Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

Un sistema smart di auto-configurazione di reti BLE Mesh: progettazione, implementazione ed analisi

Relatore:
Chiar.mo Prof.
ANGELO TROTTA

Presentata da:
KERAN JEGASOTHY

Correlatore:
Dott.
LEONARDO MONTECCHIARI

Sessione Straordinaria
Anno Accademico 2021/2022

Alla mia famiglia

Indice

Capitolo 1

Introduzione

L'Internet of Things (IoT) rappresenta una delle più grandi rivoluzioni tecnologiche degli ultimi anni, che sta trasformando il modo in cui interagiamo con il mondo che ci circonda. L'IoT consiste nell'interconnessione di dispositivi e sensori, in grado di raccogliere e scambiare dati, creando una vasta rete di informazioni e possibilità di interazione. In questo contesto, la rete wireless svolge un ruolo chiave, consentendo una connessione costante e veloce tra i dispositivi e i sistemi IoT.

Uno degli ultimi sviluppi nell'ambito della rete wireless è il Bluetooth Low Energy Mesh, una tecnologia innovativa che sta guadagnando sempre più popolarità nell'industria IoT. Il BLE Mesh consente a un numero elevato di dispositivi IoT di comunicare tra loro in modo efficiente, sicuro e affidabile, creando una rete ad-hoc che può coprire aree estese. Ciò significa che le applicazioni IoT possono essere utilizzate in ambienti ancora più ampi, inclusi i grandi edifici, i campus universitari e le città intere.

Tuttavia, la configurazione dei parametri della rete BLE Mesh è un'operazione complessa e laboriosa da eseguire manualmente. In quanto, è necessario calibrare i parametri per ogni deployment, il quale richiede un notevole sforzo in termini di tempo.

Il presente studio si occupa della progettazione ed implementazione di un sistema automatico per la configurazione delle reti BLE Mesh. Tale sistema è composto da due fasi: una fase iniziale di raccolta dati per acquisire informazioni sulla condizione della rete e una successiva fase in cui vengono applicati alcuni algoritmi di intelligenza artificiale, fra cui algoritmi basati su alberi, KNN e Gaussian Naive Bayes, per individuare i valori ottimali dei parametri necessari per garantire un livello di prestazione richiesto dall'utente.
Al fine di permettere all'utente una facile interazione con il sistema, è stata realizzata una dashboard, mediante la quale l'utente può avviare il processo di autoconfigurazione ed impostare i valori desiderati ai singoli parametri.

Successivamente, sono stati realizzati due setup per raccogliere informazioni sulla prestazione della rete BLE Mesh modificando i valori dei parametri. In seguito, è stato condotto uno studio sulla prestazione della rete BLE Mesh, esaminando l'impatto della modifica dei parametri regolabili in tempo reale attraverso il sistema.

In sintesi, questa tesi si concentrerà sulla progettazione ed implementazione di un sistema automatico per la configurazione di alcuni parametri del BLE Mesh, con uno studio della prestazione della rete in relazione alle variazioni dei valori dei suddetti parametri.

Capitolo 2

Stato dell'arte

2.1 Internet of thing

L'Internet of Things (IoT) è una rete di dispositivi interconnessi che utilizzano le tecnologie della comunicazione per scambiare dati tra loro senza necessità di interazione umana [2]. I dispositivi IoT possono essere incorporati in oggetti di uso quotidiano, come elettrodomestici, automobili, dispositivi wearable, sensori e dispositivi di automazione industriale. Essi possono raccogliere e utilizzare i dati da questi dispositivi per migliorare l'efficienza, l'automazione e la prevenzione dei problemi.
In generale, l'IoT è composto da tre elementi principali:

- Dispositivi: i dispositivi IoT sono dotati di un microcontrollore o di un microprocessore, una connessione di rete e una serie di sensori e attuatori. I sensori raccolgono i dati dall'ambiente circostante, mentre gli attuatori eseguono azioni in base ai dati raccolti. Inoltre, questi dispositivi sono caratterizzati da dimensioni ridotte, bassa potenza e spesso sono alimentati da batterie.

- Reti: le reti IoT possono utilizzare diverse tecnologie di comunicazione, tra cui Wi-Fi, Bluetooth, Zigbee, NFC, o rete cellulare per comunicare tra loro e con la rete. Inoltre, l'IoT utilizza protocolli di comunicazione standard come MQTT, CoAP, HTTP e WebSockets, per consentire la comunicazione tra i dispositivi.

- Servizi cloud: i servizi cloud sono utilizzati per elaborare e archiviare i dati raccolti dai dispositivi IoT. Essi possono utilizzare tecnologie come il Machine Learning e l'elaborazione delle immagini per analizzare i dati raccolti e generare informazioni utili.

Quindi l'IoT può essere utilizzato in una vasta gamma di ambiti e applicazioni, alcuni dei quali includono:

- Smart home: l'IoT può essere utilizzato per automatizzare il controllo dell'illuminazione, della temperatura e dell'approvvigionamento energetico in una casa. Ad esempio, i sensori possono essere utilizzati per rilevare la presenza delle persone in una stanza e regolare la temperatura di conseguenza, o per accendere e spegnere le luci in base alla quantità di luce naturale presente.

- Smart city: l'IoT può essere utilizzato per monitorare e migliorare la qualità dell'aria, il traffico, il consumo energetico e la sicurezza in una città. Ad esempio, i sensori IoT possono essere utilizzati per rilevare la qualità dell'aria in tempo reale e inviare allarmi alle autorità locali in caso di superamento dei limiti di sicurezza, o per ottimizzare il traffico in base al flusso di veicoli in tempo reale.

- Industry 4.0: l'IoT può essere utilizzato per migliorare la manutenzione predittiva, l'automazione dei processi e la sicurezza del lavoro in un'industria. Ad esempio, i sensori IoT possono essere utilizzati per monitorare la salute di una macchina e inviare un allarme quando necessario, o per ottimizzare la produzione in base ai dati raccolti dai sensori.

- Sanità: l'IoT può essere utilizzato per monitorare la salute e il benessere degli individui. Ad esempio, i dispositivi wearable possono raccogliere dati sull'attività fisica, il sonno e la frequenza cardiaca e inviare questi dati a un medico per una valutazione.

- Natura: l'IoT può essere utilizzato per monitorare le condizioni del terreno, delle piante e degli animali, per ottimizzare l'irrigazione, la fertilizzazione e la gestione delle risorse.

Questi sono solo alcuni esempi di come l'IoT può essere utilizzato, ci sono molte altre applicazioni e ambiti di utilizzo, dall'automotive alla logistica, fino all'energia. Quindi, si può affermare che l'IoT sta continuando a evolversi e a espandersi, offrendo nuove opportunità per migliorare la qualità della vita, l'efficienza e la produttività.

Tuttavia, presenta anche sfide in termini di sicurezza e privacy, poiché i dispositivi interconnessi possono rappresentare una vulnerabilità per la sicurezza informatica se non protetti correttamente. È quindi importante che gli sviluppatori e i produttori di questi dispositivi considerino la sicurezza fin dalla progettazione e che gli utenti adottino misure di sicurezza appropriate per proteggere i propri dispositivi e dati personali.

2.2 Bluetooth

Il Bluetooth [15] è uno standard di comunicazione wireless che permette di trasmettere dati su brevi distanze, generalmente entro i 10 metri. È una tecnologia a basso consumo energetico, utilizzata principalmente per collegare dispositivi mobili come smartphone, tablet, altoparlanti, cuffie, orologi smart, mouse e tastiere senza fili con altri dispositivi. Per la comunicazione, il Bluetooth utilizza una banda di frequenza di 2,4 GHz divisa in 79 canali per evitare interferenze e supporta velocità di trasferimento dati fino a 2 Mbps. I dati vengono suddivisi in pacchetti e trasmessi in modo asincrono, utilizzando tecniche di crittografia per garantire la sicurezza della comunicazione.
Il Bluetooth ha diverse versioni, con miglioramenti nelle prestazioni e nuove funzionalità introdotte con ogni nuova versione. Attualmente, la versione più recente è il Bluetooth 5.2, che supporta una lunga portata, bassa latenza, audio di alta qualità e basso consumo energetico.

In conclusione, il Bluetooth consente la connessione di dispositivi mobili su brevi distanze mediante la trasmissione di dati suddivisi in pacchetti in modo asincrono, garantendo allo stesso tempo la sicurezza della comunicazione e utilizzando tecnologie avanzate per migliorare le prestazioni e le funzionalità.

2.3 Bluetooth Low Energy

Il Bluetooth Low Energy (BLE) è una versione a basso consumo di energia del classico Bluetooth che è stata progettata per dispositivi che richiedono una connessione wireless a consumo ridotto, come ad esempio fitness tracker, smartwatch e sensori IoT. Il BLE utilizza il medesimo protocollo di comunicazione wireless a corto raggio del Bluetooth, ma con alcune importanti differenze che ne migliorano l'efficienza energetica e l'efficacia nella trasmissione dei dati [5].

Dal punto di vista tecnico, questa tecnologia utilizza un modello di connessione centrato sulla gestione efficiente dell'energia e sulla bassa latenza. Questo modello prevede la trasmissione dei dati a intervalli regolari, con lunghe pause tra una trasmissione e l'altra, che riducono significativamente il consumo energetico. Inoltre, il BLE supporta la modalità di sospensione, in cui i dispositivi possono entrare in uno stato a basso consumo energetico quando non vengono utilizzati.
Oltre a ciò, supporta anche una serie di tecnologie di sicurezza per garantire che le comunicazioni siano protette da interferenze esterne e da attacchi malintenzionati.

In sintesi, BLE rappresenta un importante passo avanti rispetto al Bluetooth tradizionale, offrendo una soluzione di comunicazione wireless a basso consumo energetico per una vasta gamma di dispositivi IoT e wearable. Con la sua combinazione di modelli di connessione efficienti, larghezze di banda ridotte e tecnologie avanzate di gestione della

potenza, questa tecnologia è in grado di garantire una trasmissione affidabile dei dati con un basso consumo energetico.

2.4 Bluetooth Low Energy Mesh

Il Bluetooth Low Energy Mesh [3] è un'evoluzione del BLE standard che permette di creare reti wireless a basso consumo energetico per dispositivi IoT (Internet of Things). In particolare, estende la portata delle reti BLE, consentendo la connessione tra più dispositivi in modo da creare una rete completa e interconnessa. Questo rende possibile la creazione di soluzioni IoT scalabili e affidabili, adatte a molteplici applicazioni, come ad esempio la domotica, l'illuminazione smart e la gestione dell'energia/risorse [4].

2.4.1 I vantaggi del BLE Mesh

In seguito sono riportati i vantaggi principali del BLE Mesh:

- Copertura Wireless Estesa: il BLE Mesh utilizza una tecnologia mesh wireless per creare una rete di dispositivi connessi che consente una copertura wireless estesa. Ciò significa che i dispositivi possono comunicare tra loro anche se non sono direttamente connessi ad un router o ad un gateway. Questa architettura mesh rende possibile la comunicazione tra numerosi dispositivi in modo efficiente e affidabile, aumentando la copertura e la resilienza della rete.

- Robustezza Superiore: la robustezza è un aspetto critico delle reti wireless, poiché la qualità del segnale può influire sulla qualità dei dati trasmessi. Il BLE Mesh utilizza una architettura che garantisce una robustezza superiore rispetto alle reti wireless convenzionali, poiché ogni dispositivo può fungere da nodo di ripetizione e rendere la rete più resistente agli errori/ guasti. In questo modo, se un dispositivo viene disconnesso o danneggiato, la rete può continuare a funzionare utilizzando gli altri dispositivi come nodi di ripetizione.

- Efficienza Energetica: il BLE Mesh è molto efficiente dal punto di vista energetico, grazie alla sua capacità di accendere e spegnere automaticamente i dispositivi non necessari. Questo rende possibile una lunga durata della batteria per i dispositivi connessi, rendendo più conveniente l'utilizzo di soluzioni IoT a lunga durata. Inoltre, utilizza un basso consumo energetico per la trasmissione dei dati, che lo rende ideale per l'utilizzo in soluzioni con risorse energetiche limitate.

- Ampia Compatibilità: il BLE Mesh supporta una vasta gamma di dispositivi e sistemi operativi, il che rende facile la creazione di soluzioni IoT a basso costo e facili da utilizzare. Inoltre, la tecnologia BLE Mesh è compatibile con molte

piattaforme e dispositivi mobile, rendendola una soluzione ideale per l'integrazione con sistemi esistenti.

- Sicurezza: la sicurezza dei dati è un aspetto critico delle reti IoT, e il BLE Mesh offre una elevata sicurezza dei dati utilizzando protocolli di crittografia avanzati. Questi protocolli proteggono le informazioni sensibili durante la trasmissione, garantendo che i dati rimangano sicuri e privati.

- Scalabilità: il BLE Mesh è progettato per essere facilmente scalabile, il che significa che è possibile espandere la rete in base alle esigenze dell'utente. È possibile connettersi a un numero illimitato di dispositivi, il che rende la tecnologia ideale per soluzioni IoT di grandi dimensioni. Inoltre, offre una configurazione semplice e una gestione efficiente dei dispositivi connessi, rendendo possibile una facile gestione delle soluzioni IoT a lungo termine.

- Costo: il BLE Mesh è una soluzione a basso costo rispetto ad altre tecnologie mesh, poiché utilizza una tecnologia standardizzata che è stata largamente implementata in molte piattaforme e dispositivi. Ciò rende possibile la creazione di soluzioni IoT a basso costo e accessibili a una vasta gamma di utenti.

In conclusione, il BLE Mesh offre una serie di vantaggi, tra cui una copertura wireless estesa, robustezza superiore, efficienza energetica, ampia compatibilità, sicurezza dei dati e scalabilità. Questi vantaggi rendono il BLE Mesh una tecnologia ideale per il settore dell'Internet of Things.

2.4.2 Funzionamento del BLE Mesh

Il funzionamento del BLE Mesh [8] è basato sull'architettura mesh, dove ogni dispositivo funge da nodo di trasmissione e di ripetizione. Ciò significa che ogni dispositivo può comunicare direttamente con altri dispositivi nella rete e trasmettere i dati ai dispositivi non in grado di comunicare direttamente l'uno con l'altro.
Quando un dispositivo invia un messaggio, questo viene trasmesso a uno o più dispositivi vicini che a loro volta lo trasmettono ai dispositivi successivi, fino a quando il messaggio non raggiunge il destinatario. Questo processo garantisce che i dati vengano trasmessi anche se uno o più nodi sono temporaneamente non disponibili o non raggiungibili.
Il BLE Mesh utilizza un sistema di routing dinamico per selezionare il percorso ottimale per la trasmissione dei dati, tenendo conto della qualità del segnale e della capacità dei nodi nella rete. Inoltre, supporta la comunicazione multicast, che consente di inviare un messaggio a più destinatari contemporaneamente.
Questa tecnologia utilizza protocolli di crittografia avanzati per garantire la sicurezza dei dati trasmessi. Ad esempio, utilizza la crittografia AES per proteggere i dati durante la

trasmissione e l'autenticazione tramite chiavi pubbliche e private per garantire che solo i dispositivi autorizzati possano accedere alla rete.

Quindi, il BLE Mesh è una tecnologia di rete wireless affidabile che utilizza un'architettura mesh per garantire la trasmissione affidabile e sicura dei dati tra i dispositivi nella rete. Inoltre, utilizza un sistema di routing dinamico e protocolli di crittografia avanzati per garantire la sicurezza dei dati e la qualità delle comunicazioni nella rete.

2.4.3 Applicazioni del BLE Mesh

Il BLE Mesh è una tecnologia versatile che può essere utilizzata in una vasta gamma di applicazioni. Ecco alcune delle principali applicazioni del BLE Mesh [8]:

- Building Automation: Il BLE Mesh è una soluzione ideale per la creazione di una rete wireless per il controllo dei sistemi di riscaldamento, ventilazione e climatizzazione in edifici commerciali o residenziali. Questa tecnologia consente agli utenti di controllare questi sistemi da qualsiasi punto dell'edificio tramite un'unica applicazione mobile. Ad esempio, gli utenti possono regolare la temperatura dell'edificio, tenere traccia dei consumi energetici e molto altro ancora, tutto da un'unica posizione centralizzata.

- Illuminazione a LED: Il BLE Mesh può essere utilizzato per creare una rete wireless per il controllo di gruppi di luci a LED. Questa tecnologia consente agli utenti di accendere e spegnere le luci, regolare la luminosità, creare scenari d'illuminazione personalizzati e molto altro ancora, tutto da un'unica posizione centralizzata tramite un'unica applicazione mobile. In questo modo, gli utenti possono ottenere una maggiore flessibilità e controllo sulle luci della propria casa o dell'edificio, e anche risparmiare sui costi energetici grazie a un uso più efficiente dell'illuminazione.

- Salute e benessere: Il BLE Mesh è una soluzione ideale per la creazione di una rete wireless per il monitoraggio della salute e del benessere. Ad esempio, questa tecnologia può essere utilizzata per monitorare la frequenza cardiaca, l'attività fisica e altri parametri relativi alla salute e al benessere. Questo consente agli utenti di tenere traccia delle proprie attività fisiche e di monitorare la propria salute in modo più efficace. Inoltre, questa tecnologia può essere utilizzata per creare una rete wireless per il controllo dei dispositivi medici, come gli apparecchi per la somministrazione di insulina o i sensori di pressione arteriosa. In questo modo, gli utenti possono monitorare la propria salute in modo più efficiente e ottenere una maggiore flessibilità nella gestione delle proprie condizioni mediche.

- Retail: Il BLE Mesh è una soluzione ideale per la creazione di una rete wireless per il monitoraggio dei prodotti in un negozio. Ad esempio, questa tecnologia può essere utilizzata per monitorare il livello di scorte dei prodotti, tenere traccia

dei movimenti dei clienti e molto altro ancora. In questo modo, i negozi possono ottenere una maggiore flessibilità e controllo sui propri prodotti, e possono anche aumentare la soddisfazione dei clienti grazie a una maggiore disponibilità dei prodotti e un servizio più efficiente.

Quindi, il BLE Mesh offre una vasta gamma di applicazioni in molti settori, tra cui l'automazione domestica, l'edilizia, l'illuminazione, il retail, la salute e il benessere. Grazie alle sue caratteristiche come la flessibilità, l'efficienza e la scalabilità, sta diventando una soluzione sempre più popolare per molte applicazioni.

2.4.4 Architettura del BLE Mesh

Lo stack del BLE Mesh è composto da 8 elementi che sono riportati in seguito [35]:

- Application Layer: questo livello è dove le applicazioni vengono eseguite sul dispositivo BLE Mesh. Utilizza il modello di programmazione fornito dal livello Model Layer per accedere ai servizi della rete.
- Model Layer: questo livello fornisce un modello di programmazione comune per l'accesso ai servizi della rete BLE Mesh. Include modelli di servizi pre-definiti per funzioni comuni come l'illuminazione e la climatizzazione. I modelli possono essere estesi o personalizzati per soddisfare le esigenze specifiche dell'applicazione.
- Foundation Model Layer: questo livello definisce le operazioni di base che possono essere eseguite all'interno della rete BLE Mesh. Ad esempio, include le funzioni per la configurazione e il controllo dei nodi della rete.
- Access Layer: questo livello gestisce l'accesso alle informazioni all'interno della rete BLE Mesh, incluso il controllo degli accessi e la gestione delle autorizzazioni. Utilizza la sicurezza fornita dal livello Security Layer per proteggere le informazioni sensibili.
- Upper Transport Layer: questo livello gestisce la trasmissione affidabile dei pacchetti di dati tra i nodi della rete utilizzando protocolli di trasporto sicuri. Il protocollo di trasporto sicuro della rete BLE (SIG Mesh Transport Security) viene utilizzato per garantire la sicurezza e l'affidabilità dei dati trasmessi.
- Lower Transport Layer: questo livello gestisce le connessioni tra i nodi della rete, incluso il mantenimento dei collegamenti e la gestione delle collisioni dei segnali. Anche qui il protocollo di trasporto sicuro della rete BLE (SIG Mesh Link Layer Security) viene utilizzato per garantire la sicurezza e l'affidabilità dei dati trasmessi.

- Network Layer: questo livello fornisce la sicurezza per la rete BLE Mesh, gestendo la crittografia e l'autenticazione dei dati. La crittografia avanzata come l'algoritmo Advanced Encryption Standard (AES) viene utilizzata per proteggere i dati trasmessi all'interno della rete.
- Bearer Layer: questo livello definisce come trasportare i messaggi all'interno della rete.

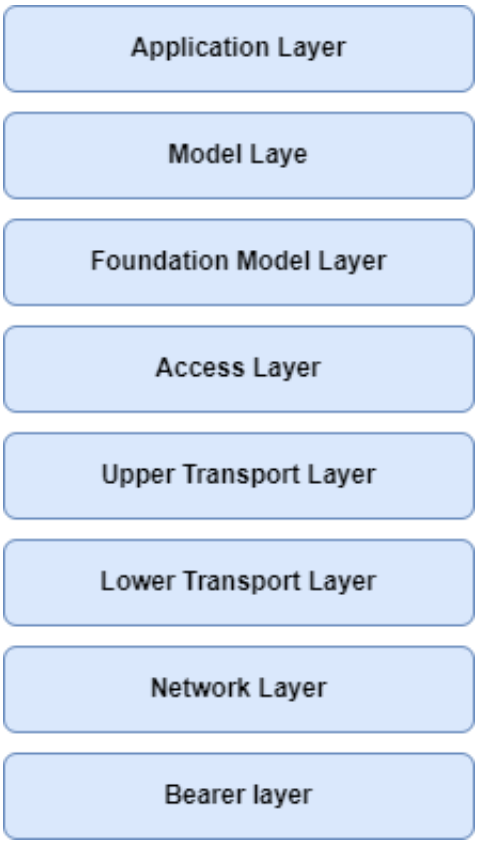


Figura 2.1: Architettura del BLE Mesh.

2.4.5 Le componenti principali del BLE Mesh

In seguito, sono riportate le componenti principali che possono essere individuate all'interno di una rete BLE Mesh [36]:

- **Nodi**: sono i dispositivi che compongono la rete BLE Mesh e sono in grado di comunicare tra di loro. Ogni nodo è dotato di un indirizzo unicast univoco che gli permette di essere identificato all'interno della rete. Possono essere dispositivi diversi, come sensori, attuatori, gateway e dispositivi mobili.

- **Elementi**:
  - Low-Power Feature: I nodi che hanno delle limitazioni di alimentazione possono beneficiare della funzionalità di basso consumo per ridurre il tempo di attivazione della radio e risparmiare energia. In particolare, i nodi a basso consumo (LPN) collaborano con i nodi friend.
  - Friend Feature: I nodi che non hanno restrizioni di alimentazione sono ottimi candidati per diventare nodi Friend. Questi sono in grado di memorizzare i messaggi in arrivo e gli aggiornamenti di sicurezza destinati ai nodi a basso consumo (LPN). Successivamente, quando il LPN è disponibile, il nodo friend consegna i messaggi memorizzati destinati al LPN.
  - Relay Feature: I nodi di inoltro hanno il compito di ricevere e ritrasmettere i messaggi, permettendo così di estendere la portata della rete grazie al passaggio dei messaggi da un nodo all'altro. La capacità di un nodo di assumere questa funzione dipende dalla fonte di alimentazione e dalla sua capacità di calcolo.
  - Proxy Feature: I nodi proxy hanno la funzione di consentire la trasmissione e la ricezione di messaggi mesh tra i nodi mesh GATT e Bluetooth. Questo tipo di nodo richiede una fonte di alimentazione solida e adeguate risorse computazionali per adempiere il suo ruolo.

  Alcuni nodi presentano una struttura più complessa rispetto ad altri, in quanto sono costituiti da più parti indipendenti denominate elementi. Ciascun nodo è costituito da almeno un elemento, chiamato «elemento primario», ma può anche includere elementi aggiuntivi. Gli elementi sono composti da entità che definiscono la funzionalità del nodo e le condizioni dell'elemento stesso.
  All'interno di ogni nodo, ciascun elemento ha un indirizzo univoco denominato «indirizzo unicast», che permette di indirizzare specificamente quell'elemento.

- **Modelli e stati**: I modelli sono contenuti all'interno degli elementi e ogni elemento deve avere almeno uno di essi. I modelli definiscono e implementano la funzionalità e il comportamento del nodo, mentre gli stati definiscono la condizione degli elementi.
  Il BLE Mesh consente di definire stati compositi, ovvero stati che sono formati da due o più valori. Ad esempio, una lampada che cambia colore consente di modificare la tonalità indipendentemente dalla saturazione o dalla luminosità del colore. Inoltre, il concetto di stati legati si riferisce alla situazione in cui un cambiamento in uno stato provoca un cambiamento in un altro stato. Uno dei legami più comuni è quello tra gli stati di livello e gli stati On/Off: se il livello passa da 0 a 1, anche lo stato On/Off passa da off a on.
  È importante sottolineare che ogni modello all'interno di BLE Mesh possiede un

identificatore univoco. In particolare,i modelli Bluetooth SIG hanno un identificatore a 16 bit, mentre quelli dei fornitori sono a 32 bit (composti da un identificatore dell'azienda a 16 bit e da un identificatore del modello assegnato dal fornitore a 16 bit). Questo assicura che ogni modello sia indirizzabile in modo univoco all'interno della rete.
La comunicazione all'interno di una rete mesh Bluetooth avviene attraverso l'utilizzo di messaggi e un'architettura client-server. La funzione del server consiste nel mettere a disposizione gli stati di un determinato elemento. Uno degli stati più comuni è l'interruttore binario, che può essere acceso o spento. Un esempio di modello di server semplice è il modello generico di server on/off, il quale include lo stato dell'interruttore, che rappresenta l'attivazione o la disattivazione.
Questa architettura client-server dà origine a tre tipi di modelli:

  - Modello server: il modello è formato da uno o più stati che possono estendersi su uno o più elementi. Esso definisce i messaggi che il modello può trasmettere o ricevere, e stabilisce il comportamento dell'elemento in base a questi messaggi.
  - Modello client: definisce l'insieme di messaggi che il client può utilizzare per richiedere, modificare o utilizzare gli stati corrispondenti di un server.
  - Modello di controllo: i modelli di controllo sono caratterizzati dalla presenza di funzionalità multiple e possono comprendere uno o più dei seguenti elementi:
    * modelli del cliente
    * modelli del server
    * logiche di controllo (regole e comportamenti) coordinando le interazioni tra i modelli con cui si connette.

- **Indirizzi**: Esistono quattro tipi di indirizzi

  - Indirizzi unicast: Durante il processo di provisioning, ogni elemento in un nodo riceve un indirizzo unicast, il quale ha validità per la durata della presenza del nodo sulla rete. Gli indirizzi unicast possono comparire sia nel campo dell'indirizzo di origine che in quello di destinazione di un messaggio. I messaggi inviati agli indirizzi unicast vengono elaborati esclusivamente dall'elemento destinatario.
  - Indirizzo virtuale: sono un insieme di elementi associati ad un etichetta UUID specifica; questi indirizzi possono essere pubblicati o sottoscritti. L'UUID dell'etichetta è un valore a 128 bit associato a più elementi che possono provenire da uno o più nodi.
  - Indirizzo non assegnato: gli elementi non configurati o gli elementi senza indirizzi designati hanno un indirizzo non assegnato. Dato che questi elementi

non dispongono di un indirizzo univoco, non possono essere utilizzati nella messaggistica.

- Indirizzo di gruppo: sono un tipo di indirizzo multicast presente nella rete BLE mesh, utilizzati per rappresentare più elementi da uno o più nodi. Esistono due tipologie di indirizzi di gruppo:
  - Indirizzi assegnati dinamicamente: 0xC000-0xFEFF
  - Indirizzi fissi – assegnati da Bluetooth SIG:
    - Riservato per uso futuro (RFU): 0xFF00-0xFFFB
    - Tutti i proxy: 0xFFFC (Invia a tutti i nodi con funzionalità proxy abilitata)
    - Tutti i friend: 0xFFFD (Invia a tutti i nodi con la funzionalità Friend abilitata)
    - Tutti i relè: 0xFFFE (Invia a tutti i nodi con funzionalità di relè abilitata)
    - Tutti i nodi: 0xFFFF (Inviato a tutti i nodi)

- **Messaggi**: All'interno della rete BLE Mesh, la cominucazione avviente tramite lo scambio di messaggi. Questi messaggi possono essere di due tipi
  - Messaggi di controllo: messaggi relativi al funzionamento della rete BLE Mesh.
  - Messaggi di accesso: possono essere utilizzati dai modelli client per recuperare o impostare i valori di stato nei modelli server, oppure possono essere impiegati dal server per segnalare i valori di stato.
    Questo tipo di messaggio può essere di due tipi:
    - Messaggio di accesso riconosciuto: vengono trasmessi e ricevuti da tutti gli elementi destinatari. Di solito, ne segue una risposta che in genere è un messaggio di stato.
    - Messaggio di accesso non riconosciuto: a questo tipo di messaggio non è prevista una risposta.

  Infine, i messaggi sono identificati da codici operativi che hanno parametri associati. Ogni codice operativo identifica l'operazione del messaggio.

- **Sicurezza dei messaggi**: In una rete BLE Mesh, ogni messaggio viene protetto mediante l'uso di NetKey e AppKey per la crittografia e l'autenticazione. Le NetKey vengono utilizzate per la comunicazione a livello di rete e, in una rete mesh senza sottoreti, tutte le comunicazioni utilizzano la stessa chiave di rete.
  Le AppKey, invece, vengono utilizzate per i dati dell'applicazione. Alcuni nodi

all'interno della rete potrebbero gestire applicazioni specifiche che richiedono un accesso limitato a dati sensibili. In tali casi, questi nodi sono associati a una specifica AppKey. Le diverse aree che utilizzano diverse AppKey possono includere la sicurezza (accesso all'edificio, accesso alla sala macchine e all'ufficio del CEO), l'illuminazione (piano di produzione, luci esterne dell'edificio e passaggi pedonali) e i sistemi HVAC.

