POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

A.a. 2022/2023

Progettazione e implementazione di un prodotto software d'interconnessione di macchinari in ottica Industry4.0: SmartEDGE4.0

Tesi svolta presso Orchestra s.r.l

Relatore:
Prof. Giovanni Malnati
Dott. Walter Ferrarese
Ing. Giancarlo Degli Esposti

Candidato:
Riccardo Di Dio

Indice

Indice delle figure

Capitolo 1

Introduzione

1.1 Presentazione

Le nuove fabbriche 4.0 stanno sostituendo le vecchie macchine, apportando nuove tecnologie sempre più integrate ai processi di produzione, nel più ampio scenario dell'era digitale. Lo scambio di dati e le informazioni ottenute dalla digitalizzazione dei processi in fabbrica sono la chiave per la vera digitalizzazione che permette l'integrazione tra le Operation Technologies (OT) e i sistemi ICT. Fino ad ora tale integrazione si è rilevata molto scarsa, soprattutto nell'ambito delle PMI sprovviste di personale adeguato al trattamento dei dati e la digitalizzazione dei processi produttivi. Connettere una macchina non è solo "attaccare una spina" e collegarla a un sistema con una porta digitale; e se una macchina è in grado di "parlare", può anche far capire come renderla migliore e come prevedere cosa accadrà sulla base di analisi specifiche attraverso algoritmi di intelligenza artificiale e machine learning.

La rivoluzione tecnologica e di business in atto attraverso la progressiva diffusione nelle imprese del paradigma Industria 4.0 investe sempre più in profondità anche la strumentazione di processo. Quest'ultima include tutti quegli strumenti di misurazione, regolazione, controllo indispensabili per una corretta gestione dei processi industriali e, in molte realtà aziendali costituisce anche un vasto patrimonio di

tecnologia legacy, stratificata in anni di attività e investimenti.

Figura 1.1: Industria 4.0 [1]

ORCHESTRA, attraverso lo SmartEDGE$^{4.0}$ installato a bordo macchina, prevede la trasmissione e l'elaborazione dei dati grezzi in informazioni pronte all'uso direttamente dove i dati vengono generati. Nascono così nuovi servizi di prossimità attraverso una nuova User Experience per gli operatori ed i manutentori, User Experience anche attraverso l'uso di Realtà Aumentata (AR), Realtà Mista (MR) e Realtà Estesa (ER). Ne trarrebbero beneficio tutti i processi aziendali che coinvolgono lo Shopfloor: la produzione, la manutenzione, il controllo qualità, la logistica. Gli operatori sul campo possono sfruttare interfacce personalizzate in funzione della tipologia di processo messe a disposizione dagli Edge devices che già comunicano con la macchina per leggere i dati.

1.2 Contesto aziendale

Il nome dell'azienda deriva dalla sua capacità di "ri-accordare i processi di produzione delle PMI manifatturiere sullo spartito dell'Industria 4.0", concetto chiave racchiuso nel marchio Retuner® della suite dei prodotti ORCHESTRA. L'azienda, nata nel 2016 per fornire sistemi di monitoraggio e controllo remoto di qualsiasi macchina e impianto nuovo o esistente, ha puntato fin dall'inizio su un'architettura definita Cloud Driven Edge Computing che permette di riconfigurare dinamicamente lo stack IIoT, aggiungendo intelligenza e servizi digitali ad ogni livello.

ORCHESTRA ha sviluppato e integrato nella propria offerta il dispositivo SmartEDGE$^{4.0}$, in grado di stabilire una comunicazione senza fili con il cloud aziendale, e trasmettere dati utilizzabili per vari servizi digitali. Quest'ultimo consente d'integrare tutti i dispositivi di campo in una piattaforma cloud-based; è costituito da una completa gamma di prodotti e servizi, ed è finalizzato a fornire componenti e soluzioni di facile installazione.

Qui, la possibilità di adottare, da parte degli operatori addetti all'assistenza e agli interventi tecnici sul campo, dispositivi mobili come tablet e smartphone in grado di connettersi alla SmartHINGE$^{4.0}$, e accedere immediatamente a tutti i dati relativi alla strumentazione d'impianto e ai dispositivi di campo. Tale componente software semplifica di molto la routine del workflow quotidiano, consente all'operatore un risparmio di tempo del 10-20% [11], che può essere impiegato per gestire altri problemi e compiti di particolare importanza.

Lavorare nelle condizioni ottimali aumenta anche la produttività e favorisce la generazione di idee per il miglioramento continuo. Le tecnologie daranno una spinta sempre maggiore al cambiamento del lavoro. IoT, impianti connessi e uso di algoritmi di elaborazione dati faranno sì che gli ingegneri e addetti ai processi non utilizzino il loro tempo nella raccolta dati ed elaborazione, ma possano concentrarsi sulla interpretazione delle informazioni usando la propria creatività con il fine di applicare in modo efficace il miglioramento continuo dei processi.

Sul versante della gestione dei processi produttivi, l'offerta di Retuner®è completata dal MiniMES$^{4.0}$, una Web App multilingua, accessibile da qualsiasi piattaforma per pianificare, tracciare e controllare la produzione in tempo reale. Collegato agli SmartEDGE$^{4.0}$ attraverso "cerniera intelligente" SmartHINGE$^{4.0}$ costituita da piattaforma IIoT di Orchestra installata nel cloud dei clienti, il MiniMES$^{4.0}$ diventa non solo un sistema intelligente per la gestione della produzione con funzionalità di autoapprendimento, ma anche un supporto attivo e proattivo verso gli operatori. MiniMES$^{4.0}$ può essere utilizzato da qualsiasi tipo di produzione sia discreta che di processo. Può essere installato sul server di fabbrica o su piattaforme IIoT di terze parti. Questa sua caratteristica ha aperto la strada a collaborazioni e integrazioni con piattaforme internazionali [6].

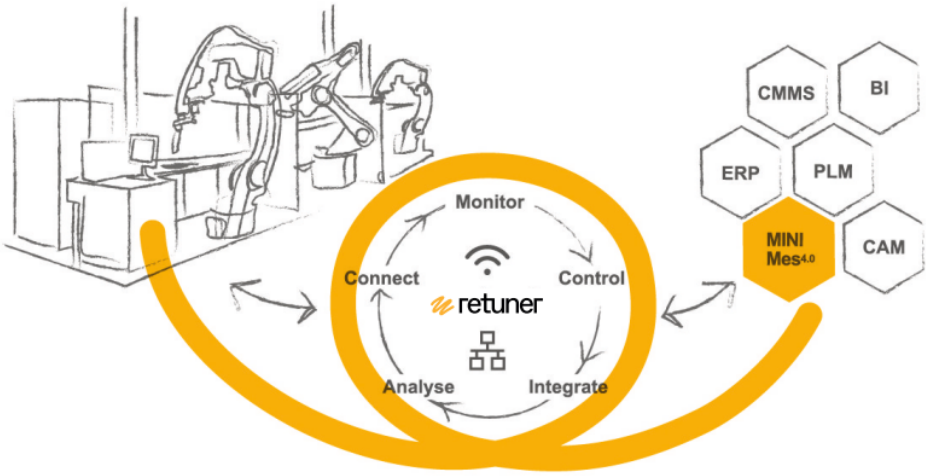


Figura 1.2: Retuner® suite [14]

1.3 Obiettivo

Accompagnare le imprese, soprattutto quelle di piccole e medie dimensioni, verso un percorso di innovazione e formazione, dando loro la possibilità di utilizzare una strumentazione sofisticata. La missione di ORCHESTRA è la transazione tecnologica, una possibilità attuabile attraverso la creazione di un solido software che fornisce l'interconnessione di sistemi informatici di fabbrica dando al cliente un'interfaccia tra uomo e macchina semplice e intuitiva. Approccio metodico graduale, con miglioramento incrementale, partendo dalla raccolta, visualizzazione, analisi e ottimizzazione dei dati per il singolo processo fino a coinvolgere l'intera supply chain [6]. La normativa vigente n. 57/L, allegato A (Articolo 1, comma 9) prevede l'ottenimento e il mantenimento dei seguenti requisiti in merito all'interconnessione di macchinari e impianti industriali:

> "Tutte le macchine utensili devono essere dotate delle seguenti caratteristiche:
>
> - Controllo per mezzo di CNC (Computer Numeric Control) e/o PLC (Programmable Logic Controller);
> - Interconnessione ai sistemi informatici di fabbrica con caricamento

da remoto di istruzioni e/o part program;

- Integrazione automatizzata con il sistema logistico della fabbrica o con la rete di fornitura e/o con altre macchine del ciclo produttivo;
- interfaccia tra uomo e macchina semplici e intuitive;
- rispondenza ai più recenti parametri di sicurezza, salute e igiene del lavoro."

Il processo di digitalizzazione degli impianti è iniziato introducendo un software integrato per il monitoraggio in tempo reale e da remoto degli impianti interconnessi. Come secondo step è stato invece avviato un progetto di manutenzione predittiva rintracciando dai dati raccolti gli allarmi generati dall'impianto. In questo modo la sostituzione di pezzi e interventi di manutenzione avverranno prima di trovarsi in emergenza, riducendo così i tempi di fermo macchina/impianto produttivo.

Riportando una citazione di Franco Perico in "Prevedere per essere più efficienti" [8], amministratore delegato di Automac:

> "Una programmazione predittiva del funzionamento delle macchine con relativa manutenzione rende più efficiente non solo la produzione, evitando interruzioni non previste, ma anche l'approvvigionamento dei materiali commerciali, come materie prime, motori e avvitatori elettrici, che richiedono tempo per il rifornimento e che, essendo costosi, diventerebbero costi "fissi" in magazzino in attesa di essere utilizzati."

Inoltre, sarà sviluppata una piattaforma IoT dotata di interfaccia HMI (Human-Machine Interface) per la segnalazione da remoto degli stati di funzionamento e dei guasti, con risparmi di tempo grazie alla tempestività nelle comunicazioni e alla manutenzione da remoto. È importante effettuare un'analisi accurata dei dati dell'impianto, dei guasti ricorrenti e la necessità di aggiungere o meno sensori additivi nei punti critici della macchina. L'analisi dei dati raccolti, precisa Francesco Braghin, professore del Politecnico di Milano, richiede tecniche software in grado di garantire la correttezza del dato rendendoli uniformi e utilizzabili per compiere previsioni attendibili nella logica di causa-effetto e quindi intervenire efficacemente e tempestivamente.

1.4 Risultato ottenuto

Lo SmartEDGE$^{4.0}$ Next Generation, capace di trasformare i dati grezzi in informazioni pronti all'uso già nel luogo in cui vengono prodotti, vale a dire bordo macchina, e di integrare tecnologie abilitanti come la realtà aumentata e l'advanced manufacturing, per proporre nuove user experience a tutti gli addetti di stabilimento. L'integrazione di algoritmi di intelligenza permette allo SmartEDGE$^{4.0}$ NG di supportare una nuova generazione di servizi digitali di prossimità in svariati campi di applicazione: manutenzione predittiva, on-condition, sicurezza degli operatori in prossimità, controllo qualità bordo macchina, processi manuali assistiti da robot collaborativi.

Figura 1.3: Elettronica SmartEDGE$^{4.0}$

Si tratta di servizi a valore aggiunto capaci di dare supporto attivo e in tempo reale agli operatori di produzione e ai manutentori degli asset produttivi, ma anche di offrire un valido aiuto nella formazione di nuovi operatori e nell'aggiornamento del personale già formato.

Capitolo 2

Stato dell'arte

2.1 Piano Nazionale di Ripresa e Resilienza

Il Piano Nazionale di Ripresa e Resilienza (PNRR) vale 221,1 miliardi di euro da impiegare nel periodo 2019-2026, di cui 191 miliardi di euro messi a disposizione sui fondi Next Generation EU. Le risorse vengono erogate per il 36% tramite sovvenzioni a fondo perduto e per il 64% tramite prestiti a tasso agevolato. Il 25% delle risorse è dedicato alla digitalizzazione, il 40% agli investimenti per il contrasto di cambiamento climatico è più del 10% alla coesione sociale.

Il PNRR prevede inoltre un ampio programma di riforme, ritenute necessarie per facilitare la sua attuazione e contribuire alla modernizzazione del Paese e all'attrazione degli investimenti. Il testo del PNRR si articola in sei missioni che rappresentano i pilastri dell'intero Piano:

1. Digitalizzazione, innovazione, competitività, cultura;
2. Rivoluzione verde e transizione ecologica;
3. Infrastruttura per una mobilità sostenibile;
4. Istruzione e ricerca;
5. Inclusione e coesione;

6. Salute.

I provvedimenti di interesse per le imprese ricadono in particolare nelle prime due missioni.

Le misure di incentivazione fiscale del Piano Transizione 4.0, anticipate dalla Legge di Bilancio 2021, prevedono il riconoscimento di tre tipologie di crediti di imposta alle imprese che investono in:

- Beni capitali, tra cui beni materiali e immateriali direttamente connessi alla trasformazione digitale dei processi produttivi cosiddetti 'beni 4.0';
- Ricerca, sviluppo e innovazione;
- Attività di formazione per acquisire o consolidare la conoscenza di tecnologie rilevanti (l'analisi dei Big Data e dei dati, l'interfaccia uomo-macchina, IoT, l'integrazione digitale dei processi aziendali, la sicurezza informatica).

L'investimento sostiene la realizzazione di investimenti in macchinari, impianti e attrezzature per prodotti di avanguardia [8].

2.2 Soluzioni sul mercato

L'industria manufatturiera e di processo adotta tecnologie capaci di raccogliere, storicizzare ed elaborare dati allo scopo di monitorare e comandare processi produttivi. La trasformazione digitale, causa e figlia della rivoluzione 4.0, ha il merito di aver introdotto nuove tecnologie (IIoT, cyber-physical system, Edge Computing, Big Data, Cloud, etc..) molte delle quali di derivazione informatica (IT).

Alcune delle principali aziende che offrono una soluzione per l'interconnessione di macchinari in ottica Industria 4.0 sono:

- Weidmüller: supporta le aziende a sviluppare il potenziale delle applicazioni industriali IoT attraverso soluzioni basate sulle specifiche esigenze e ad integrarle con successo nelle strutture esistenti.

Figura 2.1: Weidmuller gateway [11]

Attraverso una preelaborazione dei dati con tecnologia IoT Edge, effettua analisi dei flussi di dati con conseguente riduzione dei costi grazie alle seguenti funzionalità:

  - Programmazione attraverso piattaforma web accessibile via browser;
  - Acquisizione e pre-elaborazione dei dati con u-control web grazie a funzionalità IoT;
  - Gateway IoT per soluzioni industriali con gestione di reti con protocollo di comunicazione Modbus-RTU.

- EMERSON: DeltaV Spectral PAT offre un processo architetturale semplice con maggiori prestazioni portando i dati direttamente nel sistema di controllo per eseguire analisi in tempo reale. Inoltre, fornisce le seguenti funzionalità:
  - Esegue calcoli attraverso un modello chemiometrico, ovvero modelli descrittivi e analisi esplorativa dei dati;

– Fornisce previsioni di qualità basate su modelli predefiniti incorporati nel blocco funzionale Delta standard;

– Utilizza il protocollo di comunicazione OPC-UA.

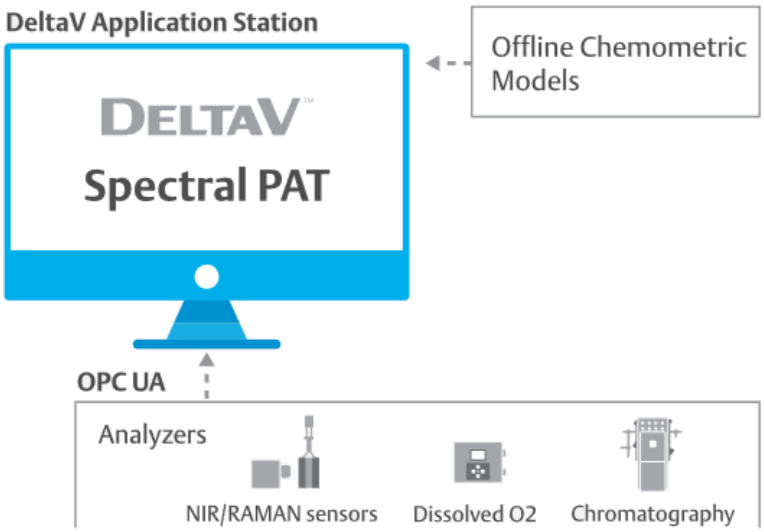


Figura 2.2: Delta V gateway [7]

- Alleantia: fornitori di software indipendenti basati su IIoT per migliorare l'attività, raggiungendo i massimi livelli di qualità e manutenzione predittiva su modelli predefiniti dell'impianto. I System Integrator si occupano dell'analisi delle macchine e delle esigenze del cliente, indagano sui dati che devono essere raccolti, identificando il numero dei sensori necessari. I System integrators Partners sono la chiave per la creazione di soluzioni flessibili e pronte all'uso adattate ad ogni contesto industriale [2].

2.3 Richiesta del mercato

Le aziende sopra citate e molte altre si focalizzano sul fornire un servizio personalizzato di interconnessione della macchina limitando pertanto il vero potenziale dell'interconnessione 4.0. Il passo ulteriore consiste nell'offrire al cliente un unico dispositivo hardware e software connesso alle reti IT e OT di fabbrica che permetta di far comunicare le macchine di produzione indipendentemente dal loro numero e

dal loro protocollo di comunicazione.
Da questo concetto nasce l'idea di un prodotto unico e programmabile direttamente dal cliente per estrapolare le informazioni utili all'analisi del rendimento ed efficienza della macchina; il software non si limita ad una semplice interconnessione ma permette di offrire servizi aggiuntivi per le varie macchine in modo da integrare in un unico pacchetto un sistema di interconnessione semplice, modulare e scalabile.

