Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il management

Progettazione e sviluppo di un sistema IoT con interfaccia mobile per il monitoraggio ambientale della sala server

Relatore:
Chiar.mo Prof.
Marco Di Felice

Correlatore:
Dott.
Marco Giordani

Presentata da:
Davide Boschi

II sessione
Anno Accademico 2024/2025

Abstract

Negli ultimi anni la crescente complessità delle infrastrutture digitali e la loro dipendenza da condizioni ambientali stabili hanno reso sempre più evidente la necessità di soluzioni affidabili per il monitoraggio in tempo reale di parametri critici. Le sale server, cuore pulsante dei servizi informatici, richiedono sistemi capaci di prevenire anomalie dovute a variazioni di temperatura, umidità, qualità dell'aria, vibrazioni o rumore, garantendo continuità operativa e riducendo il rischio di guasti improvvisi.

Alla luce di questa esigenza, la tesi presenta la progettazione e la realizzazione di un prototipo IoT embedded, basato su microcontrollore ESP32 e integrato con la piattaforma di monitoraggio Zabbix, per la raccolta e l'analisi di parametri ambientali fondamentali. Accanto al dispositivo hardware è stata sviluppata un'applicazione mobile Android che, tramite connessione Bluetooth Low Energy, consente al personale tecnico di configurare e supervisionare i dispositivi in modo diretto e intuitivo, anche in locale.

La validazione in scenari realistici ha evidenziato l'affidabilità del prototipo, la coerenza delle misurazioni e la capacità di notificare tempestivamente eventuali anomalie. La soluzione si è dimostrata robusta, replicabile e pronta per essere estesa con nuove funzionalità.

Questo lavoro si inserisce nel più ampio contesto dell'innovazione tecnologica applicata al monitoraggio ambientale, ponendo le basi per futuri sviluppi orientati all'integrazione di ulteriori sensori, alla realizzazione di versioni multipiattaforma e all'adozione del sistema nelle infrastrutture informatiche dell'Università di Bologna.

Indice

Elenco delle figure

Listings

Capitolo 1

Introduzione

Negli ultimi anni il tema del monitoraggio ambientale ha assunto un ruolo di crescente rilevanza, sia in ambito domestico sia all'interno di infrastrutture tecnologiche e ambienti controllati, come sale server, laboratori e data center. L'aumento della complessità dei sistemi informatici, unito alla progressiva dipendenza da reti e servizi digitali, rende imprescindibile garantire condizioni ambientali stabili, al fine di prevenire guasti e malfunzionamenti dovuti a fattori quali temperatura, umidità o eventi acustici anomali. In tale contesto, l'adozione di soluzioni basate sull'Internet of Things (IoT) si è affermata come una strategia efficace per raccogliere dati in tempo reale e supportare processi decisionali tempestivi.

Il paradigma IoT consente infatti di connettere dispositivi embedded, sensori e attuatori a piattaforme di gestione centralizzate, integrando il monitoraggio continuo con sistemi di allerta e, in prospettiva, anche con meccanismi di automazione. Nel caso specifico delle sale server, l'impiego di sensori intelligenti contribuisce non solo a mantenere condizioni operative ottimali, ma anche a garantire la continuità dei servizi e a ridurre i costi legati ad attività di manutenzione preventiva e correttiva.

La presente tesi nasce da una concreta esigenza manifestata dai tecnici del Dipartimento di Informatica – Scienza e Ingegneria dell'Università di Bologna, interessati a sperimentare un sistema prototipale leggero, robusto e integrabile con l'infrastruttura già esistente. A partire da tali richieste, è stato sviluppato un dispositivo embedded basato su microcontrollore ESP32, progettato per rilevare parametri ambientali chiave e comunicare con la piattaforma di monitoraggio Zabbix.

Il sistema proposto si compone di tre elementi fondamentali: un firmware per la scheda ESP32, capace di gestire sensori di temperatura, umidità e rumore; un'infrastruttura di raccolta e visualizzazione dei dati su Zabbix, corredata da item, trigger e dashboard dedicate; e un'applicazione mobile sviluppata in Kotlin con architettura MVVM, che consente l'interazione diretta con i dispositivi tramite Bluetooth Low Energy (BLE).

L'obiettivo perseguito è stato quello di realizzare un prototipo affidabile e facilmente replicabile, capace di fornire dati ambientali in tempo reale e di segnalare prontamente

eventuali anomalie al personale tecnico. L'applicazione mobile, inoltre, offre funzionalità di configurazione dei parametri di rete e di gestione del dispositivo, mettendo a disposizione un'interfaccia intuitiva e sicura per l'interazione locale.

La tesi è articolata in cinque capitoli principali. Il secondo capitolo presenta lo stato dell'arte, con una panoramica sul paradigma IoT, sulle principali tecnologie di connettività e sulle applicazioni esistenti per il monitoraggio ambientale. Il terzo capitolo illustra gli aspetti progettuali del sistema, distinguendo i requisiti hardware da quelli software. Il quarto capitolo è dedicato all'implementazione pratica, comprendente lo sviluppo del firmware ESP32, l'integrazione con la piattaforma Zabbix, lo sviluppo dell'applicazione mobile e la realizzazione del contenitore prototipale in stampa 3D. Il quinto capitolo riporta i risultati delle attività di validazione, con particolare attenzione all'affidabilità delle comunicazioni e alla qualità dei dati raccolti. Infine, nelle conclusioni, vengono discussi i limiti del prototipo e delineate le possibili evoluzioni future, evidenziando come il contributo principale del lavoro consista nella realizzazione di un sistema completo (hardware, software e mobile) pienamente integrato con infrastrutture esistenti.

Capitolo 2

Stato dell'arte

2.1 Il paradigma dell'Internet of Things

Il concetto di Internet of Things (IoT), in italiano Internet delle Cose, rappresenta uno dei paradigmi tecnologici più influenti e trasformativi degli ultimi due decenni. Il termine è stato coniato per la prima volta nel 1999 da Kevin Ashton, ricercatore del MIT (Massachusetts Institute of Technology), che ne descrisse la natura come un'estensione diretta di Internet agli oggetti fisici, capaci di interagire tra loro e con l'ambiente circostante[8].

L'idea di fondo è che ogni oggetto possa acquisire un'identità digitale e, attraverso la connettività, contribuire a un sistema informativo distribuito e globale. Le applicazioni spaziano da dispositivi indossabili e assistenti domestici intelligenti, fino a sistemi complessi in ambito aziendale e industriale, quali linee di produzione automatizzate, veicoli connessi per la logistica e reti di sensori per il monitoraggio ambientale.

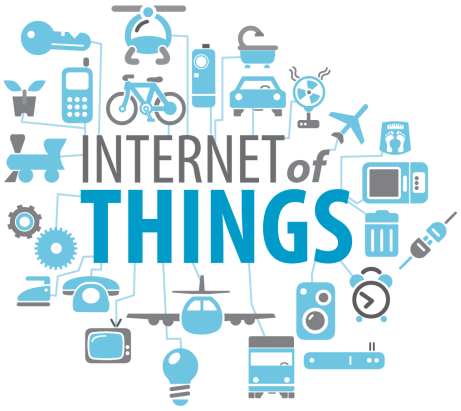

Figura 2.1: Rappresentazione concettuale del paradigma IoT[18].

2.1.1 Evoluzione storica

L'evoluzione del paradigma IoT può essere suddivisa in diverse fasi storiche, ciascuna caratterizzata da specifiche tecnologie abilitanti.

La prima fase, risalente ai primi anni 2000, è stata dominata dall'impiego della tecnologia RFID (Radio Frequency Identification), che ha reso possibile l'identificazione automatica degli oggetti e la tracciabilità delle merci nelle supply chain industriali, con applicazioni nei sistemi di logistica e gestione dei magazzini.

Con la diffusione degli smart devices, come smartphone e sensori intelligenti, l'IoT si è progressivamente esteso a contesti quotidiani, introducendo soluzioni come termostati connessi e dispositivi indossabili per il monitoraggio della salute.

Negli anni successivi, l'avvento del cloud computing ha fornito l'infrastruttura necessaria per raccogliere, elaborare e archiviare grandi quantità di dati generati dai dispositivi connessi. Il cloud ha permesso lo sviluppo di piattaforme centralizzate in grado di gestire milioni di sensori e attuatori, abilitando la creazione di ecosistemi IoT scalabili, ad esempio nelle smart city e nei sistemi di trasporto intelligenti.

In epoca più recente, l'attenzione si è spostata verso il concetto di edge computing, che porta capacità di calcolo e analisi vicino alla sorgente dei dati, riducendo latenza e congestione di rete [6]. Questa transizione rappresenta una risposta alle esigenze sempre più stringenti di applicazioni in tempo reale, come i veicoli autonomi, la robotica collaborativa e il monitoraggio ambientale avanzato.

2.1.2 Caratteristiche principali

L'Internet of Things si caratterizza per una serie di tratti distintivi che ne delineano la portata innovativa e ne giustificano l'ampia diffusione. In primo luogo, la connettività pervasiva consente ai dispositivi di interagire attraverso una molteplicità di protocolli e tecnologie di rete, tra cui Wi-Fi, Bluetooth Low Energy, ZigBee, LoRaWAN e, più recentemente, il 5G. Questa varietà di soluzioni permette di coprire scenari che spaziano dalle comunicazioni locali, come quelle tra smartwatch e smartphone tramite BLE, fino ai contesti industriali, in cui sensori distribuiti possono sfruttare LoRaWAN per trasmettere dati a lungo raggio.

