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

MINISTERO DELLO SVILUPPO ECONOMICO

Ricerca di Sistema elettrico

Sviluppo modulo di interfacciamento con ApioDB e rilevazione automatica di comandi malevoli e anomalie in un sistema di supervisione energetica in ambito Smart Buildings

Dario Masucci, Martina Botticelli

ROMA TRE
UNIVERSITÀ DEGLI STUDI

Report RdS/PAR2017/098

SVILUPPO MODULO DI INTERFACCIAMENTO CON APIODB E RILEVAZIONE AUTOMATICA DI COMANDI MALEVOLI E ANOMALIE IN UN SISTEMA DI SUPERVISIONE ENERGETICA IN AMBITO SMART BUILDINGS

D. Masucci (Università Roma Tre), M. Botticelli (Università delle Marche)

Settembre 2018

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Annuale di Realizzazione 2017
Area: Efficienza energetica e risparmio di energia negli usi finali elettrici e interazione con altri vettori energetici
Progetto: D.1 Tecnologie per costruire gli edifici del futuro
Responsabile del Progetto: Stefano Pizzuti, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione "Riconoscimento di anomalie di funzionamento e sviluppo di strumenti orientati alla robustezza informatica negli edifici"

Responsabile scientifico ENEA: Stefano Pizzuti

Responsabile scientifico Università Roma Tre: prof. Stefano Panzieri

Indice

Elenco delle figure

1. Introduzione

L'attività di ricerca e sviluppo condotta nell'ambito del presente accordo di collaborazione tra ENEA e

Università Roma Tre ha proseguito alcuni studi iniziati nelle precedenti annualità (RdS/PAR2017) inerenti la gestione energetica di edifici terziari. In particolare, le linee di attività riprese e ulteriormente sviluppate sono state:

- 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.

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 di queste funzionalità 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.
E' con queste finalità che si è svolta l'attività di ricerca che ha riguardato 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 fancoil dislocati all'interno di un edificio. Inoltre, a seguito dell'accordo di fornitura tecnologica hardware e software stipulato tra l'ENEA e l'azienda APIO srl, è previsto lo sviluppo di un modulo di interfacciamento con il database ApioDB per assicurare la completa funzionalità del sistema e la continuità con la logica del supervisore pre-esistente.
Le attività sono state delineate con le seguenti modalità:

- Sviluppare un modulo di interfacciamento e invio dati da ApioDB a SmartTownDB;
- Valutare la messa in opera della procedura di Fault Detection definita nella precedente annualità;
- Sviluppare un applicativo per la rilevazione automatica di anomalie in un sistema di climatizzazione elettrica;
- Sviluppare un modulo per il riconoscimento di comandi malevoli.

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".

2. Sviluppo modulo di interfacciamento e invio dati da ApioDB a SmartTownDB.

Lo scopo di questa attività è lo sviluppo di modulo di interfacciamento per il recupero dei dati di consumo elettrici, termici e di comfort dell'edificio F40, sito C.R. ENEA Casaccia, a livello di edificio, piano e singola stanza. I dati attualmente confluiscono ad un DBApio dedicato su un Server Enea. Nella seguente attività verrà effettuato l'invio di questi ultimi dal DBApio ad una piattaforma di elaborazione e diagnostica dei dati in un contesto più ampio di Smart Village. Quest'ultima costituisce una piattaforma integrata ICT alla quale confluiscono dati di diversa natura ed è quindi il fulcro delle interazioni tra le diverse infrastrutture dello Smart Village, permettendo un'acquisizione a analisi d'insieme sui dati. Tale piattaforma integrata comprende un apposito DB, chiamato SmartTownDB, al quale i dati dovranno quindi confluire, dopo essere stati prelevati ed elaborati nell'opportuno formato, per permettere azioni successive di analisi e diagnostica.
A seguito dell'accordo di fornitura tecnologica hardware e software tra l'ENEA e l'azienda APIO srl, sono stati effettuati una serie di interventi di ammodernamento del sistema di monitoraggio e attuazione attivo nell'edificio F40.
I nuovi dispositivi APIO, che sono disponibili a seguito di questi interventi, presentano un sistema di storage dei dati centralizzato e proprietario.

Per assicurare la completa funzionalità del sistema e la continuità con la logica del supervisore pre-esistente è stato quindi opportuno integrare in qualche modo la nuova struttura di storage con quella utilizzata fino ad ora dal sistema di supervisione stesso.
Il sistema di automazione APIO prevede un tempo di campionamento e di controllo di circa 15 minuti, perciò le tabelle del database dedicato, *APIOdb*, verranno modificate ogni 15 minuti.
È stato quindi possibile definire una routine attiva sul server *smarttown* che ogni 15 minuti, con sfasamento di qualche secondo rispetto al ciclo di attivazione dei dispositivi APIO, che interroghi il database di APIO in cerca di nuovi records memorizzati, aggiornando eventualmente il database *smarttowndb* nelle varie tabelle.
Così facendo sarà possibile mantenere attive le routine di controllo definite sul server ENEA.
Dopo un'attenta analisi del database *smarttowndb* è emerso che le tabelle interessate dagli aggiornamenti sono:

- *historian*, che contiene lo storico delle misure rilevate (fermo dal 2016);
- *historian_control_actions3*, storico dei comandi impartiti dal supervisore attivo nell'edificio F40;
- *scada*, contiene le informazioni "statiche" che identificano ogni dispositivo di misura o di attuazione presente nell'edificio F40 e l'ultimo valore di input o di output registrato.
- *Relazione_scada_apio*, che è una nuova tabella creata appositamente per questa attività con lo scopo di mappare la corrispondenza tra le vecchie property mappate con ID in scada e le nuove property presenti nelle varie tabelle del DBApio.

Di seguito (Figura 1) si riportano le tabelle sopraelencate con diagramma ER:

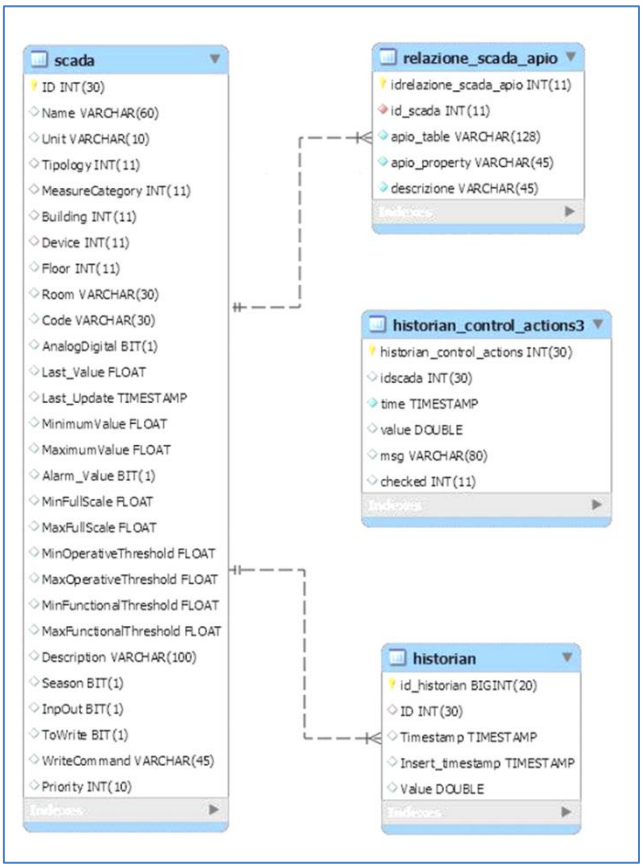

Figura 1 - Tabelle interessate dagli aggiornamenti in smarttowndb

Nel DBApio sul server Enea, i dati sono organizzati in tabelle, una per ogni sensore. Pertanto i campi presenti in ogni tabella variano a seconda della tipologia di sensore. Ogni campo di una tabella rappresenta

infatti una property del particolare sensore. Nel DBApio i nomi delle tabelle sono costituiti da: IdSensore_IdBoard a cui il sensore è associato.
Di seguito (Figura 2) si riporta un esempio di una delle tante tabelle presenti nel DBApio sul server Enea che si riferisca ad un Meter presente nel quadro elettrico del primo piano dell'edificio F40 (ID = 34), associato al gateway di edificio (ID = 837cc430-bf0a-4477-a4c2-a28e52d21046). La terminazione "avg15" presente alla fine del nome della tabella, è indice che la tabella che stiamo prendendo in considerazione si riferisce a dati mediati sui 15 minuti. Infatti sono presenti in altrettante tabelle organizzate allo stesso modo ma che non contengono la dicitura finale "_avg15", che contengono i dati puntuali restituiti real-time a variazione dal sensore. I dati mediati sui 15 minuti che prendiamo in considerazione per questa attività, sono quindi frutto di elaborazioni quartorarie utilizzando appositi algoritmi sviluppati.

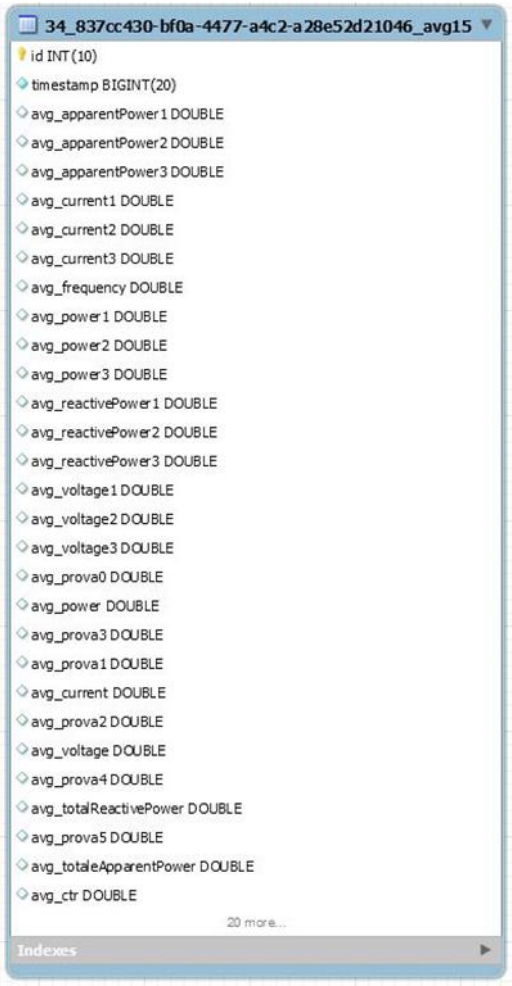

Figura 2 - Esempio tabella Meter su DBApio

2.1 Introduzione tabella di mapping in smarttownDB: "relazione_scada_apio"

L'attività in oggetto ha quindi consistito nell'implementazione di un modulo di interfacciamento e invio dati da ApioDb a SmartTownDB. Tale modulo è stato realizzato attraverso l'implementazione di una script in Java.
Tale script sarà in esecuzione sul Server Enea, e provvederà quindi alla migrazione dei dati tra i due DB ogni 15 minuti. Infatti i dati puntuali, provenienti dalla sensoristica dell'edificio F40, vengono collezionati in real-time ma elaborati per l'aggregazione al quarto d'ora tramite uno script che elabora i dati del quarto d'ora precedente e alimenta le tabelle preposte alla raccolta di dati quartorari (diverse per ogni sensore, vedi

paragrafo precedente). Sono proprio queste infatti le tabelle contenenti i dati quartorari di interesse che si vogliono migrare verso smarttownDB.
Si è resa necessaria l'introduzione di una nuova tabella all'interno di smarttownDB chiamata *"relazione_scada_apio"* che è stata creata appositamente per questa attività con lo scopo di mappare la corrispondenza tra i vecchie property mappate con ID in scada e le nuove property presenti nelle varie tabelle del DBApio. Di seguito (Figura 3) vediamo un piccolo scorcio di tale tabella.

idrelazione_scada_apio	id_scada	apio_table	apio_property	descrizione
1	118	0001_837cc430-bf0a-4477-a4c2-a28e52d21046_avg15	avg_temperatura	temperatura stanza 107
2	169	34_837cc430-bf0a-4477-a4c2-a28e52d21046_avg15	avg_power	potenza attiva luci piano 1
3	270	43_837cc430-bf0a-4477-a4c2-a28e52d21046_avg15	avg_power	potenza attiva luci piano 2

Figura 3 - Scorcio tabella "relazione_scada_apio"

Notiamo cose essa contiene 5 campi:

- Idrelazione_scada_apio: che è semplicemente un id progressivo.
- Id_scada: corrisponde all'ID assegnato nel precedente sistema ad una determinata property acquisita e che sono mappati nella tabella "scada".
- Apio_table: identifica la tabella in ApioDB corrispondente al sensore che rileva quella specifica misura (per la costruzione del nome si veda il paragrafo precedente).
- Apio_property: come tale property è nominata nella tabella Apio, ossia il nome della colonna in cui cercare quella property nella Apio_table.
- Descrizione: contiene una breve descrizione della misura in oggetto.

Successivamente è stato inserito un sesto campo che si è reso necessario per l'individuazione della tipologia di dato da mappare: misura booleana, misura generica o azione.
I campi della nuova tabella "relazione_scada_apio", devono essere inseriti manualmente da un amministratore nella fase di start up del processo, in quanto devono essere immesse informazioni che devono essere note a priori e che fungono appunto da mapping tra il nuovo ed il vecchio sistema.