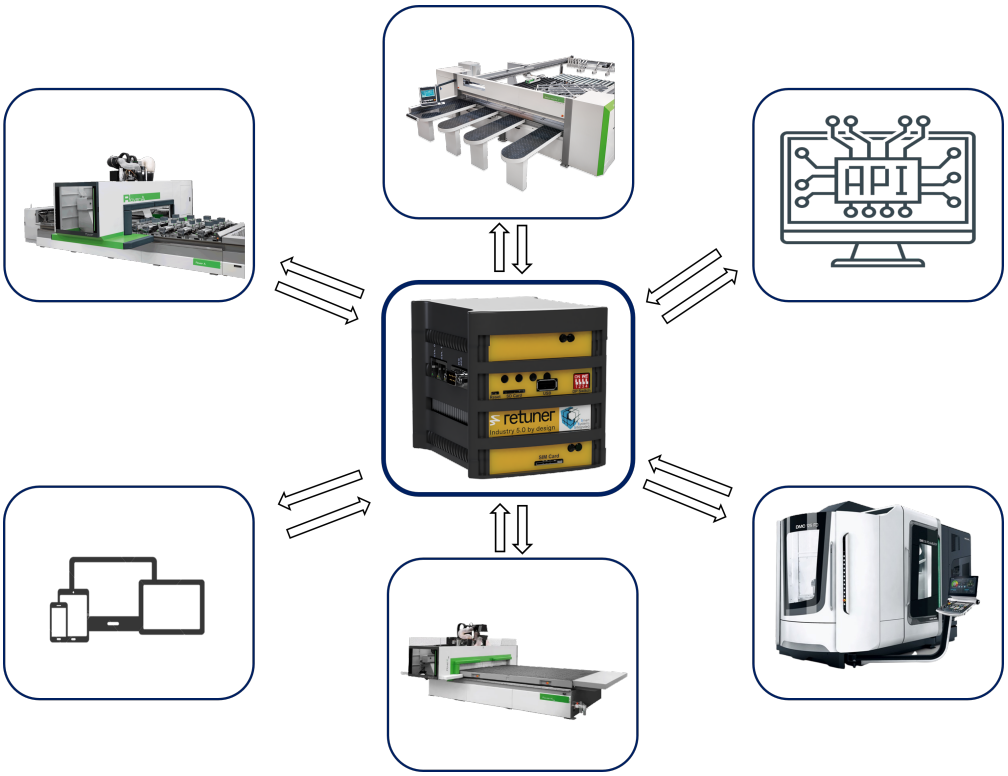


Figura 2.3: Richiesta del mercato

Capitolo 3

Caso di studio: Retuner® SmartEDGE$^{4.0}$

3.1 Descrizione del prodotto

SmartEDGE$^{4.0}$ è la componente distribuita sia come hardware che come software di Retuner®(suite dei prodotti ORCHESTRA) la cui funzione è interconnettere macchinari, impianti e sensori nuovi ed esistenti, analizzando segnali, variabili e messaggi da protocolli differenti. Fornisce il livello più basso per la connettività delle macchine mediante l'elaborazione a bordo di regole e algoritmi specifici in grado di filtrare, integrare e calcolare la trasmissione dei dati grezzi in informazioni affidabili e pronte all'uso. SmartEDGE$^{4.0}$ integra come sistema operativo Debian Bullseye permettendo l'inserimento di software proprietario e di partner terzi, scambia dati e informazioni wired e wireless con la piattaforma software IIoT Retuner® SmartHINGE$^{4.0}$ installata sul server della rete di fabbrica per condividerli ulteriormente con sistemi ICT aziendale.

Figura 3.1: Elettronica SmartEDGE$^{4.0}$

Integra al suo interno un modulo in grado di abilitare servizi aggiuntivi sviluppati da terze parti per aumentare ulteriormente il dettaglio di analisi qualità delle singole macchine in modo da fornire un unico punto di raccolta dati e invio verso l'esterno di informazioni significative, già elaborate e pronte per la visualizzazione su interfacce HMI (Human-Machine Interface).

3.2 Modello di sviluppo del software

Dopo una completa analisi architetturale del progetto è stato scelto di svilupparlo utilizzando come linguaggio Python e di optare per una metodologia di processo a spirale.

Si presuppone che lo sviluppo di un'applicazione sia un ciclo iterativo che viene ripetuto fino al raggiungimento dell'obbiettivo prefissato. Il modello a spirale minimizza il rischio dei grandi progetti software valutando periodicamente il prodotto intermedio. Il suo approccio incrementale e iterativo include anche la valutazione

periodica dei rischi sotto forma di bozze di prototipi, analisi o simulazioni. Tale approccio permette la mitigazione dei rischi in fase di sviluppo.

Fino allo stato finale, il progetto di un software viene sottoposto continuamente al ciclo del modello a spirale, costituito dai seguenti passaggi:

- Descrizione delle condizioni: un modello a spirale inizia con la valutazione di quali obbiettivi devono essere associati ai singoli passaggi dello sviluppo software;
- Valutazione di alternative: in questa fase vengono analizzati i punti critici per l'avanzamento dello sviluppo del progetto software;
- Sviluppo: dopo la fase di valutazione dei rischi, si prosegue con l'effettivo sviluppo del software, sebbene permangano inevitabili rischi residui.
- Pianificazione del ciclo successivo: concluso un ciclo, si procede subito con la pianificazione del ciclo successivo.

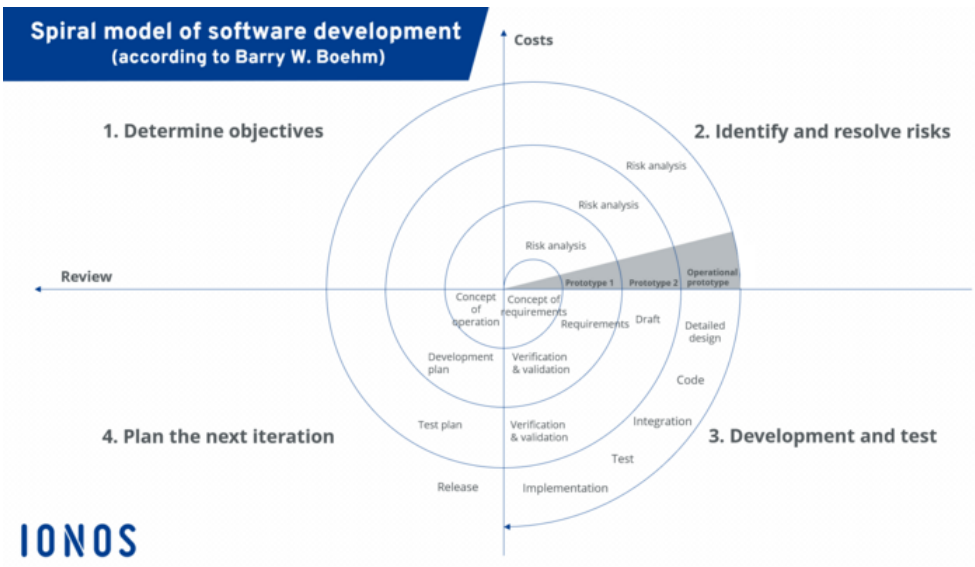


Figura 3.2: Modello a spirale [9]

A supporto del modello utilizzato per lo sviluppo del prodotto [9], è stato utilizzato il diagramma di Gantt per rappresentare e visualizzare graficamente le scadenze e l'avanzamento del processo di sviluppo. Il concetto alla base del diagramma di Gantt è stato formalizzato dall'ingegnere americano Henry L. Gantt, che l'ha sviluppato come metodo per descrivere la pianificazione della produzione e il carico delle risorse nelle fabbriche. Si tratta di un grafico a barre orizzontali necessario per controllare lo stato di sviluppo del progetto e analizzare gli eventuali ritardi di lavoro, sotto il controllo del project manager che schedulerà le attività man mano che vengono implementate. Nella fase iniziale facilita la pianificazione attraverso la definizione della timeline, delle dipendenze tra le attività da fare, assegnando le risorse e stimando le tempistiche [18].

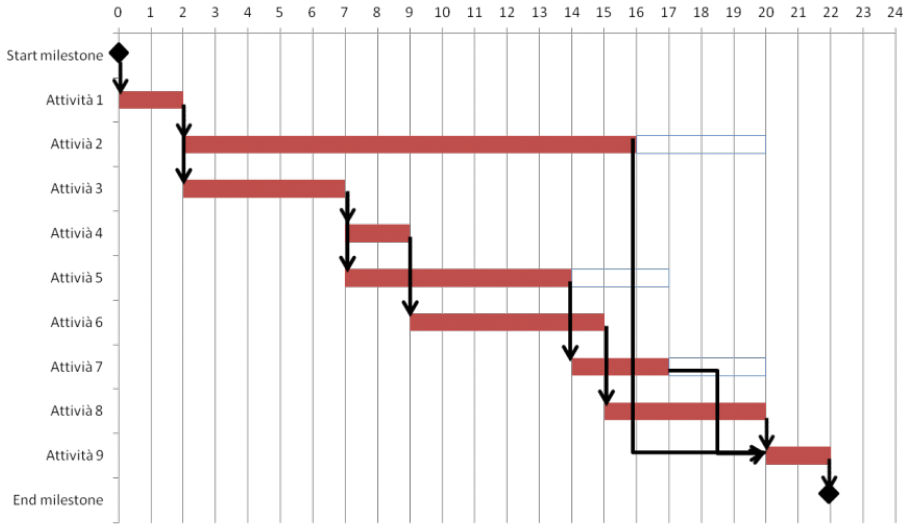


Figura 3.3: Diagramma di Gantt [5]

3.3 Funzionalità attese

Garantire l'interconnessione 4.0 delle macchine di produzione in modo da fornire al cliente una panoramica tempestiva ed esaustiva dello stato di lavorazione. Nello specifico occorre realizzare un prodotto software programmabile direttamente dal cliente per l'interconnessione degli impianti. Il software deve, nello specifico:

- Implementare i principali protocolli di comunicazione delle macchine di produzione come UPC-UA, Modbus, MTConnect e MoxaApi;
- Permettere l'interconnessione multipla degli impianti di produzione in modo rapido, trasparente e semplice per il cliente. Il sistema dovrà permettere la connessione di tipologie differenti di macchinari che comunicano in modi differenti tra loro;
- Storicizzare i dati raccolti dalle macchine in un repository locale in modo da garantire la persistenza dell'informazione;
- Invio dei dati endogeni verso lo SmartHINGE$^{4.0}$ appena il dato è pronto;
- Gestire eventuali anomalie di connessione tra SmartEDGE$^{4.0}$ e SmartHINGE$^{4.0}$ garantendo l'invio del dato endogeno non appena la connessione venga ripristinata;
- Implementare un modulo denominato 'Servizi digitali' in grado di analizzare, comprendere e calcolare una sequenza di operazioni matematiche complesse definite su file di configurazione slegandole sulle singole macchine permettendo così l'aggiornamento dinamico della logica desiderata dalla macchina;
- Acconsentire a servizi terzi di accedere a dati grezzi e/o elaborati richiedendoli direttamente allo SmartEDGE$^{4.0}$, in modo da avere un unico nodo centrale accessibile tramite richieste REST API;
- Integrare un sistema di autenticazione tra lo SmartEDGE$^{4.0}$ e il MiniMES$^{4.0}$ per l'invio dei parametri ricetta sulla macchina. Questo permetterà la scrittura di nuove configurazioni macchina tramite REST API con il gestionale MiniMES$^{4.0}$.

Figura 3.4: Smart System Integreated [13]

Capitolo 4

Architettura

4.1 Notazioni utilizzate

Nella fase di progettazione del software sono stati utilizzati differenti modelli allo scopo di marcare ed evidenziare le criticità del software in modo di analizzarle e di risolverle alla radice. Nello specifico il modello BPMN ha permesso di evidenziare il flusso generale del processo, gli attori coinvolti e le interazioni che essi hanno con il prodotto SmartEDGE$^{4.0}$. Per dettagliare maggiormente il flusso è stato utilizzato il modello dei casi d'uso che, assieme all'architettura software, ha permesso di analizzare il carico di lavoro dei singoli componenti e le attività svolte da ognuno di esse. Il modello UML, invece, ha permesso di visualizzare gli oggetti coinvolti e le strutture dati necessarie per realizzare il software.

4.1.1 Business Process Modelling Notation

Il Business Process Modelling Notation (BPMN) è un metodo costituito da diagrammi di flusso che modella dall'inizio alla fine le fasi di un processo aziendale pianificato. Elemento chiave per la gestione dei processi aziendali, illustra visivamente una sequenza dettagliata di attività e flussi di informazioni necessari per completare il processo. Il suo scopo è creare modalità per migliorare l'efficienza, tenere conto delle nuove circostanze o ottenere un vantaggio competitivo.

Ad alto livello, il BPMN si rivolge ai stakeholder e alle altre parti interessate in un processo aziendale per acquisire comprensione attraverso una rappresentazione visiva dei passaggi facili da comprendere. Su un piano di coinvolgimento più ampio, si rivolge alle persone che implementeranno il processo, fornendo dettagli sufficienti per consentire un'implementazione precisa. La rappresentazione grafica consente una comunicazione e una collaborazione più semplici per raggiungere l'obiettivo di un processo efficiente che produce un risultato di alta qualità [**?**].

Nel caso in esame, oggetto della presente trattazione, il processo inizia quando il prodotto software Retuner® SmartEDGE$^{4.0}$ effettua l'operazione di autenticazione verso il MiniMES$^{4.0}$ per abilitare i modulo denominato web hooks. In seguito, il sistema attende la richiesta di esecuzione di una delle tre operazioni:

1. Lettura dato macchina;
2. Scrittura dato in macchina;
3. Abilitazione di un servizio.

Nel caso in cui l'operazione non sia valida, il sistema terminerà l'esecuzione. L'implementazione generale del sistema è riportato in figura 4.1. Gli attori coinvolti nel sistema sono:

1. Retuner®: SmartEDGE$^{4.0}$;
2. Macchinari di produzione;
3. SmartHINGE$^{4.0}$;
4. MiniMES$^{4.0}$.

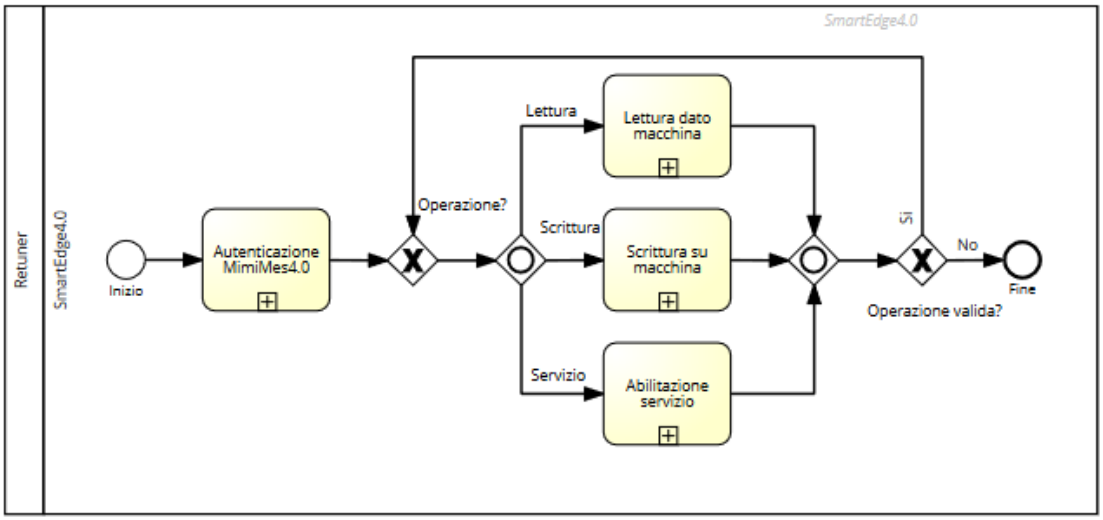


Figura 4.1: Modello BPMN generale

In seguito verranno analizzati nel dettagli i quattro blocchi presenti nel diagramma in figura 4.1

- Procedura di autenticazione verso il MiniMES$^{4.0}$: sequenza di operazioni che permette di creare un canale di comunicazione unidirezionale MiniMES$^{4.0}$ - SmartEDGE$^{4.0}$. Questo permette di ricevere dal MiniMES$^{4.0}$ le ricette. Le ricette sono una sequenza di variabili primitive inserite dall'utente che devono essere caricate in macchina per la sua parametrizzazione al fine di eseguire la lavorazione correttamente. Alcuni esempi di parametri sono: il codice della commessa, il lotto della lavorazione, la quantità di prodotti da produrre, il programma da avviare, etc. Per abilitare la ricezione delle ricette occorre la sottoscrizione ai web-hooks. I web hooks sono una procedura implementata nella componente MiniMES$^{4.0}$ che permette di segnalare tramite sottoscrizione l'invio di una ricetta da eseguire in macchina. Nello specifico lo SmartEDGE$^{4.0}$ richiede, tramite chiamate API RESTFUL. Più avanti verrà analizzato nel dettaglio il processo di esecuzione.

  Il processo inizia quando il prodotto software Retuner®, SmartEDGE$^{4.0}$, effettua una richiesta di registrazione verso il MiniMES$^{4.0}$ fornendo le credenziali di autenticazione. Il MiniMES$^{4.0}$, verificate le credenziali e, se queste

risultano corrette, conferma l'autenticazione. Nel caso in cui il sistema risulti offline o risponda negativamente, il processo termina. Superata la procedura di autenticazione, il software SmartEDGE$^{4.0}$prepara una richiesta di iscrizione ai web hooks. Confermata la sottoscrizione da parte del MiniMES$^{4.0}$, il sistema attende l'invio di una ricetta da scrivere in macchina, verifica la correttezza delle informazioni ricevute dal MiniMES$^{4.0}$e procede nell'inviarle direttamente alla macchina. Quest'ultima procedura si ripete fino al termine dell'iterazione.

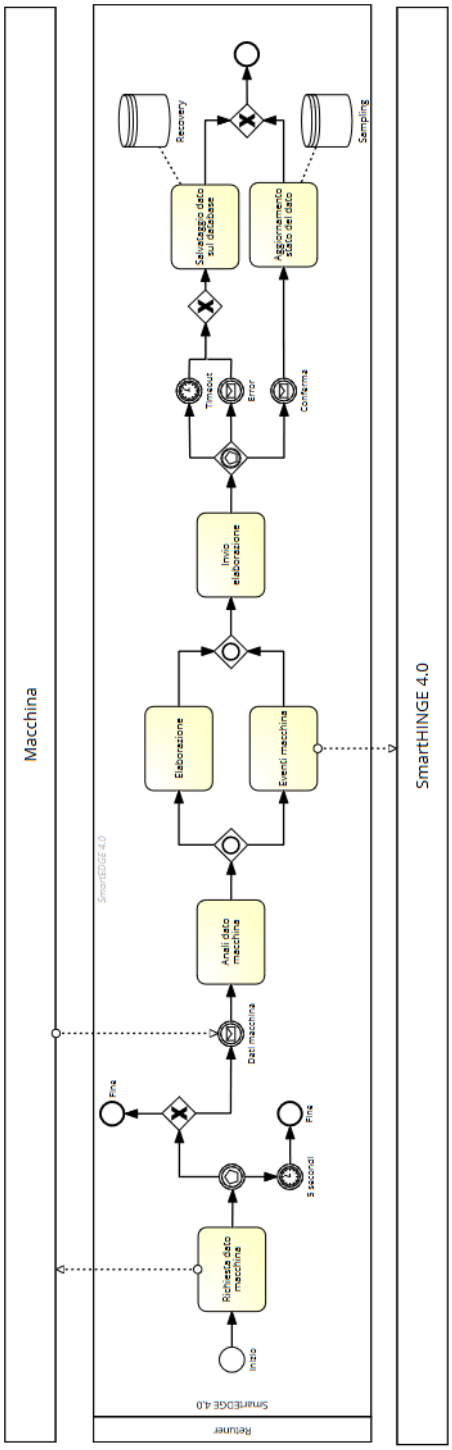


Figura 4.2: Autenticazione con il MiniMES$^{4.0}$

- Operazione di lettura dati grezzi della macchina e invio dell'elaborazione verso lo SmartHINGE$^{4.0}$: sequenza di operazioni che definisce il flusso di gestione della raccolta ed elaborazione del dato grezzo in macchina trasformandolo in informazioni elaborate da inviare alla SmartHINGE$^{4.0}$. Per dato grezzo si intende l'informazione primitiva che la macchina legge dai sensori installati al suo interno come ad esempio sensori di temperatura, sensori per il monitoraggio dello stato macchina, sensori per il tracciamento delle coordinate degli assi, sensori per segnalare l'inizio di una nuova lavorazione, sensori per l'abilitazione dell'erogazione dell'acqua, etc. Nonostante la semplicità del dato, forniscono informazioni estremamente preziose come lo stato attuale della macchina, il conteggio cumulativo e complessivo dei pezzi o prodotti lavorati, gli intervalli delle temperature registrate dalla macchina per tracciarne la rapidità di variazione, il calcolo della quantità di acqua utilizzata, etc. Da qui il software MiniMES$^{4.0}$ avrà a disposizione tutte le informazioni necessarie per ricavare il rendimento della macchina tramite i tempi di attività della macchina in modalità WORK diviso il tempo totale di attività della macchina, il consumo energetico della macchina, le percentuali di carico di lavoro delle macchine e molto altro. In seguito, verrà analizzato nel dettaglio il processo di esecuzione.