Un secondo elemento chiave è rappresentato dalla sensoristica distribuita, che consente la raccolta continua di informazioni dall'ambiente circostante. La presenza capillare di sensori apre la strada a un ampio spettro di applicazioni, dalla domotica (ad esempio il monitoraggio di temperatura e umidità nelle smart home) fino al controllo delle condizioni ambientali nei data center o negli impianti produttivi.

Un ulteriore aspetto fondamentale è l'interoperabilità: la capacità di far dialogare dispositivi eterogenei e provenienti da ecosistemi differenti costituisce un requisito imprescindibile per la realizzazione di soluzioni IoT su larga scala. Tale obiettivo è perseguito mediante l'adozione di standard comuni e middleware dedicati [21], insieme a protocolli

come MQTT e CoAP che rappresentano strumenti diffusi per favorire la compatibilità e l'integrazione tra piattaforme diverse.

Infine, l'IoT non si limita alla mera acquisizione di dati, ma ne abilita l'analisi e la storicizzazione attraverso algoritmi di machine learning, tecniche di big data analytics e sistemi di supporto alle decisioni. L'elaborazione e la correlazione delle informazioni su larga scala consentono di individuare pattern e anomalie, rendendo possibili applicazioni avanzate come la manutenzione predittiva o la gestione intelligente delle smart city.

Questi elementi distintivi rendono l'IoT non soltanto un paradigma tecnologico, ma anche un fenomeno socio-economico in grado di ridefinire modelli di business, abitudini quotidiane e processi produttivi.

2.1.3 Limiti attuali

Nonostante i progressi compiuti negli ultimi anni, l'adozione su larga scala dell'Internet of Things presenta ancora numerose sfide e criticità. Un primo aspetto riguarda la sicurezza e la privacy: la raccolta e la trasmissione continua di dati sensibili espongono a rischi di cyberattacchi e violazioni della riservatezza. Sono frequenti, ad esempio, le segnalazioni di vulnerabilità in dispositivi domestici connessi o nei sistemi di tracciamento sanitario, a conferma di come la protezione dei dati costituisca un elemento imprescindibile [19].

Un ulteriore limite è rappresentato dalla scarsa standardizzazione. L'assenza di un quadro normativo e tecnologico universalmente riconosciuto ostacola l'interoperabilità tra piattaforme differenti, generando frammentazione del mercato. In questo scenario, diverse alleanze industriali e l'adozione di protocolli comuni, come MQTT o CoAP, stanno cercando di ridurre il divario e favorire l'integrazione tra sistemi eterogenei.

La scalabilità costituisce un'altra criticità rilevante. La gestione di miliardi di dispositivi connessi richiede infrastrutture in grado di supportare volumi crescenti di dati e connessioni, senza comprometterne le prestazioni. Architetture distribuite e reti di nuova generazione, come il 5G, rappresentano alcune delle soluzioni oggi disponibili che mirano a rispondere a queste esigenze.

Infine, il consumo energetico rimane un problema significativo, soprattutto in contesti caratterizzati da risorse limitate. La durata delle batterie e l'efficienza energetica dei dispositivi IoT sono fattori determinanti per la sostenibilità dei sistemi nel lungo periodo. Tecnologie come le reti LPWAN (Low Power Wide Area Network) e l'energy harvesting stanno emergendo come strategie promettenti per affrontare questa sfida.

La capacità di risolvere tali criticità sarà determinante per il consolidamento e la diffusione futura dell'IoT, dalle applicazioni domestiche e personali fino ai sistemi industriali e alle infrastrutture critiche su larga scala.

2.1.4 Dati di mercato

Secondo stime recenti, il numero di dispositivi connessi a livello globale passerà da circa 9,7 miliardi nel 2020 a oltre 29–30 miliardi entro il 2030, quasi triplicando nell'arco di un decennio [24, 14, 25]. Tale espansione è trainata dalla diffusione di soluzioni IoT in molteplici settori, dalla smart home alle smart city, fino al settore automobilistico. Parallelamente, il mercato globale dell'IoT, stimato circa 928 miliardi di dollari nel 2020, è previsto superare i 4.200 miliardi entro il 2030, con un tasso di crescita annuo composto stimato tra il 12% e il 30% [13].

2.1.5 Integrazione con l'Intelligenza Artificiale

L'integrazione tra Internet of Things e Intelligenza Artificiale (IA) rappresenta un'evoluzione significativa: dal semplice rilevamento reattivo si passa a forme avanzate di percezione e decisione automatizzata. I dati generati da reti di sensori possono essere analizzati mediante tecniche di machine learning, come la classificazione delle anomalie, il forecasting o il rilevamento di outlier, sia in cloud sia in edge, con l'obiettivo di ridurre la latenza e il consumo di banda. Questa sinergia consente di trasformare la l'ingente volume di dati generati dall'IoT in informazioni a valore aggiunto, supportando decisioni autonome e predittive. Tra gli scenari applicativi più rilevanti si annoverano l'ottimizzazione dei consumi energetici, la manutenzione predittiva di infrastrutture complesse e l'individuazione precoce di situazioni anomale o critiche. L'adozione di pipeline di Machine Learning Operations (MLOps) e di piattaforme di monitoraggio avanzate permette di chiudere il ciclo che parte dalla raccolta dei dati, passa attraverso la loro trasformazione in conoscenza utile e si traduce infine in azioni concrete, generando benefici tangibili in termini di efficienza, sicurezza e qualità dei servizi [21, 16].

2.2 Applicazioni dell'IoT

2.2.1 Ambiti applicativi

Le potenzialità dell'IoT sono estremamente ampie e trasversali: i suoi ambiti di applicazione spaziano dalla sfera personale a quella industriale, fino a coinvolgere interi ecosistemi urbani ed energetici. Di seguito vengono presentati alcuni dei settori più significativi in cui l'IoT ha trovato e continua a trovare applicazione.

Domotica avanzata (Smart Home)

In ambito domestico, l'IoT si è affermato con dispositivi intelligenti capaci di migliorare comfort, sicurezza ed efficienza energetica. Assistenti vocali, termostati connessi, sistemi di illuminazione automatizzati e videocamere di sorveglianza consentono agli utenti

di monitorare e controllare l'abitazione in tempo reale, riducendo i consumi e ottimizzando l'esperienza quotidiana. L'interoperabilità tra diversi ecosistemi (Amazon Alexa, Google Home, Apple HomeKit) ha ampliato ulteriormente le possibilità di integrazione, favorendo una diffusione sempre più capillare.

Smart City

In ambito urbano, le città intelligenti rappresentano un campo privilegiato di applicazione dell'IoT, con soluzioni mirate al miglioramento della qualità della vita. Sistemi di gestione del traffico, monitoraggio ambientale, illuminazione pubblica intelligente e raccolta rifiuti ottimizzata sono solo alcuni esempi. L'integrazione dei dati provenienti da sensori distribuiti consente alle amministrazioni di pianificare interventi più efficaci, ridurre l'inquinamento e garantire maggiore sostenibilità delle infrastrutture, migliorando la vivibilità urbana.

Industria 4.0

Nel settore industriale, l'IoT costituisce uno dei pilastri della cosiddetta quarta rivoluzione industriale. I sistemi cyber-fisici, le macchine interconnesse e la manutenzione predittiva permettono di incrementare produttività e qualità, riducendo al contempo i costi di gestione. La disponibilità costante di dati provenienti da linee di produzione e macchinari abilita l'impiego di algoritmi di machine learning per rilevare anomalie e ottimizzare i processi in tempo reale. Questo approccio segna il passaggio da un modello reattivo a uno proattivo, basato su previsioni accurate e automazione intelligente.

Settore sanitario (Healthcare)

Nel settore sanitario, le applicazioni IoT stanno rivoluzionando la diagnosi e il monitoraggio dei pazienti. Dispositivi indossabili (wearables), sensori biomedicali e strumenti di telemedicina permettono un controllo continuo dei parametri vitali, riducendo la necessità di visite ospedaliere frequenti. Esempi concreti includono smartwatch, cerotti intelligenti e sensori impiantabili. L'integrazione con piattaforme di analisi dei dati facilita la rilevazione precoce di patologie e consente lo sviluppo di terapie personalizzate, favorendo un sistema sanitario più efficiente e centrato sul paziente.

Trasporti e logistica

La logistica e la mobilità connessa rappresentano un altro ambito di grande rilevanza. L'IoT abilita il tracciamento in tempo reale delle merci lungo tutta la supply chain, migliorando trasparenza e riducendo i rischi di smarrimento o contraffazione. Nei trasporti, sensori installati su veicoli e infrastrutture stradali contribuiscono alla sicurezza e all'ottimizzazione dei flussi di traffico. L'adozione di veicoli a guida autonoma rappresenta

una prospettiva futura, in cui l'IoT sarà essenziale per la comunicazione veicolo–veicolo (V2V) e veicolo–infrastruttura (V2I).