2.2 Sviluppo modulo di interfacciamento e invio dati da ApioDB a SmartTownDB.

L'attività in oggetto ha quindi consistito nell'implementazione di un modulo di interfacciamento e invio dati da ApioDb a SmartTownDB. Tale modulo è stato realizzato attraverso l'implementazione di una script in Java.
Tale script sarà in esecuzione sul Server Enea, e provvederà quindi alla migrazione dei dati tra i due DB ogni 15 minuti. Infatti i dati puntuali, provenienti dalla sensoristica dell'edificio F40, vengono collezionati in real-time ma elaborati per l'aggregazione al quarto d'ora tramite uno script che elabora i dati del quarto d'ora precedente e alimenta le tabelle preposte alla raccolta di dati quartorari (diverse per ogni sensore, vedi paragrafo precedente). Sono proprio queste infatti le tabelle contenenti i dati quartorari di interesse che si vogliono migrare verso smarttownDB.
Di seguito (Figura 5) una semplificazione del modulo sviluppato.

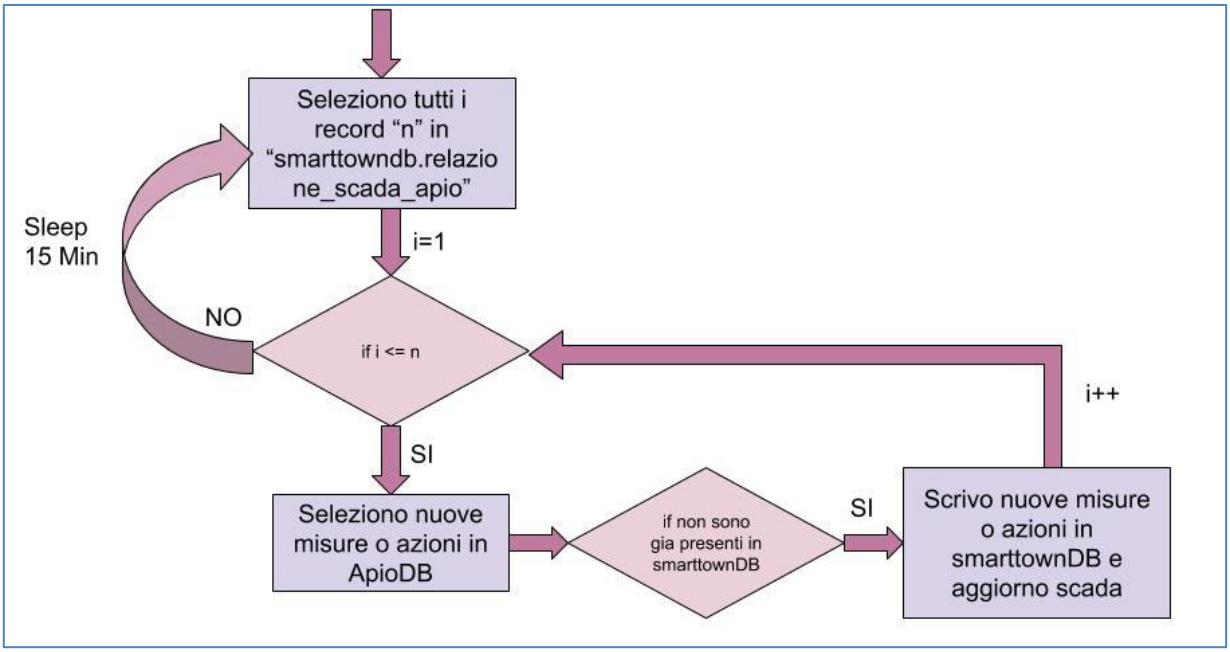

Figura 4 - Migrazione dati da ApioDB a smarttownDB

Il primo step che effettua l'algoritmo sviluppato è quello di caricarsi un file di configurazione, che si è provveduto a creare, all'interno del quale scrivere i settaggi dello script: parametri di connessione a smrttownDb, parametri di connessione a ApioDB, intervallo di esecuzione dello script e il numero di record da valutare nel DB Apio (che approfondiremo meglio successivamente).
Una volta caricati i settaggi, viene lanciato un loop che si esegue ogni 15 minuti (tempo modificabile nei settaggi). All'interno di tale loop avviene la migrazione dei dati quartorari da ApioDB a SmartTownDB.
La prima connessione viene fatta al database smarttown, dove vengono selezionati tutti i campi di interesse nella nuova tabella introdotta "*relazione_scada_apio*". Per ogni record presente in tale tabella viene a questo effettuata una chiamata al DB Apio, con lo scopo di selezionare I nuovi dati acquisiti per la misura in questione dal DB APio. A questo punto viene fatta una verifica di sicurezza per per vedere se tali dati sono già presenti in smarttownDB, e se non ci sono vengono inseriti. Ad ogni nuovo inserimento viene anche fatta un update sulla tabella di smarttown "scada" per aggiornare i campi "last_value" e "last_update" rispettivamente con l'ultimo valore letto e il timestamp corrispondente.
A questo punto viene valuto il record successivo della select su "relazione_scada_apio".
Di seguito (Figura 5) un esempio di record inserito nella tabella smarttowndb.hostorian, dove vediamo una misura di "potenza attiva luci piano 2" (ID = 270) che corrisponde per l'appunto a id_scada= 270 come possiamo vedere dalla tabella "scada" in Figura 3. Il "Timestamp" è quello a cui il dato quartorario si riferisca, mentre l' "Insert timestamp " è il timestamp di inserimento del dato. Infine il campo "Value" contiene il valore della misura acquisita. "Id_historian" è un semplice id progressivo.

id_historian	ID	Timestamp	Insert_timestamp	Value
30272926	270	2018-05-25 20:45:00	2018-05-25 21:39:22	0.258

Figure 1 - Esempio di dato inserito in "smarttowndb.historian"

Come abbiamo visto, lo storico dei dati confluisce quindi da ApioDB alla tabella "historian" di smarttownDB.
Il nuovo sistema Apio installato nell'edificio F40, non consente solo il monitoraggio dell'edificio stessa, ma effettua anche attuazioni, come ad esempio accensione/spegnimento luci e fan-coil.
Nel caso di dati inerenti ad azioni di controllo, i dati, invece che confluire in historian, vengono collezionati su un apposita tabella chiamata "historian_control_action3" sempre all'interno di smarttowndb.

Le attuazioni in apio vengono collezionate come booleani corrispondenti all'azione effettuata, che vengono associati all'azione stessa, identificata come una qualsiasi property del sensore. Pertanto, tali azioni di controllo vengono trattate ed acquisite esattamente allo stesso modo di una qualsiasi misura e quindi mappate in "relazione_scada_apio", ma saranno storicizzate come valori 0/1 all'interno della tabella "historian_control_action3" in smarttowndb, anzi che "historian".
Di seguito (Figura 6) un esempio di record inserito nella tabella "smarttowndb.historian_control_action3", dove vediamo un azione di controllo "spengo la luce corridoio piano 0", con ID = 436 che corrisponde per l'appunto a id mappato in scada. Il "time" è il timestamp a cui l'azione di controllo si riferisce e il campo "value" contiene il valore booleano dell'azione eseguita. "historian_control_actions" è un semplice id progressivo. Infine il campo "msg" contiene la descrizione dell'azione effettuata.

historian_control_actions	idscada	time	value	msg
387	436	2018-05-10 11:00:58	0	spengo la luce corridoio piano 0

Figure 2 - Esempio di azione di controllo inserita in "smarttowndb.historian_control_action3"

3. Valutazione messa in opera della procedura di Falut Detection per il sistema elettrico

L'attività di ricerca e sviluppo condotta nell'ambito del precedente accordo di collaborazione tra ENEA e Università degli Studi Roma Tre ha consistito nella definizione e nell'implementazione di una procedura di fault detection in grado di verificare automaticamente e in tempo reale l'avvenuta attuazione di comandi sequenziali impartiti da un sistema di controllo e supervisione.

La procedura che è stata realizzata può essere riassunta schematicamente 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.

E' stata quindi realizzata 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.
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

Tabella 1 - Corrispondenze comando/misura

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.

3.1 Fase di sperimentazione

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.
La regola di diagnostica è stata implementata all'interno dell'architettura ICT sviluppata nel PAR (RdS/2012/053) e quindi realizzata tramite codice Java affinché potesse essere verificato il suo funzionamento non solo teoricamente ma anche in un caso reale.
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.

Le informazioni che si è ritenuto opportuno mantenere nel file.log 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.

Durante la sperimentazione si è tenuto traccia delle grandezze di seguito elencate, e si sono valutate le misure che queste hanno assunto in fase di messa in opera:

- Numero di comandi da controllare;
- Numero di anomalie segnalate;
- Tempo computazionale impiegato per l'esecuzione della procedura.

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.

Di seguito, nella figura 1, è riportato il grafico ottenuto dall'analisi delle anomalie segnalate dalla procedura, rispetto ai comandi impartiti dal supervisore.

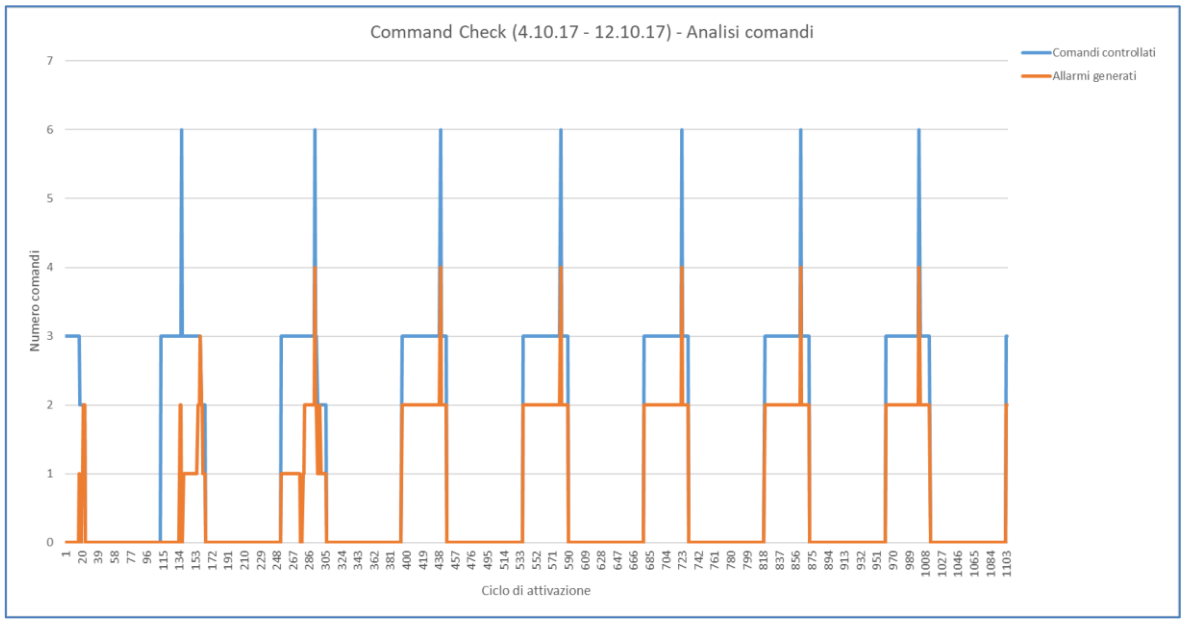

Figura 5 - Analisi dei comandi (4-12.10.17)

Sull'asse delle ascisse si ha il ciclo di attivazione, sull'asse delle ordinate il numero di comandi. In blu si osservano i comandi controllati, mentre in arancione le anomalie riconosciute.

Nella figura 2 è rappresentato il tempo computazionale che l'applicazione di fault detection impiega per espletare un ciclo di attivazione. Questo corrisponde al tempo in cui la procedura occupa il canale di comunicazione per effettuare le operazioni di lettura e scrittura con il database.

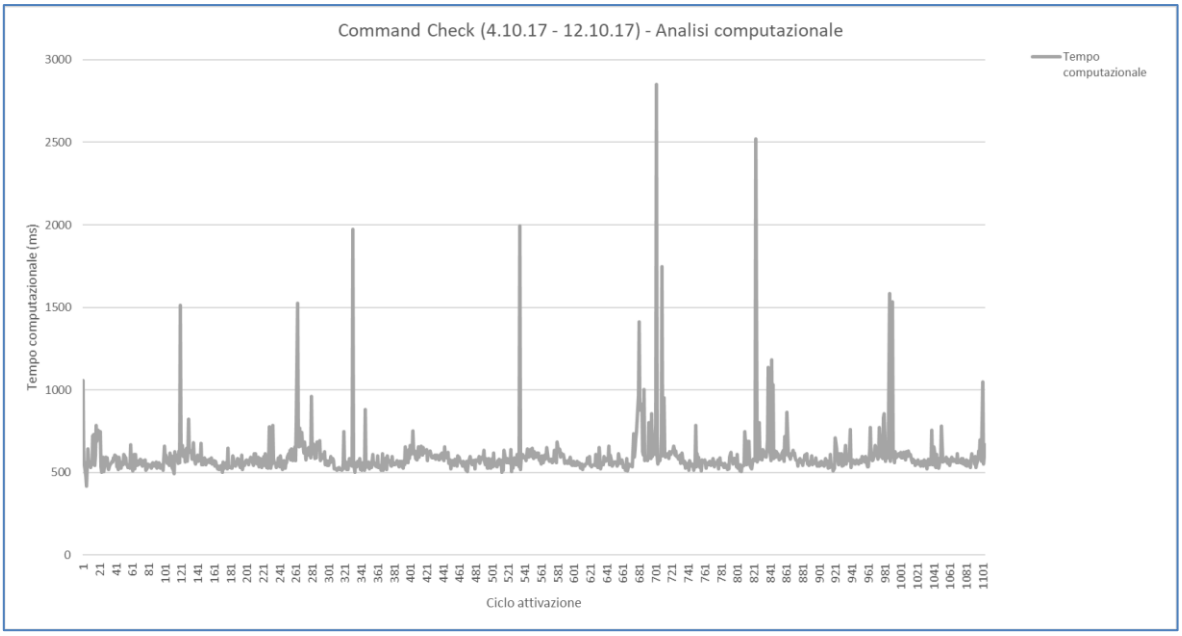

Figura 6 - Analisi del tempo computazionale (4-12.10.17)

Tramite la figura 3 si vuole mettere in evidenza il rapporto tra i comandi valutati e le anomalie di funzionamento riscontrate. Queste ultime sono state quindi suddivise in base al comando sulle quali sono state rilevate.