- **Scambio di messaggi**: In una rete mesh, per permettere ai nodi di comunicare tra di loro, viene utilizzato il paradigma «publish-subscribe». La fase di invio di un messaggio da parte di un nodo a uno o più dispositivi viene chiamata «Publishing». Un nodo che intende ricevere messaggi inviati a indirizzi specifici configura il proprio sistema di ricezione attraverso una procedura chiamata «Subscribing», la quale richiede di sottoscriversi a tali indirizzi (i quali sono memorizzati nella «subscriber list»). I messaggi possono essere destinati a un indirizzo «unicast» oppure pubblicati attraverso un indirizzo di «gruppo» o «virtuale».
  I nodi hanno la possibilità di inviare messaggi «unsolicited» o in risposta ad altri messaggi. Quando un nodo invia una risposta, utilizza come indirizzo di destinazione l'indirizzo del mittente originario. Per quanto riguarda i messaggi «unsolicited», questi possono essere inviati utilizzando come indirizzo di destinazione un «publish address», che può essere uno tra i seguenti indirizzi:

  – unicast
  – gruppo preconfigurato
  – virtuale

  Ogni modello presente all'interno di un nodo ha un singolo «publish address».
  Per quanto riguarda la ricezione dei messaggi, ogni istanza di un modello ha la possibilità di sottoscriversi ad uno o più indirizzi di gruppo o virtuali. Ogni volta che un messaggio viene pubblicato su un indirizzo di gruppo o virtuale a cui il modello è iscritto, verrà elaborato dal nodo. Inoltre, un messaggio viene elaborato anche quando il suo indirizzo di destinazione corrisponde ad un indirizzo «unicast» di un elemento che appartiene al dispositivo.
  Infine, per consentire al nodo di ricevere i messaggi pubblicati in diversi gruppi, è possibile che un nodo abbia più sottoscrizioni per ogni singola istanza del modello di un elemento.

2.4.6 Provisioning

All'interno di una rete BLE Mesh, il concetto di provisioning è uno dei più importati. Consente, tramite lo scambio di informazioni tra un dispositivo unprovisioned e un provisioner, di aggiungere il primo alla rete mesh. Inoltre, permette anche la configurazione

dei nodi esistenti e la visualizzazione dello stato della rete. In genere, il ruolo di provisioner viene ricoperto da smartphone con apposite applicazioni, oppure da dispositivi dedicati. In commercio, sono presenti diverse applicazioni disponibili sia per IOS che per Androdi; in seguito sono riportate le principali:

- **nRF Mesh**: è un'applicazione open source sviluppato da Nordic Semiconductor, progettato per essere utilizzato con i microcontrollori della famiglia nRF5x di Nordic Semiconductor. É anche possibile utilizzare tale applicazione con dispositivi di altri fornitori, come ad esempio l'ESP32. Quest'applicazione, include anche una serie di esempi e modelli che possono essere utilizzati come punto di partenza per la creazione di soluzioni personalizzate.

- **Silicon Labs**: è un applicazione che oltre alle operazioni di provisioning include anche una serie di strumenti avanzati per la diagnostica e il debug della rete, tra cui la visualizzazione dei pacchetti in transito e la risoluzione dei problemi di connettività.

- **EspBleMesh**: le funzionalità offerte da questa applicaizioni sono molto simili a quelle offerte da Silicon Labs. In più esso, supporta anche la configurazione remota dei nodi, il che significa che è possibile gestire la rete da qualsiasi luogo con una connessione Internet.

Figura 2.2: Loghi delle applicazioni di Provisioning

2.4.7 Standardizzazione del BLE Mesh

La standardizzazione del BLE Mesh [9] è un fattore cruciale per garantirne la compatibilità, l'interoperabilità e la diffusione su larga scala. L'ente di standardizzazione responsabile per il BLE Mesh è il Bluetooth Special Interest Group (SIG), un'organizzazione senza scopo di lucro che gestisce lo sviluppo e la standardizzazione della tecnologia Bluetooth. Il BLE Mesh è stato standardizzato come parte del Bluetooth Core Specification Version 5.0, che è stata pubblicata nel 2016. Questo standard specifica i requisiti per la creazione e la gestione di una rete BLE Mesh, comprese le specifiche per i nodi della rete, la modalità di trasmissione dei dati e la gestione della sicurezza. Il processo di sviluppo degli standard Bluetooth è aperto e partecipativo, con la partecipazione di

aziende, organizzazioni e individui che rappresentano un'ampia gamma di settori e interessi. Questo processo garantisce che gli standard vengano sviluppati in modo equo e trasparente e che tengano conto delle esigenze della comunità Bluetooth. Una volta che uno standard è stato pubblicato, le aziende possono utilizzarlo per sviluppare prodotti e soluzioni BLE Mesh compatibili. La conformità agli standard viene verificata attraverso un programma di certificazione del prodotto Bluetooth, che garantisce che i prodotti soddisfino i requisiti specificati.

Quindi, la standardizzazione del BLE Mesh è gestita dal Bluetooth SIG e gli standard sono sviluppati attraverso un processo aperto e partecipativo. Questa standardizzazione garantisce la compatibilità, l'interoperabilità e la diffusione su larga scala del BLE Mesh.

2.4.8 Sviluppo dell'ecosistema del BLE Mesh

L'ecosistema del BLE Mesh sta rapidamente evolvendo per soddisfare le crescenti esigenze dei clienti e dei settori industriali. Questo ecosistema include una vasta gamma di librerie e API per gli sviluppatori, moduli hardware e soluzioni di sicurezza [10].

Librerie e API per gli sviluppatori

Per supportare lo sviluppo di soluzioni BLE Mesh, sono disponibili molte librerie e API open-source e proprietarie. Queste librerie e API consentono agli sviluppatori di creare facilmente applicazioni e servizi BLE Mesh, offrendo una vasta gamma di funzionalità, tra cui la gestione dei nodi della rete, la gestione dei messaggi e la sicurezza. In seguito sono riportati alcuni esempi di librerie e API disponibili:

- **ESP-IDF BLE Mesh**: questa libreria open source è sviluppata da Espressif Systems e fornisce un insieme di funzionalità per sviluppare soluzioni BLE Mesh con dispositivi ESP32. Include un'API per la configurazione dei nodi, l'invio e la ricezione di messaggi, e la gestione della rete.
- **nRF5 SDK for Mesh**: questa libreria open source è sviluppata da Nordic Semiconductor e fornisce un insieme completo di funzionalità per sviluppare soluzioni BLE Mesh con dispositivi nRF5. Include un'API per la configurazione dei nodi, l'invio e la ricezione di messaggi, e la gestione della rete.
- **Bluetooth Mesh Profile Specification**: questa API proprietaria è sviluppata dal Bluetooth Special Interest Group (SIG) e fornisce un insieme di funzionalità standard per la creazione di soluzioni BLE Mesh con diversi dispositivi.
- **MeshKit**: questa libreria proprietaria è sviluppata da Silicon Labs e fornisce un insieme completo di funzionalità per sviluppare soluzioni BLE Mesh con dispositivi EFR32. Include un'API per la configurazione dei nodi, l'invio e la ricezione di messaggi, e la gestione della rete.

Moduli hardware

Il mercato dei moduli hardware per il BLE Mesh sta crescendo rapidamente per soddisfare le esigenze dei produttori di dispositivi IoT. Questi moduli offrono una soluzione pronta all'uso per la creazione di dispositivi BLE Mesh, offrendo una gamma completa di funzionalità e supporto per lo sviluppo di soluzioni personalizzate. In seguito sono riportati alcuni esempio di moduli hardware compatibili con il Bluetooth mesh:

- Espressif ESP32
- Nordic Semiconductor nRF52
- Texas Instruments CC2640/CC2650
- Silicon Labs Wireless Gecko

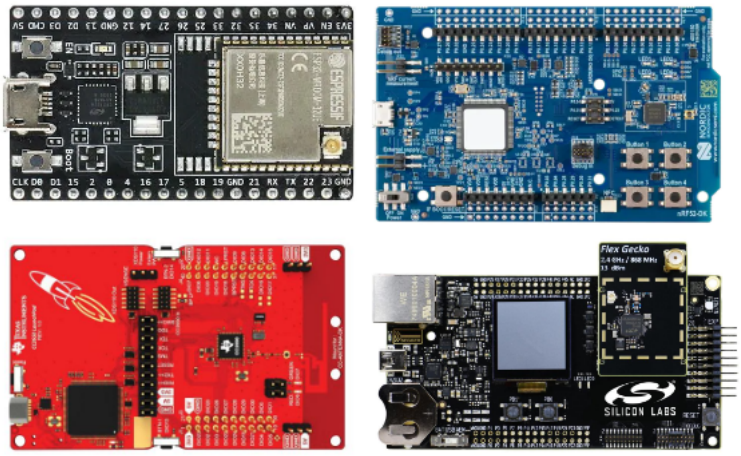


Figura 2.3: Dispositivi compatibili con il BLE Mesh

Quindi, l'ecosistema del BLE Mesh sta rapidamente evolvendo per soddisfare le crescenti esigenze dei clienti e dei settori industriali. Questo ecosistema include una vasta gamma di librerie e API per gli sviluppatori, moduli hardware e soluzioni di sicurezza, che garantiscono uno sviluppo rapido e affidabile delle soluzioni BLE Mesh.

2.4.9 Limiti e sfide del BLE Mesh

Il BLE Mesh è una tecnologia wireless emergente che sta guadagnando popolarità nell'Internet of Things (IoT). Tuttavia, come ogni nuova tecnologia, presenta anche alcuni limiti e sfide che devono essere superati per garantirne il successo. In seguito sono riportati i principali limiti del BLE Mesh [8]:

- **Coesistenza con altre tecnologie wireless**: il BLE Mesh deve coesistere con altre tecnologie wireless come Wi-Fi, Zigbee e Z-Wave. Ciò significa che i dispositivi BLE Mesh devono essere in grado di operare in un ambiente sovraccarico e coesistere con altre tecnologie wireless senza interferire con loro.
- **Gestione della larghezza di banda**: la larghezza di banda disponibile per il BLE Mesh è limitata e questo può essere un problema per l'utilizzo di molte applicazioni IoT che richiedono una larghezza di banda elevata.
- **Scalabilità**: il BLE Mesh è progettato per gestire grandi reti di dispositivi IoT. Tuttavia, con un aumento del numero di dispositivi nella rete, cresce anche la complessità delle operazioni di gestione della rete. È importante che le soluzioni BLE Mesh siano scalabili per poter gestire efficacemente grandi reti.
- **Sicurezza**: la sicurezza è un aspetto cruciale delle reti IoT e il BLE Mesh non fa eccezione. È importante che le soluzioni BLE Mesh forniscano un livello adeguato di sicurezza per proteggere i dati sensibili e prevenire eventuali attacchi.

2.4.10 Futuro del BLE Mesh

Il BLE Mesh è una tecnologia in rapida evoluzione che sta attirando sempre più attenzione da parte di aziende e sviluppatori. Con la crescente domanda di soluzioni di rete affidabili e scalabili per l'Internet of Things (IoT), il BLE Mesh sta diventando un'opzione sempre più popolare per la connessione di dispositivi smart e di automazione domestica [11].

Tendenze emergenti

Con l'aumento della popolarità del BLE Mesh, ci si aspetta che cresca la domanda di soluzioni hardware e software avanzate, tra cui dispositivi dotati di BLE Mesh integrato e librerie software più sofisticate. Inoltre, con la crescente attenzione sulle soluzioni ecosostenibili, si prevede che il BLE Mesh venga utilizzato sempre più spesso per la gestione di sistemi di automazione energetica.

Opportunità di mercato

Con l'aumento della domanda di soluzioni IoT, ci si aspetta che il mercato del BLE Mesh continui a crescere rapidamente nei prossimi anni. Ciò potrebbe portare a nuove opportunità di business per aziende che sviluppano hardware e software per questa tecnologia.

Sviluppi futuri

È probabile che il BLE Mesh continui ad evolversi per soddisfare le esigenze sempre crescenti dei clienti. Ci si aspetta che la tecnologia venga perfezionata per supportare una maggiore quantità di dispositivi e una maggiore velocità di trasmissione dei dati. Inoltre, si prevede che il BLE Mesh continui a integrarsi con altre tecnologie, come 5G e AI, per fornire soluzioni sempre più avanzate per l'IoT.

In conclusione, il futuro del BLE Mesh è molto promettente. Con un'espansione del mercato e una continua evoluzione della tecnologia, ci si aspetta che questa tecnologia giochi un ruolo sempre più importante nell'Internet of Things e nell'automazione domestica nei prossimi anni.

2.5 Internet of Things e Artificial Intelligence

IoT (Internet of Things) e AI (Artificial Intelligence) sono due tecnologie interconnesse che stanno rivoluzionando il modo in cui le persone e le aziende interagiscono con il mondo digitale. L'IoT si riferisce alla connessione di oggetti fisici (come sensori, dispositivi e macchine) alla rete Internet, al fine di raccogliere e scambiare dati in tempo reale. L'AI, d'altra parte, è una tecnologia che consente ai computer di effettuare compiti che richiederebbero normalmente l'intelligenza umana, come il riconoscimento delle immagini, la comprensione del linguaggio naturale e la prevedibilità [12].
La combinazione di IoT e AI offre molte opportunità per creare soluzioni intelligenti che possono migliorare la vita delle persone e aumentare l'efficienza delle aziende. Ad esempio, i sensori IoT possono raccogliere grandi quantità di dati sulle condizioni ambientali, sulle prestazioni delle macchine e sui comportamenti degli utenti, che possono essere analizzati dall'AI per prendere decisioni in tempo reale. Questo può essere utilizzato per diverse applicazioni, tra cui la automazione delle case, la logistica, la diagnostica medica, la sicurezza delle infrastrutture critiche e molto altro ancora.
Inoltre, l'AI può essere utilizzata per migliorare la sicurezza dell'IoT, ad esempio rilevando e prevenendo eventuali minacce alla sicurezza dei dispositivi connessi. Ad esempio, l'AI può essere utilizzata per analizzare i comportamenti di rete ed identificare eventuali attività anomale che potrebbero indicare una minaccia alla sicurezza. Oppure, può essere utilizzata per prevedere la manutenzione dei dispositivi IoT, migliorando la loro affidabilità e la durabilità.
Inoltre, l'IoT e l'AI possono lavorare insieme per creare sistemi autonomi che possono prendere decisioni e agire in modo indipendente. Ad esempio, i sistemi di automazione industriale possono utilizzare l'IoT per raccogliere dati sulle condizioni delle macchine e l'AI per analizzare questi dati e prendere decisioni su come ottimizzare le prestazioni delle macchine.

In sintesi, la relazione tra IoT e AI è molto stretta e profonda, poiché entrambe le tecnologie hanno la possibilità di potenziarsi a vicenda e creare soluzioni più smart e avanzate. L'IoT fornisce la capacità di raccogliere grandi quantità di dati dal mondo reale, mentre l'AI fornisce la capacità di analizzare questi dati e trarne informazioni utili.

2.5.1 Cos'è l'Intelligenza Artificiale

L'Intelligenza Artificiale (AI) è una branca dell'informatica che si occupa di creare sistemi che possono compiere compiti tipicamente associati all'intelligenza umana, come il ragionamento, la risoluzione dei problemi, l'apprendimento automatico e la comprensione del linguaggio [6].
L'AI si basa su algoritmi avanzati e su vasti quantitativi di dati, che gli permettono di migliorare continuamente le proprie prestazioni. Con l'aumentare della potenza di calcolo e della quantità di dati disponibili, l'AI sta diventando sempre più avanzata e sta trasformando molte industrie, dalla sanità all'entertainment.

2.5.2 Sfide: Sicurezza dei dati, privacy e responsabilità nell'interconnessione IoT ed AI

L'interconnessione tra IoT ed AI rappresenta una grande opportunità per migliorare la qualità della vita e rendere le attività quotidiane più efficienti. Tuttavia, essa presenta anche sfide significative relative alla sicurezza dei dati, alla privacy e alla responsabilità. Queste sfide devono essere affrontate e risolte per garantire che le soluzioni IoT ed AI siano sicure, affidabili e utilizzabili in modo responsabile.

Sicurezza dei dati

Con una quantità sempre crescente di dispositivi connessi, è cruciale garantire la sicurezza dei dati raccolti attraverso questi dispositivi. Ciò significa proteggere i dati da eventuali attacchi informatici, come il furto di dati o la manipolazione non autorizzata. Quindi, i dispositivi IoT e AI devono essere progettati in modo da minimizzare la vulnerabilità ai rischi di sicurezza.

Privacy

La quantità di dati raccolti e elaborati attraverso i dispositivi IoT ed AI rende la privacy un aspetto critico da considerare. È importante garantire che i dati siano gestiti in modo sicuro e che le informazioni personali non vengano utilizzate per scopi indesiderati. Ciò significa che i proprietari dei dati devono essere in grado di controllare i propri dati e

che le aziende che raccolgono ed elaborano questi dati devono seguire rigorose politiche di tutela dei dati.

Responsabilità

In caso di problemi tecnici o di sicurezza, è fondamentale identificare chi sia responsabile per la risoluzione del problema e per prevenire futuri incidenti. Ad esempio, se un dispositivo IoT o una soluzione AI causano danni, è importante determinare chi sia responsabile per questi danni e come sia possibile prevenirli in futuro.

2.5.3 Tipi di Intelligenza Artificiale

Esistono diversi tipi di Intelligenza Artificiale, ognuno dei quali si concentra su una particolare area di applicazione ed utilizza tecniche diverse per raggiungere i suoi obiettivi [6].

- Algoritmi di apprendimento supervisionato: è un tipo di intelligenza artificiale in cui l'algoritmo è addestrato utilizzando un set di dati di esempio con etichette note. Il modello formulato dall'algoritmo utilizza questi dati per fare previsioni su dati futuri non visti. Questo tipo di intelligenza artificiale è utilizzato per classificare e prevedere i dati in base alle loro caratteristiche.

- Algoritmi di apprendimento non supervisionato: è un tipo di intelligenza artificiale che non utilizza dati di esempio etichettati per addestrare il modello. Invece, cerca di scoprire automaticamente i pattern nascosti nei dati utilizzando tecniche come il clustering o la riduzione della dimensionalità. Questo tipo di intelligenza artificiale viene utilizzato per l'analisi di grandi quantità di dati e la scoperta di pattern e relazioni nei dati.

- Algoritmo di Reinforcement Learning: è un tipo di intelligenza artificiale che utilizza un processo di prova ed errore per apprendere come compiere compiti specifici. L'algoritmo è addestrato attraverso una serie di tentativi e riceve ricompense o punizioni in base alle sue azioni. Questo tipo di intelligenza artificiale viene utilizzato per problemi di ottimizzazione e prendere decisioni in ambienti incerti.

- Deep learning: è un tipo di intelligenza artificiale che utilizza una rete neurale artificiale di grandi dimensioni per imparare i rappresentazioni complesse dei dati. Questo tipo di intelligenza artificiale viene utilizzato per una vasta gamma di compiti, tra cui la classificazione, la previsione e la generazione di immagini, suoni e testo.

2.5.4 Futuro dell'IA: Tendenze e sviluppi futuri

Data la rapida crescita dell'intelligenza artificiale ci sono molte tendenze e sviluppi futuri che stanno emergendo nel settore. In seguito ne vengono riportate alcune [14]:

- AI avanzata: sta diventando sempre più sofisticata e sarà in grado di svolgere compiti ancora più complessi, come la risoluzione di problemi e la prevenzione sempre più complicati.
- Integrazione dell'AI nei dispositivi: si prevede che l'AI verrà integrata in molte più applicazioni e dispositivi, rendendo più semplice per le persone accedere e utilizzare le tecnologie di intelligenza artificiale.
- Cloud AI: l'AI basata sul cloud sta diventando sempre più diffusa, poiché permette agli sviluppatori di accedere a risorse potenti e scalabili in modo semplice e conveniente.
- Collaborazione uomo-AI: l'AI sta diventando sempre più collaborativa e sta lavorando a stretto contatto con le persone per risolvere problemi e completare compiti.

Capitolo 3

Progettazione

Il presente lavoro di tesi ha l'obiettivo di sviluppare un sistema di auto-configurazione per una rete BLE Mesh, che permetta di individuare in maniera efficiente i parametri ideali per garantire una prestazione ottimale della rete. A tal fine, è stato realizzato un'applicazione web, denominata "dashboard", che consente all'utente di interagire con il sistema in modo intuitivo e semplice.

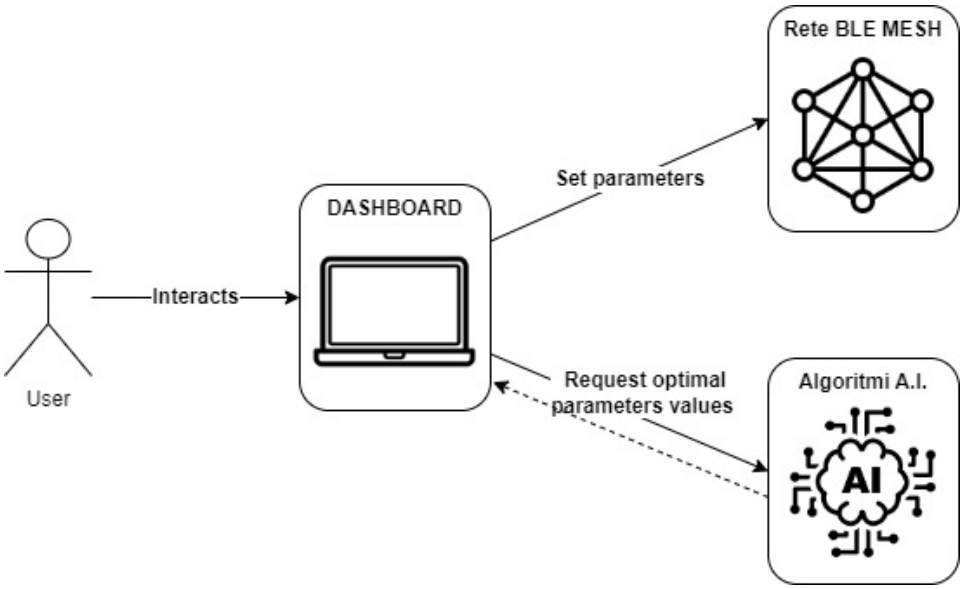


Figura 3.1: Architettura logica

La dashboard è stata progettata per svolgere diverse funzioni. In primo luogo, consente di avviare la raccolta dei dati necessari per la configurazione della rete. Questa fase di raccolta è fondamentale per raccogliere informazioni sulle prestazioni della rete

in condizioni diverse, come ad esempio la distanza tra i nodi, la densità dei nodi stessi e il carico di lavoro sulla rete. Una volta completata la fase di raccolta dati, la dashboard fornisce all'utente la possibilità di utilizzare un classificatore per individuare i parametri ottimali per raggiungere una determinata performance.
In particolare, l'utente ha la possibilità di scegliere tra diversi algoritmi di machine learning per l'analisi dei dati raccolti. Tra i modelli disponibili ci sono Decision Tree, Random Forest, Gradient Boosting, KNN e Gaussian Naive Bayes. L'utente deve specificare il Delay e/o il Packet Delivery Ratio come input per questi modelli. Inoltre, la dashboard consente all'utente di specificare anche il valore di alcuni parametri della rete BLE Mesh, come ad esempio il TTL, Potenza di trasmissione, Numero di trasmissioni e intervallo di tramissione. Nel caso in cui fossero inseriti alcuni parametri, i modelli procederanno con l'individuazione dei valori per i parametri mancanti.

Oltre ad avviare il processo di auto-configurazione, la dashboard consente anche all'utente di impostare manualmente alcuni parametri della rete, in modo da poter personalizzare la configurazione in base alle proprie esigenze e preferenze.

Per quanto riguarda la configurazione della rete, si è optato per la progettazione di una rete basata su diversi dispositivi ESP32, in grado di operare in due differenti modalità di configurazione: Nodo Client e Nodo Server.

3.1 Casi d'uso

Un diagramma dei casi d'uso è uno strumento utilizzato nell'analisi e nella progettazione del software per rappresentare le funzionalità che il sistema deve fornire, dal punto di vista dell'utente. La figura 3.2 riporta il diagramma d'uso relativa alla dashboard.

Come evidenziato dal diagramma, all'interno della dashboard, l'utente ha la possibilità di eseguire le seguenti operazioni:

- inviare al nodo client un nuovo valore per i seguenti parametri:
  - TTL
  - Potenza di trasmissione
  - Numero di trasmissioni
  - Intervallo di trasmissione

  che può essere impostato a tutti i nodi presenti nella rete oppure ad un determinato nodo
- avviare un processo di raccolta dati riguardante il setup creato dall'utente, al fine di utilizzare un algoritmo di apprendimento supervisionato volto all'individuazione

dei parametri ottimali per raggiungere specifici livelli di latenza e packet delivery ratio. Con tale approccio, sarà possibile individuare i parametri ottimali per il setup dell'utente, garantendone così la massima efficacia.

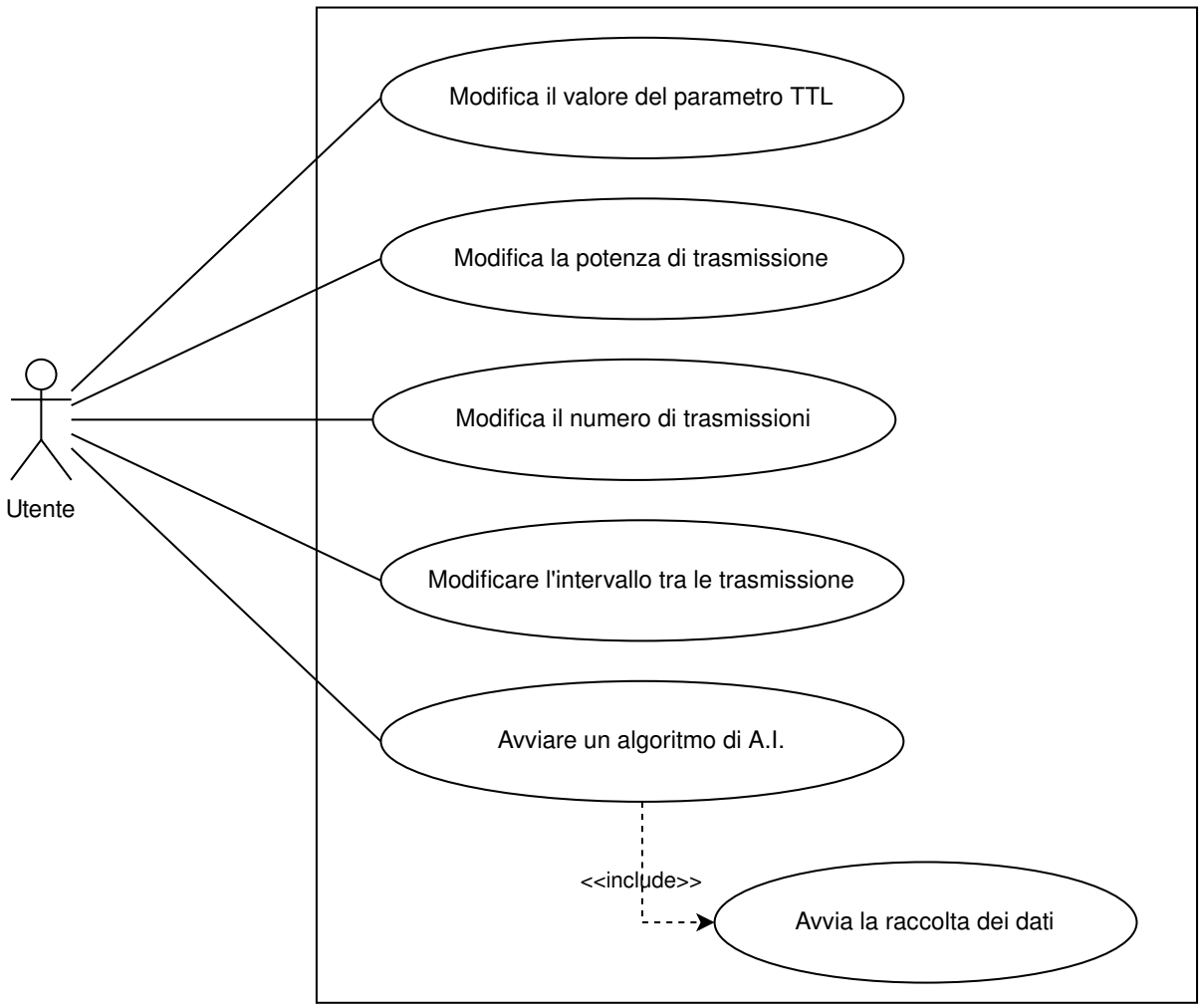


Figura 3.2: Diagramma Use Case

Successivamente, la figura 3.3 mostra la dashboard.

