, 2017-07-20 19:36:25
48, 6, 4, 27, 2017-07-20 19:37:35
49, 0, 0, 15, 2017-07-20 19:38:45
50, 6, 4, 96, 2017-07-20 19:39:55
51, 6, 4, 120, 2017-07-20 19:41:05
```

Figura 7 - Snapshot file di log

Sono quindi previsti due costruttori per sostituire il metodo *init()* e svolgere le operazioni di inizializzazione del processo:

- `CommandCheck() {…}`
  Le informazioni necessarie per stabilire la connessione al DB e per inizializzare il processo sono inserite di default, già presenti all'interno del costruttore.
- `CommandCheck(String pPath) {…}`
  Le informazioni necessarie per stabilire la connessione al DB e per inizializzare il processo sono contenute in un *file_config* il cui nome è fornito in input al costruttore.

Nel metodo *main* potrà essere richiamato una procedura *CommandCheck* tramite la riga di codice

```
CommandCheck cmdchk = new CommandCheck();
```

oppure

```
CommandCheck cmdchk = new CommandCheck('nomeFileConfig');
```

con le quali verrà richiamato il costruttore desiderato. Si richiederà la sua esecuzione tramite la riga di codice `cmdchk.start();` che richiama il metodo *run()* dell'oggetto.

I metodi implementati all'interno della classe definiscono il comportamento dell'oggetto CommandCheck e sono riassunti di seguito.

- `void connessioneDB()`
  Il metodo permette di stabilire una connessione JDBC con il database.
- `void disconnessioneDB()`
  Il metodo permette di terminare la connessione JDBC con il database.
- `int doCommandCheck(att_value, type)`
  Il metodo permette di verificare l'attuazione del commando confrontando il valore assunto dalla grandezza (*att_value*) con un valore di riferimento prestabilito, distinguendo la tipologia del comando (*type*) di accensione o spegnimento. Il valore di ritorno segnala se l'attuazione è avvenuta o meno.
- `void updateCheck(id_com)`
  Il metodo permette di aggiornare il campo '*checked*', nella tabella *historian_control_actions3*, corrispondente al comando analizzato (*id_com*) per tenere traccia dell'avvenuto controllo.
- `void setAlarm(id_scada, building, id_com)`
  Il metodo permette di aggiungere un nuovo record nella tabella *Alarm*. Ciò consente di segnalare un eventuale comando la cui attuazione non risulta essere avvenuta. Vengono fornite le principali informazioni riguardanti il comando fallito (*id_com* e *building*).
- `String getCurrentTime()`
  Il metodo restituisce il timestamp corrente, elemento necessario per la corretta esecuzione del controllo.
- `void finalize()`
  Il metodo permette di deallocare, in maniera dinamica, lo spazio di memoria occupato per l'esecuzione dell'algoritmo al termine della procedura.
- `void run()`
  Il metodo permette di gestire ed eseguire la procedura di controllo.

E' stata prevista la possibilità di avviare la procedura Command Check tramite *command line* in due diverse modalità:

- `java CommandCheck pathFileConfig`

in cui la procedura, richiamando il costruttore con argomento `CommandCheck(String pPath)` , accede al file config.properties all'interno del quale sono specificate le variabili per la configurazione iniziale dell'oggetto.

- `java CommandCheck`
  in cui la procedura, richiamando il costruttore senza argomento `CommandCheck()` , inizializza i valori delle variabili ai loro valori di default.

Ad ogni attivazione della procedura viene aggiornato il file di log per mantenere traccia di informazioni di interesse. Le informazioni memorizzate sono:

- numero progressivo del ciclo di attivazione;
- numero di comandi controllati;
- numero di allarmi generati;
- tempo necessario per l'esecuzione del processo;
- timestamp attuale.

Per completezza di informazione, di seguito si riporta la descrizione dei comandi e delle interrogazioni in linguaggio SQL definite all'interno dell'attività di rilevamento delle anomalie. In particolare si sono implementate tre interrogazioni:

- Query di aggiornamento

```
UPDATE serverDBname.commandTab
SET checked = 1
WHERE serverDBname.commandTab.historian_control_actions = id_com
```

Tramite l'UPDATE si effettua un'operazione di modifica sui dati. In particolare viene aggiornato il valore del campo "checked" della tabella "historian_control_actions3" per segnalare che il comando è stato verificato dalla procedura di rilevamento delle anomalie.