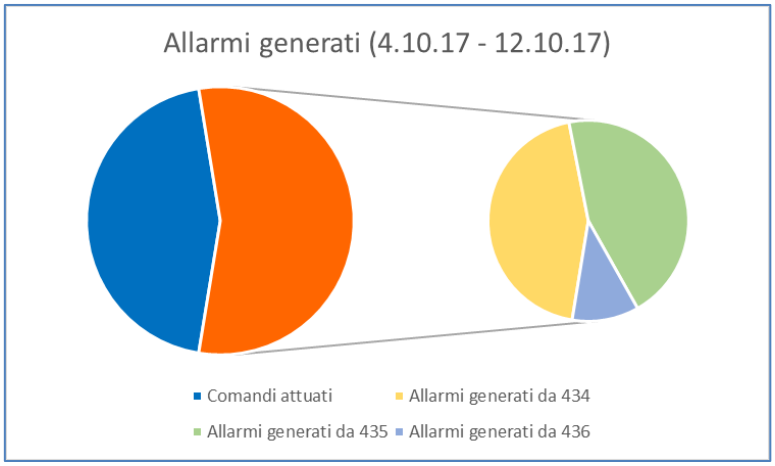

Figura 7 - Analisi degli allarmi generati (4-12.10.17)

Per una valutazione puntuale della procedura e del suo comportamento è stata effettuata un'analisi dei risultati ottenuti previa suddivisione in base al giorno di acquisizione.

Giorno 1 – mercoledì 4.10.17

Caratteristiche simulazione:

- Start time – 15:14
- End time – 23:53
- Attivazione procedura ogni 10 minuti

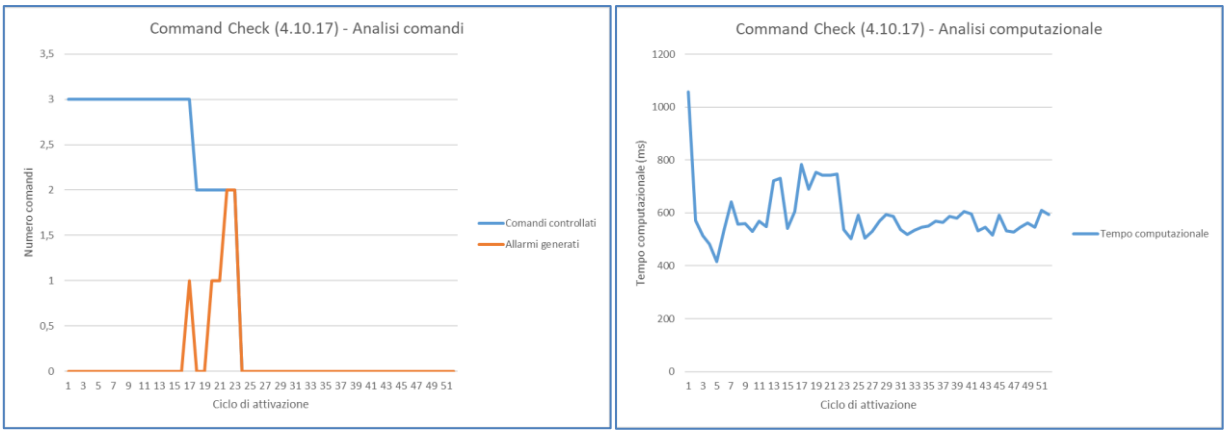

Figura 8 - Analisi dei comandi e del tempo computazionale (4.10.17)

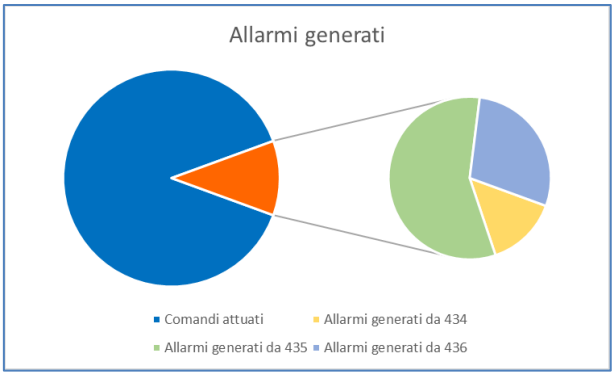

Figura 9 - Analisi degli allarmi generati (4.10.17)

Risultati della simulazione:

- Totale comandi controllati = 63
- Comandi attuati = 56
- Totale allarmi generati = 7
- Allarmi generati da 434 = 1
- Allarmi generati da 435 = 4
- Allarmi generati da 436 = 2

Giorno 2 – giovedì 5.10.17

Caratteristiche simulazione:

- Start time – 00:03
- End time – 23:58
- Attivazione procedura ogni 10 minuti

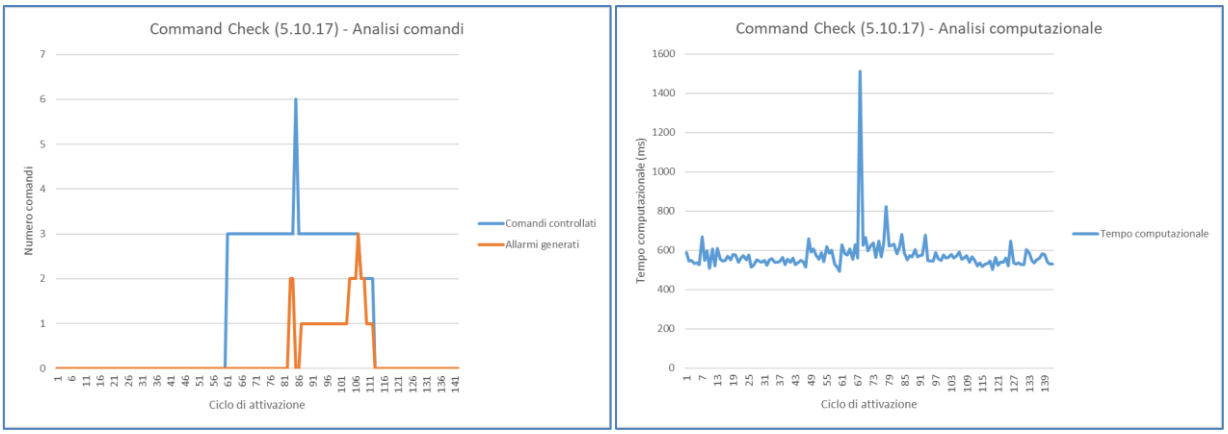

Figura 10 - Analisi dei comandi e del tempo computazionale (5.10.17)

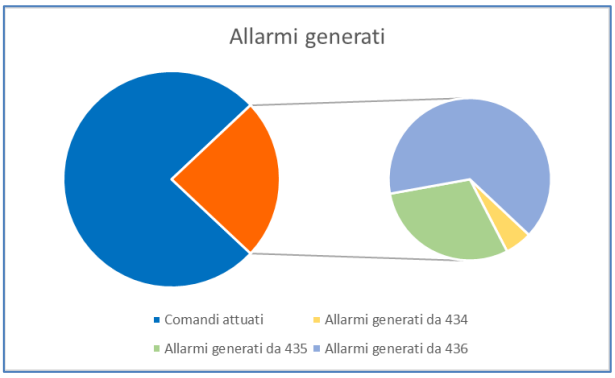

Figura 11 - Analisi degli allarmi generati (5.10.17)

Risultati della simulazione:

- Totale comandi controllati = 154
- Comandi attuati = 117
- Totale allarmi generati = 37
- Allarmi generati da 434 = 2
- Allarmi generati da 435 = 11
- Allarmi generati da 436 = 24

Giorno 3 – venerdì 6.10.17

Caratteristiche simulazione:

- Start time – 00:08
- End time – 23:53
- Attivazione procedura ogni 10 minuti

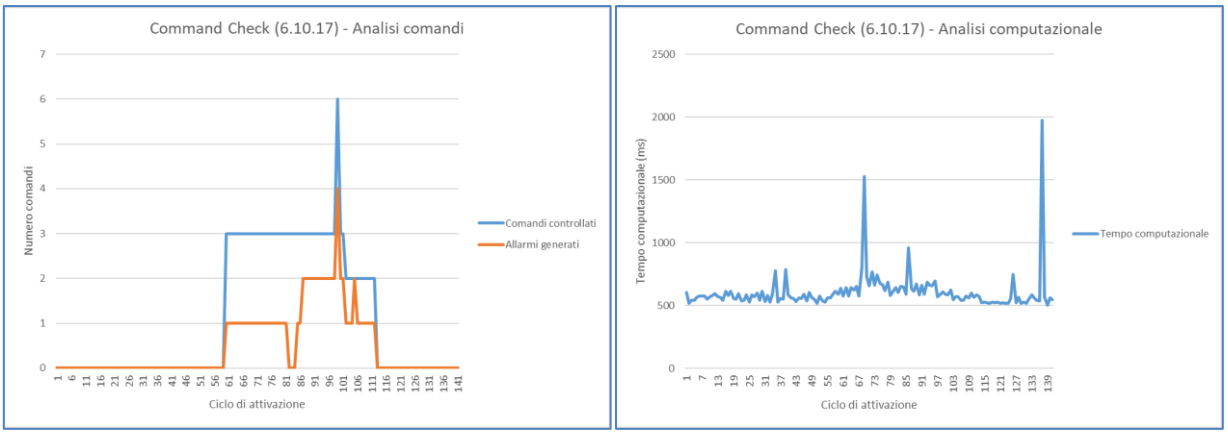

Figura 12 - Analisi dei comandi e del tempo computazionale (6.10.17)

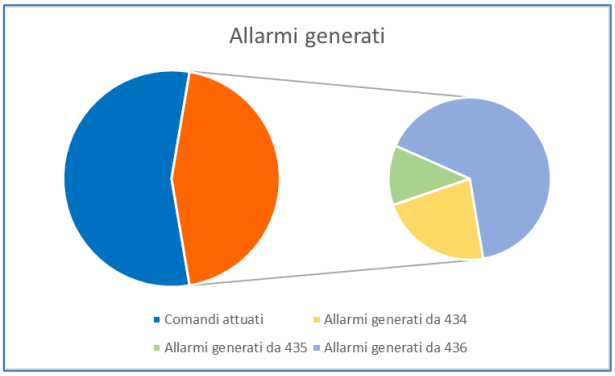

Figura 13 - Analisi degli allarmi generati (6.10.17)

Risultati della simulazione:

- Totale comandi controllati = 151
- Comandi attuati = 83
- Totale allarmi generati = 68
- Allarmi generati da 434 = 16
- Allarmi generati da 435 = 8
- Allarmi generati da 436 = 44

Giorno 4 – sabato 7.10.17

Caratteristiche simulazione:

- Start time – 00:03
- End time – 23:58
- Attivazione procedura ogni 10 minuti

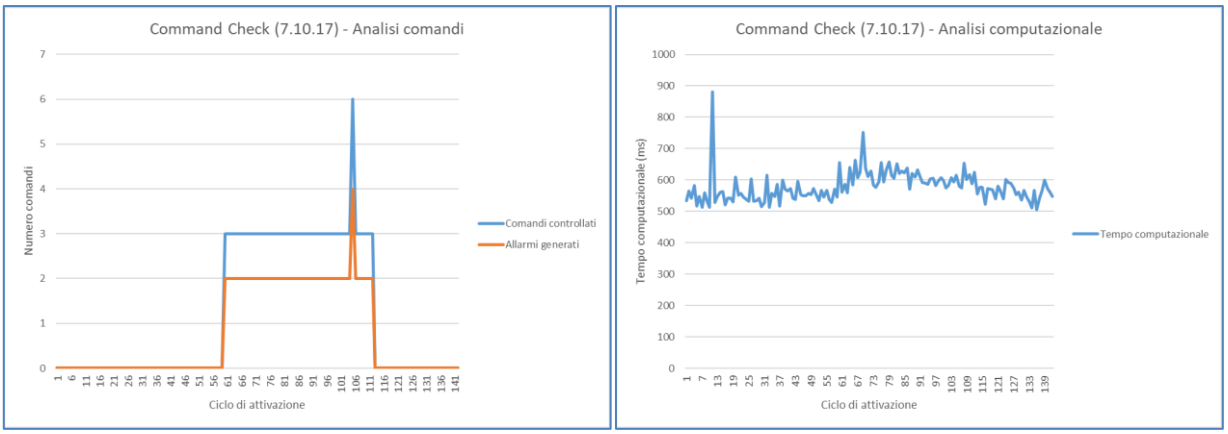

Figura 14 - Analisi dei comandi e del tempo computazionale (7.10.17)