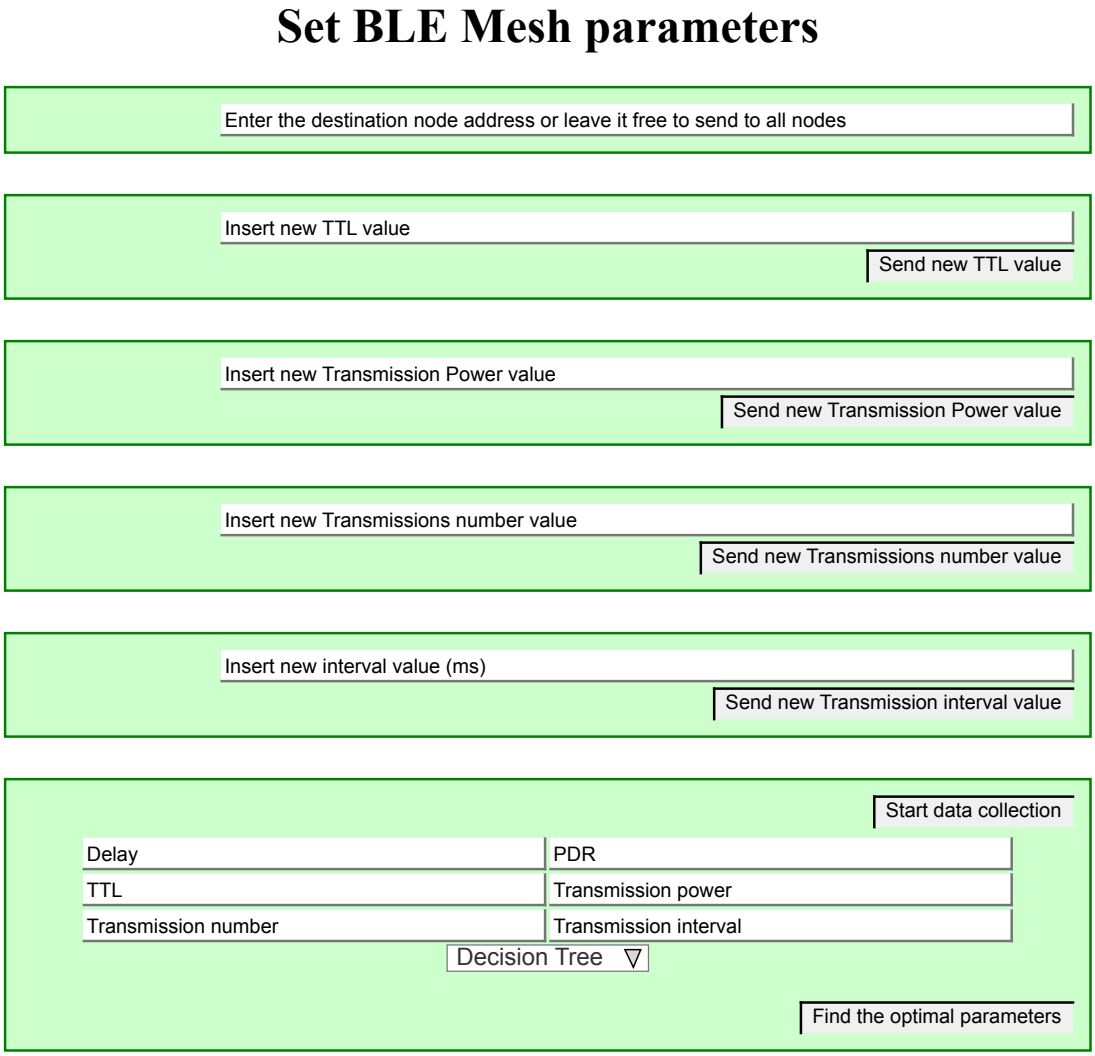


Figura 3.3: Dashboard

3.2 Parametri modificabili

In seguito, sono riportati i parametri del BLE Mesh, che l'utente ha la possibilità di impostarlo tramite da dashboard:

- TTL (Time To Live): è un valore che rappresenta il numero massimo di salti che un pacchetto di dati può fare prima di essere scartato. Quindi ad ogni passaggio da un nodo all'altro, il valore viene decrementato di uno. Quando il valore raggiunge zero, il pacchetto viene scartato e non viene più trasmesso. Questo meccanismo garantisce che i pacchetti non circolino all'infinito nella rete e che la diffusione dei

dati avvenga in modo efficiente. Il TTL può assumere i seguenti valori: 0, 1, 2, 3, 4, 5, 6 e 7.

- Potenza di trasmissione: influisce sulla portata del segnale BLE, ovvero sulla distanza massima a cui un dispositivo può comunicare con altri dispositivi BLE. Impostare un valore più elevato per la potenza di trasmissione aumenterà la portata del segnale, ma può anche aumentare il consumo energetico del dispositivo e interferire con altre comunicazioni wireless nell'area. Impostare un valore più basso per la potenza di trasmissione ridurrà la portata del segnale, ma può anche ridurre il consumo energetico del dispositivo e minimizzare l'interferenza con altre comunicazioni wireless. Possono essere assegnati 8 diverse potenze di trasmissione: -12dbm, -9dbm, -6dbm, -3dbm, 0dbm, +3dbm, +6dbm, +9dbm.

- Numero di trasmissioni: è il numero di volte che il pacchetto viene trasmesso nella rete BLE Mesh. In una situazione in cui la qualità del segnale è scarsa, potrebbe essere necessario aumentare il numero di trasmissioni per garantire che il pacchetto venga trasmesso correttamente. In altre situazioni, un numero inferiore di trasmissioni potrebbe essere sufficiente per garantire la diffusione affidabile dei dati. Questo parametro può assumere i seguenti valori: 1, 2, 3, 4, 5, 6 e 7.

- Intervallo tra le trasmissioni: è il tempo che il dispositivo deve attendere tra la trasmissione di due pacchetti consecutivi. Il tempo di attesa tra le trasmissioni può essere utilizzato per ottimizzare la diffusione affidabile dei dati nella rete BLE Mesh. Ad esempio, un tempo di attesa più lungo tra le trasmissioni può essere utilizzato per ridurre il carico sulla rete, mentre un tempo di attesa più breve può essere utilizzato per garantire che il pacchetto venga trasmesso rapidamente nella rete. Questo parametro può assumere tutti i valori che siano multiplo di 10.

3.3 Architettura raccolta dati

È stata implementata l'architettura descritta nella figura 3.4 al fine di creare un insieme di dati contenente le prestazioni della rete, ottenute variando i parametri. In questa architettura, vi è uno script scritto in Python che esegue il calcolo di tutte le possibili combinazioni di parametri. Questi parametri vengono quindi assegnati singolarmente agli ESP32 per misurare le prestazioni della rete e memorizzarle in un database MySQL.

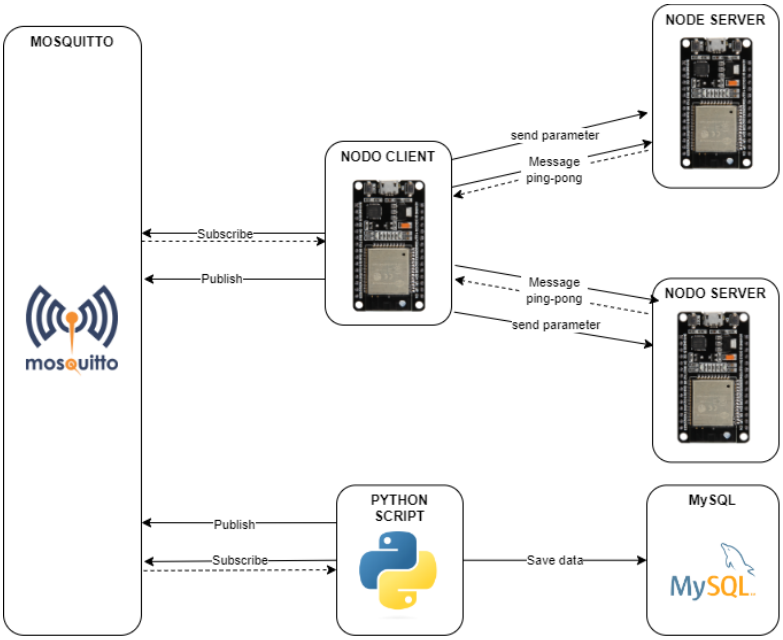


Figura 3.4: Architettura utilizzato per raccogliere i dati.

Infine, vengono utilizzati algoritmi di apprendimento supervisionati, come il Decision Tree, Random Forest, Gradiant Boosting, KNN e Gaussian Naive Bayes, per identificare in modo automatico i parametri ottimali da impostare per raggiungere un determinato livello di prestazione.

3.4 Architettura

Nella figura 3.5, viene presentata l'architettura sviluppata per il progetto. Questa architettura è composta da diverse componenti che lavorano insieme per garantire il funzionamento ottimale.
In primo luogo, abbiamo una rete BLE Mesh composta da diversi ESP32; in particolare è presente un ESP32 in modalità **nodo client** e uno o più ESP32 in modalità **node server**. Inoltre, è presnte un dispositivo mobile (Android o iOS) che utilizza l'applicazione nRF Mesh sviluppata da Nordic. Quest'ultimo ha il ruolo di **Provisioner** all'interno della rete. Successivamente, troviamo **Mosquitto**, che funge da broker per il protocollo **MQTT**, e il **Client** che si interfaccia tramite la dashboard.
Infine, il back-end è composto da **Node.js**, **Python** e **MySQL**, che forniscono le funzionalità necessarie per il corretto funzionamento dell'architettura. In particolare, node.js viene utilizzato per gestire le richieste degli utenti (in particolare si collega come client al portocollo MQTT, ed inoltre, premette di accedere allo script Python), Python viene

utilizzato per implementare gli algoritmi di intelligenza artificiale e, infine, MySQL viene utilizzato per salvare i dati.

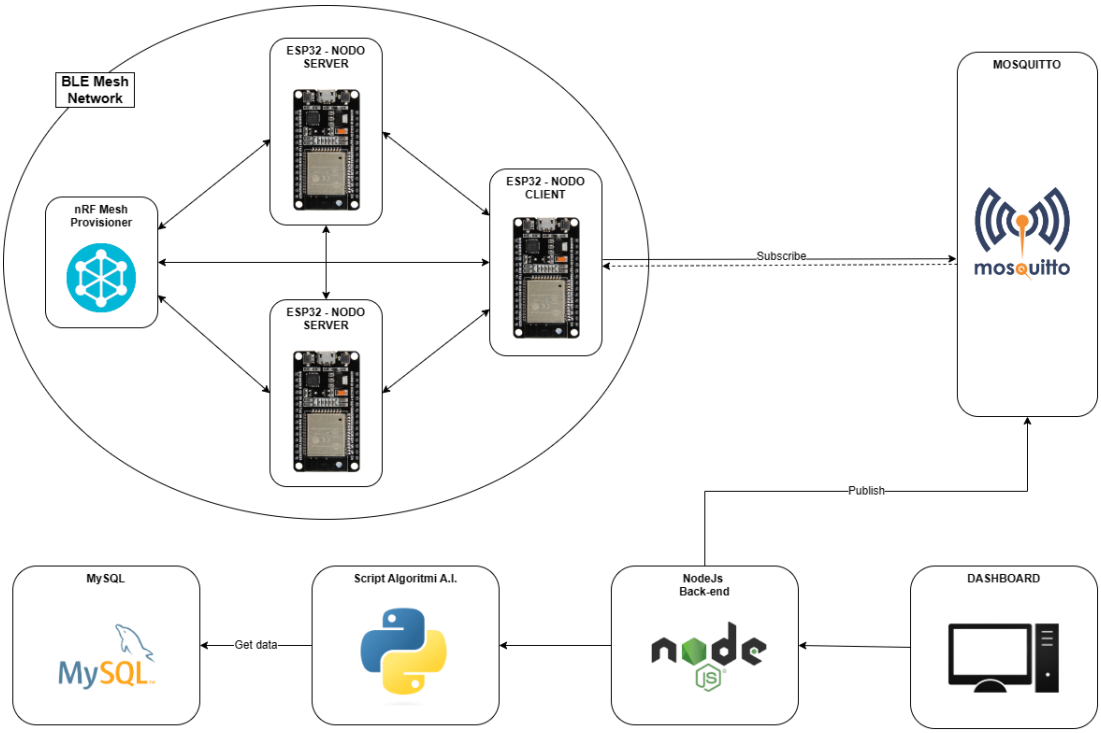


Figura 3.5: Architettura del progetto.

Questa architettura può essere divisa in 3 componenti: la rete BLE Mesh, il protocollo MQTT e l'architettura client-server.

3.4.1 Rete BLE Mesh

Nella parte della rete BLE Mesh, si possono individuare 3 compomenti:

- nRF Mesh Provisioner: è un dispositivo mobile (Android o IOS) che utilizza l'applicazione nRF Mesh. E' fondamentale per la gestione della rete BLE Mesh, poiché permette di eseguire azioni come l'aggiunta di nuovi nodi, la rimozione di nodi esistenti e la configurazione di parametri specifici. All'interno della rete può essere presente solo un unico Provisioner.
- ESP32 - Nodo server: è un dispositivo che è in grado di ricevere richieste da parte di un nodo client e di inviare una risposta attraverso un messaggio. All'interno della rete possono essere presenti uno o più nodi server.

- ESP32 - Nodo client: è un dispositivo che è in grado di inviare determinate richieste ai nodi server e di gestire i messaggi di risposta. All'interno della rete possono essere presenti uno o più nodi client.
  Inoltre, è l'unico nodo dotato di connessione ad internet, quindi permettte di creare un collegamento tra la dashboard (il client) e la rete BLE Mesh.

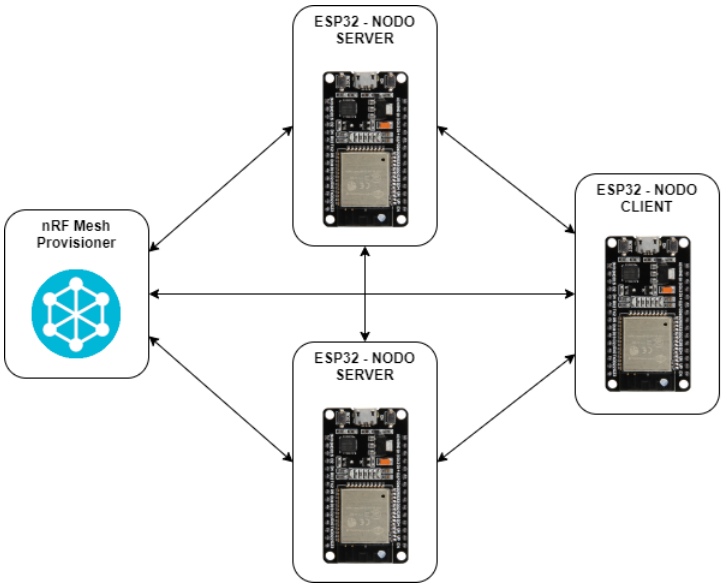


Figura 3.6: Architettura - parte BLE Mesh.

3.4.2 Protocollo MQTT

In questa architettura, viene utilizzato il protocollo di comunicazione MQTT per permettere al client, ovvero alla dashboard, di comunicare con la rete BLE Mesh. Questo protocollo fornisce una soluzione affidabile e flessibile per la trasmissione dei dati tra il client e uno o più dispositivi ESP32 che operano come nodi client. Il protocollo MQTT funziona utilizzando il modello pubblicatore-sottoscrittore, in cui il client pubblica i dati su argomenti specifici e i nodi client sottoscrivono gli argomenti per ricevere i dati pubblicati. In questo modello, il client invia i nuovi valori da assegnare ai vari parametri dei dispositivi che compongono la rete BLE Mesh utilizzando MQTT. Per l'utilizzo di MQTT in questa architettura, è stato scelto di utilizzare il broker open source Mosquitto. Il broker Mosquitto funge da intermediario per la comunicazione tra i client, riceve le pubblicazioni da parte del client (dashboard) e le distribuisce ai nodi client che hanno sottoscritto gli stessi argomenti. In questo modo, Mosquitto garantisce la consegna affidabile dei messaggi e la gestione degli argomenti e delle sottoscrizioni.

In sintesi, il protocollo MQTT e il broker Mosquitto sono stati scelti per questa architettura perché offrono una soluzione scalabile e affidabile per la comunicazione tra il client e la rete BLE Mesh, permettendo l'invio e la ricezione affidabili dei dati.

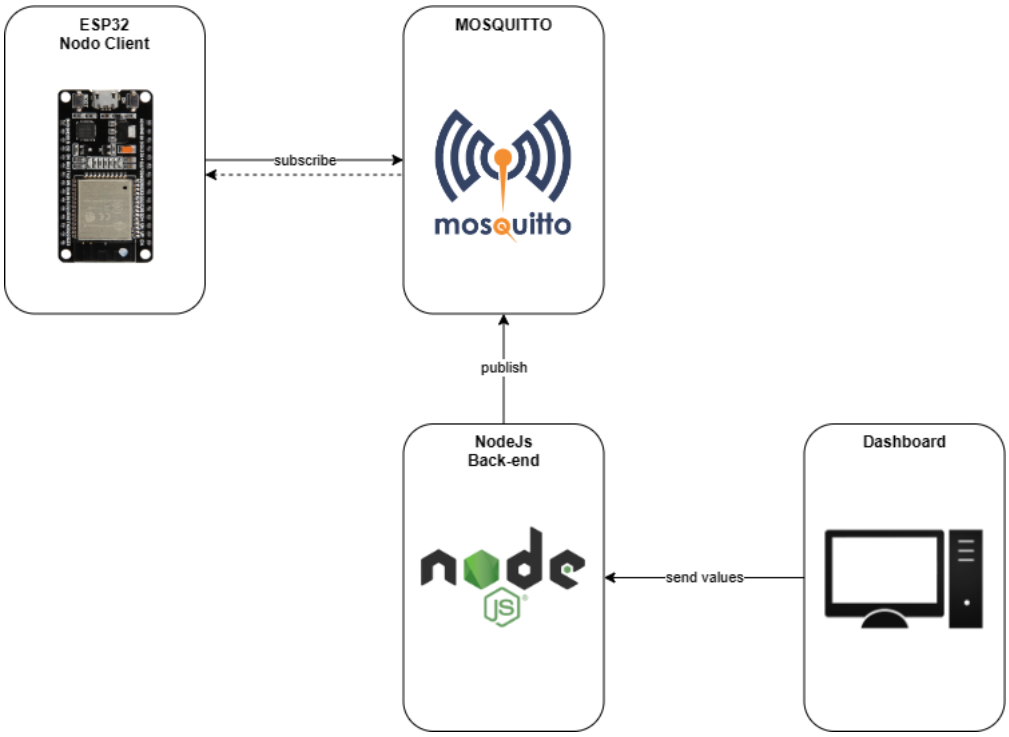


Figura 3.7: Architettura - parte protocollo MQTT.

3.4.3 Architettura client-server

L'architettura client-server è un modello di distribuzione di elaborazione in cui un'applicazione o servizio è suddiviso tra componenti che agiscono come server e componenti che agiscono come client. Nel presente sistema, la dashboard rappresenta il client, che invia richieste al server, il quale risponde con i dati richiesti. Il server side, composto da node.js, script di intelligenza artificiale e un database relazionale MySQL, è responsabile della gestione delle risorse condivise e delle operazioni di back-end, mentre il client è responsabile dell'interfaccia utente e delle operazioni di front-end.

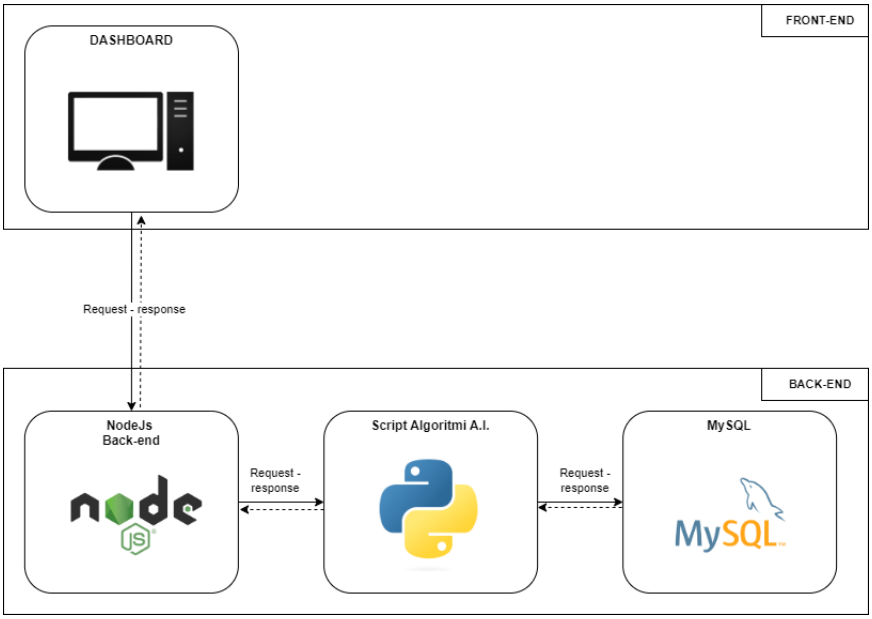


Figura 3.8: Architettura - parte client-server.

3.4.4 Database

Per archiviare i dati raccolti in modo organizzato, è stato utilizzato il database relazionale MySQL. Per questo scopo, è stato creato un database denominato «ble_mesh_parameters», contenente tre tabella:

- configurations
- parameters_performance
- parameters_pdr

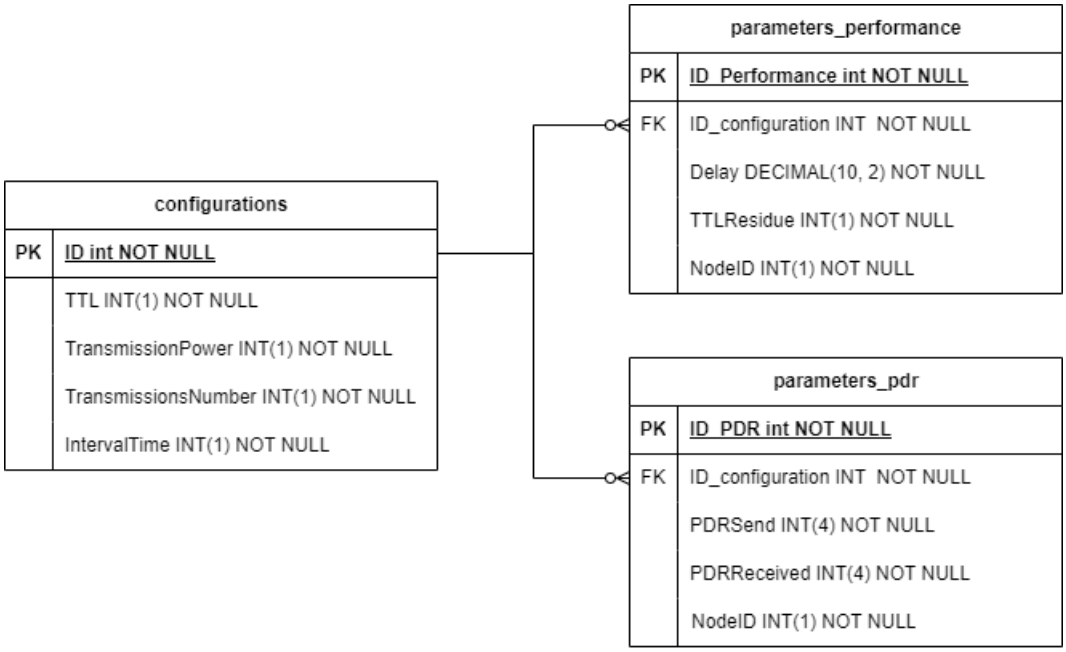


Figura 3.9: Struttura Database

La tabella «configurations» è composta dai seguenti campi:

- ID: un campo di tipo intero che funge da primary key e viene automaticamente incrementato ad ogni inserimento di nuovi dati.
- TTL: un campo di tipo intero con lunghezza massima di 1 (int(1)).
- TransmissionPower: un campo di tipo intero con lunghezza massima di 1 (int(1)).
- TransmissionsNumber: un campo di tipo intero con lunghezza massima di 1 (int(1)).
- IntervalTime: un campo di tipo intero con lunghezza massima di 4 (int(4)).

Mentre la tabella «parameters_performance» è composta dai seguenti campi:

- ID_performance: un campo di tipo intero che funge da primary key e viene automaticamente incrementato ad ogni inserimento di nuovi dati.
- ID_configuration: un campo di tipo intero che funge da chiave esterna.
- Delay: un campo di tipo decimale (decimal(10,2)).
- TTL_Residue: un campo di tipo intero con lunghezza massima di 1 (int(1)).

- NodeID: un campo di tipo intero con lunghezza massima di 1 (int(1)).

Infine, la tabella «parameters_pdr» è composta dai seguenti campi:

- ID_pdr: un campo di tipo intero che funge da primary key e viene automaticamente incrementato ad ogni inserimento di nuovi dati.
- ID_configuration: un campo di tipo intero che funge da chiave esterna.
- PDRSend: un campo di tipo intero con lunghezza massima di 4 (int(4)).
- PDRReceived: un campo di tipo intero con lunghezza massima di 4 (int(4)).
- NodeID: un campo di tipo intero con lunghezza massima di 1 (int(1)).

In tutte le tabelle i campi sono obbligatori (cioè «NOT NULL»), quindi devono sempre avere un valore specificato. Inoltre, sono state impostate alcune condizioni per determinate colonne, tra cui:

- CONSTRAINT "TTL_Ck CHECK" ("TTL" between 0 and 7): per garantire che il valore di TTL sia compreso tra 0 e 7.
- CONSTRAINT "TransmissionPower_Ck" CHECK ("TransmissionPower" between 1 and 6): per garantire che il valore di TransmissionPower sia compreso tra 1 e 6.
- CONSTRAINT "TransmissionsNumber_Ck" CHECK ("TransmissionsNumber" between 1 and 7): per garantire che il valore di TransmissionsNumber sia compreso tra 1 e 7.
- CONSTRAINT "IntervalTime_Ck" CHECK ("IntervalTime" between 20 and 1000): per garantire che il valore di IntervalTime sia compreso tra 20 e 1000.

In seguito, sono riportate delle tabelle riassuntive.

CONFIGURATIONS			
Field	**Type**	**Length**	**Constraints**
ID	int	N.D.	AUTO_INCREMENT, NOT NULL, PRIMARY KEY
TTL	int	1	NOT NULL, CHECK (TTL between 0 and 7)
TransmissionPower	int	1	NOT NULL, CHECK (TransmissionPower between 1 and 6)
TransmissionsNumber	int	1	NOT NULL, CHECK (TransmissionsNumber between 1 and 7)
IntervalTime	int	3	NOT NULL, CHECK (IntervalTime between 20 and 320)

Tabella 3.1: Informazioni riassuntive sulla tabella configurations

PARAMETERS_PERFORMANCE			
Field	**Type**	**Length**	**Constraints**
ID_performance	int	N.D.	AUTO_INCREMENT, NOT NULL, PRIMARY KEY
ID_configuration	int	N.D.	NOT NULL
Delay	decimal	10,2	NOT NULL
TTL_residue	int	1	NOT NULL

Tabella 3.2: Informazioni riassuntive sulla tabella parameters_performance

PARAMETERS_PDR			
Field	**Type**	**Length**	**Constraints**
ID_pdr	int	N.D.	AUTO_INCREMENT, NOT NULL, PRIMARY KEY
ID_configuration	int	N.D.	NOT NULL
PDRSend	int	4	NOT NULL
PDRReceived	int	4	NOT NULL

Tabella 3.3: Informazioni riassuntive sulla tabella parameters_pdr

Capitolo 4

Implementazione

In questo capitolo verranno illustrate le tenologie utilizzate, scelte implementative ed i componenti impiegati per realizzare l'architettura illustrata nel capito precedente.

4.1 Tecnologie utilizzate

Per la realizzazione della tesi, sono state impiegate diverse tecnologie:

- C
- libreria ESP-IDF
- protocollo MQTT
- HTML, CSS, JQuery
- Node.js
- Python
- MySQL
- nRF Mesh

4.1.1 C e ESP-IDF

Per sviluppare il firmware per l'ESP32 è stato utilizzato il linguaggio di programmazione C. Il C è un linguaggio di programmazione a basso livello, general-purpose e ad alte prestazioni, sviluppato negli anni '70. È stato progettato per sistemi operativi e per la programmazione di sistemi, ed è ancora oggi molto diffuso e utilizzato in molti ambiti, tra cui sistemi operativi, driver, firmware, software embedded, compilatori, e molti altri.

Inoltre, è stato utilizzato l'ESP-IDF, che è una libreria open-source creata da Espressif Systems. Fornisce un ambiente di sviluppo completo per l'ESP32, che include driver per molte periferiche comuni, un sistema operativo FreeRTOS, strumenti per la creazione di interfacce utente, supporto per protocolli di rete come WiFi, BLE, Ethernet e molto altro. Include un set di API che semplificano lo sviluppo di firmware. Queste API sono strutturate in modo tale da fornire un alto livello di flessibilità e controllo, rendendo ESP-IDF una scelta eccellente per progetti che richiedono una solida architettura software. Questa libreria utilizza la toolchain GCC per la compilazione e l'ottimizzazione del codice. Il framework include anche un ambiente di sviluppo integrato (IDE) chiamato ESP-IDF Visual Studio Code Extension che fornisce un'esperienza di sviluppo confortevole e intuitiva [31].

ESP-IDF include anche molte funzionalità avanzate come:

- Debugging: supporto per JTAG e OpenOCD per il debugging hardware e software.
- Strumenti di gestione delle configurazioni: permettono di gestire e modificare facilmente le configurazioni del progetto senza dover modificare direttamente il codice sorgente.
- Gestione della memoria: ESP-IDF fornisce una gestione avanzata della memoria, che aiuta a evitare problemi di memoria comuni come i crash del sistema.
- Supporto per OTA (Over-The-Air) update: permette agli utenti di caricare nuovi firmware sui dispositivi senza dover rimuovere il dispositivo dalla rete.

In particolare sono stati utilizzati gli API dedicati al BLE Mesh e al protocollo MQTT.

ESP-IDF: BLE Mesh

Le API dedicate al BLE Mesh forniscono supporto per la creazione di reti BLE Mesh sui microcontrollori ESP32 di Espressif Systems, che garantiscono un'alta affidabilità e prestazioni della rete. In particolare supporta la maggior parte delle specifiche Bluetooth Mesh 1.0 e 1.1, incluse le funzionalità di routing, la gestione degli indirizzi, la trasmissione sicura dei messaggi, la gestione dei gruppi, la gestione delle configurazioni e molto altro.
Inotre, include molte funzionalità avanzate, come la gestione delle operazioni su più modelli, la gestione delle pubblicazioni, la gestione dei proxies e molto altro.
Inoltre, offre un'ampia compatibilità con i dispositivi esistenti che supportano le specifiche Bluetooth Mesh. Ciò significa che è possibile utilizzare tale libreria per integrare facilmente i dispositivi ESP32 in una rete mesh esistente o per creare nuove reti mesh da zero.
Per quanto riguarda la configurazione e la personalizzazione, include strumenti per la

configurazione e la personalizzazione della rete mesh, come la configurazione degli indirizzi, la gestione delle configurazioni, la personalizzazione dei modelli e molto altro.