- Query di inserimento

```
INSERT INTO serverDBname.alarmTab (Idscada,Name,Message,building,id_control_action)
VALUES (id_scada,'BuildingLightActuationControl','CommandActuationFailed',building,id_com)
```

Attraverso questa query si andrà ad inserire un nuovo record, all'interno della tabella "alarm", che rappresenta un nuovo allarme generato e le informazioni ad esso correlate, ossia:

  - Codice identificativo del dispositivo scada che ha generato l'allarme;
  - Codice identificativo del comando che ha generato l'allarme;
  - Nome del processo di controllo interessato;
  - Breve descrizione testuale dell'allarme;
  - Codice identificativo dell'edificio in cui opera il dispositivo scada associato all'allarme.

- Query di valutazione
  Attraverso questa SELECT vengono selezionati i dati da più tabelle sfruttando un fondamentale costrutto del linguaggio SQL, ossia l'operazione di JOIN. Attraverso essa si possono mettere in relazione diverse tabelle e ottenere un risultato combinato sulla base di uno o più campi che trovano corrispondenza nelle tabelle coinvolte.

```sql
SELECT serverDBname.commandTab.historian_control_actions,
       serverDBname.commandTab.value,
       serverDBname.commandTab.idscada,
       serverDBname.relationTab.id_scada_misura,
       serverDBname.scadaTab.Last_Value,
       serverDBname.scadaTab.Building
FROM serverDBname.commandTab
     JOIN serverDBname.relationTab
          ON serverDBname.commandTab.idscada = serverDBname.relationTab.id_scada_comando
     JOIN serverDBname.scadaTab
          ON serverDBname.relationTab.id_scada_misura = serverDBname.scadaTab.ID
WHERE serverDBname.commandTab.checked = 0
     AND serverDBname.commandTab.time < 'currentTime'
     AND serverDBname.commandTab.time > 'oldTime'
```

In particolare la query avrà come risultato un'altra tabella composta dalle colonne definite nella SELECT.

Per ognuno degli ultimi n comandi che non sono stati ancora controllati, che presentano il campo "checked" uguale a zero, si hanno:

- Codice identificativo del dispositivo scada coinvolto;
- Tipologia di comando, accensione o spegnimento;
- Codice identificativo del dispositivo scada da cui prelevare la misura della grandezza associata al comando;
- Valore misurato della grandezza associata;
- Codice identificativo dell'edificio in cui opera il dispositivo scada.

historian_control_actions	value	idscada	id_scada_misura	Last_value	Building
291229	0	435	169	1.21	1
291230	0	436	270	0.33	1
291231	0	434	156	0	1
291232	0	435	169	1.21	1
291233	0	436	270	0.33	1
291234	0	434	156	0	1
291235	0	435	169	1.21	1
291236	0	436	270	0.33	1

Figura 8 - Esempio risultato query di valutazione

3.1.5 Esempio applicativo

Per rendere chiaro il procedimento eseguito dall'algoritmo si rappresenta di seguito la sua applicazione, passo per passo, in un caso esemplificativo, ma plausibilmente molto aderente alla realtà.
Si prenderanno in considerazione, contemporaneamente tre comandi inviati dal sistema di supervisione energetica:

- Comando di accensione delle luci nel corridoio a piano terra, al quale corrisponde un conseguente aumento della potenza attiva totale riguardante la linea luce del piano interessato.
- Comando di spegnimento delle luci nel corridoio al primo piano, al quale non corrisponde una conseguente diminuzione della potenza attiva totale riguardante la linea luce del primo piano.
- Comando di accensione delle luci nel corridoio al secondo piano, che avviene durante la fase di attesa della procedura che permette la stabilizzazione della potenza attiva totale riguardante la linea luce al piano interessato.

Con l'aiuto dei messaggi stampati su terminale e evidenziando le modifiche sul database si può verificare il comportamento dell'algoritmo.

1. Inizialmente si hanno le seguenti informazioni riportate nelle tabelle di interesse:

historian_control_actions	idscada	value	checked	time
854	434	100	0	2017-09-15 16:14:00
855	435	0	0	2017-09-15 16:14:00

Figura 9 - Snapshot tabella historian_control_actions3

ID	Idscada	Name	Message	building	insert_timestamp	id_control_action
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 10 - Snapshot tabella alarm

La tabella historian_control_actions3 presenta due nuovi record corrispondenti ai primi due comandi impartiti, in cui il campo "checked" è settato a zero per segnalare che non sono stati controllati.
Nella tabella alarm non vi è ovviamente traccia di allarmi riconducibili ai comandi.