Monitoraggio ambientale

Il monitoraggio ambientale costituisce un'applicazione trasversale a diversi settori. Sensori IoT consentono la rilevazione di parametri come qualità dell'aria (CO_2, particolato fine PM2.5, composti organici volatili VOC), temperatura, umidità, rumore e vibrazioni [11]. Questi dati sono fondamentali sia in ambito urbano, per valutare l'impatto dell'inquinamento, sia in contesti industriali e accademici, come il monitoraggio di sale server o laboratori. La raccolta in tempo reale abilita sistemi di allerta precoce e politiche di sostenibilità più efficaci.

Agricoltura di precisione

In agricoltura, l'IoT permette di ottimizzare risorse e produttività attraverso sensori nel suolo e stazioni meteorologiche locali che monitorano umidità, nutrienti e condizioni climatiche. Sistemi di irrigazione intelligenti e droni connessi consentono interventi mirati, riducendo sprechi e aumentando la resa delle colture. Questo approccio, noto anche come precision agriculture, rappresenta una leva strategica per la sicurezza alimentare e la sostenibilità.

Energia e smart grid

Nel settore energetico, l'IoT supporta lo sviluppo delle reti intelligenti (smart grid) attraverso contatori connessi, sistemi di bilanciamento del carico e monitoraggio dei consumi in tempo reale. Queste soluzioni favoriscono l'integrazione delle fonti rinnovabili, migliorano l'efficienza della distribuzione e aumentano la consapevolezza dei consumatori, promuovendo un modello energetico più sostenibile.

2.2.2 Ecosistemi e piattaforme

Le applicazioni IoT si basano su infrastrutture tecnologiche che possono essere di tipo commerciale o open source. Le soluzioni commerciali, come AWS IoT Core e Azure IoT Hub, offrono piattaforme scalabili per la gestione di milioni di dispositivi connessi, includendo funzionalità di sicurezza, analisi dei dati e integrazione con algoritmi di intelligenza artificiale. Le soluzioni open source, come Zabbix, Grafana, InfluxDB e Prometheus, consentono invece di realizzare sistemi di monitoraggio e visualizzazione personalizzati e flessibili, riducendo i costi e garantendo una maggiore indipendenza dai fornitori [21]. La scelta tra piattaforme commerciali e open source dipende dal contesto applicativo, dalle esigenze di scalabilità e dal livello di controllo desiderato sull'infrastruttura.

2.3 Connettività e protocolli di comunicazione

2.3.1 Connettività dei dispositivi

La trasmissione dei dati raccolti dai sensori può avvenire attraverso diverse tecnologie di rete, la cui adozione dipende dal contesto applicativo e dai vincoli di progetto. Le soluzioni cablate, come l'Ethernet, garantiscono elevata affidabilità e bassa latenza, risultando particolarmente adatte in ambienti critici quali i data center, dove la continuità di servizio rappresenta un requisito essenziale.

Accanto a queste, le tecnologie wireless a corto e medio raggio costituiscono l'opzione più diffusa nei contesti domestici e aziendali. Il Wi-Fi offre una copertura capillare e una notevole flessibilità di utilizzo, pur comportando consumi energetici elevati e una maggiore esposizione a interferenze. Il Bluetooth, soprattutto nella variante Low Energy (BLE), consente collegamenti efficienti a basso consumo, rivelandosi ideale per dispositivi indossabili e applicazioni mobili. Altre soluzioni, come ZigBee e Z-Wave, sfruttano topologie di rete mesh e sono largamente adottate nella domotica e nell'automazione degli edifici, grazie alla capacità di connettere numerosi dispositivi con consumi contenuti [3].

Per scenari che richiedono copertura su lunghe distanze, si sono affermate tecnologie dedicate a basso consumo come LoRa e il relativo protocollo LoRaWAN, particolarmente indicate per applicazioni in agricoltura di precisione, monitoraggio ambientale e smart metering [1]. In parallelo, gli standard cellulari NB-IoT e LTE-M, sviluppati dal 3GPP, permettono di sfruttare le infrastrutture mobili esistenti per connettere milioni di dispositivi distribuiti, bilanciando affidabilità e ridotto consumo energetico [26]. Infine, l'avvento delle reti 5G ha introdotto capacità senza precedenti in termini di bassa latenza, alta densità di connessioni e qualità del servizio, aprendo la strada a scenari mission-critical come le smart city, i veicoli autonomi e l'industria 4.0 [2].

2.3.2 Protocolli di comunicazione

La scelta del protocollo rappresenta un elemento chiave nella progettazione di soluzioni IoT, in quanto influisce direttamente su affidabilità, scalabilità e interoperabilità del sistema. Come illustrato in Figura 2.2, i protocolli si distribuiscono su più livelli dello stack di comunicazione:

- **Livello fisico e collegamento dati (PHY/MAC)**: tecnologie di trasmissione come Wi-Fi, BLE e ZigBee, che definiscono il mezzo di connessione e la gestione dell'accesso al canale.
- **Livello rete**: protocolli come 6LoWPAN e RPL, pensati per estendere l'uso di IPv6 a dispositivi con risorse limitate e reti caratterizzate da alta perdita di pacchetti.

- **Livello trasporto**: protocolli TCP e UDP, che garantiscono rispettivamente affidabilità e leggerezza, a seconda che si privilegino integrità dei dati o efficienza.
- **Livello applicativo**: protocolli orientati allo scambio di dati tra dispositivi e piattaforme, tra cui MQTT (leggero e basato su publish/subscribe), CoAP (ottimizzato per dispositivi vincolati), HTTP/REST (diffuso per compatibilità con sistemi IT esistenti), AMQP (adatto a scenari enterprise) e soluzioni specifiche come i passive checks di Zabbix, utilizzati per integrare i dati in piattaforme di monitoraggio [3].
- **Applicazioni industriali**: protocolli come Modbus e BACnet, che garantiscono interoperabilità con sistemi legacy di automazione e controllo.

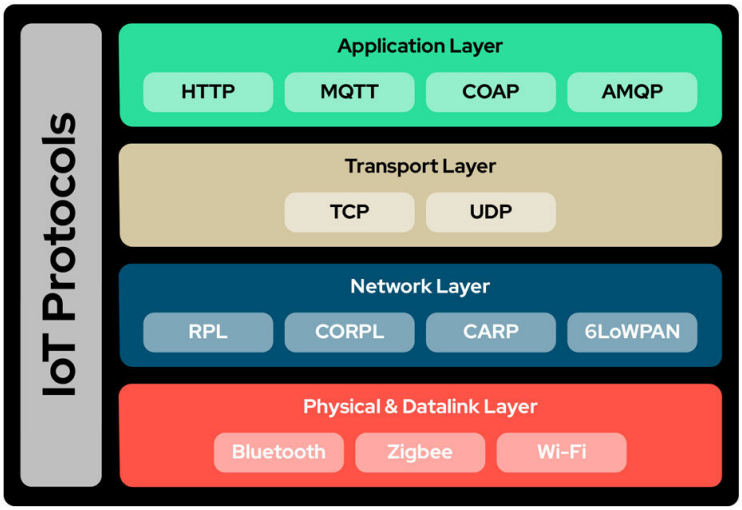

Figura 2.2: Classificazione dei principali protocolli di comunicazione IoT in base ai livelli dello stack [10].

2.4 L'IoT per le infrastrutture critiche e le sale server

Le infrastrutture critiche, come i data center e le sale server, rappresentano il cuore pulsante di numerose organizzazioni pubbliche e private. Esse ospitano sistemi informatici strategici, applicazioni mission-critical e grandi quantità di dati, la cui continuità

operativa è essenziale per garantire servizi digitali senza interruzioni. In tali contesti, il mantenimento di condizioni ambientali ottimali costituisce un requisito imprescindibile, in quanto anche brevi anomalie possono generare downtime, perdite economiche significative e danni reputazionali.

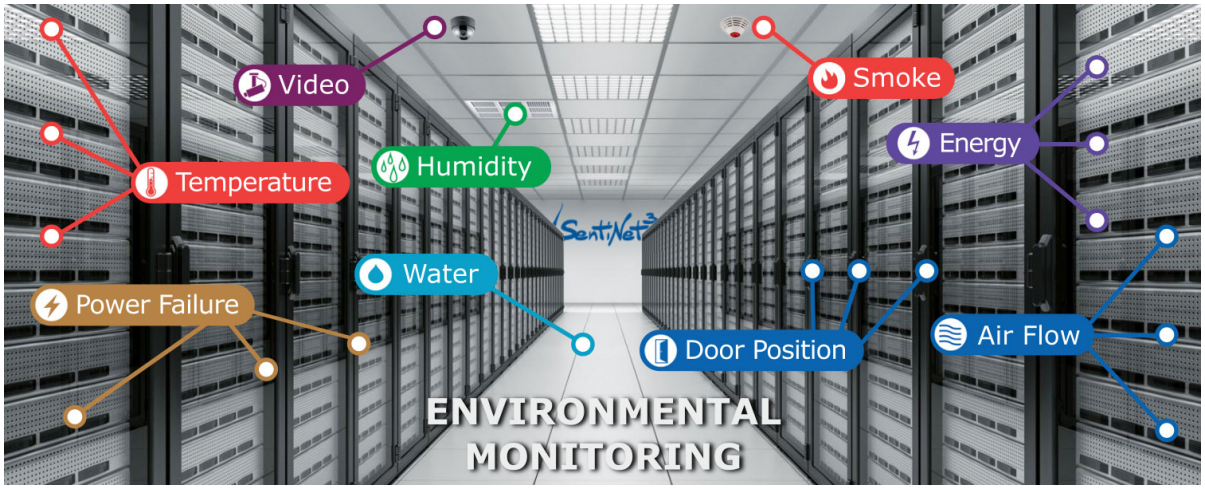

Figura 2.3: Parametri ambientali tipicamente monitorati in una sala server[22].