ESP-IDF: MQTT

ESP-IDF fornisce anche un set di API che semplificano l'utilizzo del protocollo MQTT. Questa libreria supporta la maggior parte delle specifiche MQTT 3.1.1 e MQTT 5.0, incluse le funzionalità di pubblicazione, sottoscrizione, gestione delle connessioni, gestione degli errori e molto altro. Inoltre, include anche molte funzionalità avanzate, come la gestione della qualità del servizio (QoS), la gestione dei token d'accesso e molto altro. Infine, offre anche un'ampia compatibilità con i broker MQTT esistenti, come Mosquitto, HiveMQ e AWS IoT.

4.1.2 Protocollo MQTT

Nell'architettura in questione, il protocollo MQTT viene utilizzato per garantire una comunicazione efficace tra la dashboard e il nodo client. Questo protocollo permette di stabilire una connessione affidabile e sicura tra i due dispositivi, rendendo possibile la trasmissione di informazioni in modo tempestivo e preciso. Inoltre, grazie a questa connessione, la dashboard è in grado di accedere alla rete BLE Mesh, ed modificare i parametri di tutti i nodi appartenenti alla rete BLE Mesh. L'MQTT è un protocollo di comunicazione machine-to-machine (M2M), che utilizza un modello di pubblicazione-sottoscrizione per inviare e ricevere messaggi tra client e broker. Un client può pubblicare messaggi su un argomento o un canale noto come «topic» e altri client possono sottoscriversi a questo topic per ricevere i messaggi pubblicati. Questo modello di comunicazione rende l'MQTT adatto a una vasta gamma di scenari, inclusi la telemetria, il controllo remoto e la gestione delle risorse.

Il protocollo MQTT ha alcune caratteristiche distintive che lo rendono una scelta popolare per le soluzioni IoT:

- Leggero: MQTT è progettato per essere leggero e richiedere poco spazio in termini di memoria e larghezza di banda, il che lo rende adatto a dispositivi con risorse limitate come sensori e microcontrollori.
- Affidabilità: MQTT include funzionalità per garantire la consegna affidabile dei messaggi, come la gestione delle sessioni, la riconnessione automatica e la garanzia di consegna.
- Sicurezza: MQTT include meccanismi di sicurezza come l'autenticazione e la crittografia per proteggere i messaggi durante la trasmissione.

- Scalabilità: MQTT supporta la scalabilità orizzontale, il che significa che è possibile aggiungere più broker per gestire un aumento del carico.

Inoltre, questo protocollo definisce tre tipi di entità: il client, il broker e il topic.

- Il client: è l'entità che pubblica messaggi o si sottoscrive ai topic.
- Il broker: è l'entità che gestisce la distribuzione dei messaggi ai client che hanno sottoscritto il topic. Il broker funge da intermediario tra i client che pubblicano e quelli che sottoscrivono al topic.
- Il topic: è un canale utilizzato per pubblicare messaggi e sottoscriversi . I client pubblicano messaggi su un topic e altri client possono sottoscriversi a questo topic per ricevere messaggi pubblicati.

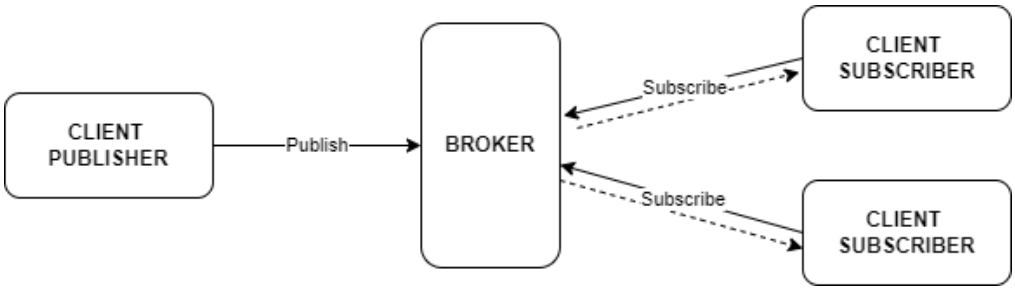


Figura 4.1: Protocollo MQTT.

Mosquitto

In questa architettura è stato utilizzato Mosquitto [28], un broker di messaggistica open source per il protocollo MQTT, ed è disponibile per diverse piattaforme, tra cui Windows, Linux, macOS, Raspberry Pi e molte altre. Può essere eseguito su un singolo server o distribuito su più nodi per garantire alta disponibilità e scalabilità.
Questo broker supporta tutte le caratteristiche principali del protocollo MQTT, come la pubblicazione ai messaggi, la sottoscrizione a determinati topic, la gestione dei topic, la qualità del servizio (QoS) e la gestione delle sessioni client. Inoltre, supporta la sicurezza tramite autenticazione e crittografia dei dati tramite SSL / TLS.
Infine, include anche molte funzionalità avanzate, come la gestione degli accessi con autorizzazioni, la gestione della configurazione tramite file di configurazione, la registrazione degli eventi tramite file di log e la capacità di estendere la funzionalità tramite plugin.

4.1.3 Front-end

Per la realizzazione del Front-end della dashboard sono stati utilizzati HTML, CSS e JQuery.

HTML

HTML [23] (Hypertext Markup Language) è il linguaggio di markup utilizzato per creare pagine web. Si basa su un insieme di etichette o tag, racchiusi tra parentesi angolari, che descrivono il contenuto e la struttura delle pagine web. Una pagina HTML è composta da una serie di elementi che descrivono il contenuto e la sua struttura. Gli elementi più comuni sono:

- Head: contiene informazioni sul documento come il titolo e i metadati.
- Body: contiene il contenuto principale della pagina.
- Headings: utilizzati per fornire intestazioni ai vari sezioni del contenuto.
- Paragraphs: utilizzati per inserire testo formattato.
- Lists: utilizzati per creare elenchi puntati o numerati.
- Links: utilizzati per creare collegamenti a pagine web esterne o interne.
- Images: utilizzati per inserire immagini.
- Tables: utilizzati per organizzare i dati in righe e colonne.

CSS

Il CSS [24] (Cascading Style Sheets) è un linguaggio di formattazione utilizzato per gestire l'aspetto e la presentazione di un documento HTML. Quindi definisce come i vari elementi HTML, come testo, immagini e tabelle, devono essere visualizzati sulla pagina. Con CSS, è possibile controllare l'aspetto di tutte le pagine di un sito web in modo centralizzato, semplificando notevolmente la gestione della formattazione. Inoltre, supporta la creazione di stili dinamici che possono cambiare in base alla risoluzione dello schermo, alla dimensione della finestra del browser e ad altri fattori. Infine, un foglio di stile CSS è composto da regole che descrivono come gli elementi HTML devono essere visualizzati. Queste regole sono composte da due parti:

- un selettore, che identifica gli elementi HTML a cui la regola si applica
- un blocco di dichiarazioni, che descrivono le proprietà di formattazione.

JQuery

JQuery [25] è una libreria JavaScript che offre una vasta gamma di funzionalità per l'elaborazione dei documenti HTML, l'animazione, l'elaborazione degli eventi, l'elaborazione delle richieste AJAX e l'interazione con il server. La sua filosofia è quella di «scrivere meno, fare di più», fornendo una sintassi concisa e intuitiva per la maggior parte delle attività comuni. In particolare, questa tecnologia viene utilizzata nella dashboard per poter visualizzare l'esito dell'invio di richiesta di modifica dei valori dei parametri.

4.1.4 Back-end

Per lo sviluppo del back-end, sono stati utilizzati Node.js, Python e MySQL.

Node.js

Node.js [26] è un ambiente di esecuzione JavaScript open source ed è stato progettato per eseguire codice JavaScript all'esterno del browser e fornisce un modo semplice e intuitivo per creare applicazioni server-side.
Utilizza un modello di I/O non bloccante e basato su eventi che lo rende adatto per le applicazioni ad alte prestazioni e scalabili. Inoltre, supporta anche una vasta gamma di librerie e moduli pre-sviluppati, noti come «pacchetti» o «module», attraverso il suo gestore di pacchetti, npm.
In fine, Node.js è molto versatile e viene utilizzato per una vasta gamma di applicazioni, come il web server, l'elaborazione di richieste HTTP, la creazione di API, l'elaborazione di dati in tempo reale e molto altro.

Per quanto riguarda lo sviluppo dell'applicazione Node.js, sono stati selezionati e utilizzati alcuni pacchetti specifici:

- Express: è un framework per le applicazioni web che semplifica la gestione delle richieste HTTP e delle risposte, fornendo un'interfaccia intuitiva per lo sviluppatore.
- jQuery: è una libreria JavaScript che facilita l'interazione con il document object model (DOM) e fornisce una vasta gamma di funzionalità per la manipolazione del contenuto della pagina web.
- MQTT: protocollo di messaggistica.
- PythonShell: è un modulo di che consente di eseguire codice Python all'interno di un ambiente Node.js. È stato sviluppato per permettere una facile integrazione tra il codice di entrambi i linguaggi, fornendo un'interfaccia semplice e intuitiva per eseguire script Python e ricevere output nella console di Node.js.

MySQL

In questo lavoro di tesi, è stato utilizzato MySQL [27] per salvare tutti i dati raccolti relativi alla prestazione della rete, al variare dei parametri.

MySQL è un sistema di gestione di database relazionali open source ed è progettato per gestire grandi quantità di dati e offre una vasta gamma di funzionalità, tra cui la gestione della sicurezza, la replica dei dati, il supporto per la gestione transazionale e molto altro ancora. Inoltre, è molto flessibile e scalabile, il che lo rende una scelta ideale per molte aziende e organizzazioni che necessitano di un robusto sistema di gestione del database.

4.1.5 Python

In questo lavoro di tesi, Python [17] è stato impiegato in tre distinte fasi.

- In una prima occasione, è stato utilizzato per creare uno script che ha permesso di modificare dinamicamente i parametri dei diversi dispositivi presenti nella rete BLE Mesh, e successivamente per raccogliere i dati relativi alle prestazioni della rete e archiviarli in un database MySQL.
- In una seconda fase, è stato utilizzato per analizzare i dati raccolti.
- In una terza fase, è stato adottato nel back-end del progetto per applicare algoritmi di Intelligenza Artificiale.

Quindi, Python è un linguaggio di programmazione di alto livello, dinamico e interpretato. Inoltre, è un linguaggio multi-paradigma, che significa che supporta diverse tecniche di programmazione, come la programmazione orientata agli oggetti, la programmazione funzionale e la programmazione procedurale.
Infine, Python fornisce una vasta gamma di librerie standard, tra cui strumenti per la manipolazione dei dati, la creazione di grafici, la creazione di interfacce utente e molto altro.
Per lo sviluppo di codice per questa tesi sono stati utilizzati diverse librerie:

- prima fase:
  - Mysql.connector: è una libreria per Python che fornisce un modo semplice e affidabile per connettersi a un database MySQL da un'applicazione Python. Questa libreria offre un'interfaccia Python per l'API di MySQL, che consente di eseguire query SQL, creare tabelle, modificare i dati e molto altro ancora. Con mysql.connector, è possibile creare una connessione a un database MySQL e quindi eseguire query SQL utilizzando una semplice API. È inoltre possibile utilizzare la libreria per connettersi a un database remoto o locale, il che la rende utile per molte applicazioni diverse.

- Paho.mqtt [19]: è una libreria che fornisce un modo semplice e affidabile per connettersi a un broker MQTT (Message Queueing Telemetry Transport) e inviare o ricevere messaggi. Con Paho-mqtt, è possibile creare una connessione a un broker MQTT, sottoscrivere argomenti e inviare messaggi in modo semplice e affidabile. La libreria supporta inoltre la connessione sicura (TLS), il che la rende utile per le applicazioni che richiedono una comunicazione sicura. Inoltre, supporta la gestione degli errori, il che significa che è possibile gestire in modo affidabile eventuali errori durante la connessione al broker MQTT. Questo è particolarmente utile per le applicazioni che devono gestire grandi quantità di dati in tempo reale e garantire la loro integrità.
- Itertools: è una libreria standard di Python che fornisce funzioni utili per lavorare con sequenze di dati, come liste, stringhe, set e molto altro ancora. Queste funzioni sono progettate per essere efficienti e facili da usare, e offrono una serie di modi per elaborare e manipolare le sequenze di dati in modo efficiente. In particolare, è stato utilizzato per creare il prodotto cartesiano di diverse liste.

- seconda e terza fase:
  - Pandas [20]: è una delle librerie più utilizzate per l'elaborazione e l'analisi dei dati. Offre strumenti potenti e intuitivi per la manipolazione e l'analisi di dati strutturati in formato tabulare come le serie temporali, le tabelle e i dataframe. Pandas è particolarmente utile per la pulizia e la preparazione dei dati, la selezione di colonne e righe specifiche, la aggregazione dei dati e l'applicazione di calcoli e statistiche. Inoltre, Pandas supporta la lettura e la scrittura di molti formati di file comuni come CSV, Excel, SQL e JSON, rendendolo facile importare e lavorare con dati provenienti da fonti diverse. I due principali tipi di strutture dati in Pandas sono il DataFrame, che rappresenta una tabella con righe e colonne etichettate, e la Serie, che rappresenta una singola colonna di dati. Entrambe le strutture forniscono una vasta gamma di metodi per l'elaborazione dei dati, tra cui il filtraggio, l'aggregazione, la trasformazione e la manipolazione delle etichette.
  - Numpy [21]: è una libreria che fornisce supporto per lavorare con array multidimensionali e matrici. Questi array multidimensionali sono la struttura di dati centrale di Numpy e sono molto più efficienti e flessibili rispetto ai tipici array di Python. Questa libreria offre una vasta gamma di funzioni matematiche e statistiche per lavorare con questi array, tra cui l'elaborazione di elementi singoli, l'aggregazione, la trasformazione e la manipolazione delle dimensioni degli array. Numpy è anche molto efficiente dal punto di vista delle prestazioni, in particolare per i calcoli su grandi quantità di dati, poiché

utilizza una combinazione di C e Python per sfruttare al meglio le risorse hardware.

  - Matplotlib [22]: è una libreria di plotting che fornisce una vasta gamma di strumenti per la creazione di grafici e visualizzazioni dei dati. Con Matplotlib è possibile creare una vasta gamma di tipi di grafici, tra cui line plot, scatter plot, bar plot, istogrammi, grafici a torta e molti altri. Inoltre, offre una gran quantità di personalizzazioni, tra cui la possibilità di modificare i colori, i titoli, le etichette dell'asse e molto altro ancora. Infine, supporta l'esportazione dei grafici in una vasta gamma di formati, tra cui PNG, PDF, SVG e molti altri, rendendo facile condividere le visualizzazioni con altre persone.

- terza fase:

  - Sklearn [18]: è una libreria open-source per l'apprendimento automatico. Fornisce una vasta gamma di algoritmi di apprendimento automatico, tra cui regressione, classificatori, clustering, riduzione della dimensionalità e molto altro ancora. Uno dei punti di forza di sklearn è la sua interfaccia uniforme, che rende semplice l'utilizzo di diversi algoritmi di apprendimento automatico senza dover apprendere un nuovo API per ogni algoritmo. Inoltre, fornisce molte funzioni utili per la valutazione e la selezione dei modelli, come la divisione delle prove e la valutazione incrociata. Infine, sklearn è molto efficiente e scalabile, il che lo rende adatto a una vasta gamma di applicazioni, dalle piccole applicazioni desktop alle grandi distribuzioni cloud.

4.1.6 nRF Mesh

In questo lavoro di tesi, è stato utilizzato l'applicazione nRF Mesh [29] sviluppata da Nordic Semiconductor, che permette di creare la rete BLE Mesh e gestire l'aggiunta e la rimozione dei dispositivi dalla rete.
nRF Mesh supporta il protocollo Bluetooth 5.1 e utilizza la crittografia end-to-end per garantire la sicurezza dei dati trasmessi. Questo rende la soluzione ideale per una vasta gamma di applicazioni, tra cui l'automazione domestica, l'illuminazione, la sicurezza e la salute.
In particolare, permette di creare facilmente reti wireless affidabili e scalabili, che possono essere gestite e configurate in modo semplice tramite app per dispositivi mobili. Inoltre, include una suite di strumenti di sviluppo completa, tra cui un framework di applicazione, una libreria di driver e un simulatore di rete, che semplificano la creazione di soluzioni innovative e personalizzate.

4.2 Hardware

Come precedentemente anticipato, per la realizzazione dell'architettura, sono stati utilizzati diversi ESP32.

4.2.1 ESP32

L'ESP32 è un microcontrollore a basso costo e a bassa potenza che combina Wi-Fi e Bluetooth in un unico chip. È stato sviluppato dalla Espressif Systems, un'azienda di semiconduttori con sede a Shanghai, in Cina, ed è diventato popolare per la sua flessibilità e le sue prestazioni.
Il microcontrollore ha un'architettura dual-core Xtensa LX6, che gli consente di eseguire due processi in parallelo a velocità fino a 240 MHz. Inoltre, ha 520 KB di memoria SRAM integrata, che lo rende adatto per una vasta gamma di applicazioni.
Uno dei suoi vantaggi è la presenza di moduli Wi-Fi e Bluetooth integrati, che consentono di connettersi a reti wireless e dispositivi Bluetooth con facilità. Inoltre, ha una serie di periferiche integrati, tra cui 34 pin di I/O, 3 UART, 3 I2C, 2 SPI, 16 canali di PWM, 2 interruttori di accensione / spegnimento, un oscillatore a cristallo, un convertitore analogico-digitale (ADC) a 12 bit, un convertitore digitale-analogico (DAC) a 8 bit e molto altro ancora.
L'ESP32 è anche noto per la sua flessibilità. Può essere programmato in una vasta gamma di linguaggi, tra cui C, C ++, Python e altri ancora. Inoltre, è compatibile con molte librerie open source, framework e ambienti di sviluppo, come il popolare Arduino IDE.
Il microcontrollore è anche adatto per l'IoT, poiché può essere alimentato da una vasta gamma di fonti, tra cui batterie e fonti di energia solare. Inoltre, il chip supporta la modalità deep sleep, che riduce notevolmente il consumo di energia, consentendo a dispositivi come sensori e attuatori di funzionare per mesi o addirittura anni con una sola carica di batteria.

Figura 4.2: ESP32

4.2.2 Altri componenti

Per poter utilizzare l'ESP32 in modo tale da avere un feedback visivo su ciò che stava accadendo, sono stati utilizzati altre due componenti:

- Breadboard: è uno strumento che permette di creare prototipi di circuiti elettronici senza la necessità di saldare i componenti.
  Questa è costituita da una base in plastica con fori in cui possono essere inseriti i componenti elettronici come resistenze, condensatori, transistor e altri dispositivi. La breadboard ha anche una serie di linee di collegamento disposte in modo specifico, che consentono di connettere i componenti in modo da creare circuiti elettronici funzionanti.

- Modulo LED-RGB: è un dispositivo che consente di controllare l'illuminazione di diodi a emissione luminosa (LED) di diversi colori (rosso, verde e blu) attraverso un microcontroller. Questo modulo è costituito da tre LED, uno rosso, uno verde e uno blu, che possono essere controllati separatamente attraverso i pin di ingresso del modulo. Utilizzando tecniche di modulazione di larghezza di impulso (PWM), è possibile variare l'intensità di ciascuno dei tre LED, permettendo di creare una vasta gamma di colori. Inoltre, i tre LED possono essere accesi e spenti in modo indipendente, consentendo di creare effetti di illuminazione complessi.

In particolare, il feedback è stato strutturato nel seguente modo: un flash con il LED blu indica che è stata eseguita una modifica di un parametro, mentre un LED rosso continuo indica che si è verificato un errore durante una modifica di un parametro. Infine, per visualizzare maggiori dettagli relativi agli errori, è necessario controllare la console dei log.

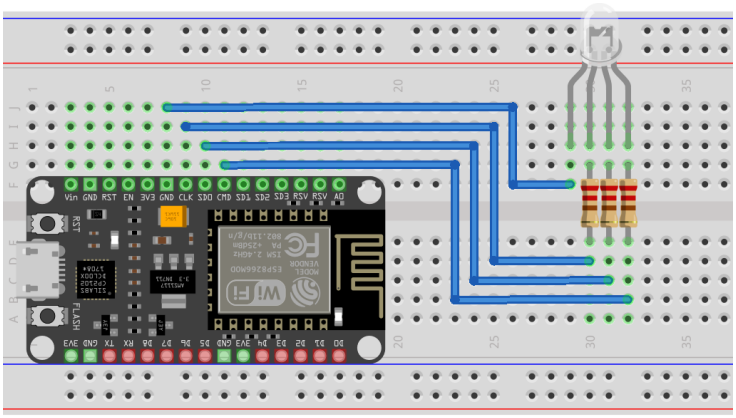


Figura 4.3: Configurazione ESP32 utilizzato

Capitolo 5

Validazione

Questo capitolo è dedicato alla presentazione dei risultati ottenuti tramite i test effettuati con due configurazioni differenti. Nella prima configurazione, i dispositivi sono stati dislocati, in modo tale che il nodo client possa raggiungere direttamenrte tutti i nodi server, mentre nella seconda configurazione, il nodo client riesce a raggiungere direttamente solo alcuni nodi server. Quindi, sono stati utilizzati dei setup realistici, che hanno permesso di simulare in modo più accurato le condizioni effettive di utilizzo del sistema. Grazie a questa scelta, i risultati ottenuti sono maggiormente significativi e rappresentativi della situazione reale, fornendo una maggiore affidabilità ai dati raccolti. Sarà quindi possibile analizzare in dettaglio le prestazioni delle configurazioni nella situazione di utilizzo effettiva, mettendo in evidenza eventuali problematiche o criticità che potrebbero verificarsi durante l'utilizzo dei sistemi.

5.1 Metriche

Per poter valutare la prestazione della rete sono stati presi in considerazione due metriche:

- Latenza (Delay)
- Packet Delivery Ratio (PDR)

5.1.1 Latenza

La latenza, uno dei concetti fondamentali nel mondo delle reti di comunicazione, è un indicatore del tempo necessario affinché un messaggio inviato da un nodo raggiunga il suo destinatario. In altre parole, la latenza misura il ritardo tra l'invio di un messaggio e la sua ricezione da parte del destinatario.
Per calcolare la latenza all'interno di una rete, viene utilizzata la misurazione in millisecondi del Round Trip Time (**RTT**), ovvero il tempo necessario per inviare un messaggio

dal nodo Client ai nodi Server e ricevere la risposta dal Server al Client. In questo modo, si ottiene una stima precisa della quantità di tempo richiesta per trasferire i dati all'interno della rete. In seguito è riportata la formula utilizzata per il calcolo della latenza.

$$latency = \frac{finalTime - startTime}{2} \tag{5.1}$$

5.1.2 Packet Delivery Ratio

Il Packet Delivery Ratio (PDR) rappresenta il tasso tra il numero di pacchetti ricevuti con successo e il numero totale di pacchetti inviati attraverso la rete. È una misura fondamentale utilizzata nelle reti di comunicazione per valutare l'efficienza e l'affidabilità della trasmissione dei pacchetti di dati.
Il valore del PDR è un indicatore importante della qualità della trasmissione dei dati all'interno di una rete. In particolare, un PDR elevato indica una maggiore affidabilità e una migliore efficienza nella trasmissione dei dati, mentre un PDR basso indica problemi di connessione e di congestione della rete. In seguito è riportata la formula utilizzata per il calcolo del PDR.

$$PDR = \frac{\sum packetReceived}{\sum packetSent} \tag{5.2}$$

5.1.3 Analisi Dati

Per l'analisi dei dati, sono stati utilizzati degli script Python in grado di acquisire le informazioni da un file CSV estratto dal database MySQL, come già illustrato in precedenza. Successivamente, è stata eseguita una fase di preprocessing al fine di preparare il dataset per la generazione dei grafici che saranno presentati di seguito. Tale procedura ha permesso di ottenere una rappresentazione visiva dei dati raccolti, rendendo più facile l'interpretazione e la valutazione dei risultati ottenuti.
Di seguito viene riportata una tabella contenente tutti i valori assegnati alle variabili durante questa fase di analisi.

TTL	2, 3, 4, 5
Potenza di trasmissione	Bassa (-12 dBm), Media (-9 dBm), Alta (-3 dBm)
Numero di trasmissioni	1, 3, 5
Intervallo di trasmissione	20ms, 100ms, 250ms, 500ms, 1000ms

Per garantire l'affidabilità dei grafici prodotti durante l'analisi, si è reso necessario raccogliere i dati relativi a ogni combinazione di parametri in diversi giorni, per un totale di 4 volte. In questo modo, si è cercato di evitare che le condizioni temporanee, come ad esempio la presenza di interferenze esterne, influenzassero i risultati dell'analisi.

5.2 Primo scenario: single-hop

Il primo scenario nel quale verrà utilizzato il sistema implementato è rappresentato da un ambiente indoor all'interno di un contesto urbano. Nello specifico, i dispositivi in oggetto sono stati dislocati all'interno di due stanze di un appartamento. La figura 5.5 riporta la planimetria dell'appartamento in questione, riportando la posizione esatta della dislocazione dei dispositivi. In particolare, è stato posizionato al centro il nodo client (indicato in rosso) ed intorno sono stati allocati i nodi server.

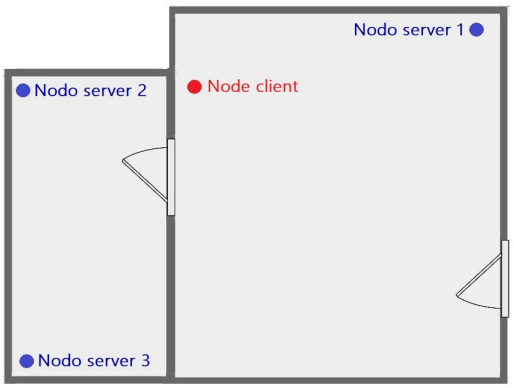


Figura 5.1: Planimetria con la rete BLE Mesh - primo scenario

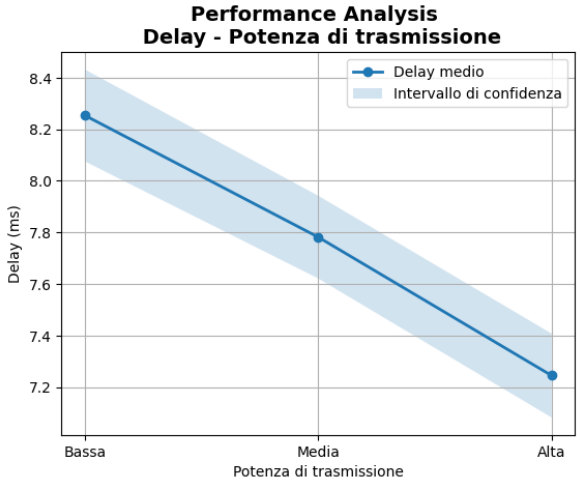
5.2.1 Single-hop: risultati

In questa sezione verranno analizzati i risultati ottenuti con la configurazione **single-hop** in relazione alle due metriche descritte in precedenza.

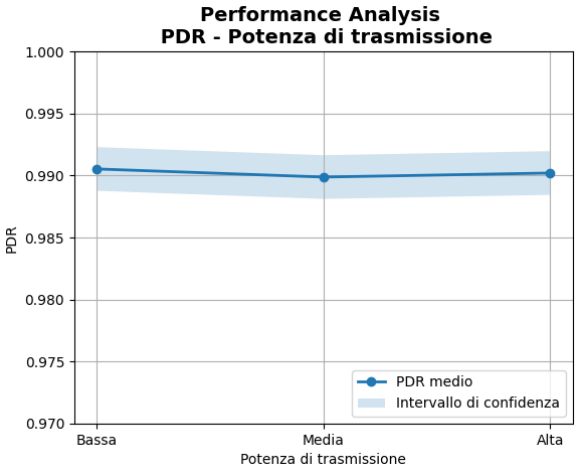
Potenza di trasmissione - Latenza

La figura 5.2a rappresenta la relazione tra la potenza di trasmissione e la latenza di rete. In particolare, sull'asse delle x, sono riportati i valori della potenza, mentre sull'asse delle y vengono riportati i valori della latenza. Dall'analisi del grafico emerge che un aumento della potenza di trasmissione si traduce in un miglioramento delle prestazioni della rete e quindi in una riduzione della latenza.

Mentre la figura 5.2b rappresenta la relazione tra la potenza di trasmissione e il PDR (Packet Delivery Ratio) della rete. Dall'analisi del grafico si evince che il valore del PDR non è influenzato dalla potenza di trasmissione.



(a) Il grafico rappresenta la relazione tra la potenza di trasmissione e la latenza.



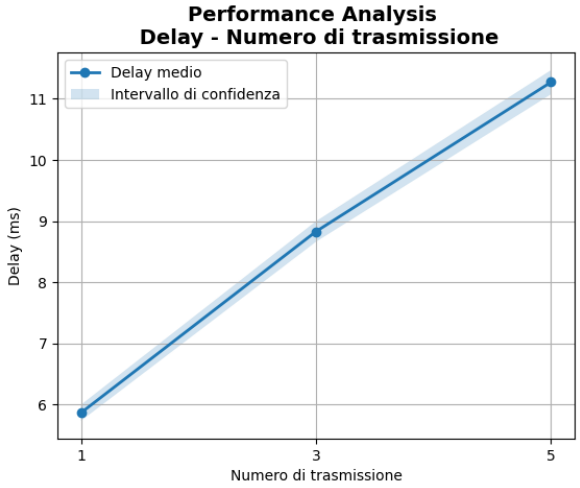
(b) Il grafico rappresenta la relazione tra la potenza di trasmissione e il PDR.