  Il processo inizia quando il prodotto software Retuner®, SmartEDGE$^{4.0}$, richiede dati relativi dai diversi sensori installati in macchina e attende la risposta del PLC o server a bordo macchina. Nel caso in cui la macchina non risponda, il processo va in time out e termina. Ricevuti i dati macchina, verranno processati dal sistema attraverso una prima fase di analisi del dato raccolto per ricercare eventi quali allarmi o eventuali anomalie da segnalare tempestivamente alla SmartHINGE$^{4.0}$. Successivamente il dato viene lavorato per inviarlo alla SmartHINGE$^{4.0}$. In caso la SmartHINGE$^{4.0}$ non risponda o segnali un errore, il sistema avvia una procedura per salvare il dato in modo da inviarlo successivamente. In caso di esito positivo, il sistema aggiorna il dato presente sul repository in locale.

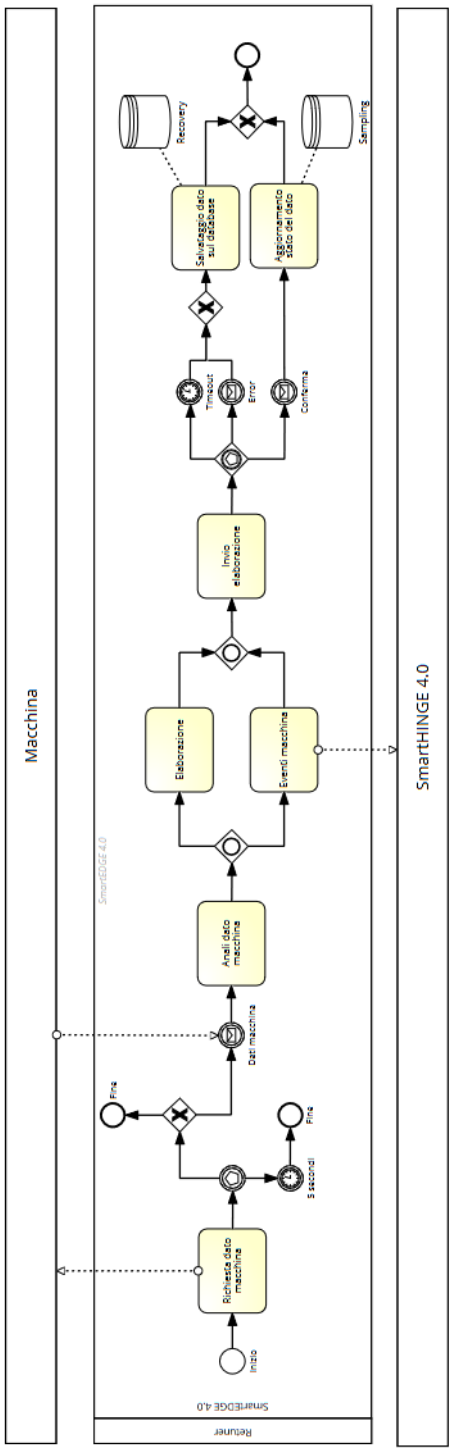


Figura 4.3: Lettura dato macchina e invio verso SmartHINGE$^{4.0}$

- Operazione di scrittura dati su macchina: la normativa vigente n. 57/L, allegato A (Articolo 1, comma 9) [19] richiede:

  > "procedura di interconnessione ai sistemi informatici di fabbrica con caricamento da remoto di istruzioni e/o part program, integrazione automatizzata con il sistema logistico della fabbrica o con la rete di fornitura e/o con altre macchine del ciclo produttivo."

  Allo scopo di ottemperare alla normativa, viene implementa la funzionalità di caricamento da remoto di istruzioni attraverso la scrittura dei parametri sulla macchina. La funzionalità permette al cliente finale la possibilità di configurare da remoto le macchine di fabbrica attraverso il software Retuner® MiniMES$^{4.0}$ o Technophylla SmartFarming$^{4.0}$ dove ogni macchina di fabbrica viene caratterizzata da una sequenza di parametri che, quando si attiva una commessa da lavorare, la macchina viene automaticamente configurata. Per commessa si intende la specifica di progettazione o di produzione da svolgere in tempi e modalità prestabiliti con il fornitore finale. In generale la commessa viene caratterizzata dal codice dell'ordine, dal codice e nome del cliente, la data di consegna e il suo stato attuale. In seguito, verrà analizzato nel dettaglio il processo di esecuzione.

  Il processo inizia quando il prodotto software MiniMES$^{4.0}$ invia la ricetta dei parametri macchina allo SmartEDGE$^{4.0}$ attraverso il meccanismo web hooks o tramite file in formato json. In entrambi i casi, il sistema SmartEDGE$^{4.0}$ verifica i dati ricevuti controllandone la validità e la correttezza e, successivamente, esegue l'operazione di scrittura delle variabili in macchina. Infine, la procedura viene terminata.

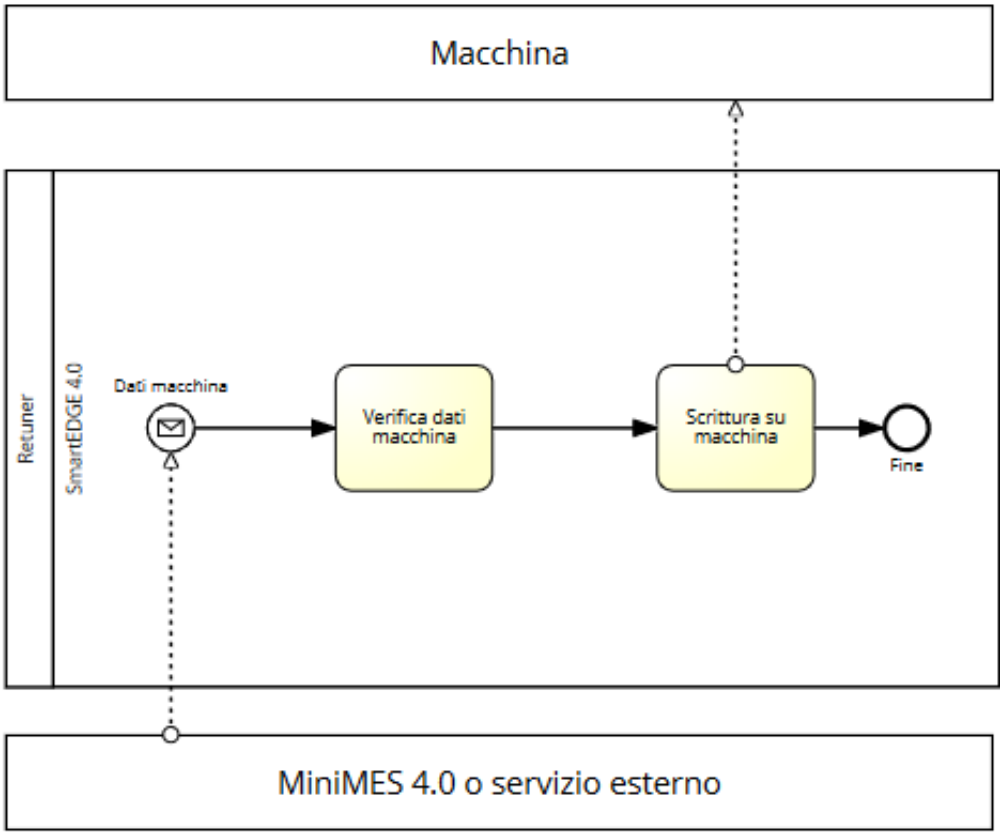


Figura 4.4: Modello BPMN per la scrittura su macchina

- Operazione abilitazione dei servizi aggiuntivi: sequenza di operazioni che definisce il flusso di esecuzione che inizia dalla richiesta di autenticazione al servizio di interfaccia macchina fino alle operazioni di richiesta di dati elaborati specifici per il servizio. Questa procedura permette di estendere la funzionalità base fornita dal MiniMES$^{4.0}$ per aggiungere ulteriori servizi alla macchina come, ad esempio, algoritmi specifici per la manutenzione predittiva o digital assistant dell'operatore/manutentore direttamente a bordo macchina. L'applicazione richiede le informazioni elaborate specifiche del servizio sulla macchina tramite richieste API senza occuparsi direttamente della formattazione ed elaborazione dei dati grezzi forniti dalla macchina e senza occuparsi direttamente dei protocolli di comunicazione della macchina di fabbrica interconnesse ma si occupa di tutto ciò lo SmartEDGE$^{4.0}$. In questo modo integrare nuove applicazioni diventa semplice, rapido e in-

dipendente dalle caratteristiche della macchina. Si entra così in ottica di industria 5.0 raggiungendo uno degli obiettivi cardine: produzione ad hoc di prodotti personalizzati sulla base delle esigenze del cliente. In seguito, verrà analizzato nel dettaglio il processo di esecuzione.

Il processo inizia quando un servizio esterno come applicazione web, mobile o altro, effettua una richiesta di abilitazione di un servizio verso lo SmartEDGE$^{4.0}$ se non lo ha richiesto in precedenza. Quest'ultimo verifica la correttezza delle credenziali e, in caso di errore, segnala al servizio che i dati immessi non sono validi e termina il processo. Viceversa, il servizio viene abilitato e il sistema attende che tale servizio invii una richiesta di scrittura o lettura dei dati macchina. Ricevuta la richiesta da parte dello SmartEDGE$^{4.0}$:

- se l'operazione richiesta è di lettura, il software elabora i dati grezzi trasformandoli in dati pronti all'uso e li invia al servizio chiamante tramite chiamata API;
- se l'operazione richiesta è di scrittura, il software elabora i dati ricevuti e li prepara per inviarli alla macchina.

Infine, la procedura termina.

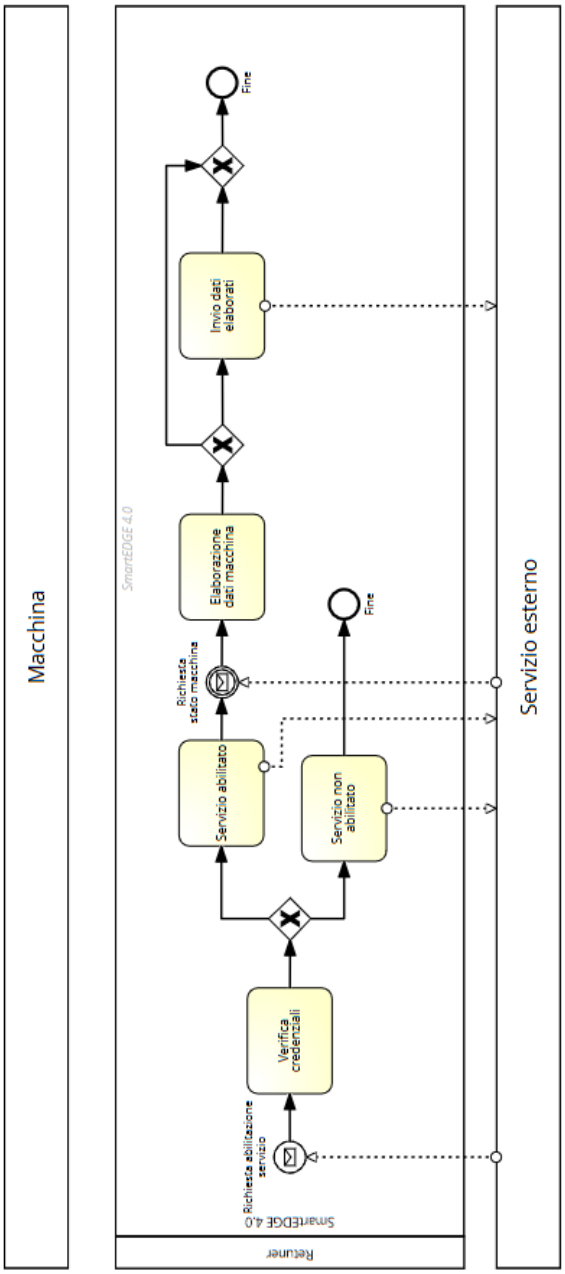
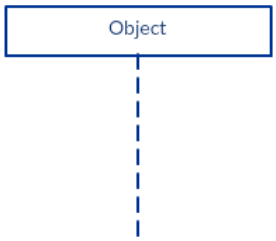


Figura 4.5: Abilitazione servizio esterno

4.1.2 Modello sequenziale

Il modello UML è un linguaggio ricco adottato in fase di progettazione del software per modellare soluzioni software, strutture applicative, comportamentali e di processi aziendali. Esistono due tipologie principali di modellazione; modellazione strutturale e modelli comportamentali. Il modello sequenziale rientra nella categoria di modelli sequenziali e viene utilizzato dagli sviluppatori per modellare le interazioni tra gli oggetti in un unico caso d'uso. Un diagramma di sequenza è strutturato in modo tale da rappresentare una linea temporale che inizia dall'alto e scende gradualmente per segnare la sequenza delle interazioni, ogni oggetto ha una colonna e i messaggi scambiati tra di loro sono rappresentati da frecce [4].

- Notazione rappresentativa della linea-vita del modello sequenziale
  - Un diagramma di sequenza è composto da diverse linee di vita posizionate orizzontalmente nella parte superiore del diagramma.

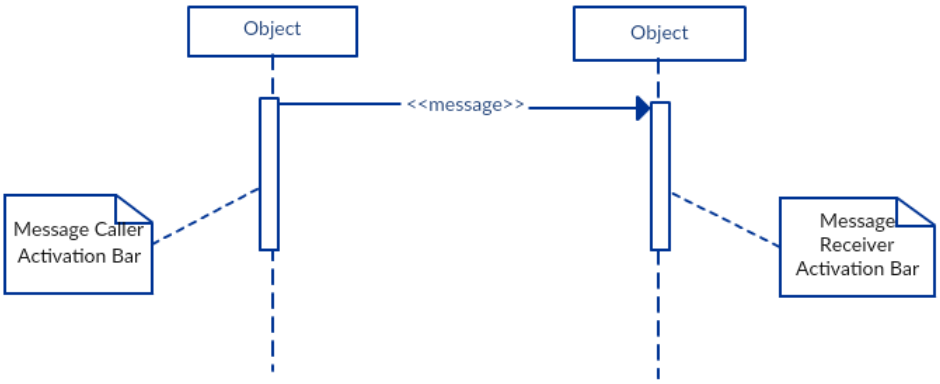


  - Le linee di vita con il simbolo di un elemento viene utilizzata quando si vuole rappresentare un attore o un sistema informatico.

  - Una line di vita con un elemento unità rappresenta i dati del sistema come il servizio clienti.

  - Una linea di vita con un elemento di confine indica un elemento di sistema o software in un sistema come ad esempio le schermate dell'interfacce utente, gateway o database

- Barra di attivazione è il riquadro posto sulla linea di vita. Si usa per indicare un'interazione tra due oggetti. La lunghezza del rettangolo rappresenta la durata degli oggetti in vita. In un diagramma di sequenza, l'interazione tra due oggetti avviene quando un oggetto invia un messaggio ad un altro.



Le entità coinvolte all'interno del modello sono le seguenti:

1. Macchina di produzione;
2. Receiver;
3. Analyzer;
4. Repository;
5. Connection;
6. SmartHINGE$^{4.0}$.

In seguito, verranno analizzati nei dettagli tre aspetti principali del processo software.

- Processo di campionamento dei dati grezzi in dati elaborati da inviare allo SmartHINGE$^{4.0}$: processo principale di esecuzione del software che prevede la raccolta dei dati grezzi forniti dalle macchine di fabbrica, la loro elaborazione attraverso un modulo denominato 'digital service' (che verrà dettagliato nel capitolo 5) fino all'invio del risultato ottenute verso lo SmartHINGE$^{4.0}$.

  Il processo inizia quando il software SmartEDGE$^{4.0}$ raccoglie il dato grezzo dalla macchina tramite il modulo 'Receiver', lo formatta opportunamente e inoltra l'informazione verso il modulo 'Analyzer'. Il modulo analizza l'informazione ricevuta ricercando eventuali allarmi o anomalia macchina in modo

da comunicarle tempestivamente alla SmartHINGE$^{4.0}$. Il dato grezzo viene inviato al modulo 'Repository' che effettuerà l'operazione di logica tramite il modulo 'digital service' sui dati raccolti e lo salva direttamente sul repository locale. Questa operazione viene fatta in modo asincrono dall'invio dei dati elaborati al modulo denominato 'Connection'. Questo modulo si occupa di raccogliere i dati elaborati dai moduli precedenti, inviarli verso lo SmartHINGE$^{4.0}$ e aggiornare il database. Nell'aggiornamento si possono verificare i seguenti scenari:

  - Lo SmartHINGE$^{4.0}$ restituisce il valore 200: il modulo aggiorna lo stato dell'elaborazione in 'HINGE';
  - Lo SmartHINGE$^{4.0}$ non risponde o segnala un errore: si salva sul database il dato non inviato in modo da garantire la persistenza e l'invio dell'informazione non appena la connessione viene ripristinata lo SmartEDGE$^{4.0}$ re-invia il dato e aggiorna il database.

- Operazione di ripristino connessione verso SmartHINGE$^{4.0}$: la procedura di comunicazione si basa sul protocollo API. L'abilitazione della richiesta da parte dello SmartEDGE$^{4.0}$ allo SmartHINGE$^{4.0}$viene confermata attraverso un token di autenticazione inserito all'interno del file di configurazione delle macchine. Ad ogni macchina viene associato uno specifico token, una sequenza alfanumerica di caratteri ottenuti dalla cifratura di un segreto all'interno del software SmartHINGE$^{4.0}$.