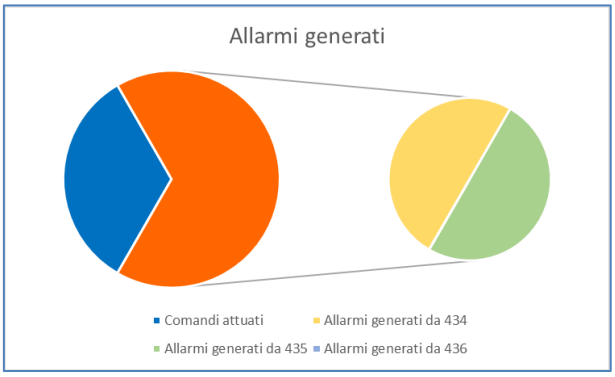

Figura 15 - Analisi degli allarmi generati (7.10.17)

Risultati della simulazione:

- Totale comandi controllati = 162
- Comandi attuati = 54
- Totale allarmi generati = 108
- Allarmi generati da 434 = 54
- Allarmi generati da 435 = 54
- Allarmi generati da 436 = 0

Giorno 5 – domenica 8.10.17

Caratteristiche simulazione:

- Start time – 00:08
- End time – 23:53
- Attivazione procedura ogni 10 minuti

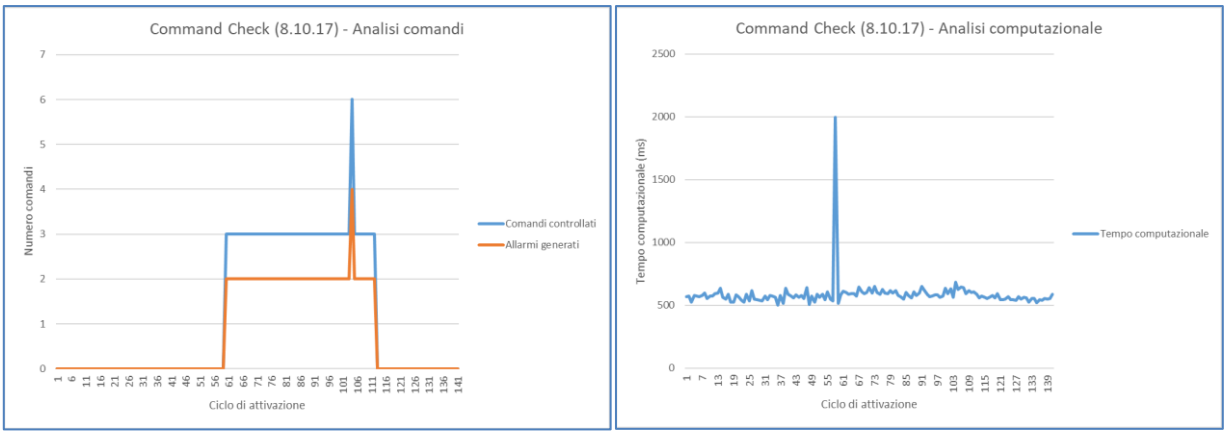

Figura 16 - Analisi dei comandi e del tempo computazionale (8.10.17)

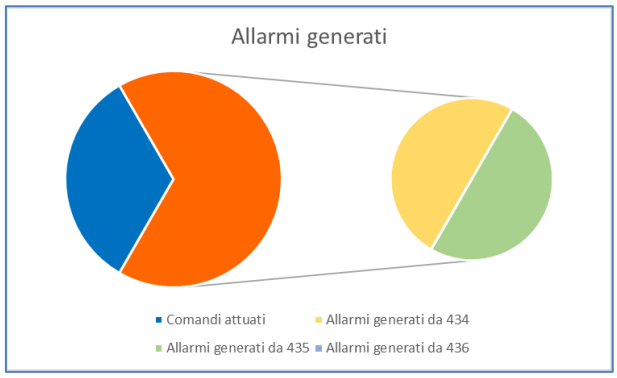

Figura 17 - Analisi degli allarmi generati (8.10.17)

Risultati della simulazione:

- Totale comandi controllati = 162
- Comandi attuati = 54
- Totale allarmi generati = 108
- Allarmi generati da 434 = 54
- Allarmi generati da 435 = 54
- Allarmi generati da 436 = 0

Giorno 6 – lunedì 9.10.17

Caratteristiche simulazione:

- Start time – 00:03
- End time – 23:58
- Attivazione procedura ogni 10 minuti

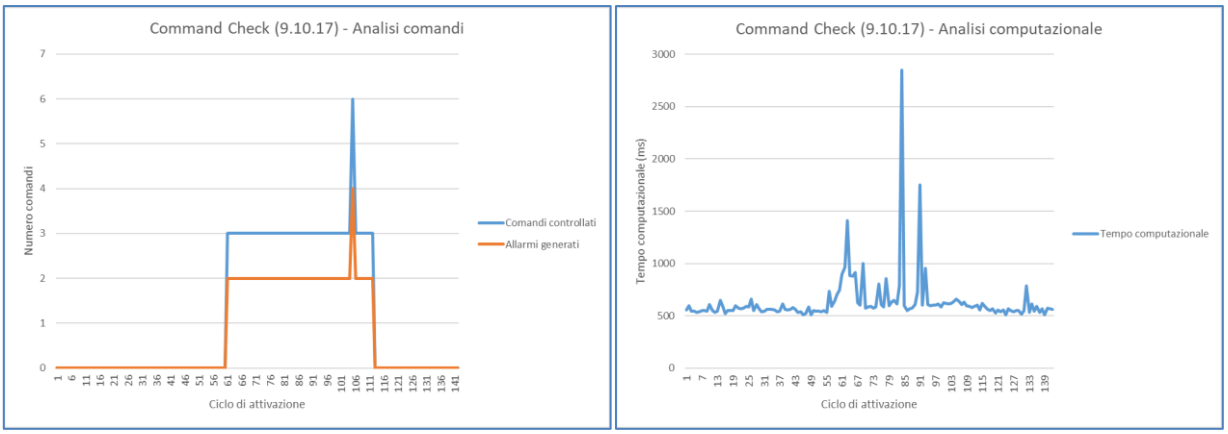

Figura 18 - Analisi dei comandi e del tempo computazionale (9.10.17)

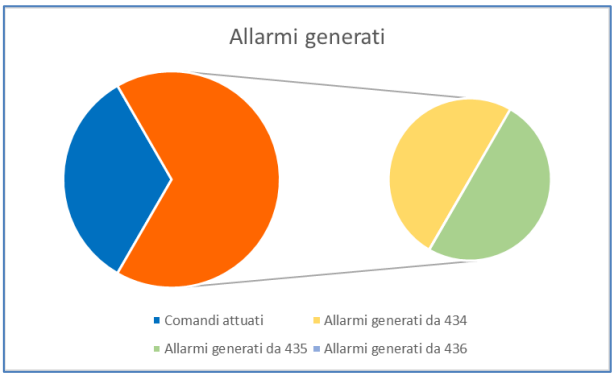

Figura 19 - Analisi degli allarmi generati (9.10.17)

Risultati della simulazione:

- Totale comandi controllati = 159
- Comandi attuati = 53
- Totale allarmi generati = 106
- Allarmi generati da 434 = 53
- Allarmi generati da 435 = 53
- Allarmi generati da 436 = 0

Giorno 7 – martedì 10.10.17

Caratteristiche simulazione:

- Start time – 00:08
- End time – 23:53
- Attivazione procedura ogni 10 minuti

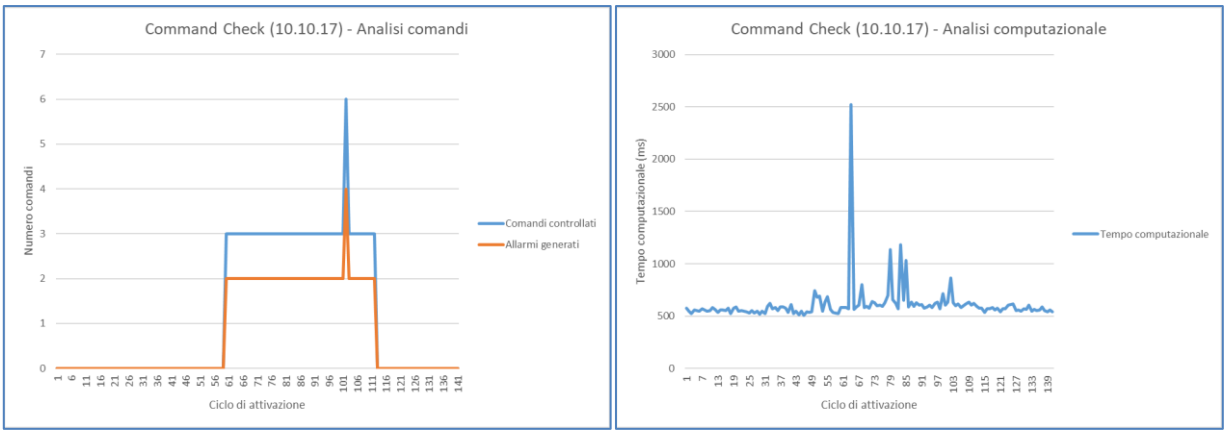

Figura 20 - Analisi dei comandi e del tempo computazionale (10.10.17)

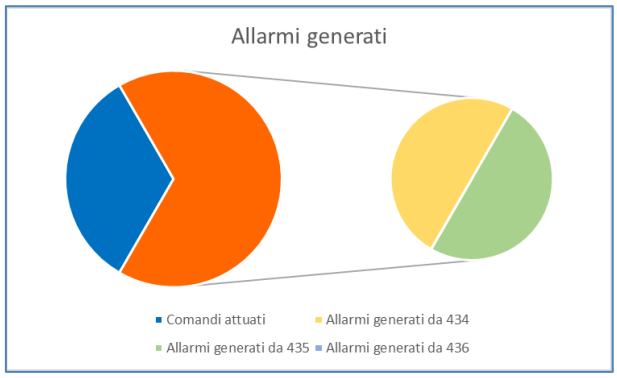

Figura 21 - Analisi degli allarmi generati (10.10.17)

Risultati della simulazione:

- Totale comandi controllati = 162
- Comandi attuati = 54
- Totale allarmi generati = 108
- Allarmi generati da 434 = 54
- Allarmi generati da 435 = 54
- Allarmi generati da 436 = 0

Giorno 8 – mercoledì 11.10.17

Caratteristiche simulazione:

- Start time – 00:03
- End time – 23:58
- Attivazione procedura ogni 10 minuti

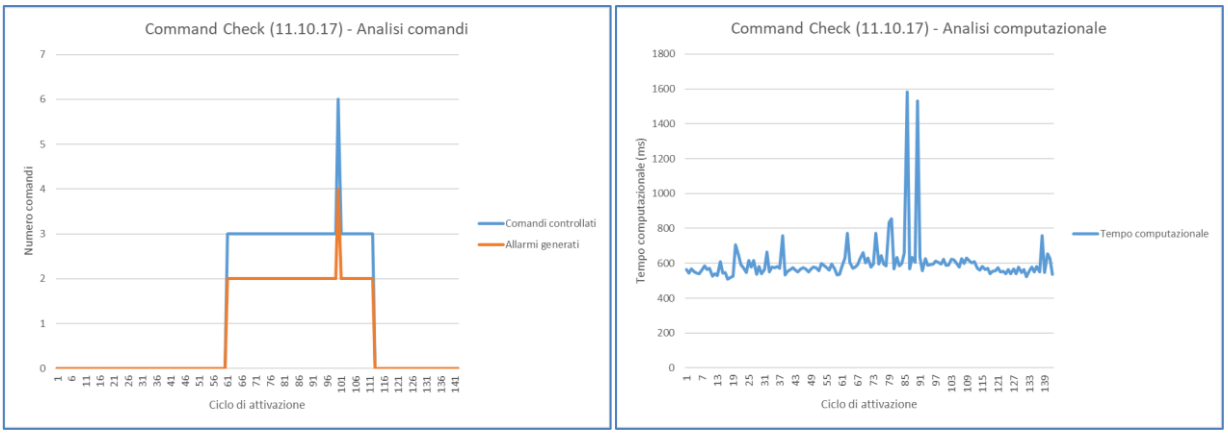

Figura 22 - Analisi dei comandi e del tempo computazionale (11.10.17)

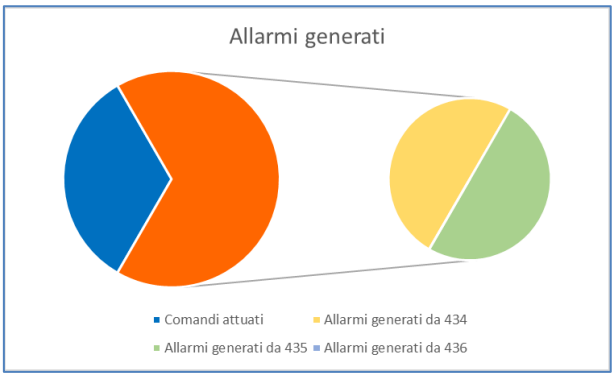

Figura 23 - Analisi degli allarmi generati (11.10.17)

Risultati della simulazione:

- Totale comandi controllati = 159
- Comandi attuati = 53
- Totale allarmi generati = 106
- Allarmi generati da 434 = 53
- Allarmi generati da 435 = 53
- Allarmi generati da 436 = 0

Giorno 9 – giovedì 12.10.17

Caratteristiche simulazione:

- Start time – 00:08
- End time – 10:19
- Attivazione procedura ogni 10 minuti

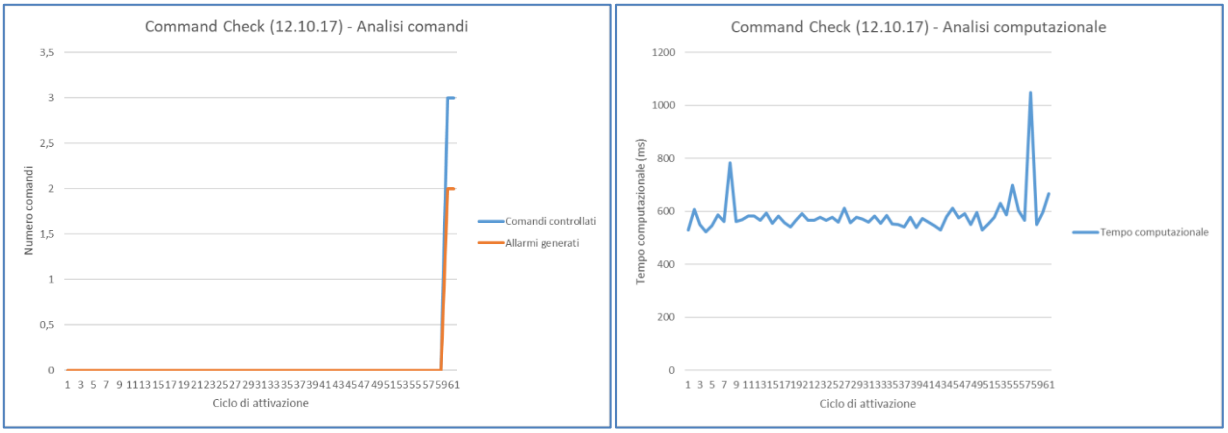

Figura 24 - Analisi dei comandi e del tempo computazionale (4.10.17)

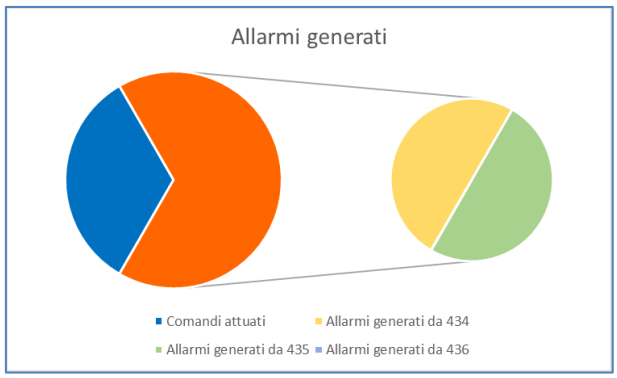

Figura 25 - Analisi degli allarmi generati (12.10.17)

Risultati della simulazione:

- Totale comandi controllati = 6
- Comandi attuati = 2
- Totale allarmi generati = 4
- Allarmi generati da 434 = 2
- Allarmi generati da 435 = 2
- Allarmi generati da 436 = 0

3.2 Analisi dei risultati

Come riportato nel precedente capitolo, si è deciso di aumentare il tempo che intercorre tra due attivazioni consecutive della procedura di fault detection passando da sessanta secondi a dieci minuti.
E' stato adottato questo accorgimento con il duplice obiettivo di diminuire le risorse computazionali impiegate, e di uniformarsi il più possibile al ciclo di funzionamento del supervisore elettrico che opera nell'edificio F40. Quest'ultimo, infatti, prevede che sia i comandi che le acquisizioni avvengono circa ogni dieci minuti.
Osservando l'analisi generale e giornaliera dei comandi si evidenzia che periodicamente la procedura *Command Check* effettua il controllo di sei comandi, invece dei tre previsti.
Questo comportamento anomalo è dovuto alla asincronicità che caratterizza l'applicazione rispetto al ciclo di funzionamento del supervisore. Ogni giorno di funzionamento, infatti, la procedura di fault detection incontrerà un ciclo di attivazione durante il quale dovranno essere controllati sei comandi. In queste situazioni è possibile che siano erroneamente generati degli allarmi dato che i primi tre comandi non verranno opportunamente processati.
E' proprio in base alla natura dei comandi si potranno avere due situazioni:

- Se i comandi effettuati sullo stesso dispositivo sono analoghi, entrambi di accensione o di spegnimento, la procedura non genera allarmi;
- In caso contrario, dovendo processare contemporaneamente due comandi discordanti sullo stesso dispositivo, la procedura produrrà dei falsi positivi.

Per poter ovviare a questo problema si propone una differente logica di attivazione della procedura. Si può infatti definire un segnale di *trigger* in grado di riconoscere la richiesta di attuazione da parte del supervisore ed avviare di conseguenza l'algoritmo di controllo.
In alternativa può essere dimezzato il tempo che intercorre tra due attivazioni consecutive della procedura portandolo da dieci a cinque minuti, apportando in questo modo modifiche minime nel codice implementato.
Dall'analisi dei comandi si osserva inoltre un gran numero di allarmi generati, ed un andamento della procedura che si ripete pressoché identico per i giorni che vanno dal 7 all'11 ottobre. Questo pattern comportamentale, inizialmente imputato ad un tuning scorretto dei parametri caratteristici dell'applicazione, è stato poi giustificato e ricondotto agli interventi tecnici e di manutenzione del sistema di supervisione effettuati proprio in quei giorni.
Dal punto di vista computazionale sono emersi dei picchi temporali che evidentemente non dipendono dal numero di comandi controllati, ma sono riconducibili ai tempi di accesso al database *smarttowndb*. Si osserva infatti che i picchi si manifestano in corrispondenza di particolari momenti in cui il traffico intenso nei canali di comunicazione verso e da i database condiziona il tempo necessario alla procedura per espletare la sua funzione.
E' da sottolineare che su oltre 1100 cicli di attivazione dell'applicazione realizzata solo in 16 casi il tempo computazionale ha superato la soglia di un secondo, stabilizzandosi comunque ad un tempo computazionale medio di circa 596 millisecondi.

4. Rilevazione automatica di anomalie in un sistema di climatizzazione elettrica in ambito Smart Building

Questa progettualità consiste nella definizione e nello sviluppo di una procedura informatica in grado di segnalare in modo automatico l'eventuale mancata attuazione dei comandi, impartiti dal supervisore attivo nell'edificio F40, riguardanti la climatizzazione elettrica.
Più nel dettaglio, si vuole verificare il corretto funzionamento dei comandi di attuazione verso i dispositivi fancoil dislocati in ogni stanza dell'edificio.

4.1 Definizione della procedura di controllo

Analogamente a quanto sviluppato durante la precedente annualità (RdS/2016/xxx), la strategia di controllo a risposta rapida consiste nel verificare la corretta comunicazione tra i dispositivi interessati dal processo ed il conseguente funzionamento del fancoil nei periodi di tempo in cui è richiesto il suo intervento per la regolazione della temperatura.
Considerando il percorso logico riportato nella figura che segue, si procede con un controllo dell'attuazione a tre livelli distinti che permette di individuare a che punto del processo si evidenzia l'anomalia, e quindi intervenire in modo puntuale sul malfunzionamento.

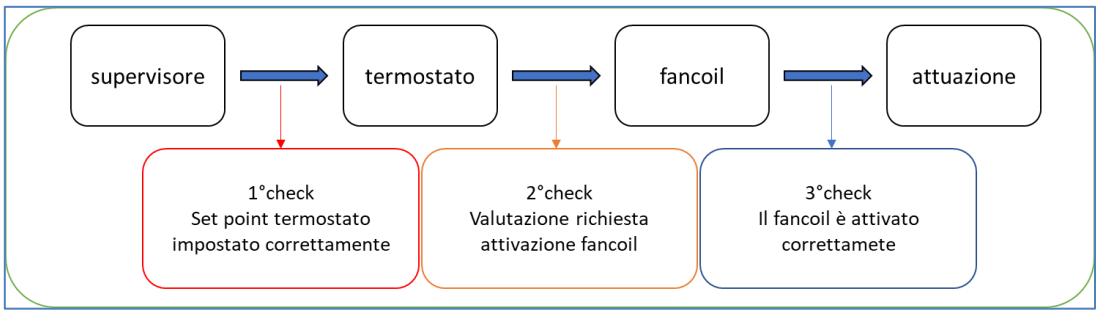

Figura 26 - Logica di controllo implementata su tre livelli

Operativamente l'obiettivo è di controllare la corretta attuazione dei comandi della tipologia:
Set_point_fancoil_stanza (SpF), Comando di set point per il termostato del fancoil.
In base all'ambiente in cui opera il fancoil si considera il comportamento delle seguenti grandezze associate:

- *Temperatura_stanza* (Ts), Misura di temperatura della stanza;
- *Set_point_termostato_stanza* (SpT), Misura impostata dal termostato associato al fancoil;
- *Status_fancoil_stanza* (Rf), Attivazione del fancoil della stanza;

In particolare, la logica implementata avrà l'obiettivo di verificare la corretta comunicazione tra i dispositivi, e di operare un confronto tra la temperatura effettiva (Ts)e quella impostata dal comando (SpF). Nel caso in cui queste non corrispondono, ci si aspetta che nel giro di pochi secondi si attivi il relè di alimentazione (Rf), del dispositivo fancoil associato al comando.
Verificando questo comportamento sarà verificata l'effettiva attuazione del comando. Al contrario, nel caso in cui il relè non risulta attivato, verrà segnalata la mancata attuazione mediante appositi allarmi.

E' possibile formalizzare come segue:

1. Si acquisiscono le informazioni riguardanti il comando impartito (SpF) – tipologia di comando, grandezze di riferimento (Ts, SpT, Rf);
2. Si acquisiscono le informazioni riguardanti le grandezze di riferimento associate – tempo necessario alla stabilizzazione;
3. Si attende il tempo necessario alla stabilizzazione delle grandezze e se ne acquisisce lo stato;

4. Si definisce l'esito del comando valutando lo stato delle grandezze ad esso associate;
5. Si segnala l'eventuale presenza di un'anomalia.

Schematizzando la regola di valutazione per il controllo si ha:

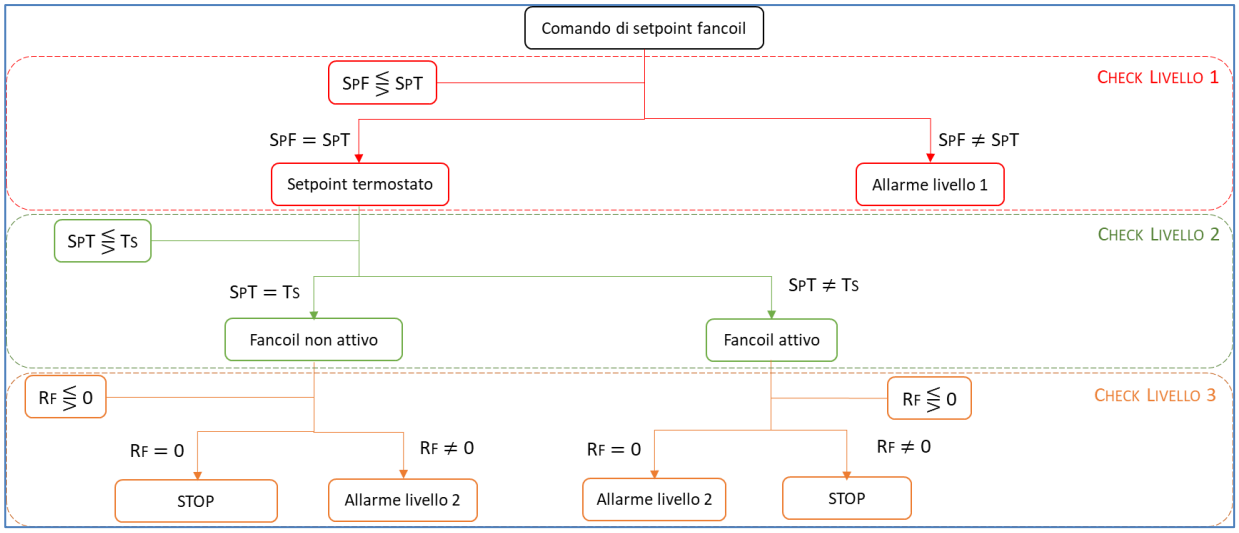

Figura 27 - Schema concettuale procedura di Fault Detection

Anche in questo caso è stato sfruttato l'utilizzo di un componente Java molto importante, ossia il thread, grazie al quale è possibile creare applicazioni in grado di garantire la condivisione in parallelo dello spazio di memoria tra i vari processi. L'altro fondamentale strumento utilizzato è MySQL che consente di creare connessioni con il database tramite l'indirizzo IP e di eseguire i comandi per la manipolazione dei dati. Tramite l'installazione di librerie dedicate è possibile 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.

Di seguito si riporta la procedura per l'accesso ai dati utilizzata dall'applicazione.

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.

4.2 Fase di implementazione della procedura

Nel caso applicativo considerato si è deciso di prendere in esame i comandi di impostazione del set point di temperatura ai dispositivi di climatizzazione elettrica presenti nelle stanze dell'edificio F40. Le grandezze associabili ai comandi coincidono con le misure registrate dai sensori integrati nei dispositivi interessati dal processo di climatizzazione. In particolare, si fa riferimento alle seguenti corrispondenze tra comando impartito e misura associata:

Tabella 2 - Corrispondenza tra comando e grandezza associata

ID Comando	Descrizione Comando		Misura Associata	ID Misura Associata
373	Set_point_fancoil_105		Temperatura_105	119
			Set_point_termostato_105	400
			Rele_fancoil_105	500
393	Set_point_fancoil_106		Temperatura_106	139
			Set_point_termostato_106	401
			Rele_fancoil_106	501
372	Set_point_fancoil_107		Temperatura_107	118
			Set_point_termostato_107	402
			Rele_fancoil_107	502

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.