Figura 5.2: Grafici single-hop: potenza di trasmissione

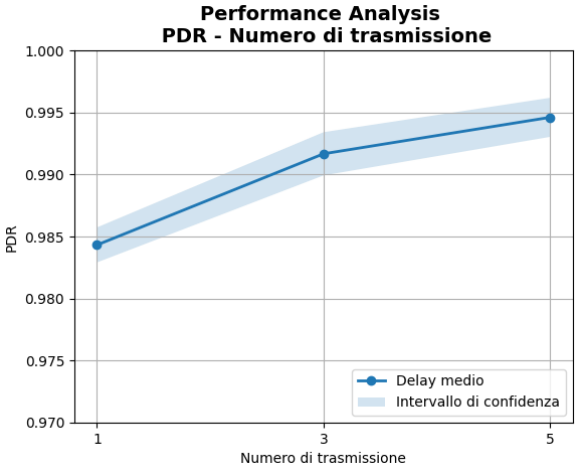
Numero di trasmissione - Latenza

La figura 5.3a rappresenta la relazione tra il numero di trasmissioni e la latenza della rete. Attraverso l'analisi di questo grafico, si può osservare che all'aumentare del numero di trasmissioni, la latenza della rete tende ad aumentare in maniera quasi lineare. Tale fenomeno implica un peggioramento delle prestazioni della rete, poiché il tempo di trasmissione dei dati risulta prolungato.
Di conseguenza, è fondamentale valutare attentamente il numero di trasmissioni che occorre effettuare per garantire un livello ottimale di prestazioni della rete. Un numero eccessivo di trasmissioni può infatti determinare un significativo aumento della latenza, con conseguente diminuzione delle prestazioni.

Mentre la figura 5.3b rappresenta la relazione tra il numero di trasmissioni e il PDR (Packet Delivery Ratio) della rete. Dall'analisi del grafico si evince che il valore del PDR non è influenzato dal numero di trasmissioni.



(a) Il grafico rappresenta la relazione tra il numero di trasmissioni e la latenza



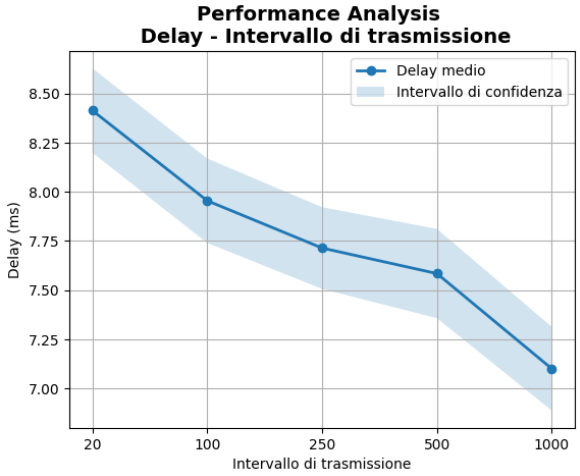
(b) Il grafico rappresenta la relazione tra il numero di trasmissioni e il PDR.

Figura 5.3: Grafici single-hop: numero di trasmissioni

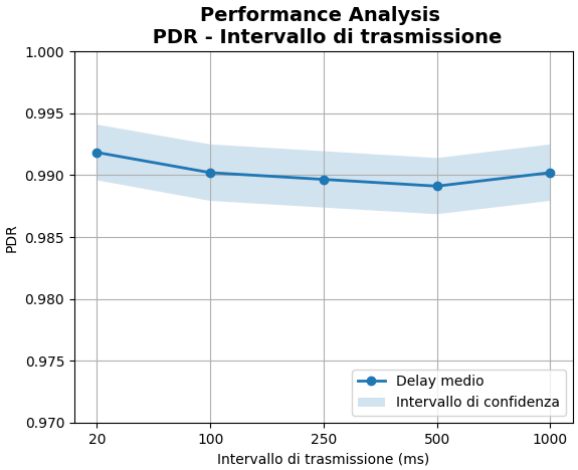
Intervallo tra le trasmissioni - Latenza

La figura 5.4a rappresenta la relazione tra l'intervallo di tempo tra una trasmissione e quella successiva e la latenza della rete. Attraverso l'analisi di questo grafico, si può osservare che all'aumentare dell'intervallo di tempo tra le trasmissioni, la latenza della rete tende a diminuire progressivamente. Questo fenomeno è riconducibile alla congestione della rete, in quanto un intervallo di trasmissione elevato permette di limitare la congestione della rete e quindi permette di ottenere una migliore latenza della rete.

Mentre la figura 5.4b rappresenta la relazione tra l'intervallo di trasmissione e il PDR (Packet Delivery Ratio) della rete. Dall'analisi del grafico si evince che il valore del PDR non è influenzato dall'intervallo di trasmissione.



(a) Il grafico rappresenta la relazione tra l'intervallo di trasmissioni e la latenza.



(b) Il grafico rappresenta la relazione tra l'intervallo di trasmissione e il PDR.

Figura 5.4: Grafici single-hop: intervallo tra le trasmissioni

5.3 Secondo scenario: multi-hop

Anche il secondo scenario nel quale verrà utilizzato il sistema implementato è rappresentato da un ambiente indoor all'interno di un contesto urbano. A differenza del primo scenario, i dispositivi sono stati dislocati all'interno di due appartamenti. Quindi, è aumentata la distanza tra i vari dispositivi. La figura 5.5 riporta la planimetria dei due appartamenti in questione, riportando la posizione esatta della dislocazione dei dispositivi. In particolare, il nodo client non è in grado di raggiungere direttamente tutti i nodi, pertanto è necessario sfruttare una caratteristica fondamentale del BLE Mesh, ovvero l'utilizzo dei nodi intermediari, per raggiungere anche i nodi più distanti.

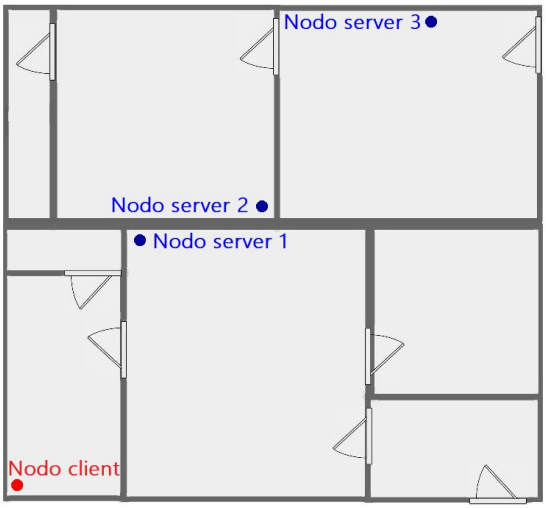


Figura 5.5: Planimetria con la rete BLE Mesh - secondo scenario

5.3.1 Multi-hop: risultati

In questa sezione verrano analizzati i risultati ottenuti con la configurazione **multi-hop** in relazione alle due metriche descritte in precedenza. In particolare, saranno esaminati i risultati associati al nodo server 3, poiché è l'unico nodo al quale il nodo client non riesce a connettersi direttamente.

Potenza di trasmissione - Latenza

La figure 5.6 rappresenta la relazione tra la potenza di trasmissione e la latenza della rete. Dall'analisi del grafico emerge che l'aumento della potenza di trasmissione migliora le prestazioni della rete, riducendo la latenza. Questo risultato è dovuto al minor numero di salti che il messaggio deve compiere per giungere al nodo server 3. Con una potenza di trasmissione bassa, infatti, il messaggio deve effettuare 3 salti, mentre con una potenza media/alta, il nodo client raggiunge il nodo server 3 impiegando solo 2 salti.

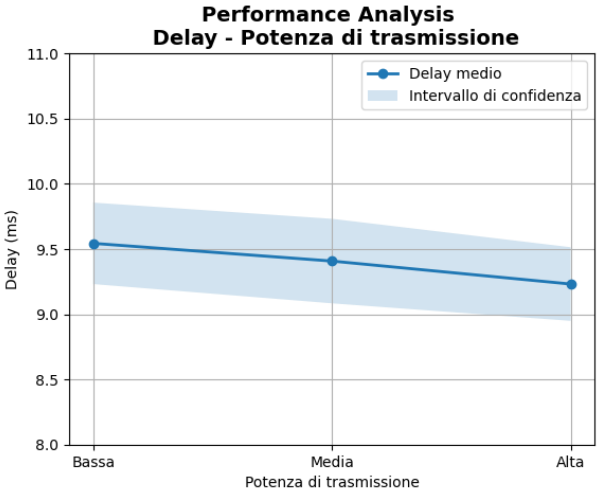
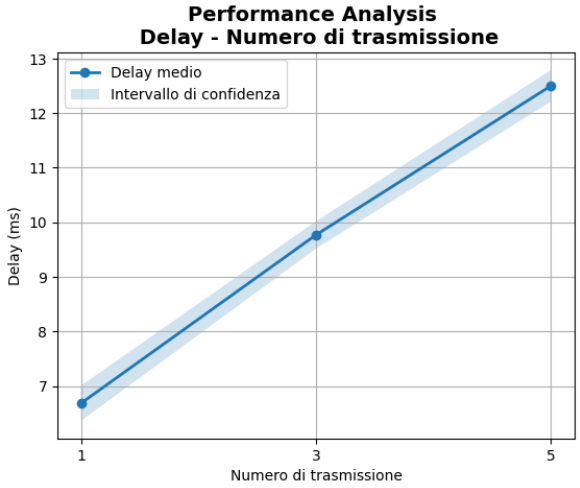


Figura 5.6: Il grafico rappresenta la relazione tra la potenza di trasmissione e la latenza.

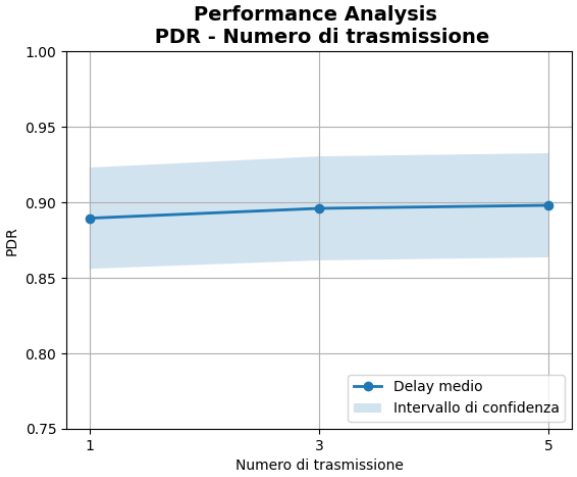
Numero di trasmissioni - Latenza

La figura 5.7a rappresenta la relazione tra il numero di trasmissioni e la latenza della rete. Dall'analisi del grafico si possono trarre le stesse conclusioni dell'analisi precedente riguardante la rete single-hop. In particolare, è emerso che incrementando il numero di trasmissioni si verifica un peggioramento della performance della rete. Tale effetto è imputabile all'aumento della congestione della rete.

Mentre la figura5.7b rappresenta la relazione tra il numero di trasmissioni e il PDR (Packet Delivery Ratio) della rete. Dall'analisi del grafico si evince che il valore del PDR non è influenzato dal numero di trasmissioni.



(a) Il grafico rappresenta la relazione tra il numero di trasmissione e la latenza.



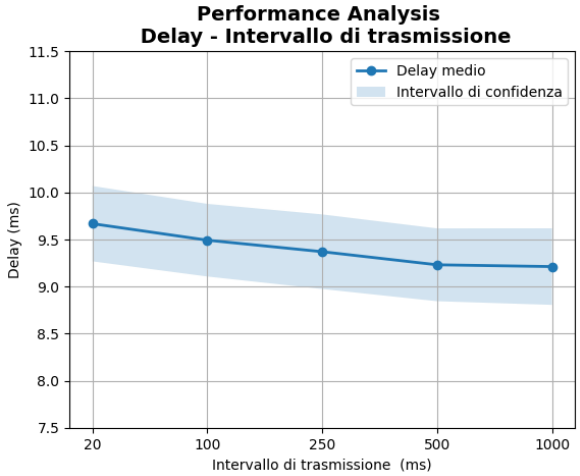
(b) Il grafico rappresenta la relazione tra il numero di trasmissioni e il PDR.

Figura 5.7: Grafici multi-hop: numero di trasmissioni

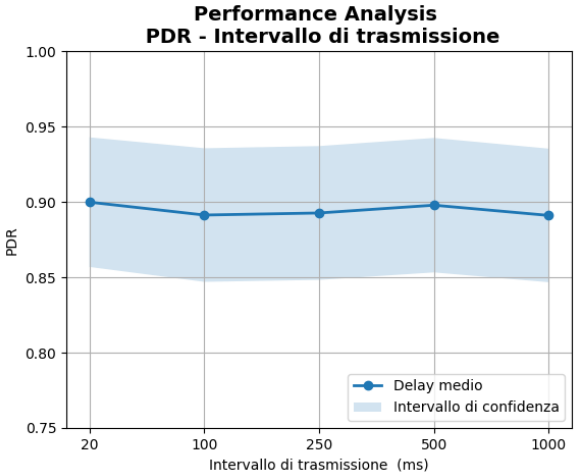
Intervallo di trasmissione - Latenza

La figura 5.8a rappresenta la relazione tra l'intervallo di tempo tra una trasmissione e quella successiva e la latenza della rete. Attraverso l'analisi di questo grafico, si possono trarre le stesse conclusioni dell'analisi precedente riguardante la rete single-hop, ovvero che all'aumento dell'intervallo di tempo tra le trasmissioni la latenza della rete tende a diminuire progressivamente. Questo fenomeno è riconducibile alla congestione della rete, in quanto un intervallo di trasmissione elevato permette di limitare la congestione della rete e quindi permette di ottenere una migliore latenza della rete.

Mentre la figura 5.8b rappresenta la relazione tra l'intervallo di trasmissione e il PDR (Packet Delivery Ratio) della rete. Dall'analisi del grafico si evince che il valore del PDR non è influenzato dall'intervallo trasmissione.



(a) Il grafico rappresenta la relazione tra l'intervallo di trasmissione e la latenza



(b) Il grafico rappresenta la relazione tra l'intervallo di trasmissione e il PDR.

Figura 5.8: Grafici multi-hop: intervallo di trasmissione

TTL - PDR

La figura 5.9 riporta due grafici che rappresentano la relazione del TTL e la latenza della rete:

- il primo grafico, prendendo in considerazione un contesto nel quale tutti i dispositivi hanno una potenza di trasmissione bassa
- il secondo grafico, prendendo in considerazione un contesto nel quale tutti i dispositivi hanno una potenza di trasmissione media/alta.

Dalle analisi dei grafici, si può dedurre che nel primo caso il messaggio non giunge al nodo server 3 se il TTL è impostato al di sotto o uguale a 2, perché a causa di una bassa potenza di trasmissione servono almeno 3 salti per raggiungere la destinazione. Nel secondo caso, con un TTL pari o superiore a 2, il messaggio viene invece ricevuto dal nodo server 3 poiché una potenza di trasmissione media/alta rende sufficienti soltanto 2 salti per raggiungere il nodo server 3.

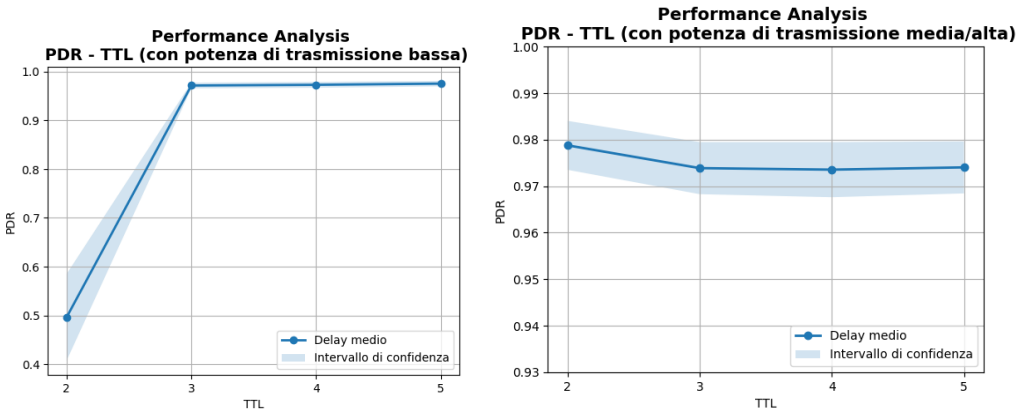


Figura 5.9: I grafici rappresenta la relazione tra TTL e la PDR

5.3.2 Conclusione

Dopo l'analisi svolta, è possibile affermare che la latenza della rete è influenzata da diversi parametri, quali la potenza di trasmissione, il numero di trasmissioni e l'intervallo tra le trasmissioni. Nel dettaglio, un incremento della potenza di trasmissione determina una diminuzione della latenza, poiché il nodo client riesce a stabilire una connessione diretta con un maggior numero di nodi server. L'aumento del numero di trasmissioni, al contrario, determina un aumento della latenza, poiché il traffico sulla rete diventa più congestionato. Invece, un incremento dell'intervallo tra le trasmissioni porta ad una diminuzione della latenza, poiché il traffico sulla rete risulta meno congestionato.

Mentre, il Packet Delivery Ratio (PDR) viene condizionato dalla potenza di trasmissione e dal Time To Live (TTL), poiché tali parametri sono in grado di influenzare l'estensione massima della rete.

5.4 Automatic configuration

Nella presente tesi sono stati impiegati cinque algoritmi di intelligenza artificiale, ovvero Decision Tree, Random Forest, Gradient Boosting, KNN e Gaussian Naive Bayes. Grazie a ciascun algoritmo, sono state generate diverse configurazioni, utilizzando soltanto valori di Dalay o PDR come input. In questa sezione, verranno illustrati i risultati ottenuti, nonché presentati dei grafici volti a misurare l'effettiva prestazione conseguita con le configurazioni ottenute tramite i predetti algoritmi di A.I. Grazie a tali grafici potremo valutare quanto i risultati ottenuti siano accurati per ciascun algoritmo.

5.4.1 Automatic configuration: Risultati

In questa sezione vengono presentati tabelle e grafici che consentono di analizzare l'efficacia degli algoritmi di Intelligenza Artificiale impiegati. In particolare, per ciascun algoritmo sono stati prodotti due tabelle e due grafici. La prima tabella illustra la configurazione che gli algoritmi di AI hanno individuato per raggiungere i livelli di latenza di 5ms, 7.5ms, 10ms, 12.5ms e 15ms. La seconda tabella invece riporta la configurazione che gli algoritmi di AI hanno elaborato per raggiungere PDR di 0.5, 0.6, 0.7, 0.8, 0.9 e 1. Infine, i due grafici mostrano il livello effettivo di prestazione della rete, utilizzando le configurazioni ottenute dai vari agloritmi di A.I.

Decision Tree

Delay:

Input	**Nodo ID**	**TTL**	**Potenza di trasm.**	**Numero di trasm.**	**Intervallo di trasm.**	**Delay effettivo**	**PDR effettivo**
5 ms	3	4	Media	1	100 ms	5,19	0,98
7,5 ms	3	3	Alta	1	250 ms	8,76	1
10 ms	3	3	Alta	3	250 ms	11,58	0,965
12,5 ms	3	3	Alta	3	500 ms	12,53	0,975
15 ms	3	3	Alta	5	1000 ms	19,58	1

Tabella 5.1: Risultati Decisione Tree - Delay

PDR:

Input	**Nodo ID**	**TTL**	**Potenza di trasm.**	**Numero di trasm.**	**Intervallo di trasm.**	**Delay effettivo**	**PDR effettivo**
0,5	3	3	Media	3	1000 ms	19	0,95
0,6	3	2	Alta	3	20 ms	7,78	0,97
0,7	3	4	Bassa	5	20 ms	8,17	0,985
0,8	3	4	Bassa	3	20 ms	6,23	0,96
0,9	3	2	Alta	5	1000 ms	12,97	1
1	3	4	Alta	5	20 ms	5,85	0,96

Tabella 5.2: Risultati Decisione Tree - PDR

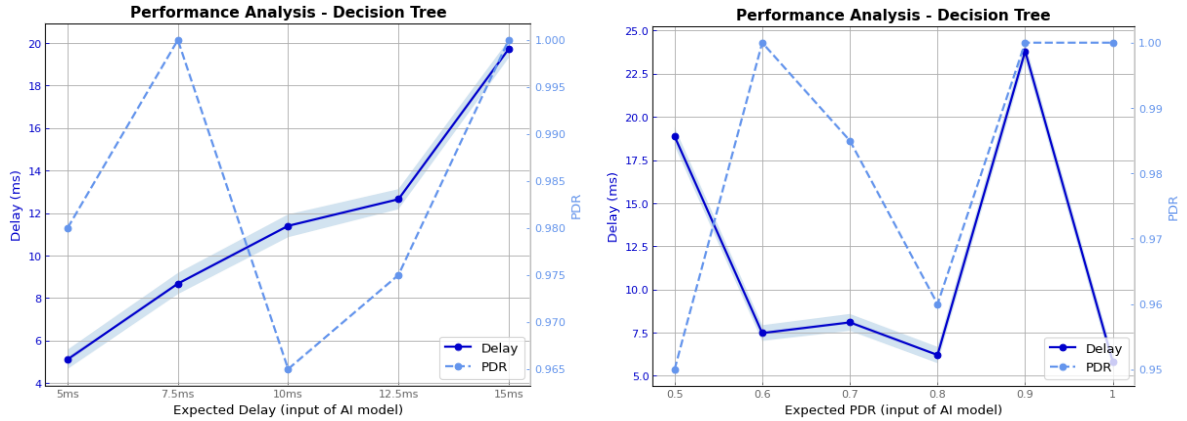


Figura 5.10: Il grafico rappresenta le performance della rete in base alle configurazioni generate dall'algoritmo Decision Tree

Random Forest

Delay:

Delay	**Node ID**	**TTL**	**Potenza di trasm.**	**Numero di trasm.**	**Intervallo di trasm.**	**Delay effettivo**	**PDR effettivo**
5	3	4	Media	1	100 ms	4,69	1
7,5	3	3	Alta	5	1000 ms	15,16	1
10	3	3	Alta	5	1000 ms	17,07	0,96
12,5	3	3	Alta	5	1000 ms	16,83	1
15	3	3	Alta	5	1000 ms	17,9	0,94

Tabella 5.3: Risultati Random Forest - Delay

PDR:

PDR	**Node ID**	**TTL**	**Potenza di trasm.**	**Numero di trasm.**	**Intervallo di trasm.**	**Delay effettivo**	**PDR effettivo**
0,5	3	4	Media	3	500 ms	8,33	0,96
0,6	3	4	Media	3	500 ms	10,64	1
0,7	3	4	Media	3	500 ms	9,64	1
0,8	3	4	Media	3	500 ms	11,23	0,98
0,9	3	4	Media	3	500 ms	10	1
1	3	4	Media	3	500 ms	8,7	1

Tabella 5.4: Risultati Random Forest - PDR

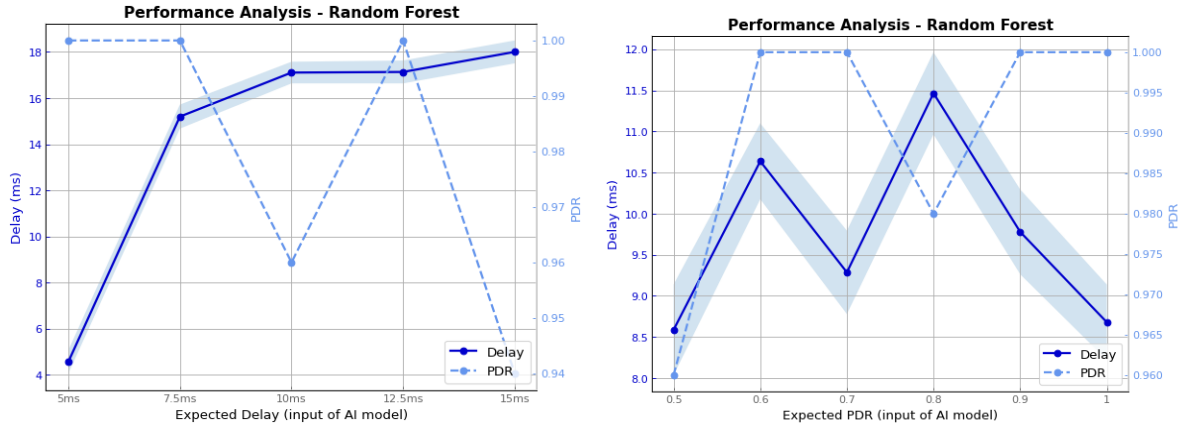


Figura 5.11: Il grafico rappresenta le performance della rete in base alle configurazioni generate dall'algoritmo Random Forest.

Gradient Boosting

Delay:

Delay	Node ID	TTL	Potenza di trasm.	Numero di trasm.	Intervallo di trasm.	Delay effettivo	PDR effettivo
5	3	4	Media	3	250 ms	7,58	1
7,5	3	4	Media	3	250 ms	8,59	1
10	3	4	Media	3	250 ms	8,86	1
12,5	3	4	Media	5	250 ms	10,89	0,92
15	3	4	Media	5	250 ms	11,65	1

Tabella 5.5: Risultati Gradient Boosting - Delay

PDR:

PDR	Node ID	TTL	Potenza di trasm.	Numero di trasm.	Intervallo di trasm.	Delay effettivo	PDR effettivo
0,5	3	4	Media	1	250 ms	3,33	0,96
0,6	3	4	Media	1	250 ms	3,87	1
0,7	3	4	Media	1	250 ms	4,07	1
0,8	3	4	Media	1	250 ms	5,05	1
0,9	3	4	Media	3	250 ms	10,01	1
1	3	4	Media	3	250 ms	7,58	1

Tabella 5.6: Risultati Gradient Boosting - PDR

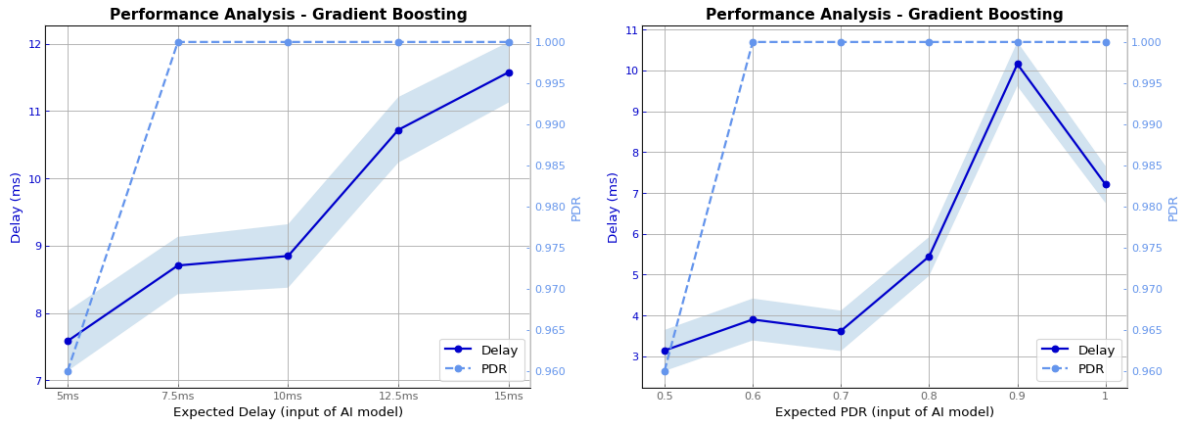


Figura 5.12: Il grafico rappresenta le performance della rete in base alle configurazioni generate dall'algoritmo Gradient Boosting

KNN

Delay:

Delay	Node ID	TTL	Potenza di trasm.	Numero di trasm.	Intervallo di trasm.	Delay effettivo	PDR effettivo
5	3	4	Media	1	100 ms	4,4	1
7,5	3	3	Alta	5	1000 ms	8,16	0,96
10	3	3	Alta	5	1000 ms	15,15	1
12,5	3	3	Alta	5	1000 ms	10,88	1
15	3	3	Alta	5	1000 ms	7,09	1

Tabella 5.7: Risultati KNN - Delay

PDR:

PDR	Node ID	TTL	Potenza di trasm.	Numero di trasm.	Intervallo di trasm.	Delay effettivo	PDR effettivo
0,5	3	2	Media	1	20 ms	9,39	1
0,6	3	2	Media	1	20 ms	6,41	1
0,7	3	2	Media	1	20 ms	6,12	0,98
0,8	3	2	Media	1	20 ms	9,56	1
0,9	3	2	Media	1	20 ms	7,03	1
1	3	2	Media	1	20 ms	8,74	1

Tabella 5.8: Risultati KNN - PDR

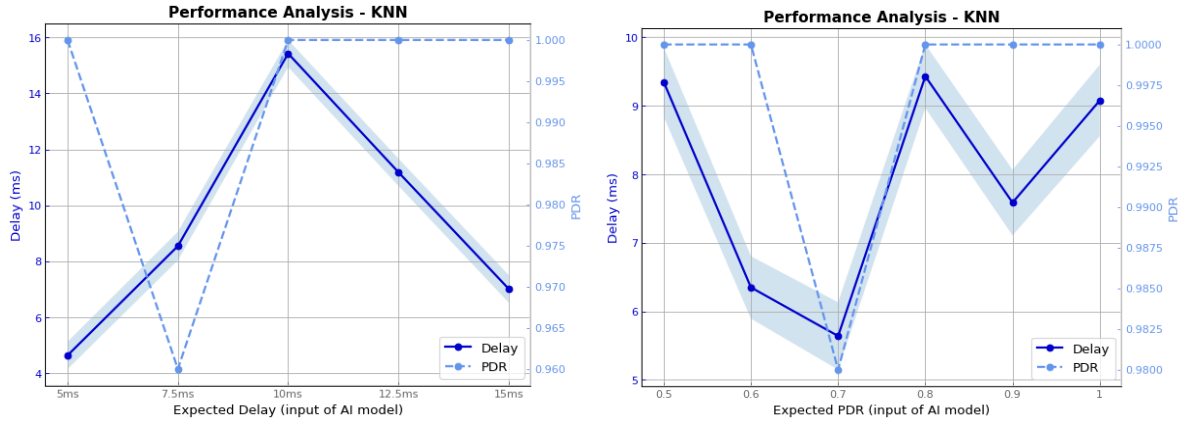


Figura 5.13: Il grafico rappresenta le performance della rete in base alle configurazioni generate dall'algoritmo KNN

Gaussian Naive Bayes

Delay:

Delay	Node ID	TTL	Potenza di trasm.	Numero di trasm.	Intervallo di trasm.	Delay effettivo	PDR effettivo
5	3	5	Bassa	1	1000 ms	3,55	1
7,5	3	5	Bassa	3	1000 ms	7,02	1
10	3	4	Bassa	3	100 ms	10,15	0,94
12,5	3	4	Bassa	3	100 ms	9,95	1
15	3	4	Bassa	5	20 ms	12,07	1

Tabella 5.9: Risultati Gaussian Naive Bayes - Delay

PDR:

PDR	Node ID	TTL	Potenza di trasm.	Numero di trasm.	Intervallo di trasm.	Delay effettivo	PDR effettivo
0,5	3	3	Bassa	5	20 ms	8,09	1
0,6	3	3	Bassa	1	20 ms	14,9	1
0,7	3	3	Bassa	1	20 ms	8,62	1
0,8	3	3	Bassa	1	20 ms	8,05	1
0,9	3	3	Bassa	1	100 ms	5,14	1

Tabella 5.10: Risultati Gaussian Naive Bayes - PDR

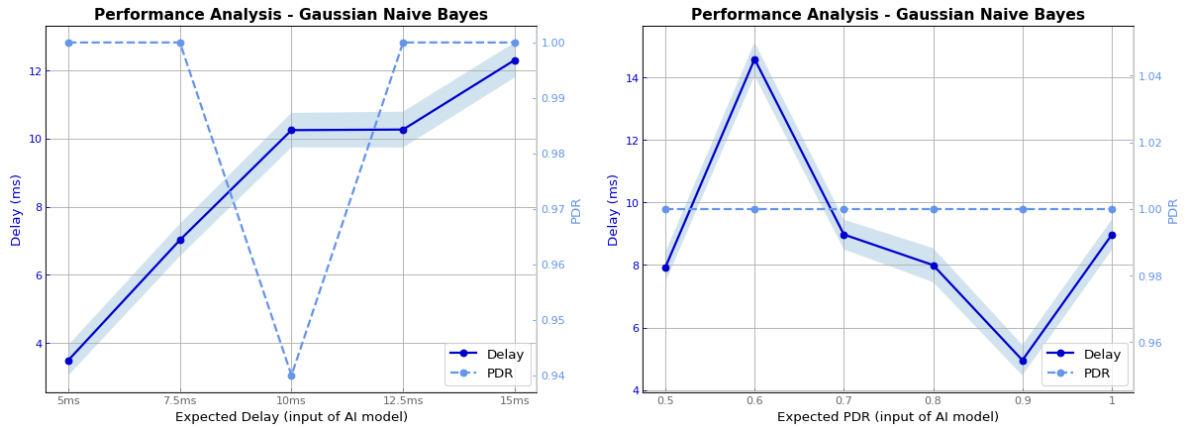


Figura 5.14: Il grafico rappresenta le performance della rete in base alle configurazioni generate dall'algoritmo Gaussian Naive Bayes

Actual vs Expected

In seguito è riportato il grafico che mettono in relazione il delay della rete con il delay atteso, che viene utilizzato come parametro di input per gli algoritmi di intelligenza artificiale. Dalla valutazione di questo grafico, si può concludere che il Gaussian Naive Bayes e gli algoritmi basati su alberi hanno dimostrato risultati eccellenti, in particolare, l'algoritmo di Decision Tree è stato in grado di identificare ottimamente i parametri necessari per raggiungere un determinato latenza.