2.4.1 Sfide ambientali

Condizioni ambientali ottimali

Secondo le linee guida ASHRAE (American Society of Heating, Refrigerating and Air-Conditioning Engineers), la temperatura raccomandata all'interno di una sala server dovrebbe oscillare tra i 18 °C e i 27 °C, con un livello di umidità relativa compreso tra il 40% e il 60% [7]. Il rispetto di tali parametri riduce il rischio di fenomeni come surriscaldamento, condensa ed elettricità statica, tutti potenzialmente dannosi per l'integrità dei sistemi elettronici.

Rischi derivanti da condizioni non controllate

La mancata gestione di temperatura e umidità nelle sale server espone a rischi concreti:

- **Surriscaldamento**: causa guasti permanenti e riduce la vita utile dei componenti.
- **Condensa**: provoca cortocircuiti e danni irreversibili ai circuiti stampati.
- **Polvere e particolato**: accelerano l'usura dei sistemi di raffreddamento e compromettono la circolazione dell'aria.

- **Downtime**: interruzioni anche brevi possono avere conseguenze critiche, specialmente in ambiti finanziari, sanitari e governativi.

In un'epoca in cui la continuità dei servizi digitali è imprescindibile, questi rischi assumono un peso crescente e richiedono soluzioni di monitoraggio affidabili e proattive [19].

Tecniche tradizionali di monitoraggio

Storicamente, la gestione ambientale dei data center è stata affidata a Building Management System (BMS) centralizzati, che integrano sensori e attuatori con l'obiettivo di mantenere condizioni ottimali negli edifici tecnologici. All'interno delle sale server, apparecchiature di raffreddamento come i CRAC (Computer Room Air Conditioning) e i CRAH (Computer Room Air Handling) hanno rappresentato le soluzioni di riferimento. Parallelamente, protocolli come lo SNMP (Simple Network Management Protocol) hanno consentito di monitorare parametri operativi delle apparecchiature IT e degli impianti di rete, fornendo agli amministratori strumenti di supervisione e allerta [15]. Nonostante l'affidabilità dimostrata per decenni, questi sistemi risultano spesso costosi, complessi da implementare e limitati nella granularità dei dati raccolti.

2.4.2 Il contributo dell'IoT

L'adozione di soluzioni IoT rappresenta un avanzamento significativo rispetto agli approcci tradizionali al monitoraggio. La distribuzione di sensori consente di raccogliere dati molto più dettagliati, fino al livello del singolo rack o di specifiche zone della sala server. Un ulteriore vantaggio è la possibilità di ricevere avvisi in tempo reale quando i valori superano le soglie impostate, riducendo i tempi di intervento. I dati non sono solo visualizzati, ma anche storicizzati, permettendo l'individuazione di andamenti ricorrenti e l'applicazione di tecniche predittive per prevenire guasti e malfunzionamenti [16]. Infine, l'integrazione con piattaforme di monitoraggio già diffuse in ambito IT riduce i costi e semplifica la gestione. Questo approccio rende i sistemi più flessibili e apre la strada a scenari in cui le azioni correttive possono essere parzialmente automatizzate, migliorando la stabilità complessiva delle infrastrutture critiche.

2.5 Dispositivi e sensori per il monitoraggio ambientale

Il monitoraggio ambientale tramite dispositivi IoT si basa sull'impiego congiunto di sensori eterogenei e sistemi di connettività, che consentono di raccogliere, trasmettere ed

elaborare dati in tempo reale. Tali strumenti rivestono un ruolo centrale in contesti critici come le sale server, dove parametri quali temperatura, umidità e rumore devono essere costantemente mantenuti entro valori ottimali al fine di garantire continuità operativa e ridurre il rischio di guasti.

2.5.1 Sensori ambientali

I sensori ambientali costituiscono l'elemento cardine dei sistemi di monitoraggio, poiché permettono di rilevare in modo continuo e preciso i parametri critici che influiscono sul corretto funzionamento delle infrastrutture IT. A seconda della grandezza fisica osservata, esistono diverse categorie di sensori che si distinguono per principio di funzionamento, accuratezza e ambito applicativo.

Un primo ambito fondamentale è rappresentato dai sensori di temperatura, che costituiscono la prima linea di difesa contro il surriscaldamento dei sistemi informatici. Le tecnologie disponibili spaziano da soluzioni economiche, come i termistori analogici, fino a dispositivi digitali più accurati e stabili, tra cui i sensori DHT11 e DHT22 (che integrano anche la misura dell'umidità), il DS18B20 con interfaccia One-Wire [11] e modelli più avanzati come LM35, TMP102 o BME280, quest'ultimo capace di rilevare anche pressione e umidità. In contesti critici, come i data center di grandi dimensioni, trovano applicazione sonde certificate industriali quali PT100 e PT1000, caratterizzate da elevata stabilità e precisione.

Accanto alla temperatura, l'umidità rappresenta un parametro altrettanto importante per prevenire fenomeni di condensa e scariche elettrostatiche. I sensori disponibili si dividono principalmente in resistivi, meno costosi ma soggetti a degrado nel tempo, e capacitivi, oggi i più diffusi per affidabilità e stabilità. Tra i modelli più noti figurano DHT11, DHT22, SHT31, BME280 e BME680, mentre in ambito professionale vengono impiegati sensori certificati come la serie Honeywell HIH, in grado di garantire misure conformi agli standard industriali [21].

Un'ulteriore categoria è costituita dai sensori di rumore e vibrazione, sempre più utilizzati in ottica di manutenzione predittiva. Moduli semplici, basati su microfoni elettrete accoppiati a comparatori come l'LM393, consentono il rilevamento di eventi acustici oltre una certa soglia. Per applicazioni più avanzate vengono adottati microfoni MEMS, sia analogici che digitali (ad esempio con interfaccia I^2S), che permettono un'acquisizione continua del segnale e la relativa analisi in frequenza. In parallelo, accelerometri e sensori piezoelettrici vengono impiegati per rilevare vibrazioni meccaniche di ventole, dischi rigidi o altri componenti rotanti, con l'obiettivo di individuare precocemente anomalie e ridurre i rischi di fermo macchina [11].

Infine, la qualità dell'aria costituisce un ulteriore elemento da monitorare negli ambienti chiusi come le sale server. Sensori dedicati consentono la rilevazione di particolato atmosferico (PM1, PM2.5, PM10) tramite moduli come SDS011 e PMS7003, la misura di gas nocivi quali CO, CO_2, NO_x e composti organici volatili (VOC) attraverso dispositivi

come MQ-135 o CCS811, nonché la valutazione complessiva della qualità dell'aria indoor grazie a sensori integrati come il BME680. Questi strumenti contribuiscono non solo a salvaguardare l'integrità degli apparati elettronici, ma anche a garantire condizioni di lavoro salubri per il personale tecnico [21].

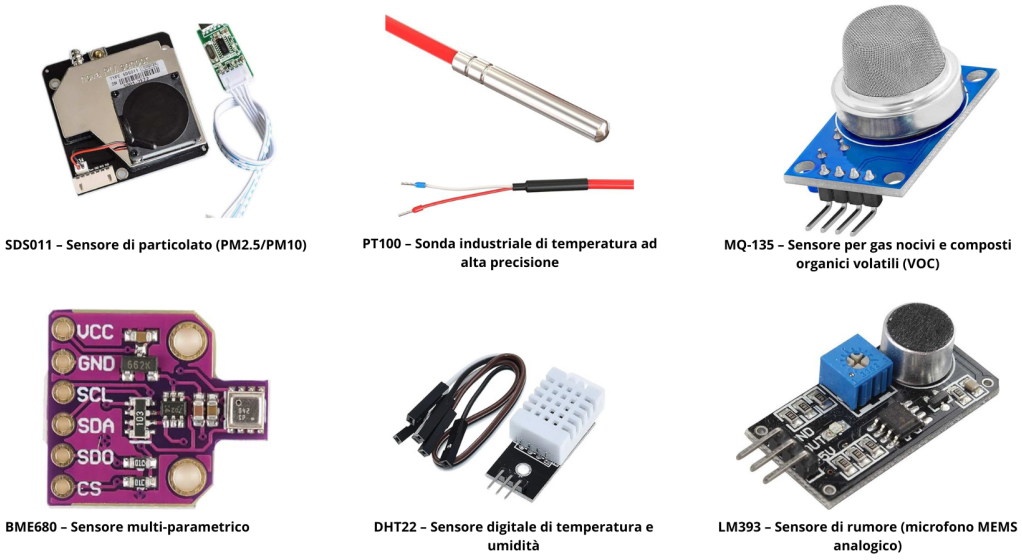

Figura 2.4: Esempi rappresentativi di sensori per il monitoraggio ambientale [4].