2. La procedura inizia il suo ciclo di attivazione e rimane in attesa che sia terminato il tempo di stabilizzazione delle grandezze (Figura 12)

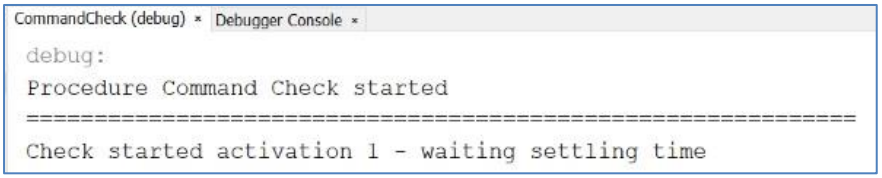

Figura 11 - Snapshot Terminal

Durante questa attesa viene impartito il terzo comando.

historian_control_actions	idscada	value	checked	time
854	434	100	0	2017-09-15 16:14:00
855	435	0	0	2017-09-15 16:14:00
856	436	100	0	2017-09-15 16:14:05

Figura 12 - Snapshot tabella historian_control_actions3

3. Terminata l'attesa, viene interrogato il database effettuando la query di valutazione che restituisce una nuova tabella con le informazioni di interesse per ogni comando

historian_control_actions	value	idscada	id_scada_misura	Last_value	Building
854	100	434	156	10.5	1
855	0	435	169	10.5	1

Figura 13 - Snapshot tabella risultato della query

Come ci si aspettava il terzo comando non viene considerato in quanto il tempo che intercorre dal suo inserimento non è sufficiente per permettere alla grandezza associata di stabilizzarsi. Verrà quindi processato nel ciclo di attivazione successivo.

4. Avviene la valutazione dei comandi impartiti dal supervisore:
   - Primo comando
     - Dal valore 100 osservabile nel campo “value” si nota che il comando è di accensione;
     - Si controlla quindi la misura della grandezza associata al comando, e si nota che questa è superiore al valore soglia preimpostato. Ciò sta a significare che l’accensione delle luci del piano terra è avvenuta correttamente.
   - Secondo comando
     - Dal valore 0 osservabile nel campo “value” si nota che il comando è di spegnimento;
     - Si controlla quindi la misura della grandezza associata al comando, e si nota che questa è superiore al valore soglia preimpostato. Ciò sta a significare che lo spegnimento delle luci al primo piano non è avvenuto correttamente.

   Dal terminale si può verificare l’avanzamento della procedura.

```
Procedure Command Check started
================================================
Check started activation 1 - waiting settling time
- Command checked...Check Update
- Command checked...Check Update...Alarm added
Commands checked : 2 - Alarms generated : 1
Execution time: (10000 + 680) millisec
Check finish - waiting next activation
```

Figura 14 - Snapshot del terminale

5. Si aggiorna ad 1 il campo “checked” della tabella historian_control_actions3 per segnalare l’avvenuto controllo dell’attuazione da parte della procedura.

historian_control_actions	idscada	value	checked	time
854	434	100	1	2017-09-15 16:14:00
855	435	0	1	2017-09-15 16:14:00
856	436	100	0	2017-09-15 16:14:05

Figura 15 - Snapshot tabella historian_control_actions3

6. Si inserisce nella tabella “alarm” l’anomalia riscontrata nell’attuazione del secondo comando.

ID	Idscada	Name	Message	building	insert_timestamp	id_control_action
478	435	BuildingLightActuationControl	CommandActuationFailed	1	2017-09-15 16:14:10	855

Figura 16 - Snapshot tabella alarm

Terminato il primo ciclo di attivazione, la procedura si pone in standby per alcuni secondi prima di iniziare un nuovo ciclo di attivazione.

1. La tabella historian_control_actions3 presenta un record corrispondente al terzo comando impartito, in cui il campo “checked” è settato a zero per segnalare che non è stato controllato.

Nella tabella alarm vi è traccia dell'allarme prodotto dall'anomalia riscontrata sul secondo comando.

2. La procedura inizia il secondo ciclo di attivazione e rimane in attesa che sia terminato il tempo di stabilizzazione delle grandezze.
3. Terminata l'attesa, viene interrogato il database effettuando la query di valutazione che restituisce la tabella con le informazioni di interesse

historian_control_actions	value	idscada	id_scada_misura	Last_value	Building
856	100	436	270	0.5	1

Figura 17 - Snapshot tabella query

Come ci si aspettava, in questo caso il terzo comando viene considerato.

4. Avviene la valutazione del comando:
   - Terzo comando
     - Dal valore 100 osservabile nel campo "value" si nota che il comando è di accensione;
     - Si controlla quindi la misura della grandezza associata al comando, e si nota che questa è inferiore al valore soglia preimpostato. Ciò sta a significare che l'accensione delle luci del piano terra non è avvenuta correttamente.

5. Si aggiorna ad 1 il campo "checked" della tabella historian_control_actions3 per segnalare l'avvenuto controllo dell'attuazione da parte della procedura.