  All'avvio del software, vengono ricaricati gli eventuali dati elaborati non inviati precedentemente alla SmartHINGE$^{4.0}$ direttamente nel modulo 'Connection'. Se l'operazione di invio del dato viene confermata, aggiorna lo stato dei dati presenti sul repository. Viceversa, li colleziona nelle variabili pronto a trasmettere i dati elaborati appena la connessione viene ripristinata.

4.1.3 Unified Modelling Language

Unified Modelling Language o UML è un linguaggio unificato di modellazione grafica utilizzata per progettare e implementare sistemi software molto complessi

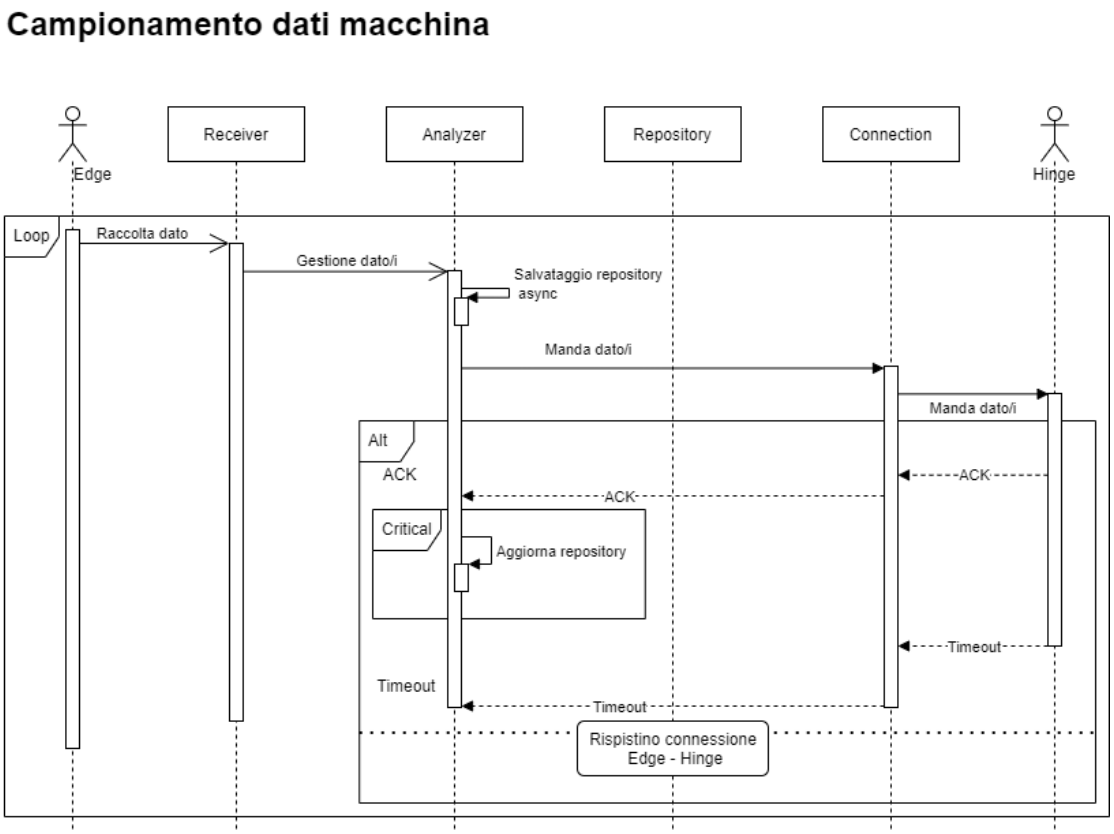


Figura 4.6: Campionamento dati macchina

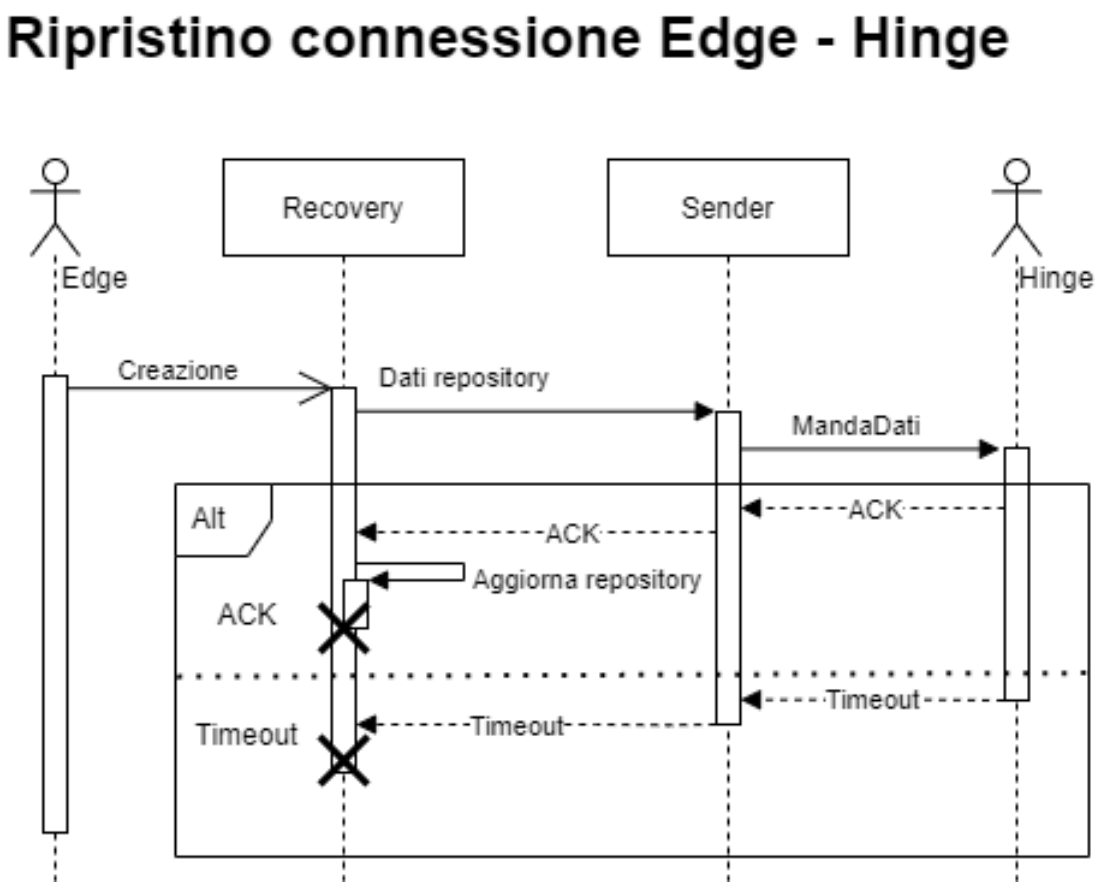


Figura 4.7: Procedura di ripristino

dal punto di via sia architetturale sia procedurale. Nel complesso, i diagrammi UML descrivono i confini, la struttura e il comportamento del sistema e degli oggetti al suo interno.

In informatica, comprende numerosi paradigmi o modelli per la risoluzione dei problemi quali ad esempio:

- linguaggi imperativi;
- linguaggi funzionali;
- linguaggi dichiarativi;
- linguaggi orientati agli oggetti (OOP).

UML è una combinazione di diverse notazioni orientate agli oggetti: progettazione, tecnica di modellazione e ingegneria del software orientata agli oggetti [10].

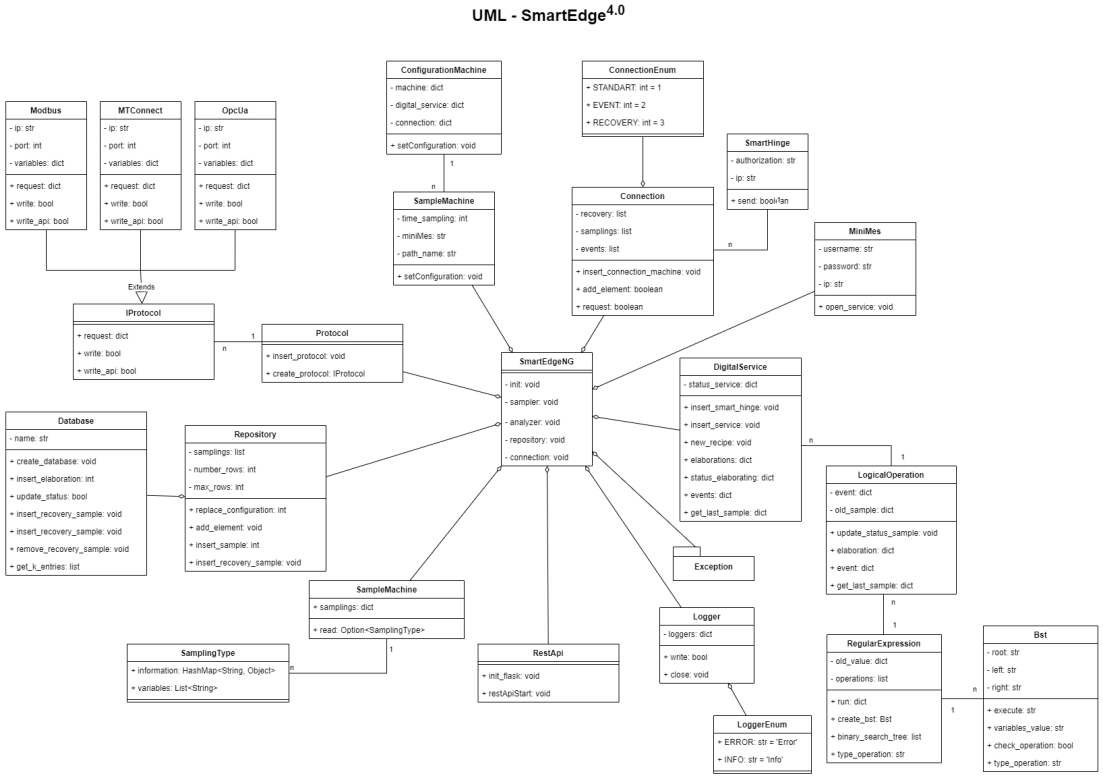


Figura 4.8: UML SmartEDGE$^{4.0}$

4.1.4 Architettura progettata

Il modello grafico in figura 4.9 evidenzia la struttura software per gestire il flusso di dati. In particolare, i differenti flussi o processi sono:

- Sampler: si occupa di stabilire la comunicazione con la macchina attraverso il protocollo di comunicazione caratteristico. Inoltre, effettua le richieste in modo campionato con frequenza variabile dal cliente ottenendo una campionatura omogenea e regolare;
- Analyzer: si occupa di recuperare il dato dal processo di lavoro precedente attraverso un buffer di sistema, effettua operazioni dinamiche per rintracciare dei fenomeni endogeni della macchina; ovvero informazioni recuperabili direttamente dalla macchina attraverso equazioni matematiche come la quantità di pezzi prodotti, lo stato attuale della macchina, la modalità di lavorazione della macchina, etc. Il dato grezzo viene inoltrato al modulo Repository e, in presenza di fenomeni esogene; ovvero informazioni non recuperabili direttamente dalla macchina ma ricavabili attraverso il controllo dei valori ricevuti dalla macchina che risultano fuori intervallo accettabile, l'anomalia viene inviata anche al processo Connection che sui occuperà di inviarlo alla SmartHINGE$^{4.0}$;
- Repository: si occupa di trasformare i dati grezzi in dati elaborati, salva il risultato sul database locale impostando lo stato del dato in 'NEW'; ovvero il dato è stato caricato nel repository ma non è ancora stato inviato alla SmartHINGE$^{4.0}$. Termina il ciclo di operazioni con la preparazione del dato da inviare al processo Connection;
- Connection: si occupa di mandare verso l'esterno i dati elaborati ricevuti dal processo precedente. Successivamente aggiorna il risultato della connessione verso lo SmartHINGE$^{4.0}$. In caso di invio con successo, aggiorna lo stato del dato inviato passandolo da 'NEW' a 'HINGE'; ovvero il dato è stato inviato correttamente alla SmartHINGE$^{4.0}$; viceversa inserisce una copia del solo dato elaborato sul database in modo da garantire la persistenza delle informazioni in caso di anomalie del sistema. Per anomalie del sistema si

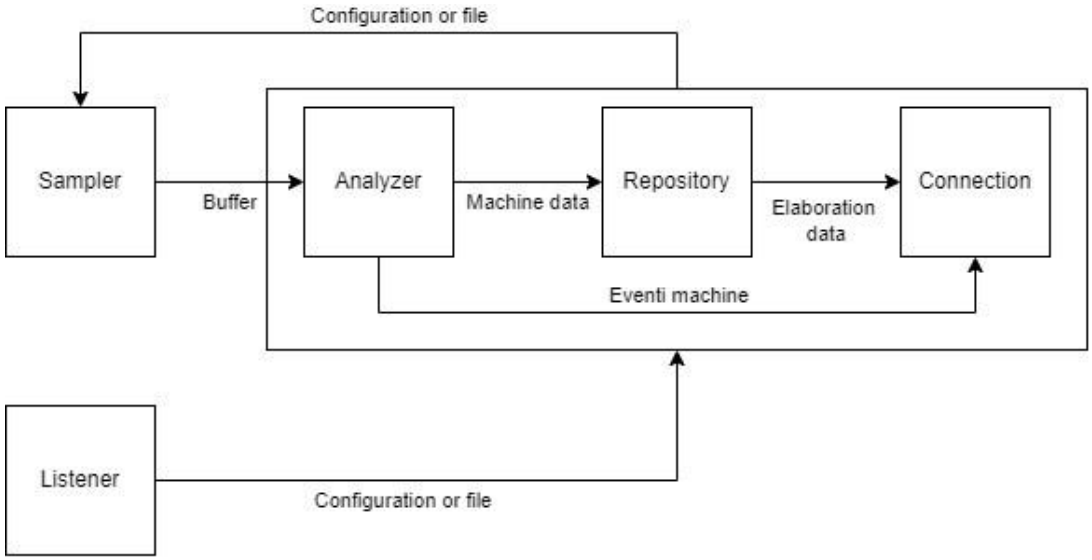


Figura 4.9: Architettura software SmartEDGE$^{4.0}$

intendono tutte gli errori che si possono verificare durante il processo di raccolta e invio del dato, ad esempio errori di connessione verso il server dove è stato installato lo SmartHINGE$^{4.0}$, calo di tensione sull'elettronica hardware dello SmartEDGE$^{4.0}$, etc.;

- Listener: web server in Flask dedicato con la funzione di ricevere comandi esterni dallo SmartHINGE$^{4.0}$, MiniMES$^{4.0}$ o servizi esterni abilitati tramite protocollo API. In questo modo è possibile configurare dinamicamente il sistema di raccolta dei dati o inviare ai servizi esterni i dati elaborati della macchina in modo rapido e semplice.

Capitolo 5

Implementazione

5.1 Strategie implementative

Il progetto SmartEDGE$^{4.0}$ nasce dall'esigenza di interconnettere macchinari di produzione, installati all'interno delle fabbriche attraverso un prodotto stabile, sicuro e scalabile. Vengono analizzate di seguito le difficoltà legate all'interconnessione di macchine diverse e le strategie implementative studiate al fine di superarle:

- Ogni macchina comunica in modi differenti come Modbus, OPC-UA, MTConnect, API Restful, etc. può abilitare l'accesso in lettura e/o in scrittura sui registri e può richiedere, in alcuni casi, l'autorizzazione tramite credenziali statiche definite dal costruttore della macchina. La ricerca di una uniformazione si complica ulteriormente nel caso in cui il costruttore di macchine fornisca librerie personalizzate per l'accesso alle informazioni. La soluzione adottata consiste nell'implementazione di una classe Protocol con le seguenti funzioni:
  - insert_protocol(imei: str, config_protocol: dict): la funzione si occupa di inserire in una lista interna alla classe la nuova macchina da interconnettere e un'istanza figlia dell'interfaccia IProtocol.

- get_protocols(): funzione che ritorna la lista delle macchine con i relativi protocolli associati

```
class Protocol:
    def __init__(self):
        self.__protocols = {}

    def insert_protocol(self, imei: str, protocol:
       dict):
        self.__protocols[imei] = Protocol.
            create_protocol(protocol)

    def get_protocols(self):
        return self.__protocols.copy()
```

La funzione Protocol.insert_protocol() si occupa di creare un'istanza della classe richiesta. Ogni classe figlia che estende l'interfaccia IProtocol deve implementare i seguenti metodi:

- request(): funzione che richiama le librerie fornite dal protocollo P per accedere in lettura ai registri dove i dati vengono storicizzati dalla macchina. In caso di errore nella comunicazione e genera un eccezione di classe ExceptionProtocol();
- write(): funzione che attende in un directory un file con estensione .json contenente le variabili da scrivere in macchina. Questa funzionalità genera una vulnerabilità grave perché l'autore del file non è definibile a priori ed è possibile utilizzarla quando il software SmartEDGE$^{4.0}$ viene installato nel server assieme allo SmartHINGE$^{4.0}$ e MiniMES$^{4.0}$ o SmartFarming$^{4.0}$, pertanto è stata deprecata a favore della funzione write_api()
- write_api(parameters: dict): funzione che riceve come argomento i parametri da scrivere in macchina. I parametri vengono inviati dal MiniMES$^{4.0}$ quando l'utente invia la ricetta in macchina e ricevuti dal

web server presente all'interno del software SmartEDGE[4.0]. Questo garantisce la provenienza del parametro riducendo il rischio di un attacco informatico.

- open(): funzione che apre la connessione con la macchina in modo da tenere la connessione sempre attiva. L'invocazione della procedura avviene al richiamo della classe figlia che definisce il protocollo. Alcune macchine prevedono l'apertura della connessione solo una volta per evitare di istanziare continuamente connessioni.
- close(): funzione che chiude la connessione con la macchina. L'invocazione della procedura avviene al richiamo del distruttore della classe

In seguito, la rappresentazione dell'interfaccia IProtocol

```
class IProtocol:
    def open(self) -> None:
        pass

    def request(self) -> dict:
        pass

    def write(self) -> None:
        pass

    def write_api(self) -> None:
        pass

    def close(self) -> None:
        pass
```

- I produttori di macchine progettano il loro prodotto integrando sensori per il monitoraggio dell'impianto. Ogni progettazione è pensata per campionare accuratamente i dati generati da ogni singolo sensore della macchina. Le informazioni che possono essere raccolte sono molteplici e differenti tra di

loro. Questo comporta una personalizzazione delle operazioni matematiche da effettuare per trasformare i dati grezzi dei sensori in informazioni significative e utili al cliente finale. Inoltre, le operazioni matematiche possono variare nel tempo e l'introduzione di nuovi servizi aggiuntivi comporta l'aggiunta di ulteriori elaborazioni. Tali funzionalità devono essere configurabili dinamicamente in funzione delle necessità del cliente, delle caratteristiche e delle lavorazioni della macchina.