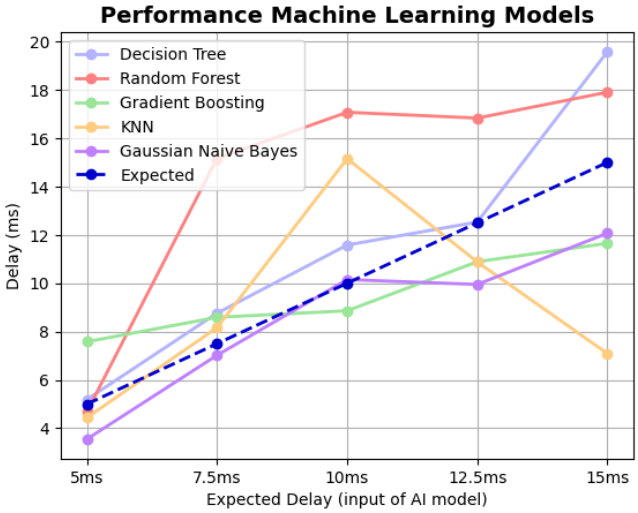


Figura 5.15: AI: Actual vs expected

Capitolo 6

Conclusione e sviluppi futuri

La tesi si concentra sulla progettazione e implementazione di un sistema di auto - configurazione per le reti BLE Mesh, un tipo di rete wireless basata sulla tecnologia Bluetooth Low Energy, che si sta sempre più diffondendo grazie alla sua capacità di supportare molteplici dispositivi con basso consumo energetico.

Il sistema di auto-configurazione sviluppato è in grado di configurare quattro parametri fondamentali per la prestazione della rete: TTL, potenza di trasmissione, numero di trasmissioni e intervallo di trasmissione. Questo sistema è composto da due fasi: nella prima fase vengono rilevate le condizioni della rete, mentre nella seconda fase vengono individuati i parametri ottimali per raggiungere le prestazioni desiderate dall'utente.
Per individuare i valori ottimali dei parametri, il sistema di auto-configurazione utilizza degli algoritmi di apprendimento automatico come algoritmi basati su alberi decisionali, Gaussian Naive Bayes e KNN.

Per interagire con il sistema di auto-configurazione, è stata sviluppata una dashboard, che consente all'utente di avviare le due fasi di auto-configurazione e di impostare i singoli parametri su tutti i dispositivi o su un singolo dispositivo. In questo modo, l'utente gestire la rete in modo efficiente e intuitivo.

Successivamente, sono stati realizzati due setup per raccogliere dati sulla performance della rete in diverse condizioni. Ed è stata fatta un'analisi sull'impatto che un determinato parametro ha sulla performance della rete. In particolare, la potenza di trasmissione, il numero di trasmissione e l'intervallo di trasmissione incidono sulla latenza della rete, mentre il TTL e la potenza di trasmissione incidono sul packet delivery ratio.

6.1 Sviluppi futuri

In questa sezione, si analizzeranno alcuni possibili sviluppi futuri per estendere il lavoro di tesi.

In primo luogo, si potrebbe considerare l'ampliamento dei parametri modificabili dall'utente all'interno dei nodi della rete Bluetooth Low Energy, allo scopo di offrire un maggiore grado di personalizzazione e flessibilità nella modifica dei parametri per ottimizzare le prestazioni della rete.

Inoltre, un'altra direzione di sviluppo interessante sarebbe quella di esplorare l'utilizzo di algoritmi di Reinforcement Learning per migliorare l'apprendimento dell'agente all'interno dell'ambiente. Il Reinforcement Learning (RL) rappresenta una tecnica di apprendimento automatico che si concentra sulla sperimentazione e l'interazione dell'agente con l'ambiente, il quale apprende a scegliere le azioni migliori in base alla funzione di ricompensa. Questo tipo di algoritmo ha dimostrato risultati promettenti in molti campi e potrebbe portare a risultati significativamente migliori nell'ottimizzazione delle prestazioni della rete Bluetooth Low Energy.

Appendice A

Dettagli implementativi

A.1 Codice comune sia al nodo client che server

I firmware utilizzati sugli ESP32 sono stati generati apportando dalla modifica dell'esempio ble_mesh_node fornito da Espressif. Questo esempio include due firmware denominati onoff_client e onoff_server. In particolare, viene utilizzato il modello onoff, il quale permette di accendere o spegnere un LED situato nel nodo server. Sia l'app nRF Mesh che il nodo client sono in grado di controllare i LED tramite un comando specifico per ogni colore (rosso, blu e verde).

A.1.1 BLE Mesh

In seguito, sono riportati tutti gli import che sono stati utilizzati per permette agli ESP32 di accedere a tutte le API messe a disposizione dalla libreria ESP-IDF che riguardano il BLE Mesh.

```
#include "esp_ble_mesh_defs.h"
#include "esp_ble_mesh_common_api.h"
#include "esp_ble_mesh_networking_api.h"
#include "esp_ble_mesh_provisioning_api.h"
#include "esp_ble_mesh_config_model_api.h"
#include "esp_ble_mesh_generic_model_api.h"
#include "esp_ble_mesh_local_data_operation_api.h"
#include "ble_mesh_example_init.h"
```

Infine, il codice riportato in seguito, permette a tutti i nodi di inizializzare il BLE Mesh.

```
static esp_err_t ble_mesh_init(void)
{
    esp_err_t err = ESP_OK;

```

```
    esp_ble_mesh_register_prov_callback(example_ble_mesh_provisioning_cb);
    esp_ble_mesh_register_generic_client_callback(example_ble_mesh_generic_client_cb);
    esp_ble_mesh_register_config_server_callback(example_ble_mesh_config_server_cb);

    err = esp_ble_mesh_init(&provision, &composition);
    if (err != ESP_OK) {
        ESP_LOGE(TAG, "Failed to initialize mesh stack (err %d)", err);
        return err;
    }

    err = esp_ble_mesh_node_prov_enable(ESP_BLE_MESH_PROV_ADV | ESP_BLE_MESH_PROV_GATT);
    if (err != ESP_OK) {
        ESP_LOGE(TAG, "Failed to enable mesh node (err %d)", err);
        return err;
    }

    ESP_LOGI(TAG, "BLE Mesh Node initialized");

    board_led_operation(LED_G, LED_ON);

    return err;
}
```

A.1.2 Modifica parametri

Il codice presentato di seguito consiste in una variabile utilizzata all'interno del protocollo BLE Mesh. Di default, questa variabile viene dichiarata come statica, il che significa che non può essere modificata durante l'esecuzione del codice. Tuttavia, è stata apportata una modifica al codice al fine di rendere possibile la modifica dei valori in essa contenuti a runtime, rimuovendo la dicitura «static».
Questa variabile è utilizzata per regolare diversi parametri del protocollo BLE Mesh, tra cui il TTL (Time To Live), il numero di pacchetti trasmessi e l'intervallo di tempo che deve trascorrere tra l'invio di un pacchetto e quello successivo. La possibilità di modificare questi parametri a runtime consente una maggiore flessibilità nella gestione del protocollo BLE Mesh, adattandolo alle esigenze specifiche dell'applicazione in esecuzione. In definitiva, questa modifica consente di rendere il protocollo BLE Mesh più personalizzabile e versatile, per garantire un'esperienza di utilizzo ottimale agli utenti finali.

```
esp_ble_mesh_cfg_srv_t config_server = {
    .relay = ESP_BLE_MESH_RELAY_DISABLED,
    .beacon = ESP_BLE_MESH_BEACON_ENABLED,
```

```
#if defined(CONFIG_BLE_MESH_FRIEND)
    .friend_state = ESP_BLE_MESH_FRIEND_ENABLED,
#else
    .friend_state = ESP_BLE_MESH_FRIEND_NOT_SUPPORTED,
#endif
#if defined(CONFIG_BLE_MESH_GATT_PROXY_SERVER)
    .gatt_proxy = ESP_BLE_MESH_GATT_PROXY_ENABLED,
#else
    .gatt_proxy = ESP_BLE_MESH_GATT_PROXY_NOT_SUPPORTED,
#endif
    .default_ttl = 7,
    .net_transmit = ESP_BLE_MESH_TRANSMIT(2, 20),
    .relay_retransmit = ESP_BLE_MESH_TRANSMIT(2, 20),
};
```

Il codice riportato di seguito consente di modificare la potenza del Bluetooth Low Energy (BLE), e di conseguenza, influisce sulla potenza della rete mesh. In pratica, la potenza del BLE e la potenza della rete mesh sono strettamente correlate, poiché la potenza del BLE influisce sulla portata dei dispositivi all'interno della rete mesh.

```
#include <esp_bt.h>

esp_err_t set_tx_power(esp_power_level_t power_level) {
    printf('sono dentro a set_tx_power');
    esp_err_t ret;
    ret = esp_ble_tx_power_set(ESP_BLE_PWR_TYPE_CONN_HDL0, power_level)
    ;
    if (ret != ESP_OK) {
        ESP_LOGE(TAG, "Failed to set BLE connection power, error code =
    %x", ret);
    } else {
        ESP_LOGI(TAG, "BLE connection power set to %d dBm", power_level
    );
        board_led_operation(LED_R, LED_ON);
    }
    return ret;
}
```

La funzione sopracitata viene richiamata nel seguente modo: in base al messaggio ricevuto dal nodo, viene individuato il nuovo valore di potenza da impostare. Nel dettaglio:

- Se il contenuto del messaggio è 1, bisogna impostare la potenza di trasmissione a ESP_PWR_LVL_N12, che corrisponde a -12dbm.
- Se il contenuto del messaggio è 2, bisogna impostare la potenza di trasmissione a ESP_PWR_LVL_N9, che corrisponde a -9dbm.

- Se il contenuto del messaggio è 3, bisogna impostare la potenza di trasmissione a ESP_PWR_LVL_N6, che corrisponde a -6dbm.
- Se il contenuto del messaggio è 4, bisogna impostare la potenza di trasmissione a ESP_PWR_LVL_N3, che corrisponde a -3dbm.
- Se il contenuto del messaggio è 5, bisogna impostare la potenza di trasmissione a ESP_PWR_LVL_N0, che corrisponde a -0dbm.
- Se il contenuto del messaggio è 6, bisogna impostare la potenza di trasmissione a ESP_PWR_LVL_P3, che corrisponde a +3dbm.
- Se il contenuto del messaggio è 7, bisogna impostare la potenza di trasmissione a ESP_PWR_LVL_P6, che corrisponde a +6dbm.
- Se il contenuto del messaggio è 8, bisogna impostare la potenza di trasmissione a ESP_PWR_LVL_P9, che corrisponde a +9dbm.

Di seguito è riportato un esempio di come viene richiamata la funzione sopra descritta.

```
if (read_data == 1) {
    set_tx_power(ESP_PWR_LVL_N12);
} else if (read_data == 2){
    set_tx_power(ESP_PWR_LVL_N9);
} else if (read_data == 3){
    set_tx_power(ESP_PWR_LVL_N6);
} else if (read_data == 4){
    set_tx_power(ESP_PWR_LVL_N3);
} else if (read_data == 5){
    set_tx_power(ESP_PWR_LVL_N0);
} else if (read_data == 6){
    set_tx_power(ESP_PWR_LVL_P3);
} else if (read_data == 7){
    set_tx_power(ESP_PWR_LVL_P6);
} else if (read_data == 8){
    set_tx_power(ESP_PWR_LVL_P9);
```

A.2 Nodo Client

Nel contesto dello sviluppo di un nodo client, è stato necessario integrare due protocolli differenti, ovvero il protocollo BLE Mesh e il protocollo MQTT. Per consentire il funzionamento simultaneo di entrambi i protocolli, è stato creato un doppio stack all'interno del nodo client, costituito proprio dai suddetti protocolli. Tuttavia, per poter inserire questo doppio stack all'interno del dispositivo ESP32, è stato necessario apportare alcune modifiche alla tabella di partizione predefinita. Grazie a questa modifica, il nodo

client è in grado di supportare entrambi i protocolli in modo efficiente e affidabile, rappresentando così una soluzione ideale per le applicazioni che richiedono la coesistenza di più protocolli all'interno di un unico dispositivo. In particolare, la nuova tabella di partizione è stata strutturata come segue:

Name	Type	SubType	Offset	Size	Flags
nvs	data	nvs	0x9000	24K	
phy_init	data	phy	0xf000	4K	
factory	app	factory	0x20000	2M	

Tabella A.1: Tabella di partizione

In pratica, è stato necessario creare un file CSV con i dati riportati nella tabella soprastante. Successivamente, per consentire al nodo Client di impostare la partizione desiderata, è stato necessario eseguire alcuni comandi tramite il terminale di ESP-IDF:

- digitare **idf.py menuconfig** per accedere al menu di configurazione
- accedere alla sezione **serial flasher config**
- accedere alla sezione **flash size** e selezionare **4M**
- ritornare al menu iniziale e selezionare **partition table**
- selezionare **custom partition table** e riportare il nome del file CSV precedentemente creato

A.2.1 Protocollo MQTT

Di seguito, sono elencati gli import necessari per utilizzare il protocollo MQTT all'interno del nodo Client.

```
#include <stdint.h>
#include <stddef.h>
#include "esp_wifi.h"
#include "esp_system.h"
#include "esp_event.h"
#include "esp_netif.h"
#include "protocol_examples_common.h"

#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
```

```
#include "freertos/semphr.h"
#include "freertos/queue.h"

#include "lwip/sockets.h"
#include "lwip/dns.h"
#include "lwip/netdb.h"
#include "mqtt_client.h"
```

Come si può facilmente notare, l'utilizzo del Protocollo MQTT richiede una connessione attiva ad una rete WiFi. Pertanto, oltre a importare le API necessarie per accedere alle funzioni del Protocollo MQTT, è fondamentale importare anche i moduli che permettono al nodo di stabilire una connessione WiFi. In particolare, è necessario importare i moduli che consentono al nodo di connettersi alla rete, gestire le credenziali di autenticazione e mantenere attiva la connessione. Una volta che la connessione è stata stabilita con successo, il nodo Client sarà in grado di utilizzare il Protocollo MQTT per scambiare informazioni con il broker di riferimento. In seguito è riportato il codice che permette al nodo client di inizializzare il modulo Wi-Fi e gestirlo.

```
static esp_err_t wifi_event_handler(void *arg, esp_event_base_t
    event_base, int32_t event_id, void *event_data)
{
    switch (event_id)
    {
    case WIFI_EVENT_STA_START:
        esp_wifi_connect();
        ESP_LOGI(TAG_MQTT, "Trying to connect with Wi-Fi\n");
        break;

    case WIFI_EVENT_STA_CONNECTED:
        ESP_LOGI(TAG_MQTT, "Wi-Fi connected\n");
        break;

    case IP_EVENT_STA_GOT_IP:
        ESP_LOGI(TAG_MQTT, "got ip: startibg MQTT Client\n");
        mqtt_app_start();
        break;

    case WIFI_EVENT_STA_DISCONNECTED:
        ESP_LOGI(TAG_MQTT, "disconnected: Retrying Wi-Fi\n");
        if (retry_cnt++ < MAX_RETRY)
        {
            esp_wifi_connect();
        }
        else
        ESP_LOGI(TAG_MQTT, "Max Retry Failed: Wi-Fi Connection\n");
        break;

```

```
    default:
        break;
    }
    return ESP_OK;
}

void wifi_init(void)
{
    esp_event_loop_create_default();
    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &
    wifi_event_handler, NULL);
    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &
    wifi_event_handler, NULL);

    wifi_config_t wifi_config = {
        .sta = {
            .ssid = WIFI_SSID,
            .password = _WIFI_PASS,
             .threshold.authmode = WIFI_AUTH_WPA2_PSK,
        },
    };
    esp_netif_init();
    esp_netif_create_default_wifi_sta();
    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    esp_wifi_init(&cfg);
    esp_wifi_set_mode(WIFI_MODE_STA);
    esp_wifi_set_config(ESP_IF_WIFI_STA, &wifi_config);
    esp_wifi_start();
}
```

Siccome la dashborad comunica con il nodo client tramite il protocollo MQTT, è necessario che quest'ultimo sia iscritto a determinati topic. In seguito,è riportato il codice che permette di gestire tale dinamica.

```
static void mqtt_event_handler(void *handler_args, esp_event_base_t
    base, int32_t event_id, void *event_data)
{
    ESP_LOGD(TAG_MQTT, "Event dispatched from event loop base=%s,
    event_id=%d", base, event_id);
    esp_mqtt_event_handle_t event = event_data;
    esp_mqtt_client_handle_t client = event->client;
    int msg_id;
    char* result[2];

    switch ((esp_mqtt_event_id_t)event_id)
    {
    case MQTT_EVENT_CONNECTED:
        ESP_LOGI(TAG_MQTT, "MQTT_EVENT_CONNECTED");
```

```
13          MQTT_CONNEECTED=1;
14
15          msg_id = esp_mqtt_client_subscribe(client, "/setTTL", 0);
16          ESP_LOGI(TAG_MQTT, "sent subscribe successful, msg_id=%d",
    msg_id);
17
18          msg_id = esp_mqtt_client_subscribe(client, "/setPower", 0);
19          ESP_LOGI(TAG_MQTT, "sent subscribe successful, msg_id=%d",
    msg_id);
20
21          msg_id = esp_mqtt_client_subscribe(client, "/setTN", 0);
22          ESP_LOGI(TAG_MQTT, "sent subscribe successful, msg_id=%d",
    msg_id);
23
24          msg_id = esp_mqtt_client_subscribe(client, "/setTI", 0);
25          ESP_LOGI(TAG_MQTT, "sent subscribe successful, msg_id=%d",
    msg_id);
26
27          msg_id = esp_mqtt_client_subscribe(client, "/setTTLWithAddress"
    , 0);
28          ESP_LOGI(TAG_MQTT, "sent subscribe successful, msg_id=%d",
    msg_id);
29
30          msg_id = esp_mqtt_client_subscribe(client, "/
    setPowerWithAddress", 0);
31          ESP_LOGI(TAG_MQTT, "sent subscribe successful, msg_id=%d",
    msg_id);
32
33          msg_id = esp_mqtt_client_subscribe(client, "/setTNWithAddress",
     0);
34          ESP_LOGI(TAG_MQTT, "sent subscribe successful, msg_id=%d",
    msg_id);
35
36          msg_id = esp_mqtt_client_subscribe(client, "/setTIWithAddress",
     0);
37          ESP_LOGI(TAG_MQTT, "sent subscribe successful, msg_id=%d",
    msg_id);
38
39          msg_id = esp_mqtt_client_subscribe(client, "/setAllParameters",
     0);
40          ESP_LOGI(TAG_MQTT, "sent subscribe successful, msg_id=%d",
    msg_id);
41
42          msg_id = esp_mqtt_client_subscribe(client, "/
    setAllParametersWithAddress", 0);
43          ESP_LOGI(TAG_MQTT, "sent subscribe successful, msg_id=%d",
    msg_id);
44          break;
45      case MQTT_EVENT_DISCONNECTED:
```

```
46          ESP_LOGI(TAG_MQTT, "MQTT_EVENT_DISCONNECTED");
47          MQTT_CONNEECTED=0;
48          break;
49
50      case MQTT_EVENT_SUBSCRIBED:
51          ESP_LOGI(TAG_MQTT, "MQTT_EVENT_SUBSCRIBED, msg_id=%d", event->
    msg_id);
52          break;
53      case MQTT_EVENT_UNSUBSCRIBED:
54          ESP_LOGI(TAG_MQTT, "MQTT_EVENT_UNSUBSCRIBED, msg_id=%d", event
    ->msg_id);
55          break;
56      case MQTT_EVENT_PUBLISHED:
57          ESP_LOGI(TAG_MQTT, "MQTT_EVENT_PUBLISHED, msg_id=%d", event->
    msg_id);
58          break;
59      case MQTT_EVENT_DATA:
60          ESP_LOGI(TAG_MQTT, "MQTT_EVENT_DATA");
61          /* qui vengono gestiti i messaggi ricevuti. Il codice verra'
    illustrato in seguito */
62          break;
63      case MQTT_EVENT_ERROR:
64          ESP_LOGI(TAG_MQTT, "MQTT_EVENT_ERROR");
65          break;
66      default:
67          ESP_LOGI(TAG_MQTT, "Other event id:%d", event->event_id);
68          break;
69      }
70  }
```

Dal codice soprariportato, si può notare che il nodo client si sottoscrive a 9 topic:

- **/setTTL**: permette di modificare il valore del parametro TTL a tutti i nodi presenti nella rete
- **/setPower**: permette di modificare la potenza di trasmissione a tutti i nodi presenti nella rete
- **/setTN**: permette di modificare il valore del parametro «Transmission Number» a tutti i nodi presenti nella rete
- **/setTI**: permette di modificare il valore del parametro «Transmission Interval» a tutti i nodi presenti nella rete
- **/setTTLWithAddress**: permette di modificare il valore del parametro TTL a un determinato nodo

- **/setPowerWithAddress**: permette di modificare la potenza di trasmissione a un determinato nodo
- **/setTNWithAddress**: permette di modificare il valore del parametro «Transmission Number» a un determinato nodo
- **/setTIWithAddress**: permette di modificare il valore del parametro «Transmission Interval» a un determinato nodo
- **/setAllParametes**: permette di modificare tutti i parametri a tutti i nodi presenti nella rete
- **/setAllParametesWithAddress**: permette di modificare tutti i parametri a un determinato nodo.

A.2.2 Gestione messaggio MQTT

Ogni volta che viene pubblicato un messaggio su un topic al quale il nodo client è sottoscritto, viene richiamato il seguente codice che permette di estrarre la stringa relativa al topic del messaggio.

```
    char read_topic[7];
    for(int i = 0; i <= event->topic_len; i++) {
        if(i == event->topic_len) read_topic[i] = '\0';
        else read_topic[i] = event->topic[i];
    }
```

Successivamente, vengono estratti il valore del messaggio ed eseguito un controllo sulla correttezza del valore:

- Il valore del TTL deve essere compreso tra 0 e 7 compreso.
- Il valore della potenza di trasmissione deve essere compreso tra 1 e 8.
- Il valore del numero di trasmissione deve essere compreso tra 1 e 7.
- Il valore dell'intervallo tra le trasmissioni deve essere un multiplo di 10.