2.5.2 Attuatori

Accanto ai sensori, i sistemi di monitoraggio includono anche attuatori, come ventole intelligenti, relè per dispositivi elettrici e sistemi automatici di raffreddamento. Questi componenti trasformano l'IoT da semplice strumento di osservazione a un vero e proprio sistema di controllo attivo, capace non solo di segnalare condizioni critiche ma anche di intervenire tempestivamente per prevenirle, migliorando così efficienza operativa e sicurezza complessiva [11].

2.6 Sistemi e soluzioni sul mercato

Il mercato del monitoraggio ambientale per infrastrutture IT e sale server offre oggi una gamma molto ampia di soluzioni, che vanno dai sistemi enterprise certificati fino alle piattaforme open source e ai dispositivi low-cost a scopo prototipale. La scelta

dipende principalmente dal livello di affidabilità richiesto, dal budget disponibile e dalla complessità dell'infrastruttura da monitorare.

Soluzioni enterprise

Le soluzioni di fascia enterprise comprendono sistemi come AKCP, NetBotz (Schneider Electric), Sensaphone e HW Group, progettati per garantire elevata affidabilità, supporto tecnico dedicato e certificazioni di sicurezza [15]. Queste piattaforme includono sensori proprietari, interfacce grafiche avanzate, integrazione con protocolli industriali e funzionalità di allerta in tempo reale. Sono pensate per grandi data center e ambienti mission-critical, dove downtime o guasti possono comportare conseguenze economiche rilevanti. Il principale limite è rappresentato dal costo elevato e dalla rigidità delle soluzioni, spesso eccessive per realtà di dimensioni ridotte.

Soluzioni open source

Il panorama open source mette a disposizione strumenti come Zabbix, Grafana, Nagios e Prometheus, che permettono di realizzare sistemi di monitoraggio personalizzabili e scalabili, capaci di integrare dati provenienti da fonti eterogenee [21]. I principali vantaggi sono la flessibilità, l'indipendenza da vendor specifici e l'assenza di costi di licenza. In contropartita, queste soluzioni richiedono competenze tecniche per installazione, configurazione e manutenzione, oltre alla disponibilità di hardware dedicato.

Soluzioni low-cost e prototipali

In ambito di ricerca, didattica e sperimentazione si diffondono dispositivi a basso costo come ESP32, Arduino e Raspberry Pi, ideali per lo sviluppo di sistemi IoT prototipali [11]. Pur non raggiungendo i livelli di affidabilità delle soluzioni enterprise, offrono personalizzazione elevata, forte supporto comunitario e costi contenuti. La possibilità di integrarli con piattaforme open source (ad esempio Zabbix o Grafana) consente di realizzare soluzioni ibride, capaci di combinare semplicità e funzionalità avanzate.

In conclusione, la scelta della soluzione più appropriata dipende dal bilanciamento tra affidabilità, costi e competenze disponibili, variabili che determinano se privilegiare un approccio di tipo enterprise, una piattaforma open source o una soluzione a basso costo e a scopo prototipale.

Capitolo 3

Progettazione

La fase di progettazione è stata il momento in cui le esigenze espresse dai tecnici del Dipartimento di Informatica sono state tradotte in specifiche concrete. L'obiettivo principale era realizzare un sistema di monitoraggio ambientale affidabile, semplice da configurare e pienamente compatibile con l'infrastruttura già in uso. Oltre al rispetto delle specifiche minime richieste, sono state introdotte estensioni progettuali per aumentarne la versatilità e predisporre il sistema a futuri sviluppi. L'approccio seguito è stato di tipo incrementale, alternando implementazioni essenziali a miglioramenti mirati a robustezza, fruibilità e manutenzione.

3.1 Requisiti

La definizione dei requisiti ha rappresentato il punto di partenza della fase di progettazione. L'obiettivo principale era quello di realizzare un sistema che fosse facilmente integrabile con l'infrastruttura esistente, garantisse stabilità operativa e richiedesse soltanto una configurazione minima, limitata ai parametri di rete essenziali.

3.1.1 Requisiti funzionali

Dal punto di vista funzionale, il sistema deve essere in grado di acquisire in maniera periodica i valori di temperatura e umidità attraverso un sensore ambientale, così da fornire una base costante di dati per il monitoraggio. Tali valori devono essere esposti alla piattaforma di supervisione mediante passive checks, in modo da garantire la piena compatibilità sia con Zabbix 6 sia con Zabbix 7. È inoltre previsto che l'utente possa configurare i principali parametri di rete (indirizzo IP, subnet mask, gateway e DNS) in modo semplice e diretto. Infine, il sistema deve essere in grado di funzionare in maniera continuativa 24 ore su 24 e 7 giorni su 7, senza richiedere interventi manuali frequenti da parte del personale tecnico.

3.1.2 Requisiti non funzionali

Accanto ai requisiti operativi, sono stati definiti anche alcuni requisiti non funzionali, orientati a garantire qualità e affidabilità in un contesto critico come quello delle sale server. Il sistema deve infatti assicurare un'elevata stabilità operativa e la capacità di proseguire nel proprio funzionamento anche in caso di temporanea indisponibilità del server di monitoraggio. Particolare attenzione è stata riservata alla robustezza delle comunicazioni: per questo motivo è stata scelta la connessione Ethernet, preferita al Wi-Fi, così da ridurre il rischio di interruzioni. Allo stesso tempo, il sistema doveva essere semplice da usare, con procedure di configurazione ridotte al minimo e accessibili anche a personale non specializzato. Infine, la compatibilità con più versioni di Zabbix è stata ritenuta un aspetto essenziale, per garantire continuità operativa anche in presenza di aggiornamenti futuri della piattaforma.

3.2 Architettura del sistema

L'architettura del sistema è stata definita sulla base dei requisiti concordati con i tecnici del Dipartimento. L'obiettivo era realizzare una soluzione semplice, robusta e facilmente integrabile nella piattaforma di monitoraggio esistente. Tre principi hanno guidato tutte le scelte progettuali: affidabilità della comunicazione, continuità operativa e compatibilità con gli strumenti già in uso.

3.2.1 Schema generale

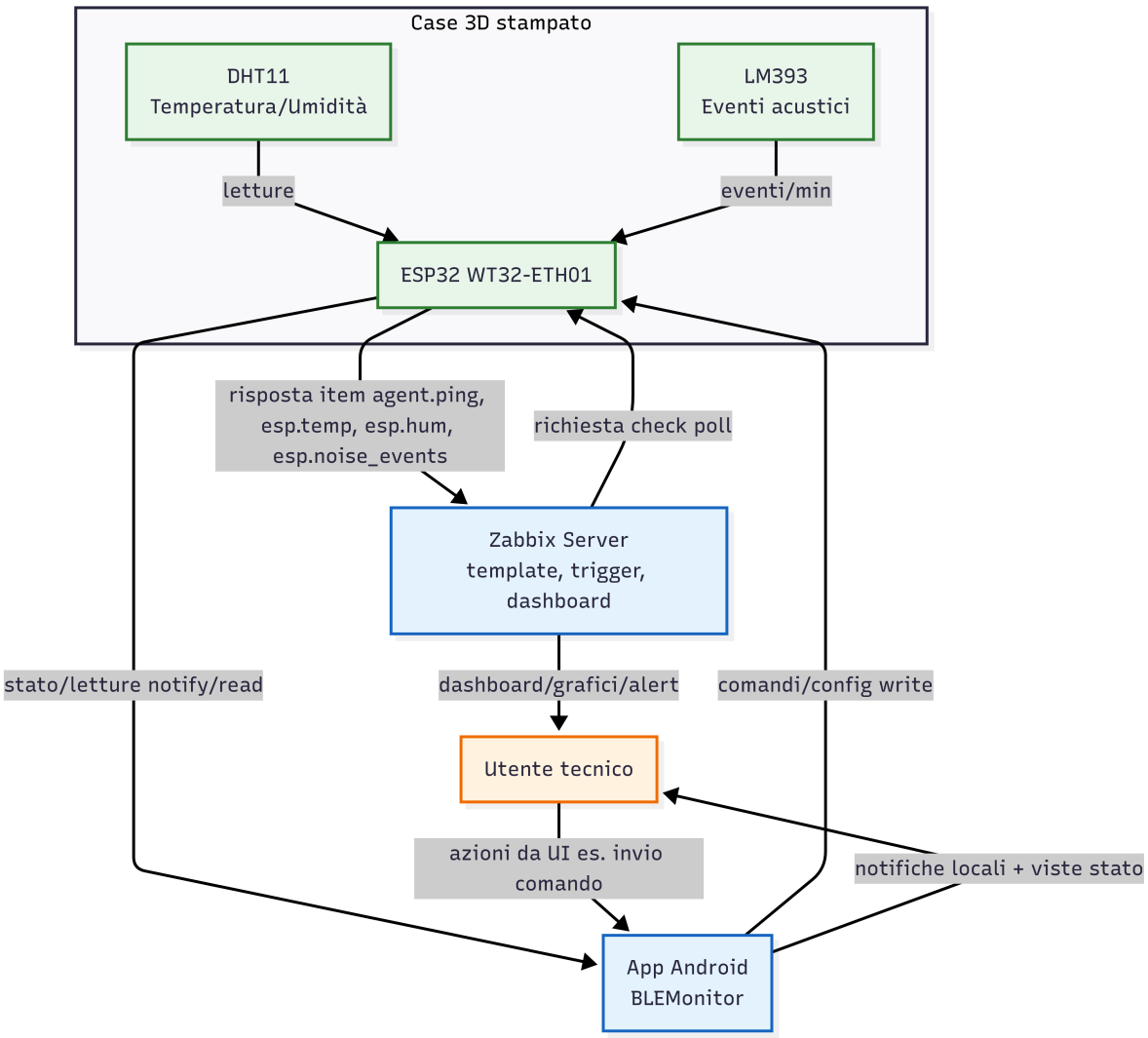

Figura 3.1: Architettura del sistema di monitoraggio ambientale

La Figura 3.1 illustra i flussi di comunicazione del sistema. L'utente tecnico interagisce sia con il server Zabbix, per consultare dashboard, grafici e ricevere alert, sia con l'applicazione mobile BLEMonitor, tramite la quale può inviare comandi e ricevere notifiche locali. L'app comunica via Bluetooth Low Energy con il microcontrollore ESP32 WT32-ETH01, integrato in un case stampato in 3D insieme ai sensori DHT11 (temperatura/umidità) e LM393 (eventi acustici). In parallelo, il server Zabbix interroga l'ESP32 tramite connessione Ethernet, ottenendo le misure ambientali per la supervisione centralizzata.