historian_control_actions	idscada	value	checked	time
854	434	100	1	2017-09-15 16:14:00
855	435	0	1	2017-09-15 16:14:00
856	436	100	1	2017-09-15 16:14:05

Figura 18 - Snapshot tabella historian_control_act3

6. Si inserisce nella tabella "alarm" l'anomalia riscontrata nell'attuazione del terzo comando.

ID	Idscada	Name	Message	building	insert_timestamp	id_control_action
478	435	BuildingLightActuationControl	CommandActuationFailed	1	2017-09-15 16:14:10	855
479	436	BuildingLightActuationControl	CommandActuationFailed	1	2017-09-15 16:14:45	856

Figura 19 - Snapshot tabella alarm

Terminato il secondo ciclo di attivazione, la procedura si pone in standby per alcuni secondi prima di iniziare un nuovo ciclo di attivazione.

```
=====================================================
Check started activation 2 - waiting settling time
- Command checked...Check Update...Alarm added
Commands checked : 1 - Alarms generated : 1
Execution time: (10000 + 612) millisec
Check finish - waiting next activation
```

Figura 20 - Snapshot Terminale

Il file di log generato al termine dei due cicli di attivazione, riportato di seguito, fornisce le seguenti informazioni:

- AC (Activation Cycle) – numero progressivo ciclo di attivazione;
- CC (number of Commands Checked) – numero di comandi controllati;
- GA (number of Generated Alarm) – quantità di allarmi generati;
- ET (Execution Time) – tempo computazionale impiegato per la valutazione dei comandi;
- T (Timestamp) – istante di inizio del ciclo di attivazione.

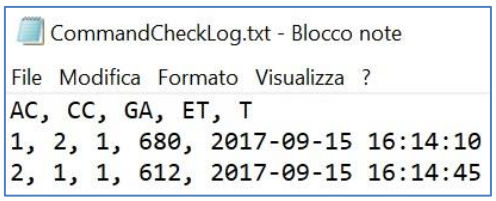

Figura 21 - File di log dopo i due cicli di attivazione

3.1.6 Prove sperimentali

La fase di test ha avuto l'obiettivo primario di verificare il corretto comportamento dell'applicazione realizzata in opposizione alle eterogenee istanze del problema affrontato.
Sono stati quindi preparati dei test di verifica per dimostrare l'efficienza dell'algoritmo e per evidenziare le proprietà che lo rendono uno strumento efficace in relazione agli obiettivi definiti.

In particolare, si è tenuto traccia delle grandezze di seguito elencate, e si sono valutate le misure che queste hanno assunto in fase di test:

- Numero di comandi da controllare;
- Numero di errori di controllo e aggiornamento
- Tempo computazionale impiegato per l'esecuzione del programma e per l'aggiornamento del file di log.

Per chiarezza di esposizione, si è deciso di suddividere i test implementati in tre classi:

- Classe I – si è controllata l'efficacia dello strumento valutando il numero di errori di controllo e aggiornamento commessi;
- Classe II – si è controllato il tempo computazionale, in millisecondi, impiegato dall'algoritmo;
- Classe III – si è controllata l'efficacia dello strumento nel caso particolare in cui avviene un comando durante il tempo di stabilizzazione delle grandezze.

Sono stati individuati e utilizzati, per le classi di test, i seguenti casi di studio perché ritenuti più rispondenti alla realtà, in riferimento ai dati storici:

- Caso di Studio 1 – Valutazione di comandi che non generano allarmi;
- Caso di studio 2 – Valutazione di comandi misti (60% dei comandi genera allarme);
- Caso di Studio 3 – Valutazione di comandi che generano solo allarmi.
- Caso di Studio 4 – Valutazione di comandi impartiti durante il *settling time* (tempo di attesa per la stabilizzazione delle grandezze associate ai comandi)

Per garantire una fase di test il più possibile esaustiva, sono stati effettuati dieci *run* per ogni classe di test sopra menzionata aumentando la dimensione del problema, ossia il numero di comandi da controllare.
I risultati ottenuti sono riportati di seguito.

Caso di studio 1

La procedura di fault detection viene applicata ad un sottoinsieme di comandi, impartiti dal supervisore, tali che la loro attuazione risulti sempre verificata e che, di conseguenza, non sia previsto il riconoscimento di anomalie e quindi la generazione di alcun allarme.
Per verificare il comportamento dell'algoritmo si è deciso inoltre di aumentare in modo progressivo la dimensione dell'istanza di input facendo si che la procedura debba valutare di volta in volta un insieme di comandi sempre più numeroso.