La soluzione adottata per rispondere al requisito è l'introduzione di un modulo denominato 'digital service'. Esso prevede la creazione di una classe DigitalService che implementa le seguenti funzioni:

- insert_new_hinge(imei: str, start_hinge: dict): funzione che inserisce nel dizionario interno alla classe una struttura a doc denominata LogicalOperation per il calcolo delle equazioni matematiche ricevute come parametro. Questa struttura si occupa di interpretare l'equazione ricevuta in ingresso e la trasforma in un albero di operazioni matematiche. L'albero è costituito da un nodo contenente l'operazione e due foglie contenenti ulteriori operazioni o variabili alfanumeriche.

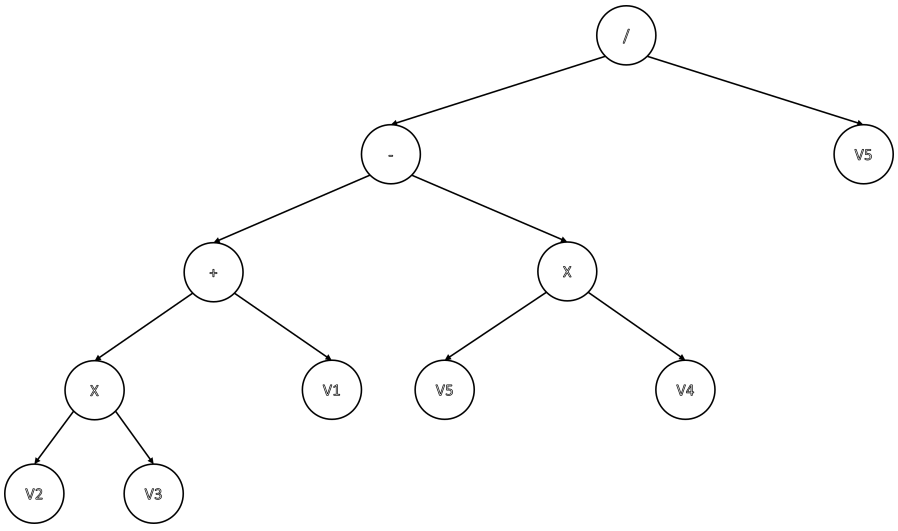


Figura 5.1: Albero operazionale

L'operazione di calcolo dell'equazione non è altro che un'esplorazio-

ne ricorsiva dell'albero partendo dal root dell'albero fino alle foglie. Consegue una complessità computazionale lineare.

$$\theta(2 * operazioni + 1) = \theta(n)$$

Viene introdotto un'equazione aggiuntiva 'MiniMes_recipe' utile nel caso di reset dei contatori dei pezzi prodotti associati a una commessa. Il settaggio del flag della ricetta viene effettuata dalla funzione new_recipe() quando arriva una nuova ricetta dal MiniMES$^{4.0}$. Per ricetta si intende una lista di parametri da inviare in macchina per impostare una nuova configurazione di lavoro.

```
def insert_smart_hinge(self, imei: str, smart_hinge
    : dict):
     temp_smart_hinge = {
          'MiniMes_recipe': {
               'mode':
                    'oneTime#cast;int␣0,event#eq;
                       $MiniMes_recipe␣!f=␣0,never',
               'value':
                    'if;$MiniMes_recipe␣=f=␣1;T;cast;
                       int␣2;F;cast;int␣0'
          }
     }

     for _key, _value in smart_hinge['elaboration'].
        items():
          temp_smart_hinge[_key] = _value

     logical_operation = LogicalOperation(
          temp_smart_hinge,
          smart_hinge["event"])
```

```
        if not self.__digital_service.__contains__(imei
            ):
            self.__digital_service[imei] = {
                "SmartHinge": logical_operation,
            }
        else:
            self.__digital_service[imei]["SmartHinge"]
                = logical_operation

        self.__status_service[imei] = {'SmartHinge':
            True}
```

- new_recipe(imei: str): funzione invocata quando arriva una nuova ricetta dal MiniMES$^{4.0}$ che imposta il valore a uno in modo da segnalare e aggiornare eventuali equazioni matematiche per i reset o i conteggi delle ricette ricevute.

- elaborations(imei: str, sample: dict): funzione che effettua l'elaborazione dettagliata dei dati restituendo il risultato delle operazioni matematiche definite dal consumatore. La procedura itera su ogni equazione creando un oggetto 'result' che conterrà i valori delle variabili da inviare verso lo SmartHINGE$^{4.0}$.

```
def elaborations(self, imei: str,
    sample: dict,
    service: str = 'SmartHinge'):
    [...]
    result = {}
    temporal_value = {}
    for _service, _value_service in self.
        __digital_service[imei].items():
        if self.__status_service[imei][_service]:
            temporal_value = _value_service.
                elaboration(sample)
```

```
            if len(temporal_value) != 0:
                result[_service] = temporal_value
    [...]

    return result
```

- events(imei: str, sample: dict): funzione analoga a elaborations() che effettua altre tipologie di operazioni matematiche con priorità maggiore. La procedura itera su ogni equazione creando un oggetto 'result' che conterrà i valori delle variabili da inviare verso lo SmartHINGE$^{4.0}$.
- insert_service(imei: str, name: str, service: dict): funzione che, ricevuti in ingresso l'identificativo della macchina, il nome del servizio e le equazioni matematiche, lo aggiunge nel dizionario dei servizi in modo da abilitarne il calcolo attraverso la funzione status_elaborate().
- status_elaborating(imei: str, service_name: str): funzione che, ricevuti in ingresso l'identificativo della macchina, il nome del servizio e le equazioni matematiche. A differenza delle funzioni events() ed elaborations(), questa procedura viene invocata solo su richiesta dal servizio esterno.

```
def status_elaborating(self, imei: str,
    service_name: str):
     return self.__digital_service[imei][
         service_name]
           .elaboration(self.get_last_sample(
               imei,
               'SmartHinge'))
```

Alcune modalità di calcolo delle equazioni matematiche permettono di creare delle variabili locali per operazioni future senza la reale necessità di inviarle all'esterno, oppure il risultato dell'elaborazione non rappresenta nuove informazioni utili e, di conseguenza, non devono essere trasmesse. La realizzazione di una variabile da elaborare avviene tramite la definizione di due caratte-

ristiche fondamentali: la modalità di calcolo e l'operazione matematica. La modalità di calcolo definisce le metodologie computazionali della variabile. Nello specifico:

- never: modalità di calcolo che prevede un'elaborazione ad ogni campione raccolto ma il risultato ottenuto non deve essere trasmesso verso l'esterno;
- always: modalità di calcolo che prevede l'elaborazione e l'invio del risultato ad ogni campione raccolto. Questa modalità è sconsigliata per evitare un flusso costante di informazioni;
- onChange: modalità di calcolo che prevede l'elaborazione ad ogni campione raccolto e l'invio solo in presenza di alterazione del valore rispetto al passo precedente. Questa modalità permette di inviare verso l'esterno solo le variazioni di valore;
- afterS N: modalità di calcolo che prevede l'elaborazione a intervalli N di campioni raccolti. Utile quando si vuole sovra-campionare una variabile;
- onChange or afterS N: modalità di calcolo equivalenti a onChange e afterS N. Utile quando si vuole trasmettere l'alterazione del valore ma informare la costanza del valore ad ogni N campioni raccolti;
- oneTime: modalità usata all'avvio dell'elaborazione per impostare un valore di inizio. Usata soprattutto per le variabili contatori dove il valore di inizio deve valere zero;
- event: modalità di calcolo che effettua una preelaborazione attraverso un filtro sulle variabili macchina. Usata soprattutto per la ricerca di fenomeni esogeni;
- reset: modalità che si attiva ad ogni calcolo dell'elaborazione. Utilizzata assieme alla modalità event per impostare un valore alla variabile subito dopo averla trasmessa verso l'esterno.

Le modalità sopra elencate possono essere opportunamente concatenate fornendo una forma di calcolo flessibile e articolata. Le operazioni matematiche, invece, rappresentano le espressioni logico-matematiche e non solo. Verrà in seguito elencata la lista delle operazioni permesse:

- comando eq: operazione che prevede un'espressione matematica;
- comando cast: operazione che formatta la variabile macchina;

```
{
    "elaboration": {
        "elaboration_01": {
            "mode": "always",
            "value": "cast;int $old.val_01"
        },
        "elaboration_02": {
            "mode": "onChange",
            "value": "cast;int $val_01"
        },
```

- comando if: operazione che permette di simulare l'operatore if per restituire un risultato a seconda di una condizione. È possibile annidare l'operatore gli operatori if;

```
{
    "elaboration": {
        "elaboration_01": {
            "mode": "onChange or afterS_5",
            "value": "if;$val_01 > $val_02;" +
                     "T;if;$val_01 > $val_03;" +
                     "T;cast;int $val_01;" +
                     "F;cast;int $val_03;" +
                     "F;if;$val_02 > $val_03;" +
                     "T;cast;int $val_02;" +
                     "F;cast;int $val_03"
        },
```

```
        }
    }
```

- comando msg: operazione che assegna una stringa;
- comando list: operazione che restituisce una lista di variabili grezze o elaborate;

```
{
    "elaboration": {
        "elaboration_01": {
            "mode": "never",
            "value": "if;$val_01 -> 20;T;msg;Allarme
                X;F;msg;_",
        },
        "elaboration_02": {
            "mode": "never",
            "value": "if;$val_01 -> 30;T;msg;Allarme
                Y;F;msg;_",
        },
        "elaboration_03": {
            "mode": "never",
            "value": "if;$val_01 -> 40;T;msg;Allarme
                Z;F;msg;_",
        },
        "elaboration_04": {
            "mode": "always",
            "value": "list;$elaboration_01
                $elaboration_02 $elaboration_03",
        },
    },
}
```

- comando sqrt, pow e devStd: operazione matematica per il calcolo della radice quadrate, elevamento a potenza e deviazione standard,

rispettivamente;

- comando avg: operatore matematica che effettua l'operazione di media.

- Le macchine possono generare direttamente o indirettamente degli allarmi macchina dovute a fenomeni endogeni quali bassa pressione dell'impianto, assenza di materie prime nel container, superamento delle soglie di temperature imposte dal fornitore della macchina, etc. Molti di questi eventi possono essere gestiti dal software effettuando un'analisi a priori delle variabili raccolte per ricercare anomalie e incongruenze con i valori attesi. Il processo di raccolta dei dati macchina prevede una fase di ricezione, una fase di elaborazione, una fase di salvataggio e una fase di invio del dato che causa un ritardo significativo per la segnalazione di eventuali anomalie macchina. Pertanto, il flusso di esecuzione deve prevedere un flusso alternativo per l'analisi e l'invio dei problemi macchina.

  La soluzione adottata prevede la separazione della fase di elaborazione del dato in due punti distinti. Nello specifico la gestione degli eventi dovrà essere gestita dal thread Analyzer. Esso effettuerà una prima elaborazione del dato raccolto ed eventuale invio del risultato al thread Connection, dove procederà a mandare il dato verso lo SmartHINGE$^{4.0}$. In seguito, il thread Analyzer procederà con la preparazione del dato che inoltrerà al thread Repository, flusso che completerà la fase di elaborazione del dato macchina.

  Verrà mostrato in seguito il frammento di codice per la gestione dei servizi digitali di prossimità.

```
def _analyzer(self):
    [...]
    result = self.__sample_machine.read()

    if result["type"] ==
        SmartEdgeNG.INFORMATION_TYPE_SAMPLE:
        if isinstance(result['value'], dict):
            events = self.__digital_service.events(
                result['imei'],
```

```
                    result["value"])

                if len(event):
                    self.__connection.add_element(
                        result['imei'],
                        result['value'],
                        Connection.ConnectionEnum.EVENT)
        [...]

    def _repository(self):
        [...]
        for sampling in self.__repository.get_samplings():
            for _imei, _value in sampling.items():
                result =
                    self.__digital_service.elaborations(
                        _imei,
                        _value)

                self.__connection.add_element(
                    _imei,
                    result,
                    Connection.ConnectionEnum.STANDARD)
        [...]
```

- L'elaborazione del dato fornisce informazioni molto importanti all'utente e ai software successivi quali SmartHINGE$^{4.0}$, MiniMES$^{4.0}$ e SmartFarming$^{4.0}$. La mancata ricezione del dato comporta una incongruenza tra il flusso di lavoro della commessa e le informazioni raccolte dalla macchina. Le anomalie possono essere molteplici come errore connessione con la macchina perché offline o non più interconnessa alla rete, errore di comunicazione verso i software esterni causati da servizio offline o l'elettronica SmartEDGE$^{4.0}$ non è connessa a Internet. Queste situazioni sono critiche per il tracciamento del flusso del lavoro. I problemi che li possono causare vanno da eventi esogeni

come un calo di tensione, rottura dello switch, dal cavo di rete e/o thrashing del server. A questi eventi possiamo integrare anche eventi endogeni quali la fase di aggiornamento del server dove sono state installate le istanze software SmartHINGE$^{4.0}$, MiniMES$^{4.0}$ e/o SmartFarming$^{4.0}$, cambio dello switch, gateway o cavo lan dovuti a guasti o aggiornamenti architetturali. Questi fenomeni devono essere gestiti dal software che dovrà provvedere a garantire la persistenza e la consistenza dei dati raccolti e tracciare lo stato del dato; ovvero sapere quali dati sono stati mandati correttamente e quali, invece, devono ancora essere inviati.

Le circostanze sopra esposte possono essere raggruppate in tre scenari:

- La macchina non risponde correttamente alle richieste oppure non risulta interconnessa in rete: il software dovrà comunicare verso lo SmartHINGE$^{4.0}$ l'anomalia riscontrata in modo che il cliente finale possa, in tempo reale, visualizzare lo stato della macchina è intervenire di conseguenza;
- Il software SmartHINGE$^{4.0}$ non risulta interconnessa in rete: il software dovrà salvare il dato all'interno del database locale in modo da poter inviare il dato appena la connessione viene ripristinata;
- l'elettronica SmartEDGE$^{4.0}$ non riesce a comunicare verso l'esterno: in questo caso il software traccerà il problema storicizzando sul database gli stati del sistema in modo da poterli comunicare in seguito. Da parte del software SmartHINGE$^{4.0}$ viene visualizzato sulla sua interfaccia lo stato offline dell'elettronica. In questo modo il consumatore si può accorgere del problema ed effettuare i dovuti controlli.

Il framework utilizzato per creare il repository locale è sqlite, un database molto leggero e veloce installato all'interno dello SmartEDGE$^{4.0}$. Per gestire correttamente il tracciamento dei dati, sono stati creati le seguenti tabelle con relativi dati:

- Configuration(ID, machine, value): tabella che tiene traccia di tutte le configurazioni in modo da legare correttamente l'elaborazione del dato

alla configurazione;

- Sampling(ID, configuration, elaboration, status, timestamp): tabella che storicizza tutte le elaborazioni effettuate e le associa a uno stato di processamento del dato;
- Recovery(ID, machine, elaboration): tabella per il salvataggio del dato che non è stato correttamente inviato alla SmartHINGE$^{4.0}$.

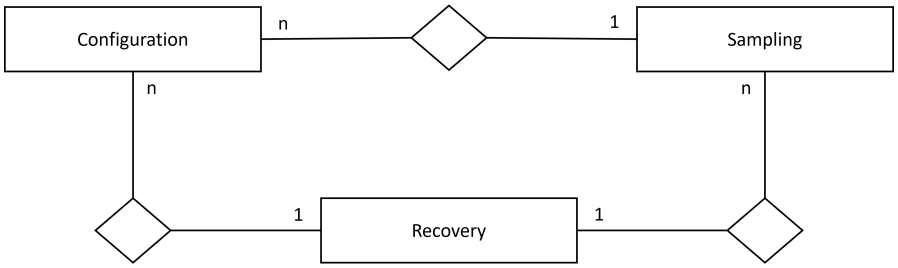


Figura 5.2: Modello entità relazione

All'interno del software, la fase di storicizzazione e aggiornamento dello stato del dato, sono state separate in due punti distinti. La fase di salvataggio è stata inserita nel thread Repository mentre la parte di aggiornamento nel thread Connection. Verrà mostrato in seguito il frammento di codice.

```
def repository(self):
    [...]
    for sampling in self.__repository.get_samplings():
        for _imei, _value in sampling.items():
            result = self.__digital_service.
               elaborations(
                _imei,
                _value)

            result['ID'] = self.__repository.
               insert_sample(
                self.__imei_config[_imei],
                self.__digital_service.get_last_sample(
```

```
                    _imei,
                    'SmartHinge'),
                'NEW' if result.get('SmartHinge')
                else 'STORED')

                if len(result) != 1:
                    self.__connection.add_element(
                        _imei,
                        result,
                        Connection.ConnectionEnum.
                            STANDARD)
        [...]

    def _connection(self):
        [...]
        for sampling in self.__connection.get_samplings():
            for _imei, _value in sampling.items():
                status, response = self.__connection.
                    request(
                    _imei,
                    _value)
                if status:
                    if _value.get('ID') is not None:
                        self.__repository.
                            update_status_sample(_value['ID'
                            ])
                else:
                    if _value.get('ID') is not None:
                        self.__repository.
                            insert_recovery_sample(
                        _value['ID'],
                        _imei,
```

```
                    _value['SmartHinge'])

        for _recovery in self.__connection.get_recovery():
            for _imei, _value in _recovery.items():
                if self.__connection.request(_imei, _value[
                    'elaboration']):
                    self.__repository.
                        update_recovery_sample(_value['
                        database_id'])
[...]
```

- Nel quadro delle iniziative intraprese dalla Commissione Europea a favore della trasformazione digitale delle imprese, Industria 4.0 ha aperto la strada alla fase Industria 5.0 o Collaborative Industry, ossia un modello di impresa che annettono prodotti software ad hoc aiutando i consumatori a cooperare efficacemente con le macchine. Necessità che il prodotto software SmartEDGE$^{4.0}$ da solo non riesce a garantire. La predisposizione di una infrastruttura interna al software permette ad applicazioni su misura di integrarsi con facilità allo SmartEDGE$^{4.0}$. Le applicazioni possono variare dall' applicazione su mobile per il controllo attivo della macchina, gli smart glasses; dispositivi che aiutano l'operatore a effettuare manutenzioni e controllo qualità a bordo macchina, servizi di videosorveglianza e molto altro. In aggiunta, i prodotti software presenti in azienda devono poter comunicare tra di loro attraverso canali bidirezionali, al fine di potersi scambiare informazioni velocemente e in modo trasparente. L'interconnessione dei prodotti Retuner® è il primo passo verso il cambio di paradigma in ottica Industria 5.0 fornendo un prodotto completo, virtualmente connesso e di facile utilizzo [16].