3.2.2 Piattaforma Zabbix

Zabbix è una soluzione open-source ampiamente adottata per il monitoraggio centralizzato di server, infrastrutture di rete e dispositivi IoT [17]. La sua architettura modulare e scalabile consente di raccogliere, elaborare, archiviare e visualizzare metriche provenienti da sorgenti eterogenee, risultando adatta sia a contesti di piccole dimensioni sia ad ambienti enterprise.

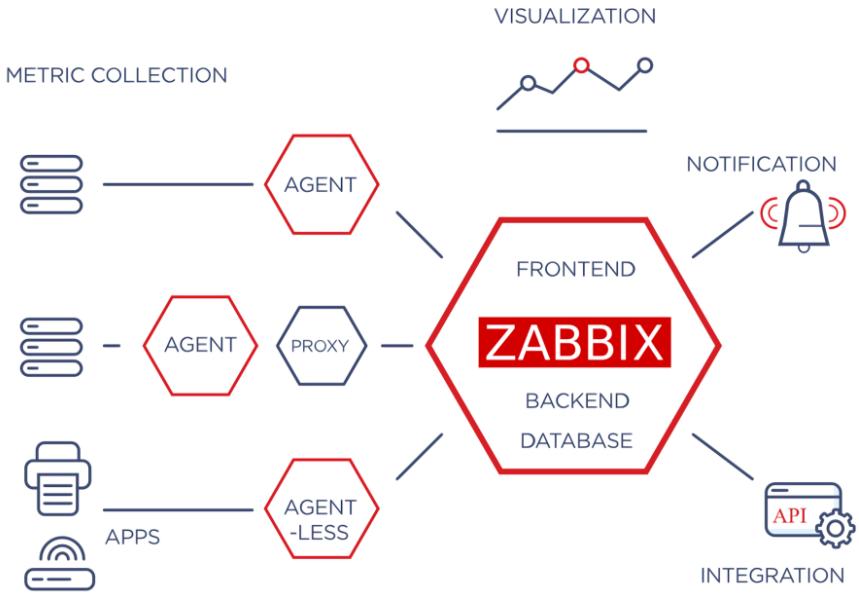

Figura 3.2: Architettura di Zabbix e dei suoi principali componenti [12]

L'architettura di Zabbix si articola attorno a quattro componenti principali. Il server rappresenta il cuore del sistema ed è responsabile dell'elaborazione e dell'archiviazione delle metriche; il database fornisce la persistenza e la storicizzazione dei dati raccolti; il frontend web mette a disposizione un'interfaccia grafica per la configurazione e la consultazione in tempo reale; infine, il proxy, opzionale, agisce come collettore intermedio nei contesti distribuiti o caratterizzati da connettività limitata.

Dal punto di vista della raccolta dei dati, Zabbix offre diversi approcci. Il più comune prevede l'utilizzo di un agent installato direttamente sul dispositivo da monitorare; in alternativa è possibile sfruttare un proxy che raccoglie i dati in maniera distribuita e li inoltra successivamente al server; in scenari più eterogenei, il monitoraggio agent-less permette invece di impiegare protocolli standard come SNMP, IPMI, SSH o HTTP.

La comunicazione tra server e agenti può essere configurata in due modalità differenti. Negli active checks, l'agente stabilisce una connessione verso il server (porta TCP 10051), ottiene la lista delle metriche da raccogliere e invia periodicamente i valori aggiornati. Nei passive checks, invece, il flusso è inverso: è il server che contatta l'agente (porta TCP 10050) e richiede in tempo reale i valori desiderati. Nel progetto sviluppato è stata adottata questa seconda modalità, che semplifica l'implementazione del dispositivo embedded: il server prende l'iniziativa e il microcontrollore si limita a rispondere, riducendo la complessità e aumentando la prevedibilità del comportamento.

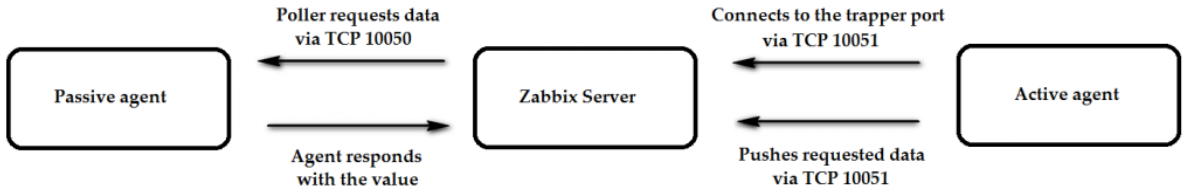

Figura 3.3: Differenza tra comunicazione mediante passive e active checks [9]

Un aspetto particolarmente rilevante di Zabbix è la possibilità di utilizzare template predefiniti. Questi consentono di definire in modo riutilizzabile item, trigger e grafici, riducendo i tempi di configurazione e minimizzando i rischi di errore. Per il progetto è stato predisposto un template dedicato al monitoraggio ambientale, comprendente item per temperatura, umidità ed eventi acustici, oltre a trigger configurati per generare notifiche automatiche al superamento delle soglie operative. Dashboard e grafici specifici completano la configurazione, offrendo una visione chiara e immediata dello stato del sistema.

Dal punto di vista della scalabilità e della sicurezza, Zabbix è stato progettato per supportare scenari complessi. L'adozione di proxy consente di distribuire il carico e gestire reti articolate, mentre il database centralizzato assicura la storicizzazione e la coerenza delle informazioni. La protezione delle comunicazioni è garantita dal supporto a TLS. Nel contesto del progetto, la scelta di utilizzare la connessione cablata unitamente ai soli passive checks ha rappresentato un compromesso efficace tra semplicità, robustezza e sicurezza.

In definitiva, all'interno del sistema realizzato, Zabbix non si limita a raccogliere dati dall'ESP32, ma costituisce il nucleo centrale delle attività di supervisione. Grazie alle funzionalità di storicizzazione, visualizzazione e notifica, esso offre un supporto decisio-

nale al personale tecnico, favorendo interventi tempestivi e contribuendo alla resilienza complessiva dell'infrastruttura informatica.

3.2.3 Modulo ESP32 WT32-ETH01

Il microcontrollore costituisce un elemento cruciale del sistema, poiché da esso dipendono la stabilità delle comunicazioni, la capacità di elaborazione dei dati e la possibilità di integrare funzionalità aggiuntive. Il sistema si basa sulla famiglia ESP32, sviluppata da Espressif Systems[23], azienda leader nel settore IoT. Dal 2008 Espressif si è affermata grazie a dispositivi caratterizzati da basso costo, buone prestazioni e un forte supporto della comunità open-source. L'ESP32, evoluzione del noto ESP8266, è oggi una delle piattaforme embedded più diffuse, grazie alla combinazione di potenza di calcolo, efficienza energetica e connettività integrata.

Dal punto di vista architetturale, l'ESP32 integra un processore dual-core Tensilica Xtensa LX6 fino a 240 MHz, con memoria interna sufficiente a gestire applicazioni complesse senza ricorrere a componenti esterni. È inoltre presente un coprocessore a bassissimo consumo, dedicato all'esecuzione di attività in background a impatto energetico minimo, caratteristica che rende il dispositivo adatto ad applicazioni continuative.

Per quanto riguarda la connettività, il microcontrollore offre nativamente Wi-Fi e Bluetooth Low Energy (BLE), risultando versatile in numerosi scenari IoT. Nel contesto specifico, tuttavia, la necessità di garantire affidabilità nelle comunicazioni ha reso preferibile l'uso dell'interfaccia cablata Ethernet. Per questo motivo è stato adottato il modello ESP32 WT32-ETH01, dotato di connettore RJ45 con PHY dedicato, che conserva tutte le potenzialità della famiglia ESP32 ma con la stabilità aggiuntiva di una connessione wired.