La tabella che segue contiene i risultati ottenuti, riportati anche nella figura 22.

n°comandi	Comandi controllati	Allarmi generati	Tempo comp.
3	3	0	66
6	6	0	80
12	12	0	86
24	24	0	150

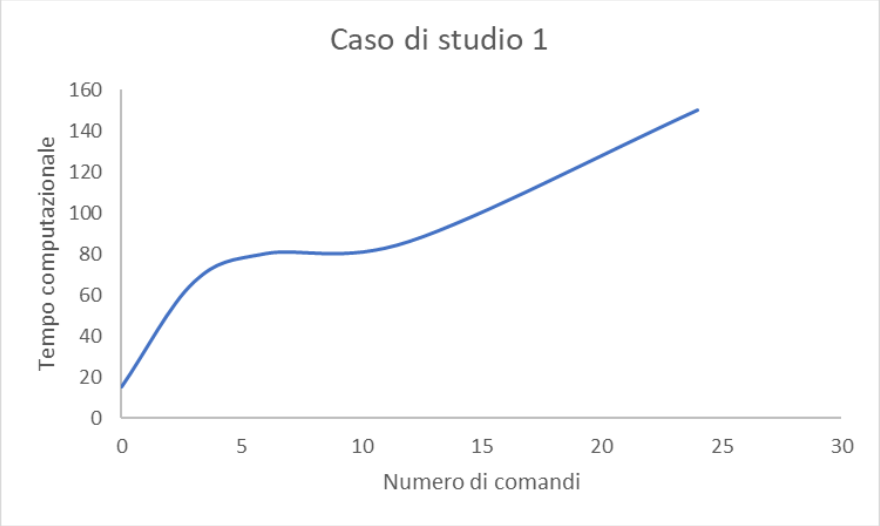

Figura 22 - Grafico dei risultati (Caso di Studio 1)

Caso di studio 2

In questo caso di studio la procedura è stata applicata ad un sottoinsieme di comandi impartiti dal supervisore tali che a circa i due terzi di essi non corrisponde un'attuazione ì. Si prevede quindi che l'algoritmo riconosca le anomalie e generi gli allarmi associati.
Anche in questo caso si è deciso di aumentare progressivamente il numero di comandi impartiti, mantenendo invariata la percentuale dei malfunzionamenti simulati, circa il 60%.
Come previsto le anomalie di funzionamento, forzate durante la simulazione, vengono tutte riconosciute e opportunamente segnalate.
Nella tabella che segue sono riportati i risultati ottenuti.

n°comandi	Comandi controllati	Allarmi generati	Tempo comp.
3	3	2	70
6	6	4	92
12	12	8	107
24	24	16	158

Nella figura 23 è rappresentato il grafico in cui sono riportati i valori della tabella.

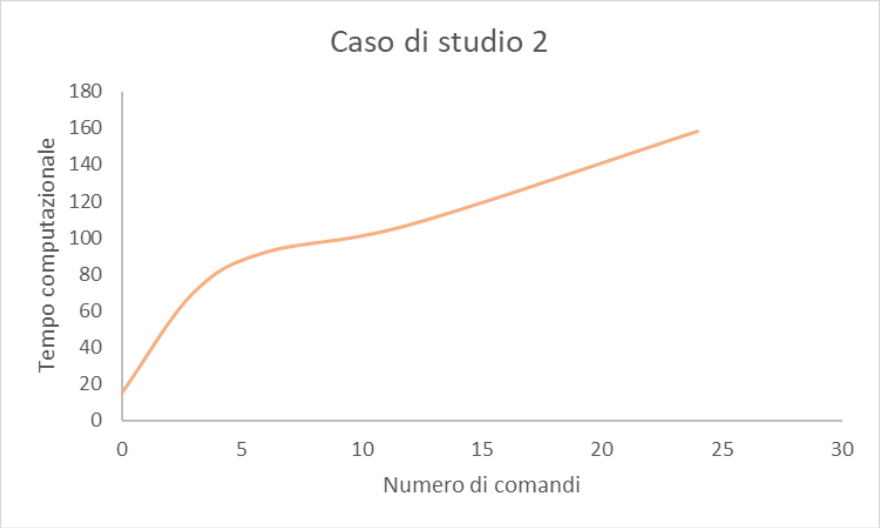

Figura 23 - Grafico dei risultati (Caso di studio 2)

Caso di studio 3

Nel terzo caso di studio si è deciso di realizzare un test massivo sulla robustezza dell'algoritmo applicando un'istanza di input che prevede solo comandi, impartiti dal supervisore, ai quali non corrisponde alcuna attuazione.
Anche in questo caso le dimensioni dell'istanza sono state fatte aumentare progressivamente.