La gestione e l'interazione con la base di dati è affidata a MySQL che costituisce una soluzione flessibile e affidabile per la gestione degli accessi e eseguire le interrogazioni sul database SmartTown.
Nella figura sottostante è riportato il diagramma EER in cui sono rappresentate le correlazioni tra le diverse tabelle interessate dall'attività di rilevamento delle anomalie.

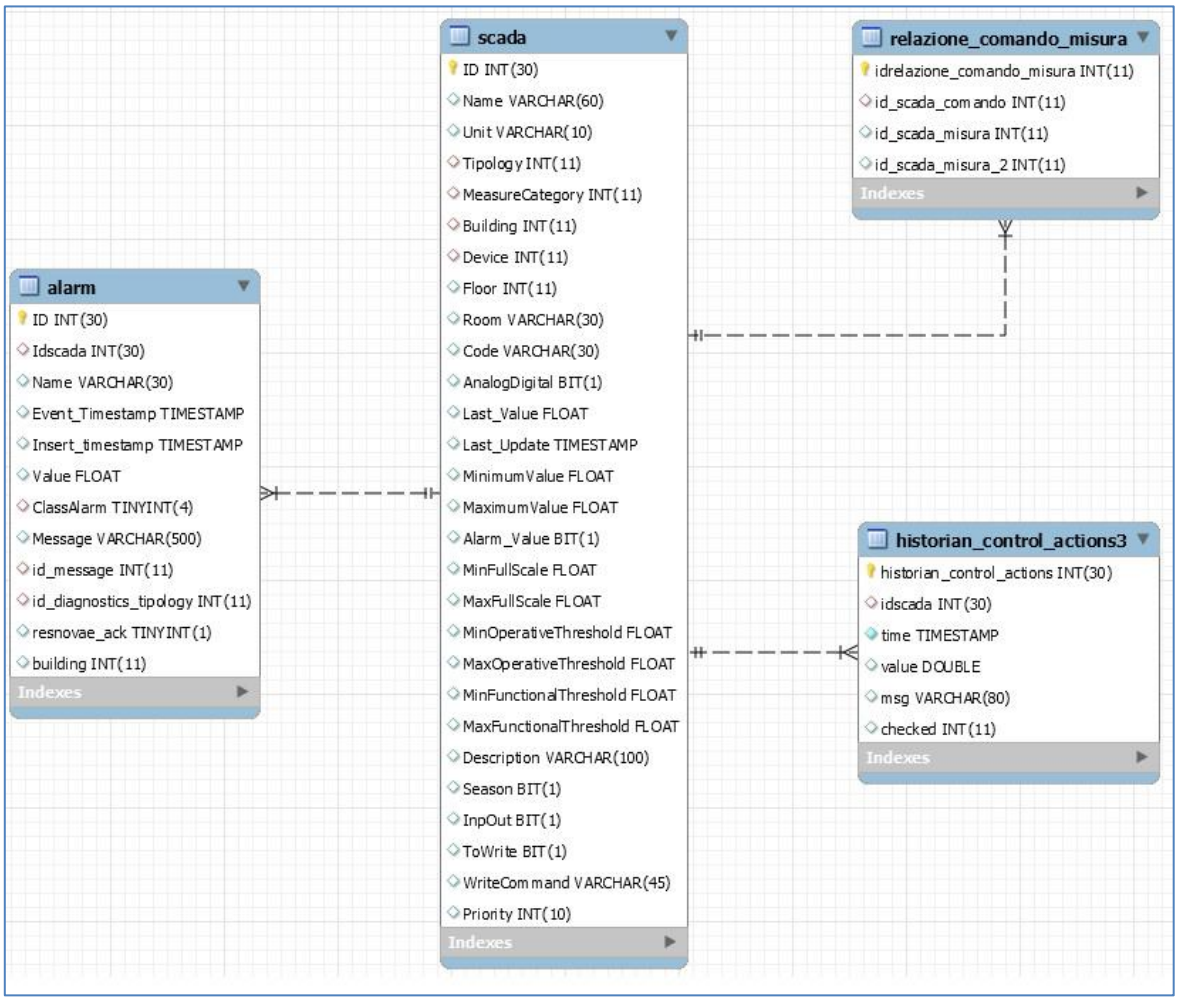

Figura 28 - Diagramma EER delle tabelle di interesse

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 il setpoin impostato dal supervisore;
    - *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. Nel caso in cui sia riferito ad un comando ne indica la tipologia (accensione/spegnimento, setpoint);
    - *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_1*, codice identificativo del dispositivo scada da interrogare per ottenere la misura della prima grandezza associata;
    - *id_scada_misura_2*, codice identificativo del dispositivo scada da interrogare per ottenere la misura della seconda grandezza associata;
    - *id_scada_misura_3*, codice identificativo del dispositivo scada da interrogare per ottenere la misura della terza grandezza associata;

Tramite le interrogazioni, o query, e i comandi, implementati utilizzando il linguaggio SQL (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 utilizzando 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.

4.3 La Procedura di controllo

Si può riassumere la procedura come segue.

Input:

- database "*smarttowndb*";
- tempo di attesa *delay = 10sec*;
- tempo riattivazione del processo *tempoCiclo = 5min*,
- tempo da sottrarre al timestamp *olderTime = 6min*.

Procedura:

1. A seguito della creazione di un oggetto *ThermalCommandCheck,* 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 il valore di set point.
    - Dalla tabella *relazione_comando_misura* si ricavano le grandezze di riferimento associate ad ogni comando identificato al passo precedente, tramite l'*idscada*, e si mantiene in memoria l'indice *id_scada_misura_n*.
    - Dalla tabella *scada* si ricava il valore assunto dalle grandezze associate ad ogni comando, tramite l'indice *id_scada_misura_n*.
3. Si effettuano le seguenti valutazioni a cascata:
    - *Set_point_fancoil* $\lessgtr$ *Set_point_termostato*;
    - *Set_point_termostato* $\lessgtr$ *Temperatura*;
    - *Status_fancoil_stanza* $\lessgtr$ 0.

Output:

- Si tiene traccia dell'avvenuto controllo sul comando, aggiornando il campo *checked* della tabella *historian_control_actions3* che corrisponde al record del comando.
- 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.

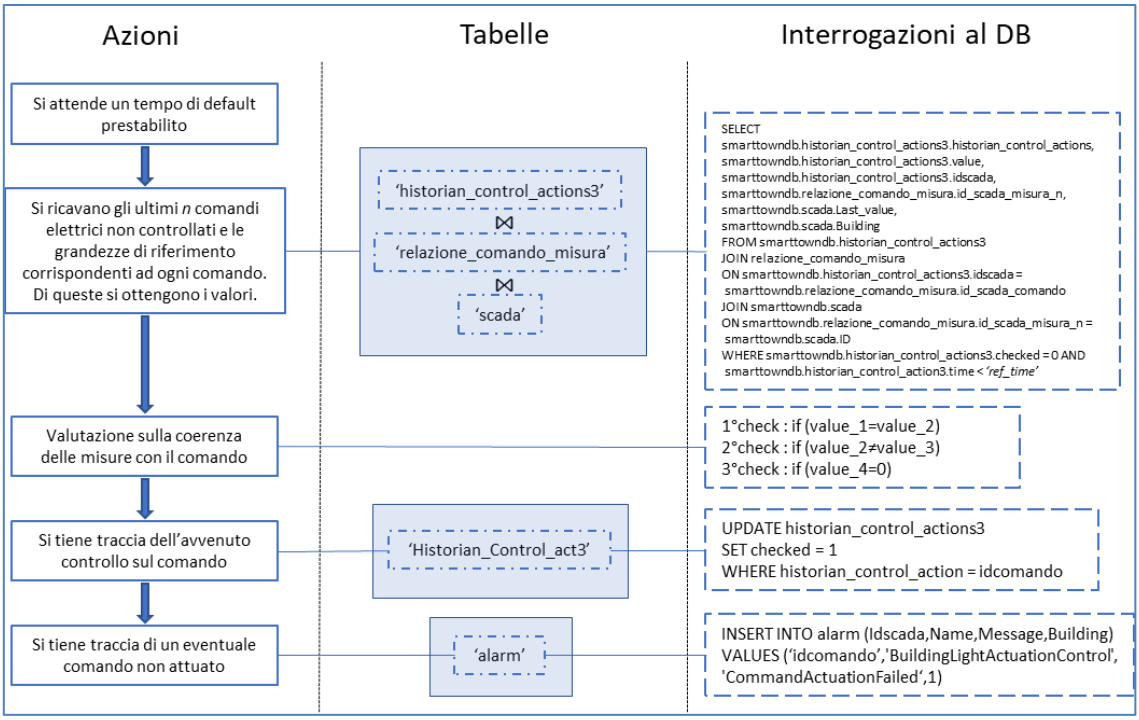

Figura 29 - Schema logico della procedura di Fault Detection

4.4 Dettagli di implementazione – La classe ThermalCommandCheck

Per rendere la procedura di fault detection portabile e riusabile è stato creato un file *config.properties* contenente i parametri di connessione al database, e altre costanti utilizzate nel processo, per evitare la loro configurazione all'interno del codice 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;
  - olderTime, millisecondi da sottrarre al timestamp corrente per ottenere un riferimento temporale.
- Informazioni per la creazione del file di log.

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. Le informazioni mantenute nel *file.log* 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.

Sono quindi previsti due costruttori che svolgono le operazioni di inizializzazione del processo:

- `ThermalCommandCheck() {…}`
  Le informazioni necessarie per stabilire la connessione al DB e per inizializzare il processo sono inserite di default, già presenti all'interno del costruttore.
- `ThermalCommandCheck(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.

I metodi implementati all'interno della classe 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.
- `String getOldTime()`
  Il metodo restituisce il timestamp corrente sottratto di una quantità che può essere definita dall'utilizzatore, elemento necessario per un'efficiente 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.

Di seguito si riporta la descrizione dei comandi e delle interrogazioni in linguaggio SQL definite all'interno del codice:

- Query di aggiornamento

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

Figura 30 - Query di aggiornamento

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)
```

Figura 31 - Query di inserimento

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 raccolta dati

Attraverso questa SELECT, ripetuta n volte (n = numero di grandezze associate al comando analizzato) 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.