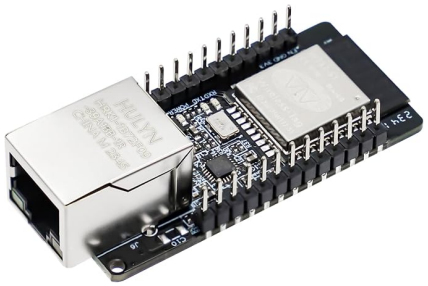

Figura 3.4: Modulo ESP32 WT32-ETH01 con supporto Ethernet e BLE [4]

Le caratteristiche principali del modulo possono essere sintetizzate come segue:

- integrazione del connettore RJ45 con PHY Ethernet, che semplifica il cablaggio e riduce la necessità di componenti esterni;

- mantenimento del supporto BLE, impiegato per configurazione e supervisione locale tramite app Android;
- disponibilità di numerosi pin GPIO, utili al collegamento diretto di sensori e attuatori;
- compatibilità con ambienti di sviluppo diversi, dall'Arduino IDE al framework ESP-IDF, supportati da un vasto ecosistema di librerie open-source.

Dal punto di vista energetico, il modulo implementa modalità di risparmio che consentono di bilanciare consumi ed esigenze operative, rendendolo idoneo a un funzionamento continuativo (24/7) con consumi contenuti. Un ulteriore punto di forza è il solido ecosistema software: l'SDK ufficiale ESP-IDF, insieme a driver e librerie costantemente aggiornati, semplifica l'implementazione di funzionalità avanzate come Ethernet, BLE e protocolli di monitoraggio (HTTP, MQTT e TCP). Il contributo della comunità open-source riduce i tempi di sviluppo e favorisce la standardizzazione delle soluzioni.

In sintesi, l'adozione dell'ESP32 WT32-ETH01 ha permesso di coniugare la robustezza della connessione Ethernet con la flessibilità della comunicazione BLE in un'unica piattaforma compatta, efficiente e affidabile. Queste caratteristiche lo rendono particolarmente adatto al monitoraggio ambientale delle sale server, dove affidabilità, continuità operativa e semplicità di gestione sono requisiti imprescindibili.

3.2.4 Sensori ambientali

Il sistema integra sensori semplici ed economici, selezionati per il buon compromesso tra costo, disponibilità e facilità di integrazione. Sono stati adottati due componenti principali: il DHT11 per la misura di temperatura e umidità, e il modulo LM393 per il rilevamento di eventi acustici.

Sensore DHT11

Il DHT11 (Figura 3.5) è un sensore digitale capace di misurare temperatura e umidità relativa. Il principio di funzionamento si basa su due meccanismi distinti: un termistore a semiconduttore per la temperatura, la cui resistenza varia in funzione delle condizioni ambientali, e un sensore capacitivo per l'umidità, costituito da due elettrodi separati da un materiale igroscopico che modifica la propria costante dielettrica in base al livello di umidità dell'aria.

Le variazioni fisiche vengono elaborate da un circuito integrato interno, che le converte in un segnale digitale trasmesso al microcontrollore mediante un'interfaccia seriale a filo singolo. Questo approccio semplifica l'integrazione, poiché il microcontrollore riceve direttamente valori numerici senza dover gestire segnali analogici grezzi.

Pur garantendo prestazioni più limitate rispetto a modelli avanzati (come DHT22 o SHT31), il DHT11 si è rivelato adeguato al prototipo per il basso costo, la buona affidabilità e la vasta disponibilità di librerie software.

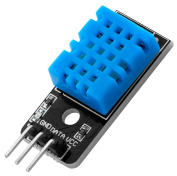

Figura 3.5: Sensore DHT11 per la misura di temperatura e umidità

Sensore LM393

Il LM393 (Figura 3.6) è un modulo basato su comparatore, ampiamente impiegato per il rilevamento di suoni o vibrazioni. È composto da due elementi principali: un microfono elettrete, che trasforma le variazioni di pressione sonora in un segnale elettrico proporzionale all'intensità del rumore, e un comparatore LM393, che confronta tale segnale con una soglia di riferimento regolabile tramite trimmer.

Quando il livello sonoro supera la soglia impostata, il comparatore produce un'uscita digitale leggibile dal microcontrollore come evento logico (0/1). Alcune versioni del modulo includono anche un'uscita analogica, utile per stimare più precisamente l'ampiezza del segnale.

Nel progetto, il sensore è stato utilizzato in modalità digitale, con lo scopo di rilevare e contare eventi acustici anomali (ad esempio rumori meccanici imprevisti) entro finestre temporali predefinite. In questo modo, il monitoraggio ambientale della sala server è stato arricchito con un parametro diagnostico aggiuntivo, complementare ai dati di temperatura e umidità.

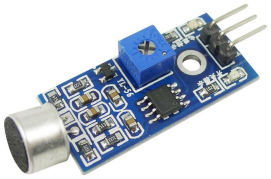

Figura 3.6: Sensore LM393 per il rilevamento di eventi acustici

3.2.5 Case 3D e alimentazione

Per garantire protezione, ordine e facilità di installazione, il dispositivo è stato racchiuso in un case stampato in 3D, progettato su misura per ospitare la scheda ESP32 e i sensori. L'adozione di un contenitore personalizzato consente di mantenere ordinato il cablaggio, assicurare un montaggio stabile e ridurre il rischio di danni accidentali ai componenti elettronici. Inoltre, la realizzazione additiva ha permesso di predisporre alloggiamenti specifici per ciascun elemento, rendendo il dispositivo più compatto, sicuro e facilmente replicabile. Un ulteriore vantaggio risiede nella semplificazione delle operazioni di manutenzione e validazione, poiché tutti i componenti risultano accessibili senza compromettere la protezione da polvere e urti esterni.

Dal punto di vista energetico, il sistema è alimentato tramite connettore USB-C, scelto in sostituzione del più datato micro-USB. Questa soluzione garantisce una maggiore robustezza meccanica, la reversibilità del connettore e la piena compatibilità con gli alimentatori moderni, contribuendo così alla praticità d'uso in contesti operativi reali. Il modulo di alimentazione integra inoltre una modalità TTL, che permette l'accesso diretto alla porta seriale del microcontrollore. Tale funzionalità si rivela particolarmente utile nelle fasi di sviluppo e debug, consentendo il monitoraggio in tempo reale dei log e la diagnosi immediata di eventuali anomalie, senza la necessità di interfacce aggiuntive.

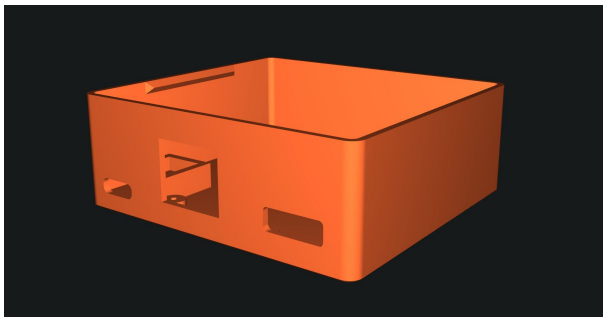

Figura 3.7: Progetto del case stampato in 3D con alloggiamento per ESP32 e sensori [20]

3.2.6 Applicazione Android e comunicazione BLE

Accanto al monitoraggio centralizzato fornito da Zabbix, è stata progettata un'applicazione Android in grado di comunicare con i dispositivi tramite Bluetooth Low Energy (BLE). L'obiettivo principale è supportare il personale tecnico, offrendo uno strumento rapido e intuitivo per configurare e monitorare i dispositivi direttamente in loco, senza dover accedere alla rete o al server centrale.

In particolare, l'applicazione consente di:

- **Configurare i parametri di rete** (indirizzo IP, subnet mask, gateway e DNS), così da rendere il dispositivo pronto alla connessione con Zabbix.
- **Supervisionare i sensori in tempo reale**, verificando il corretto funzionamento del sistema e ottenendo un riscontro immediato durante la manutenzione.
- **Controllare lo stato operativo**, interrogando il dispositivo per ricevere informazioni come l'uptime e lo stato delle connessioni Ethernet e BLE, utili per la diagnostica.
- **Gestire più dispositivi in parallelo**, associandoli a sale server differenti: l'app diventa così anche uno strumento organizzativo, capace di fornire una panoramica sintetica degli ambienti monitorati.

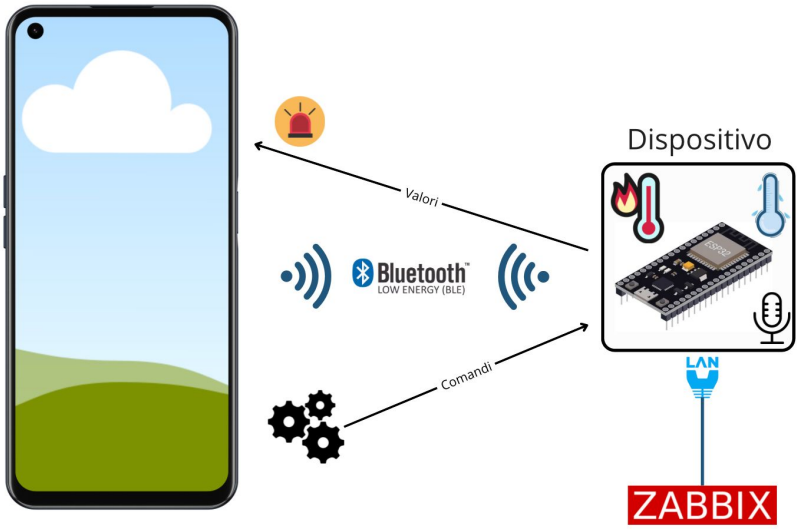

Figura 3.8: Prototipo concettuale dell'app Android per configurazione e monitoraggio

Capitolo 4

Implementazione

Il presente capitolo illustra come le scelte progettuali siano state tradotte in un sistema funzionante, mettendo in evidenza le tecnologie adottate e le modalità con cui ciascun componente è stato realizzato. L'obiettivo è mostrare la corrispondenza tra i requisiti individuati in fase di progettazione e le soluzioni effettivamente implementate.