Nel caso in cui il valore del messaggio non sia corretto, non viene effettuato la modifica in nessun nodo ed inoltre, viene accesso il led rosso presente nel nodo client.

```
        if (strcmp(read_topic, "/setTTL") == 0) {
            // Impostare il nuovo valore al nodo client ed inviarlo
   agli altri nodi
            int read_data;
            sscanf(event->data, "%d", &read_data);
```

```
5             printf("TTL: %d", read_data);
6
7             if (read_data>=0 && read_data<=7) {
8                 config_server.default_ttl = read_data;
9                 uint8_t parametro = 0;
10                uint8_t valore = read_data;
11                char address[7] = "0xFFFF";
12                SendNewParameterValue(parametro, valore, address);
13            } else {
14                board_led_operation(LED_R, LED_ON);
15            }
16        }
17
18        if (strcmp(read_topic, "/setPower") == 0) {
19            int read_data;
20            sscanf(event->data, "%d", &read_data);
21            printf("TP: %d", read_data);
22
23            if (read_data == 1) {
24                set_tx_power(ESP_PWR_LVL_N12);
25            } else if (read_data == 2){
26                set_tx_power(ESP_PWR_LVL_N9);
27            } else if (read_data == 3){
28                set_tx_power(ESP_PWR_LVL_N6);
29            } else if (read_data == 4){
30                set_tx_power(ESP_PWR_LVL_N3);
31            } else if (read_data == 5){
32                set_tx_power(ESP_PWR_LVL_N0);
33            } else if (read_data == 6){
34                set_tx_power(ESP_PWR_LVL_P3);
35            } else if (read_data == 7){
36                set_tx_power(ESP_PWR_LVL_P6);
37            } else if (read_data == 8){
38                set_tx_power(ESP_PWR_LVL_P9);
39            } else {
40                printf("Si    verificato un errore");
41                board_led_operation(LED_R, LED_ON);
42            }
43
44            uint8_t parametro = 1;
45            uint8_t valore = read_data;
46            char address[7] = "0xFFFF";
47            SendNewParameterValue(parametro, valore, address);
48        }
49
50        if (strcmp(read_topic, "/setTN") == 0) {
51            int read_data;
52            sscanf(event->data, "%d", &read_data);
53            printf("TN: %d", read_data);
```

```
54
55              if (read_data>=1 && read_data<=7)  {
56                  TN = read_data;
57                  config_server.net_transmit = ESP_BLE_MESH_TRANSMIT(TN,
    TI);
58                  config_server.relay_retransmit = ESP_BLE_MESH_TRANSMIT(
    TN, TI);
59
60                  uint8_t parametro = 2;
61                  uint8_t valore = read_data;
62                  char address[7] = "0xFFFF";
63                  SendNewParameterValue(parametro, valore, address);
64              } else {
65                  board_led_operation(LED_R, LED_ON);
66              }
67          }
68
69          if (strcmp(read_topic, "/setTI") == 0) {
70              int read_data;
71              sscanf(event->data, "%d", &read_data);
72              printf("TI: %d", read_data);
73
74              if (read_data>=20 && read_data<=1000)  {
75                  TI = read_data;
76                  config_server.net_transmit = ESP_BLE_MESH_TRANSMIT(TN,
    TI);
77                  config_server.relay_retransmit = ESP_BLE_MESH_TRANSMIT(
    TN, TI);
78
79                  uint8_t parametro = 3;
80                  uint8_t valore = read_data;
81                  char address[7] = "0xFFFF";
82                  SendNewParameterValue(parametro, valore, address);
83              } else {
84                  board_led_operation(LED_R, LED_ON);
85              }
86          }
87
88          if (strcmp(read_topic, "/setTTLWithAddress") == 0) {
89              char read_data_string[10];
90              strncpy(read_data_string, event->data, event->data_len);
91              read_data_string[event->data_len] = '\0';
92              printf(read_data_string);
93              char* result[10]; // 0: Address      1: valore parametro
94              splitString(read_data_string,result);
95              printf(" 1: %s", result[0]);
96              printf(" 2: %s", result[1]);
97              uint8_t val = atoi(result[1]);
98              printf(" 2: %d", val);
```

```
99              SendNewParameterValue(0, val, result[0]);
100         }
101
102         if (strcmp(read_topic, "/setPowerWithAddress") == 0) {
103             char read_data_string[10];
104             strncpy(read_data_string, event->data, event->data_len);
105             read_data_string[event->data_len] = '\0';
106             printf(read_data_string);
107             char* result[10]; // 0: Address     1: valore parametro
108             splitString(read_data_string,result);
109             printf(" 1: %s", result[0]);
110             printf(" 2: %s", result[1]);
111             uint8_t val = atoi(result[1]);
112             printf(" 2: %d", val);
113             SendNewParameterValue(1, val, result[0]);
114         }
115
116         if (strcmp(read_topic, "/setTNWithAddress") == 0) {
117             char read_data_string[10];
118             strncpy(read_data_string, event->data, event->data_len);
119             read_data_string[event->data_len] = '\0';
120             printf(read_data_string);
121             char* result[10]; // 0: Address     1: valore parametro
122             splitString(read_data_string,result);
123             printf(" 1: %s", result[0]);
124             printf(" 2: %s", result[1]);
125             uint8_t val = atoi(result[1]);
126             printf(" 2: %d", val);
127             SendNewParameterValue(2, val, result[0]);
128         }
129
130         if (strcmp(read_topic, "/setTIWithAddress") == 0) {
131             char read_data_string[10];
132             strncpy(read_data_string, event->data, event->data_len);
133             read_data_string[event->data_len] = '\0';
134             printf(read_data_string);
135             char* result[10]; // 0: Address     1: valore parametro
136             splitString(read_data_string,result);
137             printf(" 1: %s", result[0]);
138             printf(" 2: %s", result[1]);
139             uint8_t val = atoi(result[1]);
140             printf(" 2: %d", val);
141             SendNewParameterValue(3, val, result[0]);
142         }
143
144         if (strcmp(read_topic, "/setAllParameters") == 0) {
145              if (Start==FALSE){
146                 Start = TRUE;
147                 PDRSend = 0;
```

```
148                 PDRReceived = 0;
149             }
150             char read_data_string[10];
151             strncpy(read_data_string, event->data, event->data_len);
152             read_data_string[event->data_len] = '\0';
153             printf(read_data_string);
154             char* result[10]; // 0: TTL  1:TP    2:TN    3:TI
155             splitString(read_data_string,result);
156             printf(" 0: %s", result[0]);
157             printf(" 1: %s", result[1]);
158             printf(" 2: %s", result[2]);
159             printf(" 3: %s", result[3]);
160             uint8_t TTLval = atoi(result[0]);
161             uint8_t TPval = atoi(result[1]);
162             uint8_t TNval = atoi(result[2]);
163             uint8_t TIval = atoi(result[3]);
164 
165 
166             if (TTLval>=0 && TTLval<=7 && TPval>=1 && TPval<=8 && TNval
    >=1 && TNval<=7 && TIval>=1 && TIval<=10) {
167                 SendNewParameterValue(0, TTLval, "0xFFFF");
168                 SendNewParameterValue(1, TPval, "0xFFFF");
169                 SendNewParameterValue(2, TNval, "0xFFFF");
170                 SendNewParameterValue(3, TIval, "0xFFFF");
171 
172                 if (TTLval == 1) {
173                     TI = 20;
174                     config_server.net_transmit = ESP_BLE_MESH_TRANSMIT(
    TN, TI);
175                     config_server.relay_retransmit =
    ESP_BLE_MESH_TRANSMIT(TN, TI);
176 
177                     uint8_t parametro = 3;
178                     uint8_t valore = TTLval;
179                     char address[7] = "0xFFFF";
180                     SendNewParameterValue(parametro, valore, address);
181                 } else if (TTLval == 2) {
182                     TI = 100;
183                     config_server.net_transmit = ESP_BLE_MESH_TRANSMIT(
    TN, TI);
184                     config_server.relay_retransmit =
    ESP_BLE_MESH_TRANSMIT(TN, TI);
185 
186                     uint8_t parametro = 3;
187                     uint8_t valore = TTLval;
188                     char address[7] = "0xFFFF";
189                     SendNewParameterValue(parametro, valore, address);
190                 } else if (TTLval == 3) {
191                     TI = 250;
```

```
192                     config_server.net_transmit = ESP_BLE_MESH_TRANSMIT(
    TN, TI);
193                     config_server.relay_retransmit =
    ESP_BLE_MESH_TRANSMIT(TN, TI);
194 
195                     uint8_t parametro = 3;
196                     uint8_t valore = TTLval;
197                     char address[7] = "0xFFFF";
198                     SendNewParameterValue(parametro, valore, address);
199                 } else if (TTLval == 4) {
200                     TI = 500;
201                     config_server.net_transmit = ESP_BLE_MESH_TRANSMIT(
    TN, TI);
202                     config_server.relay_retransmit =
    ESP_BLE_MESH_TRANSMIT(TN, TI);
203 
204                     uint8_t parametro = 3;
205                     uint8_t valore = TTLval;
206                     char address[7] = "0xFFFF";
207                     SendNewParameterValue(parametro, valore, address);
208                 } else if (TTLval == 5) {
209                     TI = 1000;
210                     config_server.net_transmit = ESP_BLE_MESH_TRANSMIT(
    TN, TI);
211                     config_server.relay_retransmit =
    ESP_BLE_MESH_TRANSMIT(TN, TI);
212 
213                     uint8_t parametro = 3;
214                     uint8_t valore = TTLval;
215                     char address[7] = "0xFFFF";
216                     SendNewParameterValue(parametro, valore, address);
217                 } else {
218                     board_led_operation(LED_R, LED_ON);
219                 }
220 
221                 if (TPval == 1) {
222                     set_tx_power(ESP_PWR_LVL_N12);
223                 } else if (TPval == 2){
224                     set_tx_power(ESP_PWR_LVL_N9);
225                 } else if (TPval == 3){
226                     set_tx_power(ESP_PWR_LVL_N6);
227                 } else if (TPval == 4){
228                     set_tx_power(ESP_PWR_LVL_N3);
229                 } else if (TPval == 5){
230                     set_tx_power(ESP_PWR_LVL_N0);
231                 } else if (TPval == 6){
232                     set_tx_power(ESP_PWR_LVL_P3);
233                 } else if (TPval == 7){
234                     set_tx_power(ESP_PWR_LVL_P6);
```

```
235                 } else if (TPval == 8){
236                     set_tx_power(ESP_PWR_LVL_P9);
237                 }
238 
239                 TN = TNval;
240                 config_server.net_transmit = ESP_BLE_MESH_TRANSMIT(TN,
    TI);
241                 config_server.relay_retransmit = ESP_BLE_MESH_TRANSMIT(
    TN, TI);
242 
243                 TI = TIval;
244                 config_server.net_transmit = ESP_BLE_MESH_TRANSMIT(TN,
    TI);
245                 config_server.relay_retransmit = ESP_BLE_MESH_TRANSMIT(
    TN, TI);
246 
247             } else {
248                 board_led_operation(LED_R, LED_ON);
249             }
250         }
```

A.2.3 Invio dei nuovi valori tramite BLE Mesh

Se tutti i controlli precedentemente illustrati sono andati a buon fine, il nodo client procede con l'invio dei nuovi valori ai nodi server tramite il protocollo BLE Mesh. Il seguente codice permette di inviare un messaggio a tutti i nodi server presenti nelle rete oppure a un determinato nodo.

```
1  void SendNewParameterValue(uint8_t parametro, uint8_t valore, char
       address[10])
2  {
3      esp_ble_mesh_generic_client_set_state_t set = {0};
4      esp_ble_mesh_client_common_param_t common = {0};
5      esp_err_t err = ESP_OK;
6 
7      ESP_LOGI(TAG, "parametro %d", parametro);
8      ESP_LOGI(TAG, "valore %d", valore);
9      ESP_LOGI(TAG, "address %s", address);
10 
11     common.opcode = ESP_BLE_MESH_MODEL_OP_GEN_ONOFF_SET_UNACK;
12     common.model = onoff_client.model;
13     common.ctx.net_idx = store.net_idx;
14     common.ctx.app_idx = store.app_idx;
15     common.ctx.addr = address;
16     common.ctx.send_ttl = 7;
17     common.ctx.send_rel = true;
18     common.msg_timeout = 0;
19     common.msg_role = ROLE_NODE;
```

```
20
21      uint8_t identificatore = 1;
22      uint8_t combinazione = ((identificatore & 0x01) << 7) | ((parametro
        & 0x03) << 5) | (valore & 0x1F);
23      ESP_LOGI(TAG, "combinazione 0x%x", combinazione);
24
25      set.onoff_set.op_en = false;
26      set.onoff_set.onoff = combinazione;
27      set.onoff_set.tid = autoincrement_value++;
28      set.onoff_set.trans_time = 0x003;
29
30      timer = esp_timer_get_time();
31      err = esp_ble_mesh_generic_client_set_state(&common, &set);
32      if (err) {
33          ESP_LOGE(TAG, "Send Generic OnOff Set Unack failed");
34          return;
35      }
36  }
```

Il messaggio che viene inviato per modificare il valore di un parametro è costituito ad una struttura dati formata da 8 bit. I primi due bit vengono utilizzati per individuare il parametro:

- 00: TTL
- 01: Potenza di trasmissione
- 10: Numero di trasmissioni
- 11: Intervallo tra le trasmissioni

I successivi 6 bit vengono utilizzati per il valore del parametro.

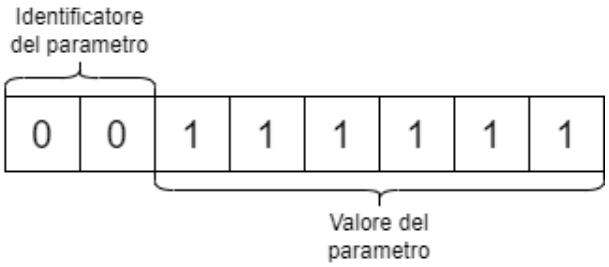


Figura A.1: Struttura del messaggio.

A.2.4 Invio messaggio «ping-pong»

Il codice riportato in seguito, viene richiamato ogni secondo e consente al nodo client di inviare un messaggio a tutti i nodi server presenti nella rete. Ad ogni messaggio, viene

associato un ID che va da 0 a 63. Inoltre, viene utilizzato un array di lunghezza pari a 64. Per ogni ID del messaggio corrisponde un indice dell'array. Perciò, ogni volta che viene inviato un messaggio, in corrispondenza all'ID - index dell'array, viene salvato il valore di «esp_timer_get_time()». Quindi questo valore viene utilizzato come se fosse un timestamp, e quindi viene utilizzato per capire quando il messaggio è stato inviato. Questo valore verrà utilizzato successivamente, per ottenere il tempo trascorso dall'invio del messaggio e la ricezione della risposta (e di conseguenza per ottenere il valore di delay del messaggio).

```
void sendPingPongMessage(void)
{
    esp_ble_mesh_generic_client_set_state_t set = {0};
    esp_ble_mesh_client_common_param_t common = {0};
    esp_err_t err = ESP_OK;

    common.opcode = ESP_BLE_MESH_MODEL_OP_GEN_ONOFF_SET_UNACK;
    common.model = onoff_client.model;
    common.ctx.net_idx = store.net_idx;
    common.ctx.app_idx = store.app_idx;
    common.ctx.addr = 0xFFFF;   /* to all nodes */
    common.ctx.send_ttl = defaultTTLValue;
    common.ctx.send_rel = true;
    common.msg_timeout = 0;
    common.msg_role = ROLE_NODE;

    uint8_t identificatoreMsg = 0;
    uint8_t MessaggioID = currentlIdWithSendTime;
    uint8_t combinazione = 0;
    combinazione |= (identificatoreMsg << 6);
    combinazione |= MessaggioID;

    set.onoff_set.op_en = false;
    set.onoff_set.onoff = combinazione;
    set.onoff_set.tid = autoincrement_value++;
    set.onoff_set.trans_time = 0x003;

    timer = esp_timer_get_time();
    idWithSendTime[currentlIdWithSendTime] = timer;
    currentlIdWithSendTime = currentlIdWithSendTime + 1;

    if (currentlIdWithSendTime>64) {
        currentlIdWithSendTime = 0;
        vTaskDelay(10000 / portTICK_PERIOD_MS);
    }

    err = esp_ble_mesh_generic_client_set_state(&common, &set);
    if (err) {
        ESP_LOGE(TAG, "Send Generic OnOff Set Unack failed");
```

```
40          return;
41      } else {
42          PDRSend = PDRSend + 1;
43      }
44  }
```

A.2.5 Ricezione messaggio «ping-pong»

Il codice seguente consente al nodo Client di ricevere la risposta dal nodo Server. Dopo aver ricevuto la risposta, viene calcolato il round trip time del messaggio. Inoltre, dal messaggio vengono estratti i valori l'identificativo del nodo Server che ha inviato la risposta e i TTL residui della richiesta inviata.

```
1  static void example_ble_mesh_generic_client_cb(
       esp_ble_mesh_generic_client_cb_event_t event,
       esp_ble_mesh_generic_client_cb_param_t *param)
2  {
3      ESP_LOGI(TAG, "Generic client, event %u, error code %d, opcode is 0
       x%04x", event, param->error_code, param->params->opcode);
4      ESP_LOGI("BLE MESH----", "opcode is 0x%04x",  param->status_cb.
       onoff_status.present_onoff);
5      switch (event) {
6
7          case    ESP_BLE_MESH_GENERIC_CLIENT_GET_STATE_EVT:
8             ...
9              break;
10         case    ESP_BLE_MESH_GENERIC_CLIENT_SET_STATE_EVT:
11             ...
12             break;
13         case ESP_BLE_MESH_GENERIC_CLIENT_PUBLISH_EVT:
14             int returnTime = esp_timer_get_time();
15             PDRReceived = PDRReceived + 1;
16
17             ESP_LOGI("BLE MESH----", "opcode is  0x%04x",  param->
       status_cb.onoff_status.present_onoff);
18             uint8_t messageReturn = param-  >status_cb.onoff_status.
       present_onoff;
19
20             uint8_t nodeID = (messageReturn >> 6) & 0x03;
21             // i primi due bit di   combinazione spostati a destra di 6
          posizioni
22             uint8_t TTLResidue = (messageReturn >>  4) & 0x03;
23             // i successivi due bit di  combinazione spostati a destra
       di 4  posizioni e poi mascherati con 0x03
24             uint8_t messageID = messageReturn & 0x0F;
25             // gli ultimi 4 bit di  combinazione mascherati con 0x0F
26
27             ESP_LOGI("BLE MESH","Messaggio ricevuto %d",messageReturn);
```

```
28              ESP_LOGI("BLE MESH","Node Id%d", nodeID);
29              ESP_LOGI("BLE MESH","TTL residue %d", TTLResidue);
30              ESP_LOGI("BLE MESH","messageID %d", messageID);
31
32              ESP_LOGI("BLE MESH", "returnTime %d", returnTime);
33              ESP_LOGI("BLE MESH", "idWithSendTime[messageID] %d",
    idWithSendTime[messageID]);
34
35              int totalTime = returnTime - idWithSendTime[messageID];
36              ESP_LOGI("BLE MESH", "totalTime %d", totalTime);
37
38              char Performances[30];
39              sprintf(Performances, "%d-%d-%d", nodeID, TTLResidue,
    totalTime);  //   Formattiamo la stringa con i tre valori   uint8_t
    separati da trattini
40              printf("Performances formattata   :  %s\n", Performances);
41
42              if(MQTT_CONNEECTED) {
43                  esp_mqtt_client_publish(client,     "/sendPerformnce",
     Performances, 0,    0, 0);
44              }
45              break;
46          case ESP_BLE_MESH_GENERIC_CLIENT_TIMEOUT_EVT:
47              ESP_LOGI(TAG,   "ESP_BLE_MESH_GENERIC_CLIENT_TIMEOUT_EVT" )
    ;
48              if (param->params->opcode ==
    ESP_BLE_MESH_MODEL_OP_GEN_ONOFF_SET) {
49                  /* If failed to get the response of     Generic OnOff
    Set, resend Generic   OnOff Set  */
50                      example_ble_mesh_send_gen_onoff_set()   ;
51              }
52              break;
53          default:
54              break;
55      }
56  }
```

A.3 Nodo Server

Il questa sezione verranno illustrati i dettagli implementativi relativi al nodo server.

A.3.1 Ricezione messaggio

Il seguente codice permette ai nodi Server di ricevere i messaggi inviati dal nodo Client. Grazie a questa funzionalità, i nodi Server saranno in grado di ricevere i messaggi contenenti informazioni cruciali relative ai parametri di comunicazione, come il TTL, la

potenza di trasmissione, il numero di trasmissioni e l'intervallo tra le trasmissioni. Una volta ricevuti questi dati, questi nodi potranno aggiornare a runtime le proprie impostazioni di comunicazione , garantendo un funzionamento fluido e affidabile dell'intero sistema.

```
static void example_ble_mesh_generic_server_cb(
    esp_ble_mesh_generic_server_cb_event_t event,
    esp_ble_mesh_generic_server_cb_param_t *param)
{
    esp_ble_mesh_gen_onoff_srv_t *srv = param->model->user_data;
    esp_ble_mesh_generic_server_cb_param_t *paramResent = { 0 };

    ESP_LOGI(TAG, "event 0x%02x, opcode 0x%04x, src 0x%04x, dst 0x%04x"
    , event, param->ctx.recv_op, param->ctx.addr, param->ctx.recv_dst);
    ESP_LOGI(TAG, "event 0x%02x, opcode 0x%04x, src 0x%04x, dst 0x%04x
    recv_ttl 0x%04x", event, param->ctx.recv_op, param->ctx.addr, param
    ->ctx.recv_dst, param->ctx.recv_ttl);

    switch (event) {
        case ESP_BLE_MESH_GENERIC_SERVER_STATE_CHANGE_EVT:
            ESP_LOGI(TAG, "ESP_BLE_MESH_GENERIC_SERVER_STATE_CHANGE_EVT
    ");
            if (param->ctx.recv_op ==
    ESP_BLE_MESH_MODEL_OP_GEN_ONOFF_SET || param->ctx.recv_op ==
    ESP_BLE_MESH_MODEL_OP_GEN_ONOFF_SET_UNACK) {
                uint8_t parametroValore = param->value.set.onoff.op_en;
                uint8_t identificatore = (parametroValore & 0x80) >> 7;
                uint8_t idMessaggio, parametro, valore;

                if (identificatore == 0) {
                    idMessaggio = parametroValore & 0x7F;
                    parametro = 0;
                    valore = 0;
                    example_handle_gen_onoff_msg(param->model, &param->
    ctx, &param->value.set);
                } else {
                    parametro = (parametroValore & 0x60) >> 5;
                    valore = parametroValore & 0x1F;
                    idMessaggio = 0;
                    setParameterFunction(parametro, valore);
                }
            }
            break;
        case ESP_BLE_MESH_GENERIC_SERVER_RECV_GET_MSG_EVT:
            ...
            break;
        case ESP_BLE_MESH_GENERIC_SERVER_RECV_SET_MSG_EVT:
            ...
            break;
```

```
36          default:
37              ESP_LOGE(TAG, "Unknown Generic Server event 0x%02x", event)
    ;
38              break;
39      }
40  }
```

A.3.2 Gestione messaggio - modifica parametro

Di seguito viene riportato il codice che consente al nodo Server di estrarre i parametri e il loro valore dal messaggio inviato dal nodo Client. Successivamente, dopo aver verificato che il valore estratto sia valido per il parametro specifico, si procederà con l'effettiva modifica.

```
1  static void setParameterFunction(uint8_t parametro, uint8_t valore){
2      ESP_LOGI(TAG, "Parametro %d", parametro);
3      ESP_LOGI(TAG, "Valore %d", valore);
4      switch(parametro) {
5          case 0:
6              printf("Hai scelto la opzione 0 - TTL.\n");
7              if (valore>=0 && valore<=7) {
8                  config_server.default_ttl = valore;
9              } else {
10                 board_led_operation(LED_R, LED_ON);
11             }
12             break;
13         case 1:
14             printf("Hai scelto la opzione 1 - PT.\n");
15             switch (valore) {
16                 case 1:
17                     set_tx_power(ESP_PWR_LVL_N12);
18                     break;
19                 case 2:
20                     set_tx_power(ESP_PWR_LVL_N9);
21                     break;
22                 case 3:
23                     set_tx_power(ESP_PWR_LVL_N6);
24                     break;
25                 case 4:
26                     set_tx_power(ESP_PWR_LVL_N3);
27                     break;
28                 case 5:
29                     set_tx_power(ESP_PWR_LVL_N0);
30                     break;
31                 case 6:
32                     set_tx_power(ESP_PWR_LVL_P3);
33                     break;
```

```
34                  case 7:
35                      set_tx_power(ESP_PWR_LVL_P6);
36                      break;
37                  case 8:
38                      set_tx_power(ESP_PWR_LVL_P9);
39                      break;
40                  default:
41                      board_led_operation(LED_R, LED_ON);
42                      break;
43              }
44              break;
45          case 2:
46              printf("Hai scelto la opzione 2 - TN.\n");
47              if (valore>=1 && valore<=7)  {
48                  TN = valore;
49                  config_server.net_transmit = ESP_BLE_MESH_TRANSMIT(TN,
    TI);
50                  config_server.relay_retransmit = ESP_BLE_MESH_TRANSMIT(
    TN, TI);
51              } else {
52                  board_led_operation(LED_R, LED_ON);
53              }
54              break;
55          case 3:
56              printf("Hai scelto la opzione 3 - TI.\n");
57              if (valore>=20 && valore<=1000)  {
58                   TI = valore;
59                   config_server.net_transmit = ESP_BLE_MESH_TRANSMIT(TN,
     TI);
60                   config_server.relay_retransmit = ESP_BLE_MESH_TRANSMIT
    (TN, TI);
61              } else {
62                   ESP_LOGI(TAG, "ERRORE TI %d", valore);
63                   board_led_operation(LED_R, LED_ON);
64              }
65              break;
66          default:
67              printf("Si    verificato un errore.\n");
68              break;
69      }
70  }
```

A.3.3 Gestione messaggio - ping pong

Il codice sottostante consente al nodo Server di gestire il messaggio «ping-pong», che viene utilizzato per valutare la prestazione della rete. Quindi, appena il nodo server riceve questo messaggio, genera un nuovo messaggio contenente l'identificativo del mes-

saggio ricevuto, l'identificativo del nodo server in questione e i TTL residui del messaggio ricevuto.

```
static void example_handle_gen_onoff_msg(esp_ble_mesh_model_t *model,
    esp_ble_mesh_msg_ctx_t *ctx,
    esp_ble_mesh_server_recv_gen_onoff_set_t *set)
{
    esp_ble_mesh_gen_onoff_srv_t *srv = model->user_data;
    printf("SONO QUI invio il messaggio?");
    ESP_LOGI(TAG, "TEST 0x%04x", set->op_en);
    ESP_LOGI(TAG, "TTLRECV 0x%04x", ctx->recv_ttl);
    switch (ctx->recv_op) {


    case ESP_BLE_MESH_MODEL_OP_GEN_ONOFF_GET:
        ...
        break;
    case ESP_BLE_MESH_MODEL_OP_GEN_ONOFF_SET:
        ...
        break;
    case ESP_BLE_MESH_MODEL_OP_GEN_ONOFF_SET_UNACK:
        ESP_LOGI(TAG, "TTLRECV 0x%04x", ctx->recv_ttl);
        uint8_t TTLResidue = (uint8_t)ctx->recv_ttl;
        uint8_t MessageId = (uint8_t)set->op_en;
        ESP_LOGI(TAG, "MessageID %d", MessageId);

        uint8_t combinazione = (nodeID & 0x03) << 6;
        // i primi due bit di nodeID spostati a sinistra di 6 posizioni
        combinazione |= (TTLResidue & 0x03) << 4;
        // i successivi due bit di TTLResidue spostati a sinistra di 4
    posizioni e combinati con gli altri due bit di nodeID
        combinazione |= (MessageId & 0x0F);
        // gli ultimi 4 bit di messageID combinati con i primi quattro
    bit di combinazione
        ESP_LOGI(TAG, "Combinazione %d", combinazione);

        esp_ble_mesh_server_model_send_msg(model, ctx,
            ESP_BLE_MESH_MODEL_OP_GEN_ONOFF_STATUS, sizeof(combinazione
    ), &combinazione);
        break;
    default:
        break;
    }
}
```

A.4 Script raccolta dati

Per poter raccogliere i dati da dare in input agli algoritmi di AI, è stato creato un codice che permette di inviare, tramite protocollo mqtt, il nuovo valore dei vari parametri. Successivamente, sempre tramite il protocollo mqtt, riceve le prestazioni della rete in seguito alla modifica del parametro. Infine, questo script permette di salvare in un database MySQL i parametri con il relativo valore di latenza e delivery ratio. In questo script sono stati utilizzati le seguenti librerie:

```
import mysql.connector
#permette di connettersi a MySQL

from paho.mqtt import client as mqtt_client
#permette di utilizzare il protocollo MQTT

import itertools
#permette di creare il prodotto cartesiano tra diverse liste
```

Il codice riportato di seguito consente di stabilire una connessione con il database denominato «ble_mesh_parameters».

```
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  database="ble_mesh_parameters"
)
```

Al fine di determinare tutte le combinazioni possibili di valori tra le variabili, è stato creato un insieme di liste per ogni parametro contenente tutti i valori che il parametro può assumere. Successivamente, tali liste sono state utilizzate per creare un prodotto cartesiano, ottenendo così un insieme di tutte le possibili combinazioni. Questa dinamica è stata realizzata dal codice sottoriportato.

```
TTLValue = [1,2,3,4,5]
TransmissionPower = [1,3,6]
TransmissionsNumber = [1,3,5]
IntervalTime= [20,100,250,500,1000]
Combination = list(itertools.product(TTLValue, TransmissionPower,
    TransmissionsNumber, IntervalTime))
```

Nello script Python in questione, si fa uso del protocollo MQTT, il quale consente di inviare i valori dei parametri al nodo Client e di ricevere (sempre dal Client) i valori di delay e delivary ratio.
Per poter inviare i nuovi valori dei parametri, viene utilizzato il topic **/setAllParame-**

ters, mentre per poter ottenere i valori di delay e packet delivery ratio, vengono utilizzati i topic:

- **/sendPerformnce**
- **/sendPDR**

Inoltre, lo script viene sottoscritto anche al topic **/getNewParameter**. Questo canale viene utilizzato dal nodo Client per richiedere la combinazione di parametri successiva.

```
client.subscribe(/sendPerformnce)
client.subscribe(/sendPDR)
client.subscribe(/getNewParameter)
```

Il corpo principale dello script, può essere diviso in 3 funzioni principali:

- acquisizione del delay
- acquisizione del packet delivery ratio
- invio della nuova combinazione di parametri

A.4.1 Acquisizione del delay

Il codice presentato di seguito consente di rilevare il dealy associato a ciascun pacchetto trasmesso.

```
if msg.topic == "/sendPerformnce":
    Performances = msg.payload.decode().split("-")
    nodeID = int(Performances[0])
    TTLResidue = int(Performances[1])
    Delay = int(Performances[2])

    mycursor = mydb.cursor()
    sql = "INSERT INTO parameters_performance (DeviceID, TTL,
   TransmissionPower, TransmissionsNumber, IntervalTime, TTLResidue,
   Delay) VALUES (%s, %s, %s, %s, %s, %s, %s)"
    val = (nodeID, Combinations[index][0], Combinations[index][1],
   Combinations[index][2], Combinations[index][3],TTLResidue,Delay)
    mycursor.execute(sql, val)
    mydb.commit()
```

A.4.2 Acquisizione del packet delivery ratio

Il codice presentato di seguito consente di rilevare il PDR associato a ciascun pacchetto trasmesso.

```
elif msg.topic == "/sendPDR":
    PDRVal = msg.payload.decode().split("-")
    PDRSend = int(PDRVal[0])
    PDRReceived = int(PDRVal[1])

    mycursor = mydb.cursor()
    sql = "INSERT INTO parameters_pdr (TTL, TransmissionPower,
   TransmissionsNumber, IntervalTime, PDRSend, PDRReceived) VALUES (%s,
    %s, %s, %s, %s, %s)"
    val = (Combinations[index][0], Combinations[index][1], Combinations
   [index][2], Combinations[index][3], PDRSend, PDRReceived)
    mycursor.execute(sql, val)
    mydb.commit()
```

A.4.3 Invio della nuova combinazione di parametri

Il codice riportato di seguito consente di inviare una nuova combinazione di parametri al nodo Client.

```
elif msg.topic == "/getNewParameters":
    try:
        index += 1
        CombinazioneInvio = str(Combinations[index][0]) + ":" + str(
   Combinations[index][1]) + ":" +  str(Combinations[index][2]) + ":" +
    str(Combinations[index][3])
        publish(client, "/setAllParameters",CombinazioneInvio)
    except:
        print("FINITO")
```

A.5 Back-end

Il back-end della dashboard è costituito da due elementi principali che sono

- node.js
- script python con algoritmi di A.I.

A.5.1 Node.js

Nello script scritto con node.js sono stati utilizzati i seguenti elementi:

```
const express = require('express');
const mqtt = require("mqtt");
const $ = require('jquery');
const PythonShell = require('python-shell')
```

La funzione principale sviluppato in Node.js consiste nel poter prendere in input i vari valori che l'utente ha inserito sulla dashboard ed inviarli tramite protocollo MQTT ai nodi client.
In seguito, è riportato l'implementazione di questa logica.

```
app.get('/setTTL', (req, res) => {
    var result = "Send new TTL value successfully"
    try {
        let inputValue = parseInt(req.query.inputValue);
        if (inputValue>0 && inputValue<8) {
            client.publish('/setTTL', inputValue.toString());
        } else {
            result = "Error: Invalid value";
        }
    } catch {
        result = "Error: New TTL not sent"
    }
    res.send(result);
});
```

Il codice sopra riportato mostra come modificare il parametro TTL, ma il processo di modifica degli altri parametri è analogo. È importante notare che il codice effettua un controllo sul valore inviato dall'utente per verificare la sua idoneità al parametro in questione. Se il valore è considerato valido, il messaggio viene inviato tramite il protocollo MQTT. Se, invece, il valore non è accettabile, l'utente riceverà un messaggio di errore.