```
SELECT smarttowndb.historian_control_actions3.historian_control_actions,
       smarttowndb.historian_control_actions3.value,
       smarttowndb.historian_control_actions3.idscada,
       smarttowndb.relazione_comando_misura.id_scada_misura_n,
       smarttowndb.scada.Last_Value,smarttowndb.scada.Room
FROM smarttowndb.historian_control_actions3
     JOIN smarttowndb.relazione_comando_misura
          ON smarttowndb.historian_control_actions3.idscada = smarttowndb.relazione_comando_misura.id_scada_comando
     JOIN smarttowndb.scada
          ON smarttowndb.relazione_comando_misura.id_scada_misura_n = smarttowndb.scada.ID
WHERE smarttowndb.historian_control_actions3.checked = 0
      AND smarttowndb.scadaTab.Tipology = 4
      AND smarttowndb.historian_control_actions3.time < '"+currentTime+"'
      AND smarttowndb.historian_control_actions3.time > '"+oldTime+"'
ORDER BY smarttowndb.historian_control_actions3.historian_control_actions DESC
```

Figura 32 - Query di valutazione

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_1	Last_Value	id_scada_misura_2	Last_Value	id_scada_misura_3	Last_Value	Room
885	21	373	500	0	119	21	400	21	105
884	21	372	502	1	118	21	402	18	107

Figura 33 - Esempio risultato query di raccolta dati

4.5 Esempio applicativo

Per rendere chiaro il procedimento eseguito dall'algoritmo si rappresenta di seguito la sua applicazione, passo per passo. Si prenderanno in considerazione, contemporaneamente due comandi inviati dal sistema di supervisione energetica:

- Comando di setpoint a 21 gradi per il fancoil della stanza 106.
- Comando di setpoint a 21 gradi per il fancoil della stanza 105.
- Comando di setpoint a 21 gradi per il fancoil della stanza 107.

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
886	393	21	0	2018-07-20 14:25:15
885	373	21	0	2018-07-20 14:25:15
884	372	21	0	2018-07-20 14:25:15

Figura 34 - Snapshot tabella historian_control_actions3 prima del ciclo di controllo

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

Figura 35 - Snapshot tabella alarm prima del ciclo di controllo

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

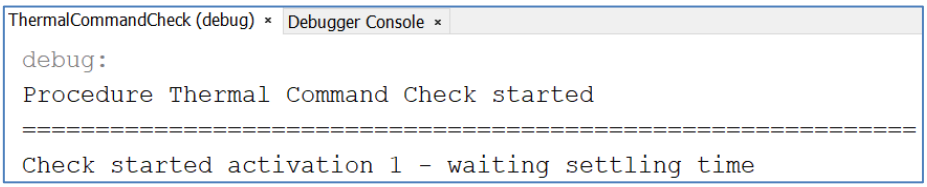

Figura 36 - Snapshot Terminal primo ciclo di controllo

3. Terminata l'attesa, viene interrogato il database effettuando la query di raccolta dati che restituisce le tabelle con le informazioni di interesse per ogni comando

historian_control_actions	value	idscada	id_scada_misura_1	Last_Value	id_scada_misura_2	Last_Value	id_scada_misura_3	Last_Value	Room
886	21	393	501	1	139	21	401	21	106
885	21	373	500	0	119	21	400	21	105
884	21	372	502	1	118	21	402	18	107

Figura 37 - Tabella di raccolta dati durante il primo ciclo di controllo

4. Avviene la valutazione dei comandi impartiti dal supervisore:

- Primo comando – stanza 105
  - Si controlla se il setpoint comandato corrisponde con quello impostato sul termostato, ossia l'ultimo valore della terza grandezza associata.
    Si ha 21 = 21.
  - Si controlla se è richiesta l'accensione del fancoil confrontando il valore impostato sul termostato con la temperatura della stanza, ossia l'ultimo valore della seconda grandezza associata.
    Si ha 21 = 21, non è richiesta l'attivazione del fancoil
  - Si controlla se il fancoil sia erroneamente attivato valutando lo stato del suo relè, ossia l'ultimo valore della prima grandezza associata.
    Si ha relè = 1, il fancoil è attivato.

- Secondo comando – stanza 106
  - Si controlla se il setpoint comandato corrisponde con quello impostato sul termostato, ossia l'ultimo valore della terza grandezza associata.
    Si ha 21 = 21.
  - Si controlla se è richiesta l'accensione del fancoil confrontando il valore impostato sul termostato con la temperatura della stanza, ossia l'ultimo valore della seconda grandezza associata.
    Si ha 21 = 21, non è richiesta l'attivazione del fancoil
  - Si controlla se il fancoil sia erroneamente attivato valutando lo stato del suo relè, ossia l'ultimo valore della prima grandezza associata.
    Si ha relè = 0, il fancoil non è attivato.

- Terzo comando – stanza 107
  - Si controlla se il setpoint comandato corrisponde con quello impostato sul termostato, ossia l'ultimo valore della terza grandezza associata.
    Si ha 21 = 19. Il comando non corrisponde.

Dal terminale si può verificare l'avanzamento della procedura.

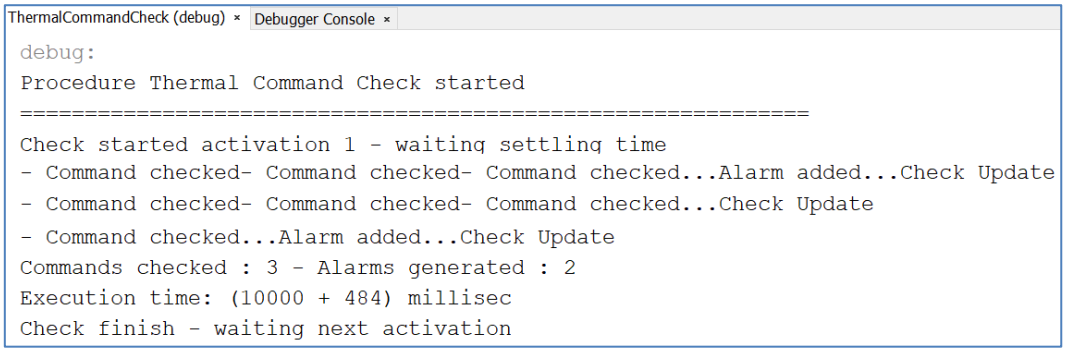

Figura 38 - Snapshot del terminale al termine del ciclo di controllo

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

historian_control_actions	idscada	value	checked	time
886	393	21	1	2018-07-20 14:25:15
885	373	21	1	2018-07-20 14:25:15
884	372	21	1	2018-07-20 14:25:15

Figura 39 - Snapshot tabella historian_control_actions3 al termine del ciclo di controllo

6. Si inseriscono nella tabella "alarm" le anomalie riscontrata nell'attuazione del secondo comando, aggiungendo come informazione il punto del processo in cui è stato rilevato il malfunzionamento.

ID	Idscada	Name	Message	building	insert_timestamp	id_control_action	room
571	372	BuildingClimaticActuationCon...	Supervisor to thermostat communication error	1	2018-07-20 14:25:25	884	107
570	393	BuildingClimaticActuationCon...	Thermostat to fancoil communication error	1	2018-07-20 14:25:25	886	106

Figura 40 - Snapshot tabella alarm dopo il primo ciclo di controllo

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

5. Rilevazione automatica di comandi malevoli

Si vuole definire una procedura in grado di analizzare lo scambio di informazioni tra il supervisore ed un attuatore ad esso associato al fine di individuare malfunzionamenti nelle attuazioni. Anomalie che sono riconducibili a probabili intrusioni informatiche. Tramite la realizzazione di algoritmi matematici per l'*anomaly detection* sarà quindi possibile riconoscere flussi sospetti o comandi malevoli e segnalare eventuali errori non legati a malfunzionamenti del supervisore.
La strategia consiste nell'attivare ciclicamente un applicativo in grado di acquisire e memorizzare in tempo reale le informazioni, in termini di parametri di funzionamento, di ogni dispositivo elettrico destinato all'illuminazione elettrica. Una volta definito lo stato del dispositivo, accenso/spento, si valuta se questo è concorde con l'ultimo comando impartito dal supervisore su quel particolare dispositivo.

5.1 Identificazione e definizione della procedura

Durante questa attività è stato studiato in modo approfondito il contesto di applicazione e gli strumenti a disposizione. Nel processo di realizzazione della strategia di controllo 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 attuazione:

1. Dispositivo elettrico associato – illuminazione;
2. Localizzazione del dispositivo – edificio, piano;
3. Tipologia dell'ultimo comando – accensione, spegnimento;

Ad ogni dispositivo è associata una grandezza di riferimento di cui è facilmente deducibile lo stato con l'obiettivo di associare ad ogni attuazione l'andamento di una data grandezza di riferimento.
Monitorando lo stato della grandezza e confrontandolo con valori soglia predefiniti è possibile stabilire lo stato del dispositivo di attuazione.
In pratica, lo stato (on/off) 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 agli attuatori riguardanti l'illuminazione generale di un piano dell'edificio coincidono con le misure registrate dal quadro elettrico generale che riporta il consumo relativo all'illuminazione di ogni piano.

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 degli ultimi 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' quindi possibile formalizzare la strategia come segue:

1. Si acquisiscono le informazioni riguardanti il particolare attuatore – comando associato, grandezza di riferimento;
2. Si acquisiscono le informazioni riguardanti la grandezza di riferimento associata – tempo necessario alla stabilizzazione, valori soglia predefiniti;
3. Si confronta lo stato della grandezza con i valori soglia, e si definisce lo stato dell'attuatore;
4. Si attende il tempo necessario alla stabilizzazione della grandezza e si determina la tipologia di comando associata all'attuatore;
5. Si confronta lo stato dell'attuatore con il comando associato, e si definisce l'esito del controllo;
6. Si segnala l'eventuale presenza di un'anomalia.

Analogamente a quanto fatto nelle precedenti attività di *anomaly detection*, per implementare questo tipo di controllo è stato sfruttato l'utilizzo di un componente Java molto importante, ossia il thread, che consente di creare applicazioni garantendo la condivisione in parallelo dello spazio di memoria tra i vari processi.

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 della coerenza tra stato dell'attuatore e comando;
6. 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:

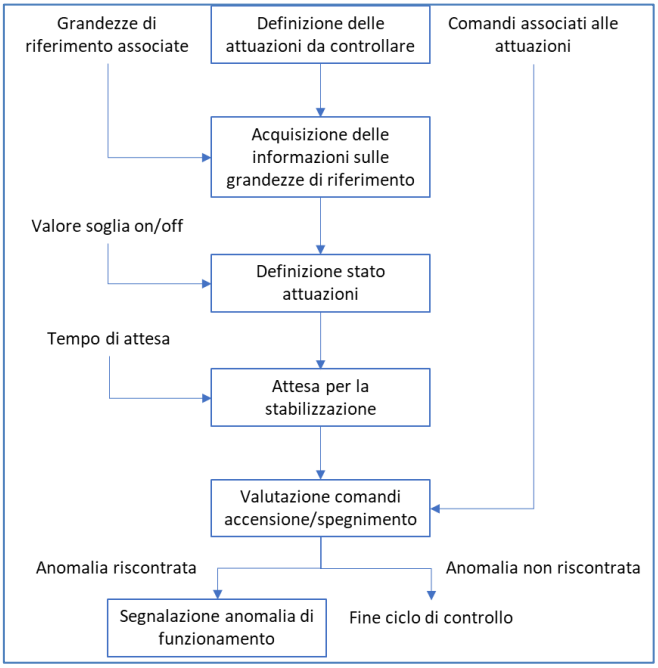

Figura 41 - Schema concettuale della procedura di Malicious Command Detection

5.2 Implementazione della procedura di Malicious Command 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 gli attuatori di illuminazione generale di ognuno dei tre piani che compongono l'edificio F40. Le grandezze associabili alle attuazioni coincidono con le misure registrate dal quadro elettrico generale.

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

Tabella 3 - Corrispondenze comando/misura

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

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.

Di seguito si riporta il diagramma EER in cui sono rappresentate le correlazioni tra le diverse tabelle interessate dall'attività di rilevamento delle anomalie.

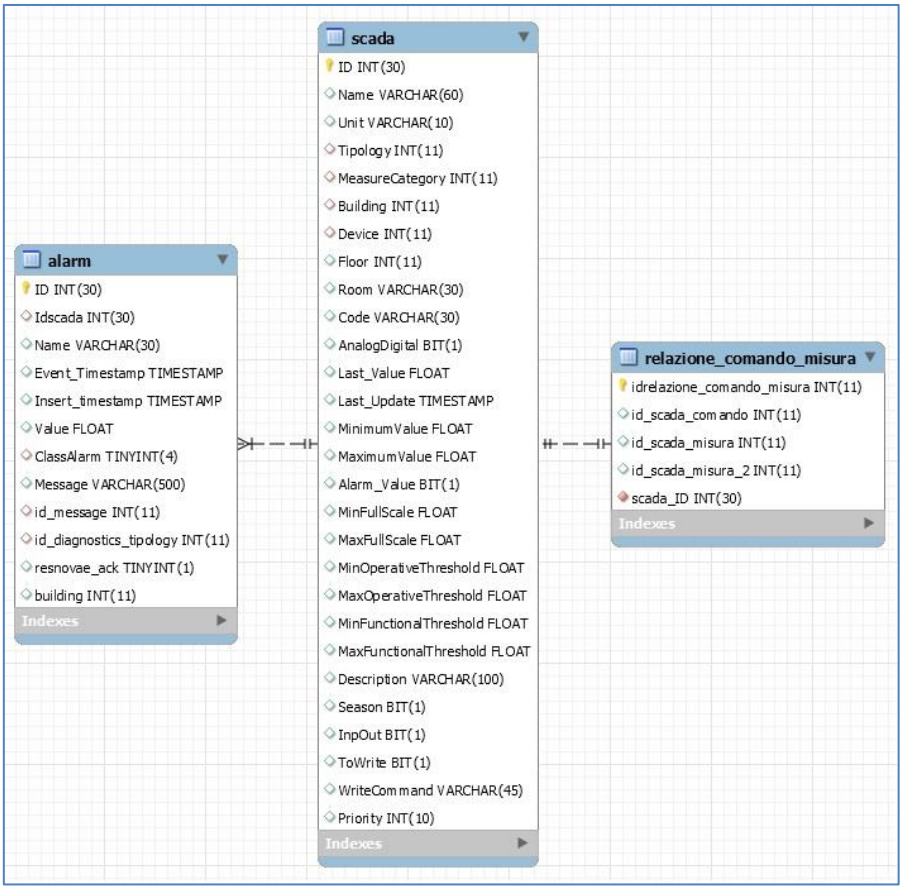

Figura 42 - Diagramma EER delle tabelle di interesse

Come nella progettualità descritta nel precedente capitolo, le tabelle e dei campi utilizzati durante la procedura sono:

- *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 sia riferito ad un comando indica la tipologia e può assumere due valori: 0 in caso di spegnimento, 1 in caso di accensione. 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;
    - *floor*, riporta il piano dell'edificio in cui è situato l'attuatore;
    - *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.

L'implementazione dell'algoritmo è stata realizzata tramite il linguaggio di programmazione orientato agli oggetti *Java*, e l'interazione con il database è avvenuta utilizzando il linguaggio SQL (Structure Query Language)

5.3 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:
  - *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 = 30min*.

Procedura:

1. A seguito della creazione di un oggetto *MaliciousCommandCheck,* si attende un tempo di default prestabilito (*delay*) che permetta la stabilizzazione delle grandezze elettriche di riferimento associate ai dispositivi di attuazione 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 *relazione_comando_misura* si ricava la grandezza di riferimento corrispondente ad ogni attuatore che si intende monitorare, 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 tramite l'indice *id_scada_misura*.
- Dalla tabella *relazione_comando_misura* si ricava il comando del supervisore associato alla particolare attuazione, tramite l'*idscada*, e si mantiene in memoria l'indice *id_scada_comando*.
- Dalla tabella *scada* si ricava il valore assunto dalla grandezza di riferimento, per ogni comando, tramite l'indice *id_scada_misura*.

3. Si confronta il valore assunto dalle grandezze di riferimento con il valore soglia prestabilito *ref_value*, per definire lo stato dell'attuatore.
4. Tenendo in considerazione la natura del comando (accensione/spegnimento) osservata tramite il campo *value*, si confronta lo stato dell'attuatore per verificare l'avvenuta attuazione del comando stesso.

Output:

- Si tiene traccia di un eventuale anomalia riscontrata aggiungendo un record alla tabella *alarm* con le informazioni del comando esaminato.

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

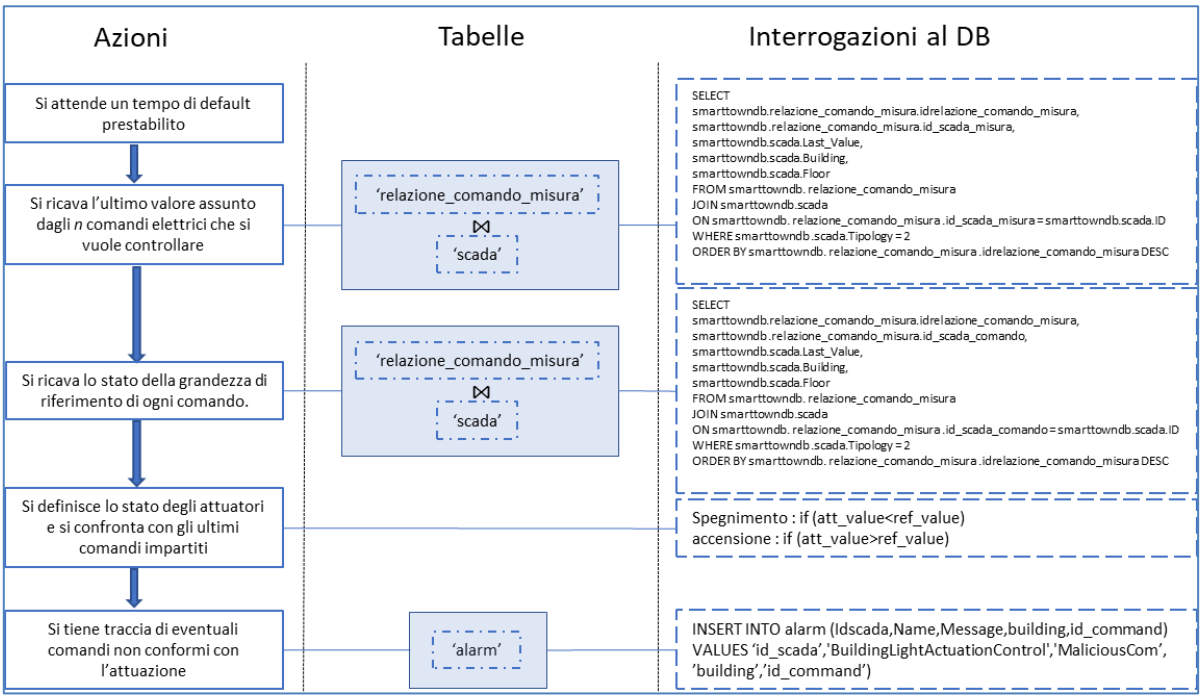

Figura 43 - Schema logico della procedura di Fault Detection

5.4 Dettagli di implementazione – La classe MaliciousElectricCommandCheck

Il comportamento della strategia 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 creato un file config.properties, contenente i parametri di connessione al database, e le 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
  - 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.

Il processo di logging creato è in grado di tenere traccia di tutti gli eventi di interesse riscontrati durante l'esecuzione della procedura.

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

- `MaliciousElectricCommandCheck() {…}`
  Le informazioni necessarie per stabilire la connessione al DB e per inizializzare il processo sono inserite di default, già presenti all'interno del costruttore nel caso in cui non sia disponibile il file di configurazione.
- `MaliciousElectricCommandCheck(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.

I metodi implementati all'interno della classe definiscono il comportamento dell'oggetto MaliciousElectricCommandCheck 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 lo stato del particolare attuatore confrontando il valore assunto dalla grandezza (*att_value*) con un valore di riferimento prestabilito, distinguendo la tipologia del comando (*type*) di accensione o spegnimento.
- `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.
- `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.

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. Sono state implementate tre interrogazioni:

- Query di inserimento