Anche in questo caso, a conferma della bontà dello strumento realizzato, la procedura riconosce i malfunzionamenti generati e memorizza nel database le informazioni associate.
Di seguito si riportano i risultati della simulazione.

n°comandi	Comandi controllati	Allarmi generati	Tempo comp.
3	3	3	86
6	6	6	104
12	12	12	114
24	24	24	160

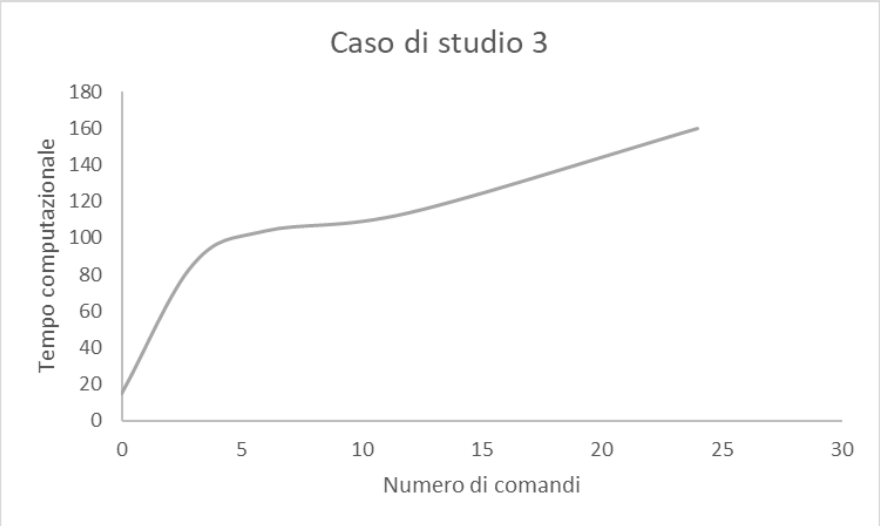

Figura 24 - Grafico dei risultati (Caso di studio 3)

Caso di studio 4

E'stato infine testato il comportamento della procedura a fronte di una particolare istanza di input. Si prevede l'inserimento di comandi sia prima che durante il settling time, ossia il tempo necessario per permettere alle grandezze associate ai comandi di stabilizzarsi.
Come da specifica i comandi impartiti dal supervisore durate il tempo di stabilizzazione non vengono controllati in quanto è concreto il rischio che le grandezze associate presentino degli andamenti ancora condizionati da componenti transitorie.
I comandi vengono quindi processati nel successivo ciclo di attivazione della procedura con la sicurezza che l'andamento delle grandezze associate sia andato a regime.

Per un'analisi più completa del comportamento dell'algoritmo sviluppato, di seguito si riportano i risultati ottenuti nei primi tre casi di studio sovrapponendo i dati ottenuti.

	Caso di studio 1			Caso di studio 2			Caso di studio 3		
n°comandi	Comandi controllati	Allarmi generati	Tempo comp.	Comandi controllati	Allarmi generati	Tempo comp.	Comandi controllati	Allarmi generati	Tempo comp.
3	3	0	66	3	2	70	3	3	86
6	6	0	80	6	4	92	6	6	104
12	12	0	86	12	8	107	12	12	114
24	24	0	150	24	16	158	24	24	160

Osservando le tabelle riguardanti i test condotti, si evidenziano i diversi comportamenti che l'algoritmo può assumere al variare dei parametri, mantenendo la coerenza con i risultati attesi.

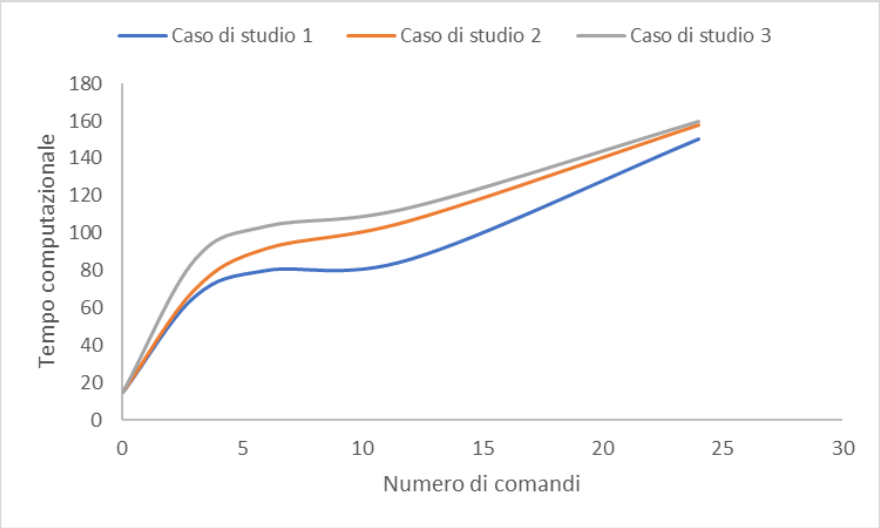

Figura 25 - Tempo computazionale dell'algoritmo

L'aumento del flusso dei dati non sovraccarica la capacità di calcolo e quindi influisce in minima parte sul tempo computazionale totale. Questo risulta essere leggermente influenzato dal numero di comandi da controllare e dal numero di allarmi che deve generare. Come evidenziato dalla figura 23, all'aumentare di questi fattori corrisponde un aumento del tempo impiegato per giungere alla soluzione, questo si mantiene comunque esiguo e accettabile ai fini del controllo.