  La risposta all'esigenza del mercato ha portato l'introduzione del framework software Flask che integra delle librerie in python per la creazione di un web server locale dove è possibile riceve delle API da altri dispositivi. In questo modo le applicazioni terze possono comunicare con il prodotto

SmartEDGE$^{4.0}$ utilizzando il protocollo API REST. Il web server Flask rimane in attesa di richieste API in ascolto sulla porta 9011 dal MiniMES$^{4.0}$, SmartHINGE$^{4.0}$ e i servizi esterni; ovvero applicazioni implementate da terze parti che possono richiedere i dati elaborati della macchina tramite richieste API. Il modulo viene avviato dalla funzione init_flask() che, ricevute due code di processo, istanzia il web server Flask. Le code servono per comunicare in entrambe le direzioni con il chiamante, ovvero il software che avvia il processo di interconnessione 4.0. Le richieste API implementate sono le seguenti:

- POST '/api/configuration': Richiesta di modifica del file di configurazione delle macchine interconnesse. Questo permette di modificare dinamicamente le proprietà di configurazione dinamicamente;
- PUT '/api/service/<service>/machine/<imei>/write': Richiesta di scrittura dati su una macchina interconnessa. Questo permette a servizi terzi di poter alterare la configurazione della macchina tramite richiesta API;
- GET '/api/service/<service>/machine/<imei>/status': Richiesta di dati elaborati per un servizio esterno su una specifica macchina. Il sistema calcola runtime l'elaborazione richiesta con l'utlimo stato dei dati raccolti dalla macchina;
- POST '/api/web_hooks/machine/write: Invio di una ricetta da parte del MiniMes4.0 per una macchina. I dati sono contenuti all'interno del body.

5.1.1 Architettura software

Il processo di avvio del software parte dalla classe SmartEdgeNG, modulo che istanzia i processi e i thread necessari per gestire il flusso di esecuzione della raccolta del dato macchina fino all'avvio della sua elaborazione allo SmartHINGE$^{4.0}$. Entrando nel dettaglio, il modulo apre un file di configurazione storicizzato in locale che contiene le informazioni per processare adeguatamente le singole macchine interconnesse.

- Sampling: passo di campionamento;
- MiniMes, SmartHinge: indirizzo IP della macchina dove sono stati installati i software;
- configurations: configurazione delle singole macchine. Ogni macchina contiene i seguenti macro-blocchi;
  - machine: caratteristiche di configurazione per la connessione verso la macchina;
  - digital_service: equazioni matematiche per la fase di elaborazione dei dati;
  - connection: token di autorizzazione verso lo SmartHINGE$^{4.0}$.

Le informazioni si riassumono in questa struttura:

```
sampling: integer

MiniMes: str

SmartHinge: str

configurations:
  machine_imei:
    machine:
      protocol: str
      variables: dict
    digital_service:
      SmartHinge:
        elaboration: dict
        event: dict
      WebApp1:
        elaboration: dict
        event: dict
    connection:
```

```
SmartHinge:
  authorization: str
```

Verificati i parametri sul file, la procedura si occupa di istanziare i seguenti oggetti:

- Oggetto Protocol() che racchiude al suo interno tutte le configurazioni dei protocolli associati per ogni macchina interconnessa;
- Oggetto DigitalService() che racchiude al suo interno tutte le strutture dati necessarie per il calcolo delle equazioni matematiche. Gestisce anche le operazioni su richiesta dai servizi digitali quando essi lo richiedono;
- Oggetto Connection() racchiude al suo interno tutte le configurazioni degli url e autorizzazione delle macchine per l'invio delle elaborazioni dati verso lo SmartHINGE$^{4.0}$;
- Oggetto Repository() che si occupa di storicizzare i dati elaborati e grezzi con relativi stati dei dati ('STORED', 'NEW' e 'HINGE') dove:
  - STORED: il dato è stato salvato e non contiene informazioni nuove da inoltrare allo SmartHINGE$^{4.0}$;
  - NEW: il dato è stato salvato ma deve essere mandato verso lo SmartHINGE$^{4.0}$;
  - HINGE: il dato è stato mandato correttamente verso lo SmartHINGE$^{4.0}$.
- Un oggetto Logger(): si occupa di scrivere su file lo stato di lavoro e gli errori riscontrati durante l'esecuzione.

```
def __init__(self, out_queue, configuration_name: str):
    [...]
    self.__configuration = Configuration(
       configuration_name)
    self.__connection = Connection()
    self.__repository = Repository()
    self.__digital_service = DigitalService()
    self.__protocols = Protocol()
```

```
        self.__logger = Logger()

        for _key, configuration in
            self.__configuration.get_configuration_machines
                ()
                .items():
            self.__connection.insert_connection_machine
                (_key,
                configuration.get_connection()["SmartHinge"
                    ])
        self.__digital_service.insert_smart_hinge(
            _key,
            configuration.get_digital_service()["SmartHinge
                "])

            for _service, _conf in configuration.
                get_digital_service()
                .items():
                if _service != 'SmartHinge':
                    self.__digital_service.insert_service(
                        _key, _service, _conf)

            self.__protocols.insert_protocol(
                _key,
                configuration.get_machine()['protocol'])
            self.__imei_config[_key] =
                self.__repository.replace_configuration(
                    _key, self.__configuration.
                    get_configuration_machine(_key))
        [...]
```

Successivamente, il sistema si avvia dopo l'invocazione della funzione statica init() che istanzia flussi paralleli di esecuzione per distribuire omogeneamente il carico e

le funzionalità da computare. Nello specifico, vengono istanziati:

- Un processo per abilitare il web server;
- Un processo per il campionamento dei dati macchina;
- Un thread per l'analisi del dato ricevuto dal processo di campionamento;
- Un thread per il calcolo della logica e il salvataggio del dato elaborato sul repository;
- Un thread per l'invio del dato verso la SmartHINGE$^{4.0}$ e l'aggiornamento del dato sul repository.

La funzione rimane in attesa di richieste esterne che il processo web_server inoltrerà tramite una coda di processo. Il web server gestisce le seguenti richieste:

- WEB HOOKS: operazione di scrittura di una ricetta ricevuta dal MiniMES$^{4.0}$ per alterare i parametri della macchina;
- STATUS: operazione richiesta da parte di un servizio esterno per recuperare l'elaborazione dei dati grezzi di una macchina di produzione. Il sistema calcola in runtime l'elaborazione specifica del servizio prendendo i dati più recenti della macchina e li restituisce, tramite code dei processi, al web server che lo invierà come risposta alla richiesta API;
- SERVICE: operazione di lettura e scrittura dei dati che un servizio esterno richiede.

```
@staticmethod
def init(configuration_name: str = ''):
    [...]
    execution_flows = [
        Process(target=init_flask, args=(flask_channel,
            flask_request)),
        Process(target=edge_software._sampler,
                args=(
                    out_queue,
                    in_queue,
```

```
                    edge_software.__configuration.
                        get_time_sampling(),
                    edge_software.get_protocols())),
        Thread(target=edge_software._analyzer, args=()),
        Thread(target=edge_software._repository, args=()),
        Thread(target=edge_software._connection, args=())
    ]

    for flow in execution_flows:
        flow.start()

    information = flask_channel.get(timeout=10)

    if information['type'] == 'WRITE':
        in_queue.put_nowait(
            {"operation": "WRITE", "imei": information['key
                '], "value": information['value']})
    if information['type'] == 'STATUS':
        flask_request.put_nowait(
            edge_software.__digital_service.
                status_elaborating(information['key'],
                information['service']))
    if information['type'] == 'WEB_HOOK':
        # Cambia il valore della variabile ricetta
        edge_software.__digital_service.new_recipe(
            information['key'])
        in_queue.put_nowait(
            {"operation": "WRITE", "imei": information['key
                '], "value": information['value']})
    [...]
```

In seguito, verranno analizzati tutti gli step del flusso di processamento del dato grezzo partendo dalla raccolta sulla macchina fino all'invio dell'elaborato verso lo

SmartHINGE$^{4.0}$.

- _sampler: Il processo di campionamento dati macchina viene istanziato come tale per slegare la fase di raccolta del dato della fase di analisi, salvataggio e invio del dato. Il processo di campionamento dati macchina si occupa di istanziare la lista delle macchine interconnesse associandole il protocollo di comunicazione, le variabili da estrapolare e il tempo di raccolta del dato. Il processo ripete la procedura secondo un passo di campionamento definito dall'utente nel file di configurazione. Questo modulo permette quindi di slegare il software con il protocollo di comunicazione rendendo le due parti indipendenti tra loro. L'informazione estrapolata viene inoltrata al modulo successivo attraverso una coda di processo. In caso di anomalie di comunicazione, il modulo lo gestisce fornendo due stati di sistema secondo il tipo di errore riscontrato: la macchina non è interconnessa in rete oppure la macchina non risponde alle richieste di comunicazione. Entrambi i casi vengono inoltrati tramite la coda.

- _analyzer: Per gestire efficacemente l'analisi del dato grezzo ricevuto dalla coda dei processi viene istanziato un thread che, ricevuto il dato, applica una logica dinamica ricercando eventuali anomalie macchina segnalandole tempestivamente al cliente finale. Viene svolta, dunque, una manutenzione predittiva a posteriore incentrata sull'analisi delle variabili che la macchina fornisce. In parallelo, il sistema predispone le variabili raccolte al modulo successivo in modo che esso possa analizzarle accuratamente estrapolando dati nuovi e significativi.

  In caso di problemi macchina dovuti ai casi elencati sopra, il sistema utilizza un timer per segnalare eventuali anomalie macchina dopo tre minuti in modo da differenziare eventuali problemi dovuti a problemi di connessione alla rete o problemi macchina.

  Il thread rimane in attesa di nuovi dati e ripete il ciclo di esecuzione sopra descritto.

- _repository: Dopo una prima elaborazione del dato, viene eseguito il modulo

denominato Servizi digitali di prossimità che, ricevuti i dati in ingresso e le equazioni da calcolare, ritorna i risultati desiderati. Questo modulo permette di rendere totalmente indipendente il software e le operazioni di logica che si desiderano calcolare. Inoltre, il software è in grado di riprogrammare le operazioni di logica senza terminare la sua esecuzione o ricompilarlo per gestire la nuova configurazione. Infine, ottenuto il risultato del modulo, occorre scrivere su database sia il dato grezzo sia il dato elaborato in modo da garantire la persistenza dei dati e l'invio di quest'ultimi in caso di anomalie di connessione.

Il ciclo termina inviando al modulo successivo l'elaborazione ottenuta.

- _connection: L'ultimo step del processo di gestione del dato macchina termina con il thread Connection. Questo flusso si occupa di recuperare l'elaborazione dello step precedente, aprire la connessione verso lo SmartHINGE$^{4.0}$ e inviare i dati elaborati della macchina. L'entry-point e il token di autenticazione cambiano a seconda della macchina. Il modulo, pertanto, associa l'autenticazione della macchina con il dato elaborato associato in modo da garantire l'invio corretto dell'informazione. Prima di terminare il ciclo effettua un aggiornamento dello stato del dato sul repository. In particolare, a seconda del risultato riportato dallo SmartHINGE$^{4.0}$:
  - ACK 200: lo SmartHINGE$^{4.0}$ ha ricevuto correttamente i dati. Viene alterata la colonna status presente sul repository associato al record in HINGE
  - Error o timeout: lo SmartHINGE$^{4.0}$ non è riuscito a salvare i dati. In questo caso viene aggiunto un nuovo record nella tabella Repository contenente l'identificativo del record del campione da ritrasmettere appena la connessione viene ripristinata
- web_server: All'interno del prodotto software viene integrato un web server necessario per interagire con il software di interconnessione 4.0 e poterlo riprogrammare e configurare il software dinamicamente. Permette inoltre di rimanere in ascolto di richieste esterne di applicazione terze che necessitano

informazioni rapide ed elaborate delle macchine. Permette di gestire i web hooks con il MiniMES$^{4.0}$.

5.1.2 Sicurezza

Il Global Threat Intelligence Report 2021 di NTT ha evidenziato come gli hacker stiano traendo vantaggio dall'attuale clima di instabilità sfruttando le vulnerabilità comuni emerse con il passaggio al lavoro remoto. Con la pandemia gli attacchi hanno ridisegnato le loro tattiche con un incremento del 88% degli esperti di IT security a segnalare il netto incremento dei rischi fra 2020 a 2021. In crescita, in particolare, le violazioni alla componente applicativa e alle applicazioni web, che rappresentano il 67% di tutte le intrusioni registrate, più del doppio degli ultimi due anni. Sin d'ora è possibile dare un'idea delle dimensioni del fenomeno ricordando che il manufacturing è balzato al secondo posto, dietro al mondo della finanza e precedendo la Sanità, per il numero complessivo di intrusioni subite nell'ultimo biennio.

Riportando la citazione di Massimo Tambani a Il Progettista Industriale in security sales specialist di NTT Italia:

> "Soprattutto presso le PMI ogni azione deve essere compresa e ragionata partendo dal presupposto che nell'era dell'economia globale tutti siamo visibili agli attaccanti, sia che questi colpiscano con il ransomware sia fermando sistemi e attività. I dati hanno un'importanza elevata per qualsiasi tipo di industria e funzione: dalla produzione alla logistica. I cybercriminali sanno come e dove colpire per fare male traendo dalle loro azioni il maggior profitto possibile. Nessuno può dirsi a tutti gli effetti fuori bersaglio e solo accettando un simile assunto si riuscirà a tutelare il business di un'impresa, indipendentemente dalla sua entità."
>
> "La manifattura ha assegnato un ruolo centrale alla gestione e al controllo delle macchine, cioè dei sistemi produttivi che costituiscono il cuore della operatività. Da qualche anno a questa parte l'accento si è fortemente spostato sul dato e sul valore, perché si è visto che da que-

> sti dipende il monitoraggio delle attività aziendali nella loro totalità, compresa la logistica."

Questa nuova realtà vede la nascita di nuove criticità alla sicurezza informatica, alla necessità di ridondare i sistemi di difesa fisica, protezione dei dati, analisi dei punti deboli di una architettura di rete e delle relative comunicazioni IIoT; per indentificare le eventuali vulnerabilità e stabilire di conseguenza le contromisure più efficienti [16].

Una prima misura importante adottata da Orchestra è stata quella di installare i prodotti Retuner® di Orchestra direttamente sul server aziendale in modo da collezionare e rendere private le informazioni dell'azienda. Questa scelta riduce il rischio di attacchi DDoS verso il server in quando questo possiede un indirizzo IP privato. In aggiunta eventuali attacchi informatici dovranno partire da dispositivi tecnologici presenti e interconnessi nella rete aziendale.

Nello sviluppo del software SmartEDGE$^{4.0}$ sono state riscontrate delle possibili vulnerabilità di sicurezza e adottate delle procedure implementative per creare limitazioni di accesso, maggiori controlli e/o sistemi di autenticazione. Nello specifico:

- Molte macchine di produzione sono prive di sistemi di autenticazione e non presentano segreti o informazioni che limitano le operazioni macchina. Questa problematica genera una criticità in quanto l'unica verifica possibile per riconoscere la macchina è quella dell'indirizzo IP privato e statico. Un prodotto hardware e/o software potrebbe sostituirsi alla macchina e inviare informazioni errate sostituendosi alla macchina.
- Il software SmartEDGE$^{4.0}$ utilizza un file di configurazione per avviare l'interconnessione con le macchine di produzione ricevuto dal software SmartHINGE$^{4.0}$. Un attaccante potrebbe alterarne il contenuto abilitando un servizio aggiuntivo non autorizzato per forzare la scrittura di dati in macchina. Questa vulnerabilità è stata risolta attraverso un sistema di autenticazione per accedere all'elettronica SmartEDGE$^{4.0}$ e la cifratura del file di configurazione.

- La funzionalità di lettura e scrittura dati in macchina presenta una vulnerabilità grave che può portare a danneggiare la macchina situata nello stabilimento del cliente. Nello specifico un attaccante può fingersi un servizio esterno per richiedere la scrittura di alcuni parametri in macchina. Per annullare ciò, il software SmartHINGE$^{4.0}$ si occupa di inviare la configurazione delle macchine con relativi servizi aggiuntivi che possono accedere ai dati macchina. Ogni servizio deve autenticarsi fornendo un segreto in modo da garantire che solo i software autorizzati possono leggere e/o scrivere informazioni macchina.
- L'introduzione di un web server permette a software esterni di effettuare delle API. Alcune di queste non possono essere effettuate da nessun software tranne per i prodotti Orchestra e Technophylla. Nello specifico sono API per l'invio di una nuova configurazione o l'abilitazione di un servizio software esterno. Per evitare che un software non autorizzato possa eseguire queste API, la tipologia di account abilitato è 'ADMIN'. Per tutte le altre tipologie verrà restituito un errore 403; utente non autorizzato.

5.1.3 Test

Intenderemo per test una porzione di codice atta a verificare in modo automatico che il codice da consegnare al committente funzioni come preventivato. Inserire la scrittura di test all'interno del flusso produttivo è una delle buone pratiche più essenziali e fruttuose, e risulta più importante anche dei design pattern, nonostante il ruolo fondamentale che hanno nella modellazione degli oggetti e per le soluzioni informatiche più evolute. Le ragioni di questa importanza sono sostanzialmente due: la prima è che solo con i test è possibile vedere se il codice effettivamente funziona, la seconda ragione è che la sola intenzione di testare un codice porta alla migliore scrittura possibile dello stesso, perché un codice che nasce con una struttura tale da poter essere testato significa, quasi sempre, aver applicato il pattern corretto nel progetto.

I test adottati all'interno del progetto ricoprono la verifica del codice e dell'architettura ma anche i test prestazionali del ciclo di elaborazione del singolo dato

raccolto.

- Test prestazionali: attraverso un dataset di dati estrapolati dalla macchina, sono stati analizzati i tempi di processamento del dato grezzo con lo scopo di cercare il passo di campionamento adeguato a campionare la macchina di produzione. I valori raccolti non includono i tempi di connessione verso la macchina e il software SmartHINGE$^{4.0}$ perché variano a seconda della connessione e dal carico della rete. Questo permette di capire il tempo effettivo di lavorazione del dato impostando così un passo di campionamento minimo oltre al quale non è possibile scendere.

  Dall'analisi sono stato ottenuti i seguenti risultati:

  - Tempo massimo: 66.672 ms
  - Tempo minimo: 10.995 ms
  - Tempo medio: 31.467 ms
  - Deviazione standard: 6.692

  Dai valori ottenuti osserviamo dalla figura 5.3 che l'affluenza dei tempi computazionali si concentrano tra i 10 e i 60 millisecondi mentre l'affluenza media del dato si aggira intorno ai 30 millisecondi. Introducendo un margine sul risultato, il tempo minimo necessario per processare il dato si aggira sui 100 ms.

  Per avere una migliore analisi sui tempi computazionali raccolti, i dati sono stati inseriti in un istogramma (vedi figura 5.4); un modello grafico utilizzato per rappresentare la distribuzione di frequenza di alcuni punti dati di una variabile.