```
INSERT INTO serverDBname.alarmTab (Idscada,Name,Message,building,id_command)
VALUES (id_scada,'BuildingLightActuationControl','MaliciousCommandIdentified',building,id_com);
```

Figura 44 - Query di inserimento

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

```
SELECT smarttowndb.relazione_comando_misura.idrelazione_comando_misura,
smarttowndb .relazione_comando_misura.id_scada_misura,
smarttowndb.scada.Last_Value,
smarttowndb.scada.Building,
smarttowndb.scada.Floor
FROM smarttowndb. relazione_comando_misura
JOIN smarttowndb.scada
ON smarttowndb. relazione_comando_misura .id_scada_misura = smarttowndb.scada.ID
WHERE smarttowndb .scada.Tipology = 2
ORDER BY smarttowndb. relazione_comando_misura .idrelazione_comando_misura DESC
```

Figura 45 – Prima query di valutazione

```
SELECT
smarttowndb.relazione_comando_misura.idrelazione_comando_misura,
smarttowndb .relazione_comando_misura.id_scada_comando,
smarttowndb.scada.Last_Value,
smarttowndb.scada.Building,
smarttowndb.scada.Floor
FROM smarttowndb. relazione_comando_misura
JOIN smarttowndb.scada
ON smarttowndb. relazione_comando_misura .id_scada_comando = smarttowndb.scada.ID
WHERE smarttowndb .scada.Tipology = 2
ORDER BY smarttowndb. relazione_comando_misura .idrelazione_comando_misura DESC
```

Figura 46 - Seconda query di valutazione

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

Per ognuno degli ultimi n comandi che si desidera controllare si hanno:

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

5.5 Esempio applicativo

Si rappresenta di seguito il procedimento eseguito dall'algoritmo, passo per passo, in un caso esemplificativo, ma plausibilmente molto aderente alla realtà.
Si prenderanno in considerazione, contemporaneamente tre attuatori da voler controllare:

- Gruppo luci per l'illuminazione del corridoio a piano terra;
- Gruppo luci per l'illuminazione del corridoio al primo piano;
- Gruppo luci per l'illuminazione del corridoio al secondo piano.

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:

ID	Name	Building	Floor	Last_Value	Last_Update	Tipology	Room
156	Potenza Attiva Tot - Linea luce OPT	1	0	0.5	2017-07-13 14:11:10	2	NULL
169	Potenza Attiva Tot - Linea Luce OP1	1	1	10.5	2017-07-13 14:11:10	2	NULL
270	Potenza Attiva Tot - Linea Luce OP2	1	2	10.5	2017-07-13 14:11:10	2	NULL
434	Comando Luci Corridoio da Supervisore PT	1	0	1	2017-07-13 14:13:19	2	NULL
435	Comando Luci Corridoio da Supervisore 1P	1	1	1	2017-07-13 14:13:19	2	NULL
436	Comando Luci Corridoio da Supervisore 2P	1	2	1	2017-07-13 14:13:19	2	NULL

Figura 47 - Snapshot tabella scada

idrelazione_comando_misura	id_scada_comando	id_scada_misura	descrizione	waiting_sec
2	434	156	luci piano 0	10
3	435	169	luci piano 1	10
4	436	270	luci piano 2	10

Figura 48 - Snapshot tabella relazione_comando_misura

ID	Idscada	Name	Message	building	insert_timestamp	id_command
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 49 - Snapshot tabella alarm prima del ciclo di controllo

La tabella *scada* contiene le informazioni riguardanti le tre attuazioni analizzate, i dispositivi di illuminazione dei tre piani, e i tre comandi associati all'illuminazione dei tre piani.
Dalla tabella *relazione_comando_misura* si ricavano le corrispondenze tra i comandi e le grandezze associate alle attuazioni da monitorare.
Nella tabella *alarm* non vi è ovviamente traccia di allarmi riconducibili a comandi malevoli.

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

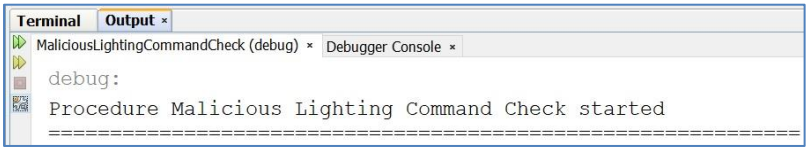

Figura 50 - Snapshot Terminal ciclo di controllo

3. Terminata l'attesa, viene interrogato il database effettuando le query di valutazione tramite le quali si ottengono due nuove tabella contenenti le informazioni di interesse per ogni comando.

idrelazione_comando_misura	id_scada_misura	Last_Value	id_scada_comando	Last_Value	Building	Floor
4	270	10.5	436	1	1	2
3	169	10.5	435	1	1	1
2	156	0.5	434	1	1	0

Figura 51 - Snapshot tabella risultato delle query di valutazione

4. Avviene la valutazione delle grandezze di riferimento per definire lo stato degli attuatori:
   - *Potenza Attiva Tot – Linea Luce 0PT < valore soglia*
     La grandezza associata è inferiore al valore soglia preimpostato. L'illuminazione del piano terra è disattiva, l'attuatore è in stato *off*.
   - Potenza Attiva Tot – Linea Luce 0P1 ≥ valore soglia
     La grandezza associata è superiore al valore soglia preimpostato. L'illuminazione del primo piano è attiva, l'attuatore è in stato *on*.
   - Potenza Attiva Tot – Linea Luce 0P2 ≥ valore soglia
     La grandezza associata è superiore al valore soglia preimpostato. L'illuminazione del secondo piano è attiva, l'attuatore è in stato *on*.
5. Avviene la valutazione sulla coerenza tra stato dell'attuatore e comando impartito dal supervisore:
   - Comando Luci Corridoio da Supervisore PT = 1

     Dal valore 1 osservabile nel campo "value" si nota che il comando è di accensione. Non vi è coerenza con lo stato dell'attuatore. Viene rilevato il comportamento anomalo.
   - Comando Luci Corridoio da Supervisore PT = 1

     Dal valore 1 osservabile nel campo "value" si nota che il comando è di accensione, in coerenza con lo stato dell'attuatore.
   - Comando Luci Corridoio da Supervisore PT = 1

     Dal valore 1 osservabile nel campo "value" si nota che il comando è di accensione, in coerenza con lo stato dell'attuatore.

   Dal terminale si può verificare l'avanzamento della procedura.

Figura 52 - Snapshot del terminale al termine del ciclo di controllo

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

ID	Idscada	Name	Message	building	insert_timestamp	id_command
568	156	BuildingLightActuationControl	MaliciousCommandIdenti...	1	2018-09-08 16:26:36	434

Figura 53 - Snapshot tabella alarm dopo il ciclo di controllo

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

6. Conclusioni

Le attività descritte in questo documento hanno consistito nella definizione ed implementazione di strumenti informatici mirati a migliorare l'efficienza e la robustezza del sistema di controllo e supervisione attivo nell'edificio F40 sito nel C.R. Casaccia.
La prima progettualità presentata riguarda l'interfacciamento di due db ad ha avuto come scopo lo sviluppo di modulo di interfacciamento per il recupero dei dati di consumo elettrici, termici e di comfort dell'edificio F40, sito C.R. ENEA Casaccia, a livello di edificio, piano e singola stanza. Per assicurare la completa funzionalità del sistema e la continuità con la logica del supervisore pre-esistente è stato quindi opportuno integrare la nuova struttura di storage con quella utilizzata precedentemente dal vecchio- sistema di supervisione.
I dati sono quindi stati inviati, dopo essere stati prelevati ed elaborati nell'opportuno formato, da un DBApio dedicato su server Enea ad un database SmarttownDB su una piattaforma di elaborazione e diagnostica dei dati pre-esistente in un contesto più ampio di Smart Village.
La seconda progettualità ha avuto come obiettivo l'attivazione della strategia di controllo di attuazione dei comandi di illuminazione dei tre piani dell'edificio F40 implementata durante la precedente annualità (RdS/2016/xxx). L'analisi dei risultati ottenuti ha evidenziato il corretto funzionamento della strategia che, durante il suo funzionamento, ha fornito risposte puntuali e in brevissimo tempo riguardanti l'insorgenza di anomalie nel processo analizzato.
Analogamente a quanto fatto per l'illuminazione dei piani, nella terza progettualità si è deciso di realizzare un applicativo informatico che fosse in grado di segnalare l'eventuale presenza di anomalie di funzionamento nel processo di controllo dei fancoil dislocati nell'edificio F40. La strategia di fault detection che si è deciso di implementare permette, con un controllo a tre livelli, non solo di riconoscere e riportare l'insorgenza di un malfunzionamento, ma anche di individuare a che punto del processo (supervisore, termostato, fancoil) si è manifestato.
La quarta progettualità presentata riguarda l'implementazione di una strategia volta ad individuare comandi malevoli che potrebbero provenire da attacchi informatici rendendo incontrollabile la rete di sensori e attuatori operanti nell'edificio F40. L'obiettivo è quindi quello di analizzare il flusso di informazioni scambiate dai dispositivi interessati nel processo e segnalare tempestivamente l'individuazione di messaggi e attuazioni non coerenti con i comandi impartiti dal supervisore.
Le ultime due progettualità sono state testate su un database di ausilio costruito ad hoc e del tutto simile a smarttowndb attivo sul server ENEA. Anche in questi casi è stato riscontrato un comportamento degli algoritmi realizzati coerente con i risultati che ci si attendeva.
E'auspicabile che, in modo analogo a quanto fatto per la progettualità di controllo sviluppata durante la precedente collaborazione, gli applicativi vengano testati sull'edificio F40, per valutarne il comportamento in un caso reale. In conclusione, le funzionalità implementate risultano particolarmente significative per i processi di monitoraggio e controllo dei dispositivi elettrici dislocati all'interno di uno smart building.
Gli algoritmi di diagnostica dei dati risultano essere di ausilio per la rilevazione automatica dei guasti e delle anomalie. Il riconoscimento e la segnalazione in tempo reale dei comportamenti anomali e dei malfunzionamenti del supervisore rappresentano un valore aggiunto tramite il quale ottenere una gestione ottimale delle risorse energetiche impiegate per la sussistenza di uno smart building.