3.1.7 Conclusioni e sviluppi futuri

In questo capitolo è stata descritta l'implementazione e la sperimentazione sull'edificio F40 delle logiche di diagnostica sviluppate per il sistema di gestione e controllo energetico attivo nello "Smart Village" della Casaccia. L'obiettivo è stato quindi l'ideazione e l'implementazione di un sistema, efficace ed efficiente, in grado di rilevare in modo automatico e in tempo reale le eventuali anomalie di controllo che possono occorrere nel sistema di supervisione energetica.

La regola di diagnostica è stata implementata all'interno dell'architettura ICT sviluppata nel PAR (RdS/2012/053) e quindi realizzata tramite codice Java affinche potesse essere verificato il suo funzionamento non solo teoricamente ma anche in un caso reale.
E'stata quindi creata un'applicazione in grado di controllare e gestire in tempo reale i dati provenienti dalla sensoristica dislocata nell'edificio F40 del Centro Ricerche Casaccia per verificare l'effettiva attuazione di un comando impartito dal sistema di supervisione.
I dati considerati non sono acquisiti direttamente dai dispositivi sul campo, ma sono mantenuti all'interno delle tabelle che caratterizzano il database "smarttowndb".

Durante il suo ciclo di attivazione la procedura realizzata esegue le funzioni descritte di seguito:

1. Preliminarmente si collega al database. Questa operazione, come quelle inerenti alla gestione della base di dati, si concretizza grazie all'utilizzo della libreria Java mySqlConnector.
2. Procede con la valutazione dell'avvenuta attuazione di un comando in relazione allo stato della grandezza di interesse ad esso associata. Si è quindi fissato un valore soglia in base al quale è possibile concretizzare l'analisi e stabilire la presenza o meno di un'anomalia di funzionamento.
3. In seguito alla valutazione viene aggiornato il campo "checked", della tabella "historian_control_actions3, riguardante il record corrispondente al comando analizzato. In questo modo è possibile indicare che il comando è stato processato.
4. L'ultima operazione eseguita prima che la procedura termini il suo ciclo è l'eventuale aggiornamento della tabella "alarm" in cui, nel caso sia riscontrato un malfunzionamento, verrà aggiunto un record in cui saranno memorizzate le informazioni riguardanti il comando ed il dispositivo che hanno generato l'anomalia, specificando l'istante in cui è avvenuta.

L'obiettivo di questo lavoro è stato raggiunto con la realizzazione di un sistema di rilevazione automatica delle anomalie applicato ai tre piani dell'edificio F40.
Con riferimento ai dati ottenuti in fase di simulazione si possono evidenziare i comportamenti attesi e trarre conclusioni circa l'utilizzo e l'efficacia della procedura proposta nella presente ricerca. Questo approccio sperimentale, testato su una casistica di eventi reali, ha lo scopo di migliorare l'efficienza delle procedure di rilevazione e segnalazione automatica delle anomalie relative al funzionamento di un supervisore energetico in ambito Smart Building.
Alcuni comportamenti inattesi della procedura verificatesi durante la fase sperimentale hanno comportato la modifica di determinati parametri raffinando l'esecuzione dell'analisi in questione.
I test effettuati hanno confermato la validità dello strumento implementato durante la ricerca. Questo risulta in grado di raccogliere, in tempi brevi, le informazioni necessarie per elaborare una valutazione dei comandi impartiti dal supervisore e segnalare eventuali anomalie di attuazione.
Il completamento di questa procedura di rilevamento delle anomalie pone le basi per una serie di possibili sviluppi futuri.
In una successiva progettualità, infatti, sarà prevista l'implementazione online della procedura sviluppata verificandone il comportamento a fronte di istanze di input acquisibili in tempo reale e provenienti dal sistema di supervisione attivo sul cluster di edifici presenti nel centro ricerche "Casaccia".