Il codice riportato in seguito consente al back-end di avviare il codice Python per la raccolta di dati sulla prestazione della rete utilizzando tutte le possibili combinazioni di parametri.

```
// Endpoint per la racolta dati
app.get('/predict', (req, res) => {

  // Eseguire il codice Python per la raccolta dati
  const options = {
    pythonPath: 'C:/Users/Keran/AppData/Local/Programs/Python/Python310
   -32', // Path di Python
    scriptPath: '../scriptRaccolaDati.py', // Path del codice Python
  }

  PythonShell.run('scriptRaccolaDati.py', options, (err, result) => {
    if (err) throw err
    res.send(result)
  })

})
```

In conclusione, il seguente codice permette al back-end di accedere ai molteplici modelli di intelligenza artificiale e consente l'esecuzione del processo che permette di individuare i valori ottimali dei parametri per un particolare nodo. Nello specifico, il seguente codice accede all'algoritmo Decision Tree.

```
// Endpoint per la predizione Decision Tree
app.get('/predict', (req, res) => {

  // Eseguire il codice Python per la predizione
  const options = {
    pythonPath: '.../Python/Python310-32', // Path di Python
    scriptPath: '.../script', // Path del codice Python
    args: [req.query.param1, req.query.param2, req.query.param3] //
     Parametri per la predizione
  }

  PythonShell.run('01_DecisionTreeDelay.py', options, (err, result) =>
   {
    if (err) throw err
    res.send(result)
  })
```

A.5.2 AI

Nel back-end è stato sviluppato un sistema di apprendimento supervisionato, ed in particolare è stato implementato un insieme di algoritmi di apprendimento basati su diverse tecniche di modellizzazione [7]:

- Algoritmi basati su alberi:
  - Decision Tree: La struttura dell'albero è composta da una radice, rami e foglie. La radice rappresenta la variabile più importante del dataset, mentre i rami rappresentano i valori possibili della variabile. Le foglie rappresentano le classi o i valori di output della variabile di destinazione.
    La suddivisione del dataset viene eseguita in modo iterativo fino a quando ogni foglia dell'albero rappresenta un set di dati omogeneo. In questo modo, l'albero può essere utilizzato per classificare nuovi dati sulla base delle decisioni prese durante la creazione dell'albero. L'algoritmo di decision tree ha diversi vantaggi rispetto ad altre tecniche di apprendimento, tra cui la facilità di interpretazione del modello creato, la capacità di gestire dati mancanti e di gestire dati sia numerici che categorici. Tuttavia, può essere sensibile al rumore presente nel dataset e può essere soggetto al problema dell'overfitting.

- Random Forest: crea un insieme di alberi decisionali, in cui ogni albero viene creato utilizzando un sottoinsieme casuale dei dati del dataset di origine. La scelta di un sottoinsieme casuale di dati aiuta a ridurre la sensibilità dell'algoritmo al rumore o a dati non rappresentativi del dataset originale. L'algoritmo di random forest ha diverse caratteristiche che lo rendono utile per la creazione di modelli predittivi accurati e stabili. In particolare, questo algoritmo è in grado di gestire dataset con molte variabili predittive e di identificare le variabili più importanti per la previsione. Inoltre, l'algoritmo è meno sensibile all'overfitting rispetto ad altri algoritmi di apprendimento supervisionato, in quanto utilizza diversi sottoinsiemi del dataset di origine per creare gli alberi decisionali.
- Gradient Boosting: L'algoritmo di Gradient Boosting crea un insieme di modelli predittivi deboli, come ad esempio alberi decisionali, in cui ogni modello viene costruito per minimizzare l'errore residuo del modello precedente. In particolare, l'algoritmo crea un modello di base e successivamente costruisce un nuovo modello che tenta di migliorare l'errore residuo del modello precedente.
  La creazione del modello avviene in modo iterativo, in cui ad ogni iterazione viene aggiunto un nuovo modello alla sequenza di modelli. Durante ogni iterazione, l'algoritmo valuta l'errore residuo del modello precedente e tenta di correggere gli errori, costruendo un nuovo modello che tenta di prevedere i residui rimanenti.
  L'algoritmo di Gradient Boosting ha diverse caratteristiche che lo rendono utile per la creazione di modelli predittivi. In particolare, questo algoritmo è in grado di gestire dataset di grandi dimensioni e di identificare le variabili più importanti per la previsione. Inoltre, l'algoritmo è meno sensibile all'overfitting rispetto ad altri algoritmi di apprendimento supervisionato, in quanto utilizza un approccio di costruzione del modello a sequenza, piuttosto che un unico modello complesso.

- KNN (K-nearest neighbors): è un algoritmo che si basa sulla vicinanza tra i punti di dati nel dataset e sulla loro etichetta di classe.
  In particolare, l'algoritmo KNN assegna una nuova istanza di dati ad una classe sulla base della maggioranza delle classi tra i suoi K vicini più vicini nel dataset di training. Il valore di K è un parametro dell'algoritmo che deve essere scelto dall'utente.
  Per trovare i K vicini più vicini di una nuova istanza di dati, l'algoritmo KNN calcola la distanza tra la nuova istanza e tutti i punti di dati nel dataset di training. La distanza può essere calcolata utilizzando diverse metriche, come ad esempio la distanza Euclidea o la distanza di Manhattan.
  Una volta che le distanze sono state calcolate, l'algoritmo KNN seleziona i K punti

di dati più vicini alla nuova istanza di dati. Successivamente, l'algoritmo determina la classe maggioritaria tra questi K punti di dati e assegna quella classe alla nuova istanza di dati.
L'algoritmo KNN può essere utilizzato per la classificazione, la regressione e l'analisi di clustering. Tuttavia, l'algoritmo KNN presenta alcune limitazioni, tra cui la necessità di avere un grande dataset di training, la dipendenza dalla scelta del valore di K e la sensibilità ai dati rumorosi.

- Gaussian Naive Bayes: Gaussian Naive Bayes è un algoritmo di apprendimento automatico utilizzato per la classificazione di dati. Il suo nome deriva dalle due parti dell'algoritmo: il teorema di Bayes e il modello Gaussian (distribuzione normale). Il teorema di Bayes viene utilizzato per calcolare la probabilità di appartenenza di un esempio ad una classe, sulla base dei dati di addestramento. Il modello Gaussian descrive la distribuzione delle feature in ogni classe. L'algoritmo assume che le feature seguano una distribuzione normale nella classe.
  Il Gaussian Naive Bayes è «naive» perché assume l'indipendenza tra le feature, il che significa che le feature non influenzano l'una l'altra. Questa assunzione semplifica il calcolo della probabilità e rende l'algoritmo facile da implementare.
  Quindi, Gaussian Naive Bayes è un algoritmo di classificazione basato su formula matematica che cerca di classificare gli oggetti in classi sulla base delle probabilità calcolate delle proprietà o delle feature degli oggetti stessi, e si avvale del modello Gaussian per descrivere la distribuzione delle feature all'interno di ogni classe.

In questo contesto, il dataset rappresenta un aspetto fondamentale per la creazione di modelli accurati e precisi. Tuttavia, per poter applicare questi algoritmi al dataset, è stato necessario procedere con la normalizzazione dei dati.

In letteratura, sono state proposte diverse tecniche di normalizzazione, tra cui la normalizzazione **Min-Max Scaling** e la normalizzazione **Z-Score**, che sono state valutate e confrontate in termini di efficienza e precisione. La normalizzazione Min-Max Scaling utilizza una formula che si basa sui valori minimi e massimi presenti nel dataset, ma la presenza di outlier può influenzare negativamente i risultati ottenuti. Invece, la normalizzazione Z-Score è immune agli outlier e fornisce risultati più affidabili. La formula utilizzata per la normalizzazione Z-Score è basata sulla media e sulla deviazione standard del dataset, che permettono di ottenere un dataset normalizzato e pronto per l'applicazione degli algoritmi di apprendimento.

$$Min - Max = \frac{X - Xmin}{Xmax - Xmin} \tag{A.1}$$

$$Z - Score = \frac{X - mean}{Std} \tag{A.2}$$

In seguito è riportato il codice relativo alla normalizzazione:

```
def scalingDataset(X, zscore):
    if (zscore == True):
        # Z normalization
        zscaler = StandardScaler()
        Xz = zscaler.fit_transform(X)
        X = pd.DataFrame(Xz, columns=X.columns)
    else:
        # Normalization
        scaler = MinMaxScaler(feature_range=(-1, 1))
        X_ = scaler.fit_transform(X)
        X = pd.DataFrame(X_, columns=X.columns)
    return X
```

Per ciascun algoritmo di Intelligenza Artificiale sono stati generati 17 modelli. Ciascun modello prende in input combinazioni di variabili differenti. In seguito sono riportate due tabelle che riportano per ciascun modello gli input che deve dare l'utente e i relativi output.

	Input	**Output**
Modello 1	Delay NodeID	TTL Potenza di trasmissione Numero di trasmissione Intervallo di trasmissione
Modello 2	PDR NodeID	TTL Potenza di trasmissione Numero di trasmissione Intervallo di trasmissione
Modello 3	Delay PDR NodeID	TTL Potenza di trasmissione Numero di trasmissione Intervallo di trasmissione
Modello 4	Delay PDR NodeID TTL	Potenza di trasmissione Numero di trasmissione Intervallo di trasmissione
Modello 5	Delay PDR DeviceID Potenza di trasmissione	TTL Numero di trasmissione Intervallo di trasmissione
Modello 6	Delay PDR NodeID Numero di trasmissione	TTL Potenza di trasmissione Intervallo di trasmissione
Modello 7	Delay PDR NodeID Intervallo di trasmissione	TTL Potenza di trasmissione Numero di trasmissione Intervallo di trasmissione
Modello 8	Delay PDR NodeID TTL Potenza di trasmissione	Numero di trasmissione Intervallo di trasmissione
Modello 9	Delay PDR NodeID TTL Numero di trasmissione	Potenza di trasmissione Intervallo di trasmissione
Modello 10	Delay PDR NodeID TTL Intervallo di trasmissione	Potenza di trasmissione Numero di trasmissione

Tabella A.2: Tabella che riporta lo schema degli input e degli output dei vari modelli di AI - prima parte

	Input	**Output**
Modello 11	Delay PDR NodeID Potenza di trasmissione Numero di trasmsissioni	TTL Intervallo di trasmissione
Modello 12	Delay PDR NodeID Potenza di trasmissione Intervallo di trasmissione	TTL Numero di trasmissioni
Modello 13	Delay PDR NodeID Numero di trasmissione Intervallo di trasmissione	TTL Potenza di trasmissione
Modello 14	Delay PDR NodeID TTL Potenza di trasmissione Numero di trasmissione	Intervallo di trasmissione
Modello 15	Delay PDR NodeID TTL Potenza di trasmissione Intervallo di trasmissione	Numero di trasmissione
Modello 16	Delay PDR NodeID TTL Numero di trasmissione Intervallo di trasmissione	Potenza di trasmissione
Modello 17	Delay PDR NodeID Potenza di trasmissone Numero di trasmissione Intervallo di trasmissione	TTL

Tabella A.3: Tabella che riporta lo schema degli input e degli output dei vari modelli di AI - seconda parte

A.5.3 Decision Tree - creazione del modello

Il codice presentato di seguito rappresenta un'implementazione dell'algoritmo Decision Tree, utilizzato per identificare i valori ottimali dei parametri TTL, Potenza di Trasmissione, Numero di Trasmissione e Intervallo di Trasmissione per una determinata configurazione. In primo luogo, i dati sono divisi in due insiemi: il set di feature X e il set di target Y. Successivamente viene eseguita una normalizzazione dei dati mediante la tecnica Z-Scale. Infine, viene generato il modello a partire dai dati normalizzati e salvato in un file con estensione *.joblib*.

```
# selezionare le colonne di input e di output
input_cols = ["Delay", "DeviceID"]
output_cols = ["TTL", "TransmissionPower", "TransmissionsNumber", "
    IntervalTime"]

# dividi le colonne di input e di output
X = dataset[input_cols]
y = dataset[output_cols]

# applicare la Standard Scaler alle colonne di input
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# creare il modello di Decision Tree e addestrarlo
model = DecisionTreeRegressor(random_state=0)
model.fit(X_scaled, y)

# Salva il modello su file
dump(model, 'Models/01_DecisionTree-Delay.joblib')
```

A.5.4 Random Forest - creazione del modello

Il seguente codice definisce la struttura del modello AI che implementa la tecnica Random Forest. Tale algoritmo condivide la medesima logica di implementazione del Decision Tree, differenziandosi unicamente per l'algoritmo adottato.

```
# selezionare le colonne di input e di output
input_cols = ["Delay", "DeviceID"]
output_cols = ["TTL", "TransmissionPower", "TransmissionsNumber", "
    IntervalTime"]

# dividi le colonne di input e di output
X = dataset[input_cols]
y = dataset[output_cols]

# applicare la Standard Scaler alle colonne di input
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)

# creare il modello di Decision Tree e addestrarlo
model = RandomForestClassifier(random_state=0)
model.fit(X_scaled, y)

# Salva il modello su file
dump(model, 'Models/01 RandomForest-Delay.joblib')
```

A.5.5 Gradient Boosting - creazione del modello

Il seguente codice definisce la struttura del modello AI che implementa la tecnica Gradient Boosting. Tale algoritmo condivide la medesima logica di implementazione del Decision Tree, differenziandosi unicamente per l'algoritmo adottato.

```
# Seleziona le feature di input e il target
input_features = ['Delay', 'DeviceID']
target_features = ['TTL', 'TransmissionPower', 'TransmissionsNumber', '
    IntervalTime']

# Crea il dataset di training e testing con tutte le feature
X = dataset[input_features]
y = dataset[target_features]

# Crea il modello per la previsione di tutti i target
model = xgb.XGBRegressor()
model.fit(X, y)

dump(model, 'Models/01 Boosting-Delay.joblib')
```

A.5.6 KNN - creazione del modello

Il seguente codice definisce la struttura del modello AI che implementa la tecnica KNN. Tale algoritmo condivide la medesima logica di implementazione del Decision Tree, differenziandosi unicamente per l'algoritmo adottato.

```
# selezionare le colonne di input e di output
input_cols = ["Delay", "DeviceID"]
output_cols = ["TTL", "TransmissionPower", "TransmissionsNumber", "
    IntervalTime"]

# dividi le colonne di input e di output
X = dataset[input_cols]
y = dataset[output_cols]

# applicare la Standard Scaler alle colonne di input
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# creare il modello di Decision Tree e addestrarlo
model = KNeighborsClassifier()
model.fit(X_scaled, y)

# Salva il modello su file
dump(model, 'Models/01 KNN-Delay.joblib')
```

A.5.7 Gaussian Naive Bayes - creazione del modello

Per creare il modello per l'algoritmo Gaussian Naive Bayes, si procede attraverso la creazione di 3 modelli intermedi. In seguito, sono riportati 4 step di codice.

- Il primo step permette di generare un modello che ricevendo in input Delay e NodeID, restituirà in output il TTL ideale.
- Il secondo step crea un modello che, avendo in input Delay, NodeID e TTL, fornirà la potenza di trasmissione migliore in output.
- Il terzo step crea un modello che, considerando in input Delay, NodeID, TTL e potenza di trasmissione, restituirà il numero di trasmissioni ottimale in output.
- Infine, il quarto step genera il modello che richiede in input Delay, NodeID, TTL, potenza di trasmissione e numero di trasmissioni, restituendo in output l'intervallo migliore tra le trasmissioni.

Step n.1:

```
# Seleziona le feature di input e il target
input_features = ['Delay', 'DeviceID']
target_features = ['TTL']

# Crea il dataset di training e testing con tutte le feature
X = dataset[input_features]
y = dataset[target_features]

# Crea il modello per la previsione di tutti i target
model = GaussianNB()
model.fit(X, y)

dump(model, 'Models/01 NaiveBayes-Delay-Step1.joblib')
```

Step n.2:

```
# Seleziona le feature di input e il target
input_features = ['Delay', 'DeviceID', 'TTL']
target_features = ['TransmissionPower']

# Crea il dataset di training e testing con tutte le feature
X = dataset[input_features]
y = dataset[target_features]

# Crea il modello per la previsione di tutti i target
model = GaussianNB()
model.fit(X, y)

dump(model, 'Models/01 NaiveBayes-Delay-Step2.joblib')
```

Step n.3:

```
# Seleziona le feature di input e il target
input_features = ['Delay', 'DeviceID', 'TTL', 'TransmissionPower']
target_features = ['TransmissionsNumber']

# Crea il dataset di training e testing con tutte le feature
X = dataset[input_features]
y = dataset[target_features]

# Crea il modello per la previsione di tutti i target
model = GaussianNB()
model.fit(X, y)

dump(model, 'Models/01 NaiveBayes-Delay-Step3.joblib')
```

Step n.4:

```
# Seleziona le feature di input e il target
input_features = ['Delay', 'DeviceID', 'TTL', 'TransmissionPower', '
    TransmissionsNumber']
target_features = ['IntervalTime']

# Crea il dataset di training e testing con tutte le feature
X = dataset[input_features]
y = dataset[target_features]

# Crea il modello per la previsione di tutti i target
model = GaussianNB()
model.fit(X, y)

dump(model, 'Models/01 NaiveBayes-Delay-Step4.joblib')
```

A.5.8 Utilizzo dei modelli (Decision Tree, Random Forest, Gradient Boosting, KNN)

Il codice in questione è in grado di effettuare il caricamento di un modello precedentemente salvato in un file con estensione *.joblib*. Dopo aver effettuato l'operazione di caricamento del modello, esso viene impiegato per individuare la configurazione ottimale. Ed infine, vengono salvati i risultati in un file *CSV*.
In questo specifico caso è riporato l'implementazione relativa all'algoritmo Decision Tree. Tuttavia, la logica impiegata per gli altri algoritmi è del tutto equivalente ad eccezione del Gaussian Naive Bayes.

```
# Carica il modello addestrato
clf = load('Models/01_DecisionTree-Delay.joblib')

# Combinazione di tutti i parametri
combinations = list(itertools.product(DelayValues, IDNodos))

output = []
for combination in combinations:
  DelayValue = combination[0]
  IDNodo = combination[1]

  # Esegui la predizione
  prediction = clf.predict([[DelayValue, IDNodo]])
  print(str(prediction))

  row = []
  row.append(DelayValue)
  row.append(IDNodo)
  row.append(prediction[0][0])
  row.append(prediction[0][1])
  row.append(prediction[0][2])
  row.append(prediction[0][3])
  output.append(row)

#print(output)
dataframeOutput = pd.DataFrame(output)
header = ['Delay (ms)', 'NodeId', 'TTL', 'Potenza di trasmissione', '
    Numero di trasmissioni', 'Intervallo tra le trasmissioni']
dataframeOutput.columns = header

# Esportare in un file CSV
dataframeOutput.to_csv('Results/01 DecisionTree-Delay.csv', index=False
    )
```

A.5.9 Utilizzo del modello Gaussian Naive Bayes

Come già menzionato, per la creazione del modello Gaussian Naive Bayes sono stati utilizzati tre modelli intermedi. Inizialmente, i quattro modelli necessari vengono caricati. Siccome, l'utente inserisce i valori di Delay attesi e NodeID, viene richiamato il primo modello. Successivamente, i valori ottimali vengono estratti combinando l'input dell'utente con l'output del passaggio precedente. Quindi, nel primo passaggio viene individuato il TTL ottimale, nel secondo passaggio la potenza di trasmissione, nel terzo passaggio il numero di trasmissioni e, infine, nel quarto passaggio l'intervallo di trasmissione.
Il codice che attua questa procedura è riportato di seguito.

```
# Carica il modello addestrato
clf1 = load('Models/01 NaiveBayes-Delay-Step1.joblib')
clf2 = load('Models/01 NaiveBayes-Delay-Step2.joblib')
clf3 = load('Models/01 NaiveBayes-Delay-Step3.joblib')
clf4 = load('Models/01 NaiveBayes-Delay-Step4.joblib')

# Combinazione di tutti i parametri
combinations = list(itertools.product(DelayValues, DeviceID))
dataframeInput = pd.DataFrame(combinations)
header = ['Delay', 'DeviceID']
dataframeInput.columns = header

prediction = clf1.predict(dataframeInput)
output = []
for index, inputRow in dataframeInput.iterrows():
    row = []
    DelayValue = float(inputRow['Delay'])
    IDNodo = int(inputRow['DeviceID'])
    TTL = min(TTLValueSet, key=lambda x: abs(x - prediction[index]))
    row.append(DelayValue)
    row.append(IDNodo)
    row.append(TTL)
    output.append(row)

dataframeOutputStep1 = pd.DataFrame(output)
header = ['Delay', 'DeviceID', 'TTL']
dataframeOutputStep1.columns = header

prediction = clf2.predict(dataframeOutputStep1)
output = []
for index, inputRow in dataframeOutputStep1.iterrows():
    row = []
    DelayValue = float(inputRow['Delay'])
    IDNodo = int(inputRow['DeviceID'])
    TTL = int(inputRow['TTL'])
    TransmissionPower =  min(TPValueSet, key=lambda x: abs(x -
   prediction[index]))
```

```
    row.append(DelayValue)
    row.append(IDNodo)
    row.append(TTL)
    row.append(TransmissionPower)
    output.append(row)


dataframeOutputStep2 = pd.DataFrame(output)
header = ['Delay', 'DeviceID', 'TTL', 'TransmissionPower']
dataframeOutputStep2.columns = header

prediction = clf3.predict(dataframeOutputStep2)
output = []
for index, inputRow in dataframeOutputStep2.iterrows():
    row = []
    DelayValue = float(inputRow['Delay'])
    IDNodo = int(inputRow['DeviceID'])
    TTL = int(inputRow['TTL'])
    TransmissionPower = int(inputRow['TransmissionPower'])
    TransmissionsNumber =  min(TNValueSet, key=lambda x: abs(x -
   prediction[index]))
    row.append(DelayValue)
    row.append(IDNodo)
    row.append(TTL)
    row.append(TransmissionPower)
    row.append(TransmissionsNumber)
    output.append(row)

dataframeOutputStep3 = pd.DataFrame(output)
header = ['Delay', 'DeviceID', 'TTL', 'TransmissionPower', '
   TransmissionsNumber']
dataframeOutputStep3.columns = header

prediction = clf4.predict(dataframeOutputStep3)
output = []
for index, inputRow in dataframeOutputStep3.iterrows():
    row = []
    DelayValue = float(inputRow['Delay'])
    IDNodo = int(inputRow['DeviceID'])
    TTL = int(inputRow['TTL'])
    TransmissionPower = int(inputRow['TransmissionPower'])
    TransmissionsNumber = int(inputRow['TransmissionsNumber'])
    IntervalTime = min(TIValueSet, key=lambda x: abs(x - prediction[
   index]))
    row.append(DelayValue)
    row.append(IDNodo)
    row.append(TTL)
    row.append(TransmissionPower)
    row.append(TransmissionsNumber)
```

```
    row.append(IntervalTime)
    output.append(row)

dataframeOutputStep4 = pd.DataFrame(output)
header = ['Delay', 'DeviceID', 'TTL', 'TransmissionPower', '
    TransmissionsNumber', 'IntervalTime']
dataframeOutputStep4.columns = header

print(dataframeOutputStep4)

dataframeOutputStep4.to_csv('Results/01 GaussinNB-Delay.csv', index=
    False)
```

Ringraziamenti

Sono profondamente grato a tutte le persone che hanno contribuito alla realizzazione di questo percorso.

Desidero innanzitutto ringraziare il mio relatore, **Angelo Trotta**, per la sua guida e il suo sostegno costante durante il mio percorso di studio. Il suo supporto è stato determinante per la realizzazione di questo lavoro, e gli sono molto grato per avermi fornito preziosi consigli e suggerimenti.

Vorrei inoltre esprimere la mia gratitudine al mio correlatore, **Leonardo Montecchiari**, per il suo prezioso contributo e per avermi assistito durante lo sviluppo di questo progetto. Senza la sua esperienza e il suo sostegno, non sarei riuscito a completare questo lavoro con successo.

Un ringraziamento speciale va alla **mia famiglia**, che mi ha sostenuto costantemente e mi ha dato la forza necessaria per affrontare le sfide che ho incontrato lungo il percorso. Il loro supporto è stato fondamentale per il raggiungimento dei miei obiettivi, e sono molto grato per tutto quello che hanno fatto per me.

Infine, ma non meno importante, desidero ringraziare i **miei amici** per il loro sostegno, le loro risate e il loro incoraggiamento. Il loro supporto morale è stato un sostegno fondamentale durante questo percorso, e non avrei potuto farcela senza di loro. Grazie di cuore per avermi spronato a dare il massimo e per essere stati al mio fianco in ogni momento.

Ancora una volta, desidero esprimere la mia sincera gratitudine a tutte le persone che hanno reso possibile la realizzazione di questo percorso. Grazie di cuore per tutto quello che avete fatto per me.

Elenco delle figure

Elenco delle tabelle

Bibliografia e sitografia

[1] Leonardo Montecchiari, Angelo Trotta, Luciano Bononi, Marco Di Felice. **Bluetooth Mesh Technology for the Joint Monitoring of Indoor Environments and Mobile Device Localization: A Performance Study**. IEEE Annual Consumer Communications Networking Conference.

[2] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M. (2015). **Internet of things: A survey on enabling technologies, protocols, and applications**. IEEE Communications Surveys Tutorials, 17(4), 2347-2376.

[3] Choi, S., Kim, S. J., Kim, S., Park, S. (2018). **BLE mesh networking: A review. IEEE Communications Magazine**, 56(9), 192-198.

[4] Hwang, D., Shin, B., Jeon, M., Kim, D., Park, C., Lee, Y. (2019). **BLE mesh networking for IoT applications: A comprehensive survey**. IEEE Access, 7, 86513-86523.

[5] Giannoulis, S., Skordoulis, D. (2019). **Bluetooth classic vs. BLE technology: A comparative review. International Journal of Computer Networks and Applications**, 6(2), 38-47.

[6] Alpaydin, E. (2010). **Introduction to machine learning** (2nd ed.). MIT Press

[7] Kotsiantis, S. B., Zaharakis, I. D., Pintelas, P. E. (2007). **Supervised machine learning: A review of classification techniques**. Informatica, 31(3), 249-268.

[8] Ahmed Wasif Reza, Yu Wang, and Chan-Hyun Youn. **Bluetooth Low Energy Mesh Networking: A Survey**. IEEE Access, 8, 2020.

[9] Ahmed Wasif Reza, Yu Wang, and Chan-Hyun Youn. **Standardization of Bluetooth Low Energy Mesh Networking: A Survey**. IEEE Communications Surveys Tutorials, 22, Numero 2, 2020.

[10] Tanweer Ali, Vasilis Chatzigiannakis, and Mohamed Younis. **Bluetooth Low Energy Mesh Networking: A Comprehensive Survey of Its Evolution**

and Ecosystem. IEEE Communications Surveys Tutorials, Volume 22, Numero 3, 2020.

[11] Muhannad T. Suleiman, Abdulkarim Al-Furaih, and Oluwaseun A. Adedugbe. **A Comprehensive Survey of Bluetooth Low Energy Mesh Networking: Advancements, Applications, and Future Directions**. IEEE Communications Surveys Tutorials, Volume 23, Numero 3, 2021

[12] Mohammad R. Islam, Farhana H. Zulkernine, and Nazim Agoulmine. **Internet of Things (IoT) and Artificial Intelligence (AI) Integration: A Review**. IEEE Internet of Things Journal, Volume 7, Numero 2, 2020

[13] Muhammad Ikram Ashraf, Khaled Salah, Imran Ahmed, and Muhammad Sher. **A Comprehensive Survey on Security of Internet of Things: Revisiting Challenges and Countermeasures**. IEEE Internet of Things Journal, Volume 7, Numero 10, 2020.

[14] Max Tegmark. **The Future of Artificial Intelligence: Opportunities and Challenges**. Journal of the American Philosophical Association, Volume 2, Numero 4, 2016.

[15] **Bluetooth Overview** - https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

[16] **Esempio espressif** - https://github.com/espressif/esp-idf/tree/master/examples/bluetooth/esp_ble_mesh/ble_mesh_n

[17] **Python** - https://www.python.org/

[18] **Scikit-learn** - https://scikit-learn.org/stable/

[19] **Paho.mqtt** - https://pypi.org/project/paho-mqtt/

[20] **Pandas** - https://pandas.pydata.org/

[21] **Numpy** - https://numpy.org/

[22] **Matplotlib** - https://matplotlib.org/

[23] **HTML** - https://html.spec.whatwg.org/

[24] **CSS** - https://www.w3.org/TR/CSS/

[25] **JQuery** - https://jquery.com/

[26] **Node.js** - https://nodejs.org/en/

[27] **MySQL** - https://www.mysql.com/it/

[28] **Mosquitto** - https://mosquitto.org/

[29] **nRF Mesh** - https://www.nordicsemi.com/Products/Development-tools/nrf-mesh

[30] **Espressif** - https://www.espressif.com/en/products/socs/esp32

[31] **ESP-IDF** - https://github.com/espressif/esp-idf

[32] **ESP-IDF Setup** - https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html

[33] **ESP-IDF VSCode** - https://docs.espressif.com/projects/esp-idf/en/v4.3.1/esp32/get-started/vscode-setup.html

[34] **Tutorial BLE Mesh** - https://docs.espressif.com/projects/esp-idf/en/v3.3.3/api-guides/esp-ble-mesh/ble-mesh-index.html

[35] **Bluetooth fundamentals parte 1** - https://www.bluetooth.com/blog/the-fundamental-concepts-of-bluetooth-mesh-networking-part-1/

[36] **Bluetooth fundamentals parte 2** - https://www.bluetooth.com/blog/the-fundamental-concepts-of-bluetooth-mesh-networking-part-2/