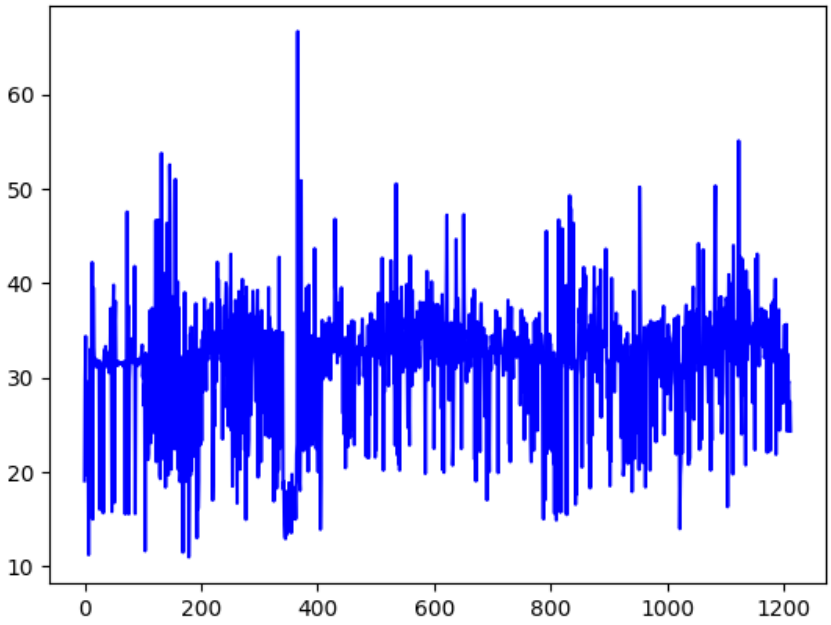


Figura 5.3: Plot dei tempi di computazione

Utilizzando un passo di campionamento che varia mediamente dai 2 ai 5 secondi, i valori dei tempi massimi e minimi ottenuti soddisfano le specifiche iniziali prefissate;

- Test architetturale: per essere tale, un test deve rispettare alcune condizioni quali:
  - Un test non deve dipendere da input esterni, ad esempio non sono accettabili test che richiedono un input all'utente o test che leggono l'ora corrente del sistema;
  - Il codice dei test deve essere strutturalmente semplice: non può contenere né iterazioni né cicli;
  - Il risultato dei test deve essere stabile; che venga eseguito cento o un milione di volte in qualunque momento: il test potrà dare esito positivo

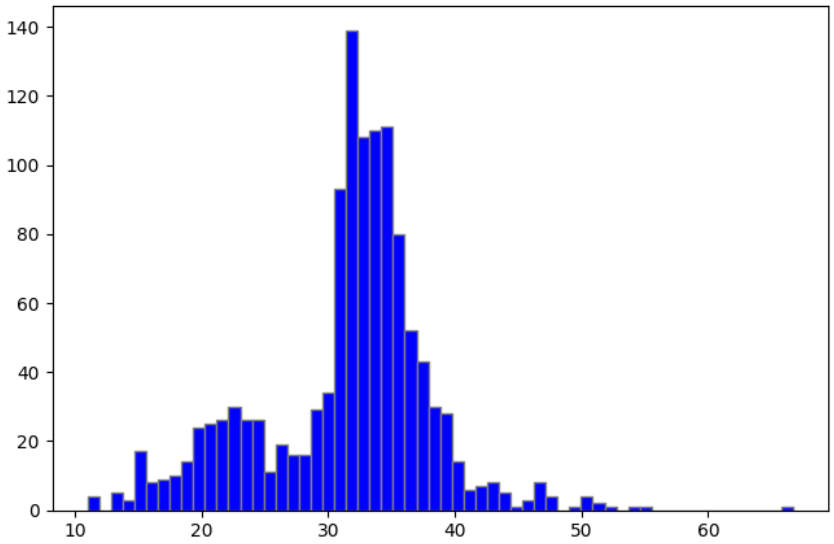


Figura 5.4: Istogramma dei tempi di computazione

o negativo ma tale risultato deve rimanere stabile e non dipendere da altri fattori.

In seguito, verrà mostrato il direttorio dei test di sistema e alcuni frammenti significativi di codice.

- test
  - configurarion
    - test_configuration.py
  - connection
    - test_connection.py
  - digital_service
    - test_digital_service.py

```
class TestNoneDigitalService(unittest.
    TestCase):
```

```
        def test_error_arguments(self):
            try:
                self.digital_service =
                    DigitalService()
                self.digital_service.
                    insert_smart_hinge(None, None)

                self.assertFalse("Exception not
                    generated")
            except ErrorInitArguments as error:
                pass
            except Exception:
                self.assertFalse("Exception class
                    incorrect")


class TestDigitalServiceMiniMes(unittest.
    TestCase):
    def setUp(self):
        self.digital_service = DigitalService
            ()
        self.digital_service.
            insert_smart_hinge('MACHINE01', {
            "elaboration": {
                "mode_event_01": {
                    "mode": "oneTime#cast;int
                        0,onChange",
                    "value": "if;
                        $MiniMes_recipe =f= 2;
                        T;cast;int 0;F;eq;
                        $mode_event_01 + (
                        $new.val_01 - $old.
```

```
                    val_01 )",
                },
                "mode_event_02": {
                    "mode": "oneTime#cast;int
                        0,event#eq;(
                        $MiniMes_recipe=f=0
                        )AND(($new.val_01
                        -$old.val_01)>0),
                        onChange",
                    "value": "eq;
                        $mode_event_02+(
                        $new.val_01-$old.
                        val_01)",
                },
            },
            "event": {}
        })

    def test_counter_object(self):
        sample = {'val_01': 0}

        result = self.digital_service.
            elaborations('MACHINE01', sample)

        self.assertEqual(len(result.items()),
            0)

        sample = {'val_01': 0}
        result = self.digital_service.
            elaborations('MACHINE01', sample)[
            'SmartHinge']
```

```
        self.assertEqual(len(result.items()),
            1)
        self.assertEqual(result.get("
            mode_event_01"), 0)

        sample = {'val_01': 1}
        result = self.digital_service.
            elaborations('MACHINE01', sample)[
            'SmartHinge']

        self.assertEqual(len(result.items()),
            2)
        self.assertEqual(result.get("
            mode_event_01"), 1)
        self.assertEqual(result.get("
            mode_event_02"), 1)

        sample = {'val_01': 1}
        result = self.digital_service.
            elaborations('MACHINE01', sample)

        self.assertEqual(len(result.items()),
            0)

        sample = {'val_01': 3}
        result = self.digital_service.
            elaborations('MACHINE01', sample)[
            'SmartHinge']

        self.assertEqual(len(result.items()),
            2)
        self.assertEqual(result.get("
```

```
            mode_event_01"), 3)
        self.assertEqual(result.get("
            mode_event_02"), 3)

        self.digital_service.new_recipe('
            MACHINE01')

        sample = {'val_01': 3}
        result = self.digital_service.
            elaborations('MACHINE01', sample)[
            'SmartHinge']

        self.assertEqual(len(result.items()),
             1)
        self.assertEqual(result.get("
            mode_event_01"), 0)

        sample = {'val_01': 4}
        result = self.digital_service.
            elaborations('MACHINE01', sample)[
            'SmartHinge']

        self.assertEqual(len(result.items()),
             2)
        self.assertEqual(result.get("
            mode_event_01"), 1)
        self.assertEqual(result.get("
            mode_event_02"), 4)
```

- test_digital_service_cast.py
- test_digital_service_eq.py
- test_digital_service_if.py

- test_digital_service_list.py
- test_digital_service_mode_event.py
- test_digital_service_mode_never.py
- test_digital_service_mode_oneTime.py
- test_digital_service_mode_reset.py
- test_digital_service_mode_msg.py
- test_digital_service_mode_pow.py
- test_digital_service_mode_sqrt.py

* logger
  - test_logger.py
* protocol
  - test_ModbusBarreUrbina.py
* repository
  - test_database.py

```
class TestSystemArchitecture(unittest.
    TestCase):
    def setUp(self) -> None:
        self.database = Database('
            machine_support.db')

    def tearDown(self):
        self.database.remove_db()

    def test_architecture(self):
        SmartEdgeNG.init(configuration_name='
            smart_edge/incorrect_conf.yml')
```

```
            self.assertEqual(self.database.
                get_number_records(), 6)
            self.assertEqual(len(self.database.
                get_recovery_samples()), 6)

            SmartEdgeNG.init(configuration_name='
                smart_edge/correct_conf.yml')

            self.assertEqual(self.database.
                get_number_records(), 12)
            self.assertEqual(len(self.database.
                get_recovery_samples()), 0)

        def test_architecture_smart_hinge(self):
            SmartEdgeNG.init(configuration_name='
                smart_edge/no_auth_smart_hinge.yml
                ')

            self.assertEqual(self.database.
                get_number_records(), 6)
            self.assertEqual(len(self.database.
                get_recovery_samples()), 6)

            SmartEdgeNG.init(configuration_name='
                smart_edge/correct_conf.yml')

            self.assertEqual(self.database.
                get_number_records(), 12)
            self.assertEqual(len(self.database.
                get_recovery_samples()), 0)
```

· test_database_dimension.py

- test_database_recovery.py

* smart_edge

  - test_architecture.py

```
class TestSystemArchitecture(unittest.
    TestCase):
    def setUp(self) -> None:
        self.database = Database('
            machine_support.db')

    def tearDown(self):
        self.database.remove_db()

    def test_architecture(self):
        SmartEdgeNG.init(configuration_name='
            smart_edge/incorrect_conf.yml')

        self.assertEqual(self.database.
            get_number_records(), 6)
        self.assertEqual(len(self.database.
            get_recovery_samples()), 6)

        SmartEdgeNG.init(configuration_name='
            smart_edge/correct_conf.yml')

        self.assertEqual(self.database.
            get_number_records(), 12)
        self.assertEqual(len(self.database.
            get_recovery_samples()), 0)

    def test_architecture_smart_hinge(self):
        SmartEdgeNG.init(configuration_name='
```

```
                smart_edge/no_auth_smart_hinge.yml
                ')

            self.assertEqual(self.database.
                get_number_records(), 6)
            self.assertEqual(len(self.database.
                get_recovery_samples()), 6)

            SmartEdgeNG.init(configuration_name='
                smart_edge/correct_conf.yml')

            self.assertEqual(self.database.
                get_number_records(), 12)
            self.assertEqual(len(self.database.
                get_recovery_samples()), 0)
```

* web_server
  - test_flask.py
* __main__.py

```
import unittest

if __name__ == '__main__':
    loader = unittest.TestLoader()
    directories = [
        'configuration',
        'connection',
        'digital_service',
        'logger',
        'protocol',
        'repository',
        'web_server',
        'smart_edge'
```

```
]

for directory in directories:
    unittest.TextTestRunner().run(unittest.
        TestLoader().discover(directory))
```

5.2 Report

L'ultimo aspetto per completate la fase di rilascio del prodotto software SmartEDGE$^{4.0}$ sono stati i test sui clienti possessori delle macchine di produzione. Assieme a loro, è stata utilizzata l'elettronica e il software SmartEDGE$^{4.0}$ interconnesso ai macchinari. Il risultato ottenuto è stata l'interconnessione di una macchina che comunica tramite protocollo OPC-UA.

I dati in oggetto prevedono il calcolo dello stato della macchina, la quantità di prodotti generati e scartati, la temperatura della macchina e la velocità di lavorazione e si basano su una giornata di lavoro. Verranno mostrati in seguito i grafici.

- Calcolo della quantità di pezzi giornalieri: il grafico in figura mostra il numero di prodotti giornalieri dell'intera giornata lavorativa;

Figura 5.5: Calcolo della quantità di pezzi prodotti

- Calcolo dei pezzi scartati: il grafico in figura mostra il numero di pezzi scartati dalla macchina a causa di deformazioni del prodotto finale;

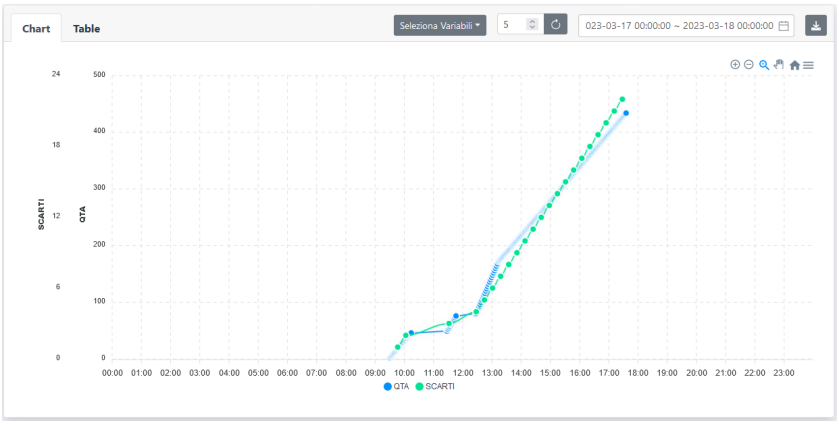


Figura 5.6: Calcolo della quantità di pezzi scartati

- Stato della macchina: il grafico in figura mostra le variazioni di stato macchina avvenuta durante la giornata lavorativa;

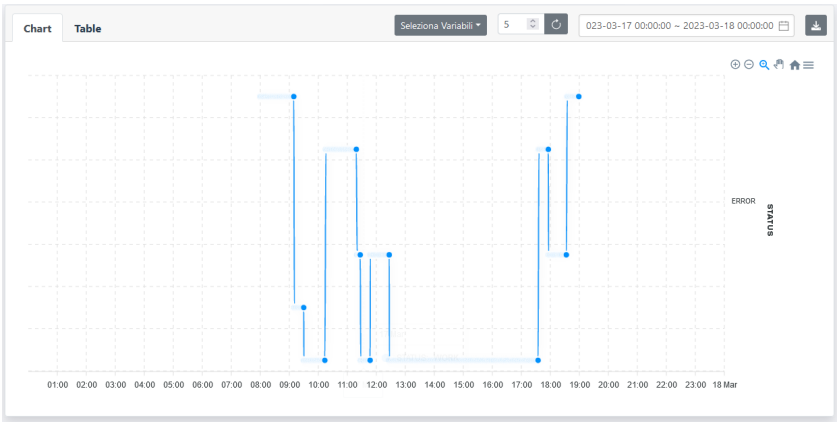


Figura 5.7: Stato della macchina

- Temperatura: il grafico in figura mostra le variazioni di temperatura avvenute durante la giornata lavorativa;

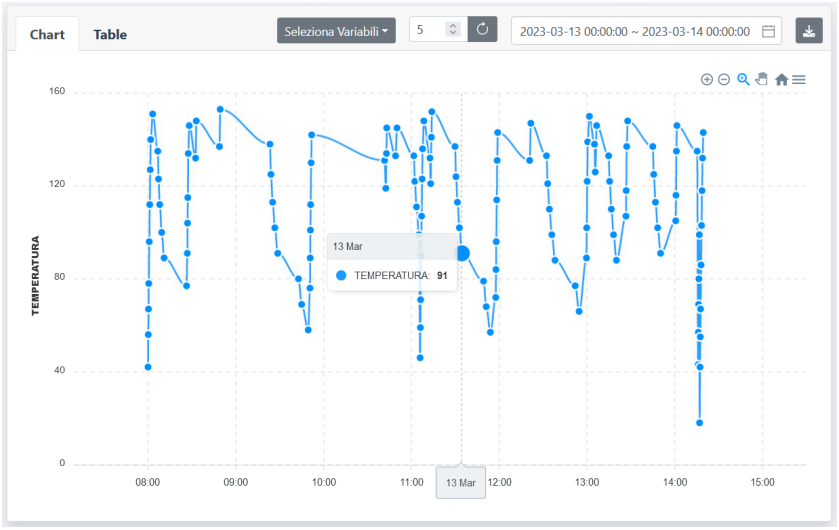


Figura 5.8: Temperatura lavorazione della macchina

- Velocità: il grafico in figura mostra le variazioni di velocità avvenute durante la giornata lavorativa;

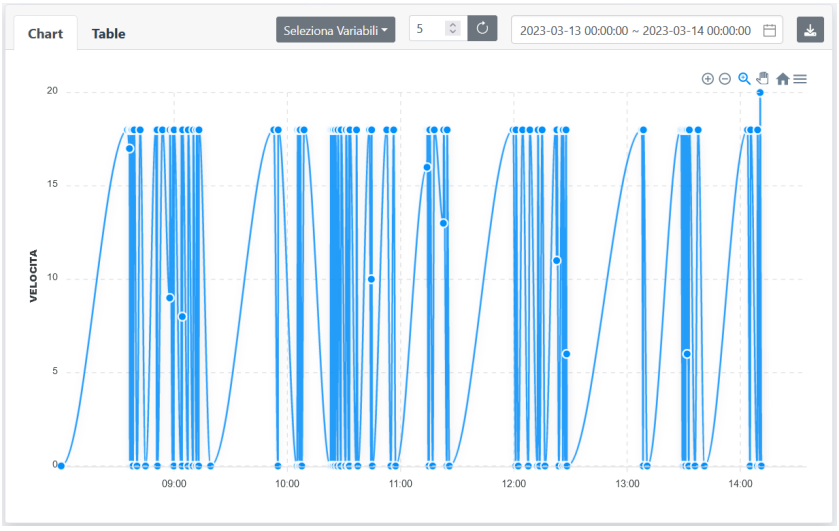


Figura 5.9: Velocità lavorazione della macchina

Capitolo 6

Servizi aggiuntivi

Rispetto a Industria 4.0, l'Industria 5.0 sarà però una Collaborative Industry con fine ultimo di rafforzare i sistemi informatici delle PMI e introdure nuove applicazioni che aiutino il consumatore a monitorare e controllare più facilmente le macchine industriali. Al centro del nuovo modello di produzione industriale ci sono:

- Umanocentrico: gli esseri umani sono al centro dei processi di produzione;
- Sostenibilità: abilitazione dei modelli di economia circolare e di efficienza energetica
- Resilienza: obiettivo di sviluppare un più alto grado di robustezza nella produzione e delle infrastrutture critiche

L'industria 5.0 persegue lo sviluppo di una società in cui è protagonista la cooperazione intelligente di macchine ed operatori umani: protagonisti sono i cobot e le applicazioni software intelligenti detti bot. I cobot sono robot collaborativi capaci di interagire con gli esseri umani in realtà lavorative condivise. I bot, con l'intelligenza artificiale, sono in grado di agire per un utente o per un altro programma con una interazione collaborativa. La nobilitazione dell'azione dell'operatore umano nella produzione e il suo valore nella definizione di qualità e personalizzazione del prodotto collocano il ruolo dell'operatore umano in una posizione fondamen-

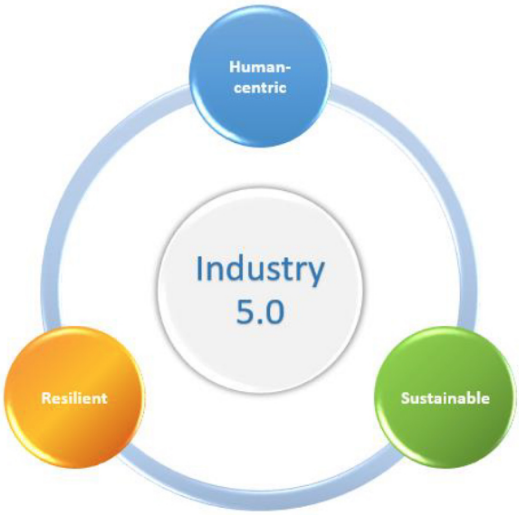


Figura 6.1: modello di produzione industriale [20]

tale dello schema Industria 5.0; un aspetto fondamentale per entrare nella visone di Industria 5.0 consiste nella produzione personalizzata di prodotti software personalizzati realizzati appositamente per rispondere alle esigenze del cliente [20].