Contestualmente una verifica e una valutazione periodica dei risultati ottenuti potrà essere di ausilio per un ulteriore perfezionamento della strategia, se necessario.
I dati prodotti potranno quindi essere utilizzati nelle fasi di Situation Assessment per fornire informazioni ad alto livello sullo stato dei dispositivi e, in generale, del sistema di supervisione.
Dal punto di vista implementativo l'algoritmo nel suo complesso risulta essere, non solo estendibile a qualsiasi edificio nel settore del terziario, ma anche integrabile con l'analisi di malfunzionamenti generati da altre tipologie di comando, come ad esempio comandi del sistema di gestione idrica o di climatizzazione degli ambienti.
Risulta infatti poco oneroso estendere questo tipo di controllo anche ad altre tipologie di comando, previa individuazione delle grandezze fisiche da monitorare associate ai comandi impartibili. E' altresì

fondamentale stabilire il tempo di attesa necessario prima che si possa evidenziare empiricamente l'attuazione prevista.
Ad esempio, nel caso in cui si voglia controllare l'avvenuta attuazione di un comando termico di riscaldamento o raffrescamento di un ambiente, sarà opportuno verificare, dopo un tempo ragionevole, il raggiungimento del set point desiderato da parte della temperatura misurata tramite sensoristica.

Riferimenti Bibliografici

[1] http://www.mercatoelettrico.org/It/default.aspx

[2] “Economic analysis case studies of battery energy storage with SAM” – N.DiOrio, A.Dobos, S.Janzou (2015) *National Renewable Energy Laboratory: Denver, CO, USA*.

[3] "A relational model of data for large shared data banks" - E.F.Codd - Communications of the ACM – Vol.13 Iss.6 Pag.377-387, June 1970 - ISSN:0001-0782 EISSN:1557-7317

[4] "A Guide to the SQL Standard" - CJ Date, H Darwen – 1987

Curriculum Vitae Autori del rapporto tecnico

Stefano Panzieri

Stefano Panzieri è professore associato presso l’Università degli Studi Roma Tre e svolge la sua attività didattica presso le facoltà di Ingegneria Elettronica, Meccanica ed Informatica dove tiene i corsi di Controllo Digitale, Controllo dei Processi e Controlli Automatici. E’ il coordinatore del laboratorio di Automatica e del laboratorio di Robotica Autonoma e Fusione Sensoriale. E’ un membro di IEEE e del gruppo di lavoro sulle Infrastrutture Critiche. I suoi interessi di ricerca sono nel campo dei sistemi di Controllo Industriale, Teoria dei Sistemi, Sistemi Complessi e CIP. E’ autore di molti articoli sperimentali che riguardano i robot mobili, i robot industriali, il controllo con apprendimento iterativo, logica Fuzzy, stima bayesiana, teoria di Dempster-Shafer.

Dario Masucci

Dario Masucci è un ingegnere dell’automazione. Lavora come collaboratore presso l'Università di Roma Tre dal 2015, quando ha conseguito la laurea. I suoi interessi di ricerca includono gli algoritmi di ottimizzazione multi-obiettivo, in particolare lo sviluppo di strumenti di supporto alle decisioni multi-criterio, la sostenibilità energetica e la definizione di modelli energetici per edifici. Negli ultimi due anni ha avuto l'opportunità di lavorare su diversi progetti europei (Grant CIPS, Grant H2020) sviluppando e progettando algoritmi decisionali multi-criterio, modelli di interdipendenza per le infrastrutture critiche e strumenti di gestione delle emergenze utilizzabili nel campo della protezione delle ICs. Attualmente, ha iniziato un rapporto di collaborazione con la Divisione Energy di ENEA nel Centro Ricerche Casaccia mirato allo sviluppo di strumenti informatici per la rilevazione di anomalie.

Fiorella Lauro

Fiorella Lauro, ha conseguito la Laurea Specialistica in Ingegneria dell'Automazione presso l'Università della Calabria. Nel lavoro di tesi, svolto presso l’unità UTTEI del Centro Ricerche ENEA di Roma, si è occupata dello sviluppo di un nuovo approccio per la modellazione dei consumi energetici degli edifici al fine di realizzare un innovativo sistema di gestione remota per la diagnostica delle anomalie e l’ottimizzazione del comportamento energetico delle reti di edifici. E’ autore di due pubblicazioni scientifiche sull’ensembling di reti neurali e su modelli ibridi applicati ai consumi energetici degli edifici. Attualmente è ricercatore a contratto presso il Politecnico di Torino e la sua attività di ricerca riguarda la modellazione dei consumi energetici degli edifici attraverso approcci metodologici inversi e l’individuazione di efficaci metodologie diagnostiche.