I servizi aggiuntivi sono la nuova funzionalità introdotta nel software SmartEDGE$^{4.0}$ che cerca di rispondere alle richieste della nuova rivoluzione industriale; aspetto molto importante in quanto molte macchine di produzione permettono l'attivazione di un solo canale di comunicazione per volta. Il prodotto SmartEDGE$^{4.0}$ diventerebbe un centro stella in grado di fungere da tramite tra le macchine e l'applicazione software attraverso una comunicazione in RESTFUL API.

Verrà in seguito descritto un servizio applicativo interconnesso con il software SmartEDGE$^{4.0}$.

6.1 Web app Smart Farming

Orchestra, in collaborazione con Technophylla, azienda partner per l'Agricoltura 4.0, ha sviluppato un'applicazione su misura per manovrare i movimenti della

macchina direttamente dal device. Nello specifico, è stata realizzata un'applicazione web installata sull'elettronica SmartEDGE$^{4.0}$ e che deve interagire con il software SmartEDGE$^{4.0}$ per la lettura dei dati macchina. L'accesso all'applicazione avverrà esclusivamente tramite l'accesso alla rete aziendale. All'avvio, il cliente dovrà autenticarsi attraverso username e password. Questo permette l'accesso al software ai soli utenti autorizzati.

Figura 6.2: Interfaccia di autenticazione

Il risultato ottenuto fornisce al cliente di gestire le seguenti modalità di lavorazione:

- Modalità manuale: questa modalità permette all'agricoltore di muovere manualmente la macchina di irrigazione e innaffiare le piante. Una volta definita la velocità di movimentazione, l'impianto si muove in base alla direzione abilitata dal device. Durante la movimentazione, l'utente attiva le elettrovalvole per iniziare l'irrigazione.

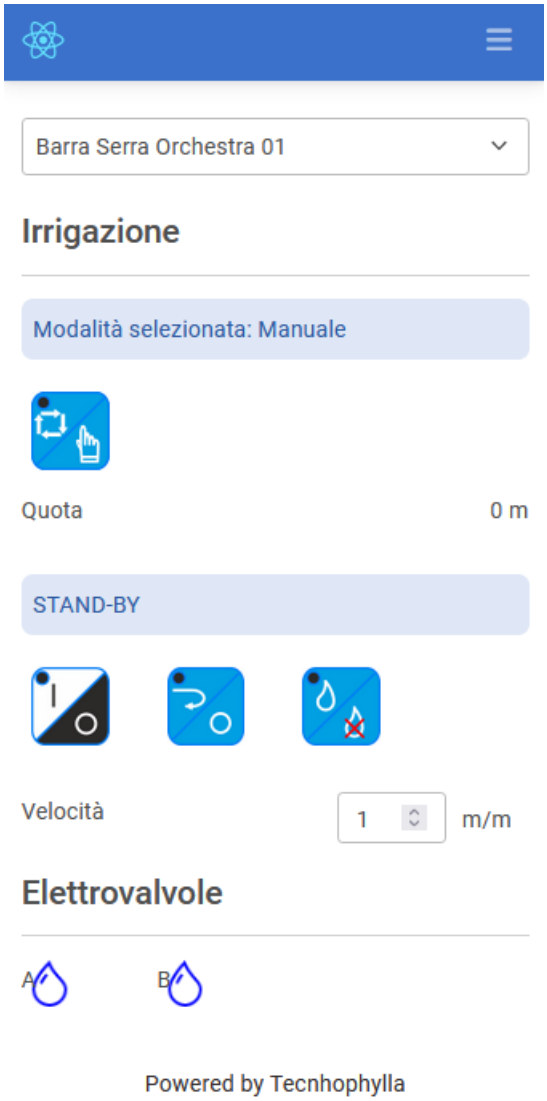


Figura 6.3: Modalità manuale

- Modalità automatica: questa modalità permette all'agricoltore di avviare automaticamente la macchina di irrigazione impostando due parametri aggiuntivi: il numero di cicli di irrigazione e i secondi di scarico; utilizzato soprattutto quando l'acqua immagazzinata è calda o da cambiare.

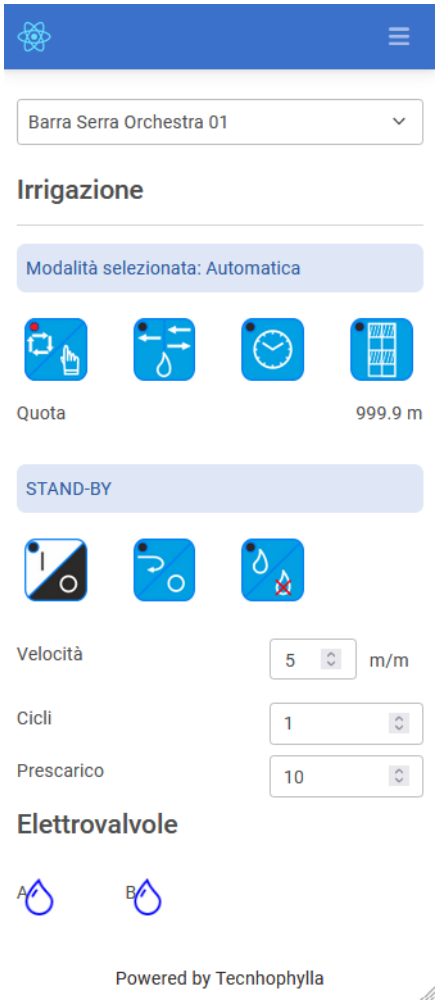


Figura 6.4: Modalità automatica

- Modalità automatica a zone: questa modalità permette all'agricoltore di avviare automaticamente la macchina impostando il programma a zone da avviare, il numero di cicli e lo scarico.

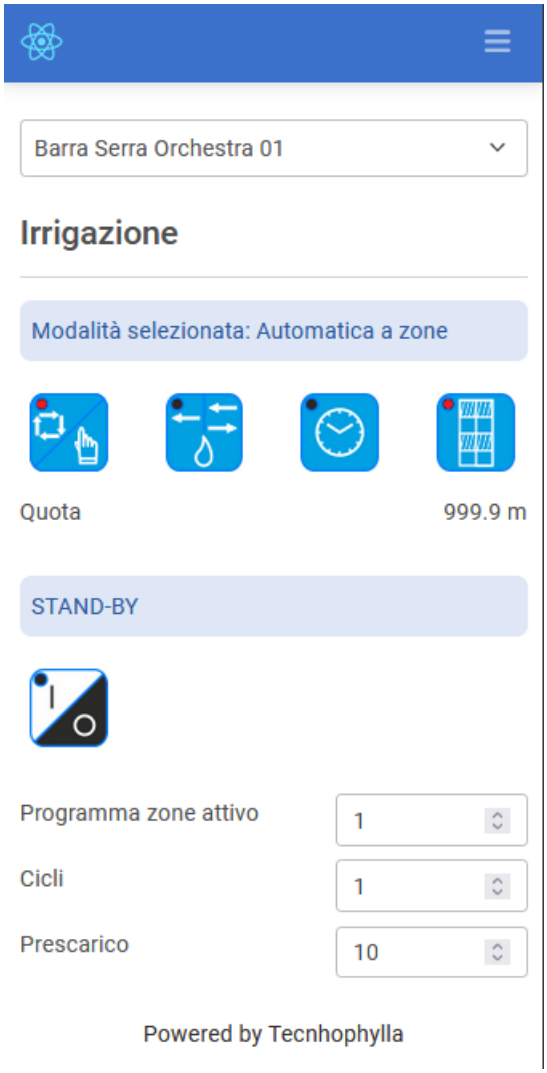


Figura 6.5: Modalità automatica a zone

- Modalità programmazione oraria: questa modalità permette all'agricoltore di impostare fino a 40 programmi di irrigazione con i seguenti parametri:
  - Abilitazione: parametro per attivare o disattivare il programma
  - Ora: orario di attivazione del programma
  - Velocità: velocità di irrigazione. Il valore è espresso in metri su minuti
  - Cicli: numero di iterazioni che la macchina dovrà eseguire

– Elettrovalvole: icone per abilitare una parte di irrigazione

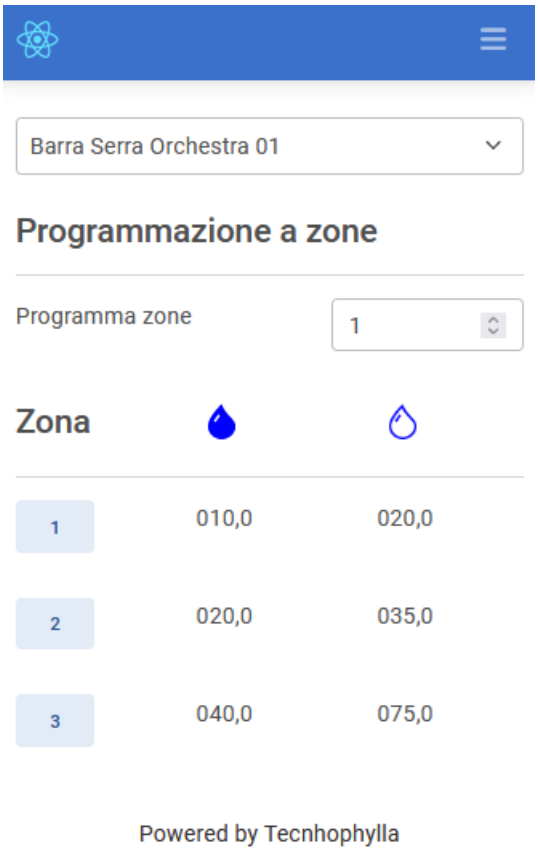


Figura 6.6: Modalità programmazione oraria

La configurazione mostrata all'utente coinciderà con quella presente nella centralina. Per avere questa sincronizzazione, l'applicazione web richiede periodicamente lo stato della macchina e invia le operazioni richieste dall'utente all'elettronica SmartEDGE$^{4.0}$ tramite apposita API. Nello specifico:

- PUT '/api/service/<service>/machine/<imei>/write': invia i nuovi parametri macchina da inserire
- GET '/api/service/<service>/machine/<imei>/status': legge lo stato attuale della macchina

Capitolo 7

Conclusioni e lavoro futuro

Orchestra è una PMI innovativa nata per servire il mercato dell'industria 4.0 fornendo soluzioni digitali con il proprio brand Retuner® alle piccole e medie imprese del settore manifatturiero, in particolare, per riuscire a interconnettere le officine e le macchine con i sistemi informatici di cui l'azienda già dispone. Lo SmartEDGE$^{4.0}$ NG installato in campo in officina raccoglie i dati da qualsiasi macchinario, apparato, sensore nuovo o esistente che, accessibile da qualsiasi apparecchio mobile o fisso, monitorano in continuo lo stato dei vari sistemi. Il passo successivo prevederà la migrazione verso la quinta rivoluzione industriale portando nuovi software ad hoc per il cliente per controllare e programmare in realtime la macchina attraverso dispositivi mobile o fisso. Questo sarà possibile attraverso l'architettura SmartEDGE$^{4.0}$ che esporta, su richiesta, i dati elaborati della macchina. Per evitare di far installare tutte le applicazioni sui dispositivi dei clienti, lo SmartEDGE$^{4.0}$ integrerà una web app dove si porterà navigare facilmente nelle varia applicazioni senza intasare il dispositivo del cliente di piccole applicazioni per controllare le differenti applicazioni (vedi figura 7.1)

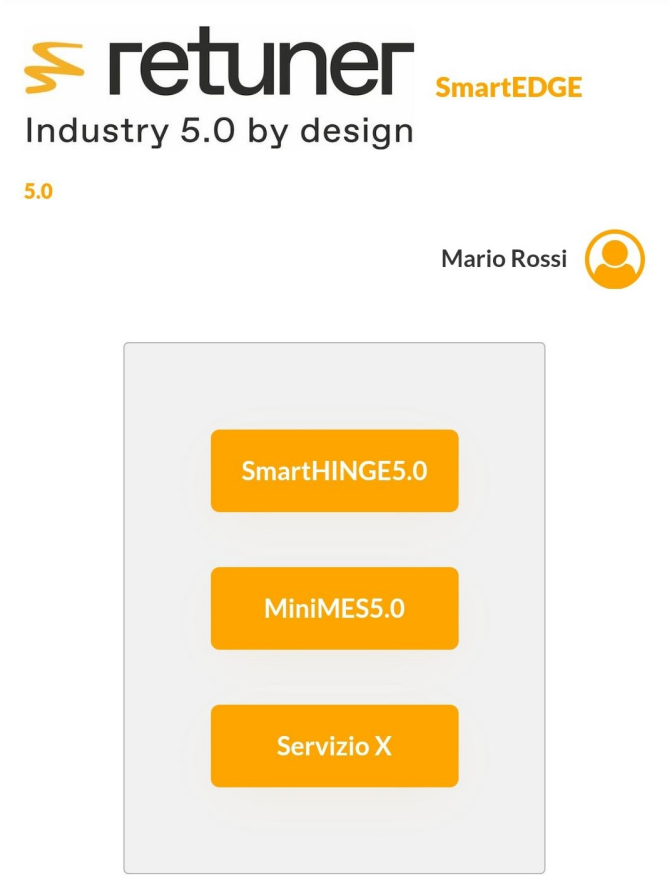


Figura 7.1: Web app SmartEDGE$^{5.0}$NG

Successivamente, avviata la fase di installazione su larga scala del prodotto SmartEDGE$^{4.0}$, verranno effettuati analisi sui dati raccolti per le singole macchine in modo da lavorare assieme ai produttori di macchinari per processare i dati elaborati alla ricerca di fenomeni esogeni. Questo apre la strada per la gestione della manutenzione predittiva della macchina e al controllo qualità del prodotto finale.

7.1 Ringraziamenti

Vorrei dedicare qualche riga a tutti coloro che mi sono stati vicini in questo percorso di crescita personale e professionale.

In primis, un ringraziamento speciale al mio relatore Malnati, per la sua immensa pazienza, per i suoi indispensabili consigli, per le conoscenze trasmesse durante tutto il percorso di stesura dell'elaborato. Ringrazio anche tutti i professori del Politecnico di Torino che, con i loro corsi didattici, mi hanno permesso di raggiungere questo traguardo.

Senza il supporto morale dei miei genitori, non sarei mai potuto arrivare fino a qui. Grazie mamma per avermi sempre supportato e per tutte le serate passate al telefono dove mi confortavi, mi motivavi ad andare avanti e per non aver mai dubitato delle mie possibilità. Grazie papà per avermi trasmesso la determinazione di non mollare mai e di credere sempre in me stesso. Ringrazio mio fratello Alessio per avermi accompagnato durate i viaggi in treno verso Torino e per la pazienza che hai sempre avuto nei miei confronti. Grazie anche a mio fratello Edoardo per vantarsi con i compagni in classe di avere un fratello grande che studia ingegneria; spero che anche tu possa intraprendere questo percorso con successo.

Un caloroso ringraziamento va anche ai compagni di corso di studio, e a coloro con cui ho legato in questi anni; amici che hanno sempre creduto in me. Ringrazio tutti loro per avermi sostenuto durante le sessioni di studio, per avermi bacchettato quando non studiavo e per tutte le serate oltre le mura del Politecnico di Torino; in particolare, voglio ringraziare Mattia, Luca, Giuseppe, Tony, Alessia e Alessandra per le giornate passate assieme, ringrazio Francesco per questi cinque anni di studi dove ci siamo anche divertiti.

Un ringraziamento va anche ai colleghi di Orchestra che mi hanno accompagnato in questo percorso lavorativo, per il caloroso affetto ricevuto in questi sei mesi di tirocinio. Ringrazio i relatori Ferrarese e Degli Esposti per questa occasione positiva e formativa.

Grazie infinite a tutti voi.

Bibliografia

[1] Michela Aliazzo. Infografica – industria 4.0 tra opportunità e sfide. https://inno3.it/2021/04/26/slowletter-netconsulting-cube-infografica-i-trend-abilitanti-lindustria-4-0-opportunita-e-sfide/.

[2] Alleantia. Alleantia isc software. https://www.alleantia.com/it/products/alleantia-isc-software/.

[3] Luana Costa. Tecno lamiera. 5:34–37, nov 2022.

[4] Creately. Tutorial sul diagramma di sequenza: Guida completa con esempi. https://creately.com/blog/it/uncategorized-it/tutorial-sul-diagramma-di-sequenza/.

[5] Informatica e ingegneria informatica. Diagramma di gantt: spiegazione e rappresentazione. https://vitolavecchia.altervista.org/diagramma-di-gantt-spiegazione-rappresentazione/.

[6] Tecnologia e Innovazione. Internet of automation. 2:3–15 48–49, mag 2022.

[7] Emerson. Tecnologia di analisi dei processi (pat) di deltav^{tm} spectral. https://www.emerson.com/it-it/automation/control-and-safety-systems/distributed-control-systems-dcs/deltav-distributed-control-system/deltav-spectral-pat.

[8] Focus. Supply chain digitale. 4:8–10, 16–19, 28–31, 68–70, nov 2021.

[9] Startup guide Ionos. Modello a spirale: il modello per la gestione dei rischi nello sviluppo di software. https://www.ionos.it/startupguide/produttivita/modello-a-spirale/.

[10] Lucidchart. Cos'è l'unified modeling language. https://www.lucidchart.com/pages/it/uml.

[11] Quine Business Publisher. 300.000 macchine collegate in rete non mentono! 2:14–20 92–94, mar 2021.

[12] Quine Business Publisher. Automazione e strumentazione, elettronica industriale. 6:9–21 46–49, set 2022.

[13] Retuner®. I prodotti di orchestra sono smart system integrated. https://www.retuner.eu/news/i-prodotti-di-orchestra-sono-smart-system-integrated/.

[14] Retuner® suite. Retuner, la soluzione che mette d'accordo l'intero sistema di fabbrica. https://www.retuner.eu/prodotti-e-soluzioni/.

[15] Zed A. Shaw. *Learn Python 3 the Hard Way.* Addison-Wesley Professional, 2017.

[16] M.G.M Motori Elettrici SpA. Il progettista industriale. 10:24–28 74–81, nov 2021.

[17] I.M.V Presse Srl. Tecno lamiera. 5:34–37, nov 2022.

[18] Cyber Tec. Diagramma di gantt: cos'è e come utilizzarlo per la pianificazione. https://blog.cybertec.it/diagramma-di-gantt.

[19] Gazzetta ufficiale. Supplemento ordinario n. 57/l, allegato a (articolo 1, comma 9). 297:89–91, dic 2016.

[20] UniversalIT. Industria 5.0: cosa c'è da sapere e come impatterà le aziende. https://universeit.blog/industria-50/.