Il capitolo è articolato in cinque parti principali: la descrizione dei collegamenti hardware e dello schema elettrico, l'implementazione del firmware per la scheda ESP32, l'integrazione con la piattaforma Zabbix, lo sviluppo dell'applicazione Android e la realizzazione del case 3D.

4.1 Dispositivo e schema elettrico

Per garantire la replicabilità del sistema, è stato predisposto uno schema elettrico che mostra i collegamenti tra la scheda ESP32 WT32-ETH01, i sensori ambientali, l'alimentazione e il modulo USB–C TTL utilizzato anche per la programmazione.

Il sensore DHT11 è stato connesso a un pin digitale della scheda, corredato da una resistenza di pull-up da 10 kΩ a 3.3 V al fine di stabilizzare il segnale dati. Il modulo LM393 è stato invece collegato a un secondo pin digitale, configurato come ingresso con interrupt per consentire la rilevazione puntuale degli eventi acustici. Entrambi i sensori condividono la stessa linea di alimentazione a 3.3 V e la massa (GND), sfruttando così un cablaggio semplice e ordinato.

L'alimentazione complessiva del sistema avviene attraverso un connettore USB-C, scelto per la sua robustezza meccanica e per la compatibilità con gli standard più recenti. Il modulo USB–C TTL fornisce i 5 V in ingresso, successivamente regolati a 3.3 V dalla scheda ESP32, e consente inoltre l'accesso diretto alla porta seriale per la programmazione e il debug in tempo reale.

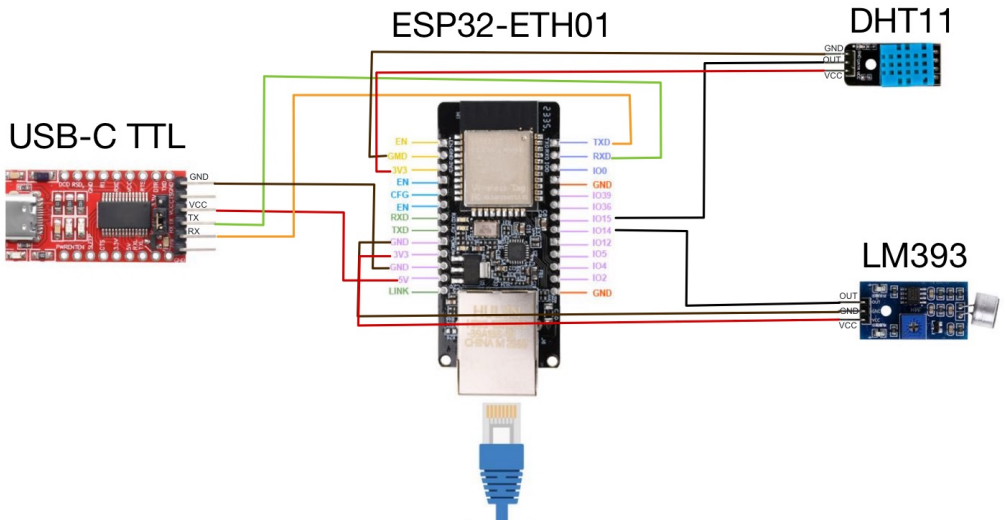

Figura 4.1: Schema elettrico del sistema: ESP32 WT32-ETH01, DHT11, LM393 e modulo USB–C TTL.

4.1.1 Programmazione della scheda

La WT32-ETH01 non dispone di una porta USB nativa né di un pulsante di reset dedicato. Per avviare la programmazione è quindi necessario accedere manualmente alla modalità bootloader, collegando il pin GPIO0 a massa e successivamente riavviando il dispositivo, ad esempio scollegando e ricollegando l'alimentazione.

Il collegamento con il modulo USB–C TTL richiede le seguenti connessioni principali:

- GPIO1 (TX) collegato al pin RX del modulo TTL
- GPIO3 (RX) collegato al pin TX del modulo TTL
- Pin 5V e GND collegati all'alimentazione fornita dal modulo TTL
- GPIO0 collegato a massa (solo durante la fase di caricamento del firmware)

Questa procedura consente di caricare correttamente il firmware tramite esptool o direttamente dall'IDE Arduino, permettendo lo sviluppo e il debug senza necessità di interfacce aggiuntive.

4.2 Implementazione del firmware ESP32

Lo sviluppo del firmware è stato condotto interamente all'interno dell'Arduino IDE [5], ambiente adatto alla prototipazione rapida grazie alla semplicità d'uso e all'ampio ecosistema di librerie. Il progetto è stato realizzato come unico file sorgente `.ino`, che integra la gestione dei sensori, la configurazione di rete, l'agente Zabbix e la comunicazione BLE.

4.2.1 Librerie utilizzate

Le principali librerie impiegate sono:

- `ETH.h`: connettività Ethernet nativa dell'ESP32.
- `WiFi.h`: inclusa per compatibilità con il core ESP32.
- `DHT.h`: gestione del sensore DHT11.
- `Preferences.h`: memorizzazione persistente (hostname, PIN, intervallo BLE, IP/-gateway/subnet/DNS).
- `BLEDevice.h`, `BLEServer.h`, `BLEUtils.h`, `BLE2902.h`: stack BLE, server, caratteristiche e descrittori.
- `ArduinoJson.h`: serializzazione/parsing JSON (supporto Zabbix 7).
- `vector`, `cstring`: utilità standard per gestione array e stringhe.

4.2.2 Struttura del firmware

Il firmware è stato sviluppato seguendo l'impostazione classica degli sketch Arduino, organizzati in due fasi principali: la funzione `setup()`, dedicata alle inizializzazioni, e la funzione `loop()`, responsabile delle attività cicliche e del funzionamento continuo del sistema.

Setup

Durante la fase di inizializzazione, il dispositivo carica innanzitutto le configurazioni persistenti tramite la libreria `Preferences`. In questo modo vengono recuperati parametri come hostname, PIN di sicurezza per l'accesso via BLE, intervallo delle notifiche e impostazioni di rete (IP, gateway, subnet e DNS). Nel caso in cui tali valori non siano presenti in memoria, vengono utilizzati parametri di default (`DEF_IP`, `DEF_GATEWAY`, ecc.), così da garantire comunque un avvio corretto.

Successivamente, vengono inizializzati i sensori. Il DHT11 è predisposto per la lettura periodica di temperatura e umidità, mentre il modulo LM393 viene configurato con un interrupt hardware, arricchito da una logica di debounce che evita conteggi errati dovuti a rumore di fondo.

Una volta attivi i sensori, si passa all'avvio della connettività Ethernet tramite le funzioni `ETH.begin()` e `ETH.config()`, che permettono di inizializzare l'interfaccia e applicare le impostazioni di rete configurate o di default. Parallelamente, viene attivato l'agente Zabbix: il firmware apre la porta TCP 10050 e si rende disponibile a gestire sia richieste in formato testuale (Zabbix 6), sia richieste incapsulate in JSON (Zabbix 7).

Infine, viene inizializzato lo stack BLE. In questa fase vengono definiti i servizi e le caratteristiche principali: notifiche per temperatura, umidità ed eventi acustici, oltre a una console di comandi con modalità write/notify. Al termine della configurazione, il modulo BLE entra in modalità advertising, rendendo il dispositivo visibile e pronto alla connessione da parte dell'applicazione Android.

Loop

La funzione `loop()` governa invece le attività cicliche. Ad ogni iterazione, il firmware legge i valori del DHT11 rispettando l'intervallo minimo richiesto dal sensore ed aggiorna il contatore degli eventi acustici, gestito tramite interrupt. In parallelo viene verificato lo stato del collegamento Ethernet con la funzione `ETH.linkUp()`; in caso di disconnessione, viene avviata una procedura di retry che reinizializza l'interfaccia e riapplica i parametri salvati.

Il ciclo operativo comprende anche la gestione delle richieste provenienti da Zabbix. Quando riceve una query, il dispositivo risponde restituendo i valori dei sensori nel formato previsto: una stringa numerica semplice per Zabbix 6 oppure un oggetto JSON incapsulato con header `ZBXD` per Zabbix 7.

Infine, la parte dedicata al BLE si occupa dell'invio periodico delle notifiche contenenti i valori di temperatura, umidità ed eventi acustici. La console di configurazione, anch'essa gestita in questa fase, è protetta da PIN ed utilizza un meccanismo di offuscamento XOR per i messaggi. Ogni comando valido ricevuto dall'utente viene elaborato e, se necessario, salvato in memoria, garantendo la persistenza delle configurazioni anche in caso di riavvio.

4.2.3 Gestione dei sensori

Il sistema utilizza due sensori principali: il DHT11 per la rilevazione ambientale e il modulo LM393 per il rilevamento di eventi acustici.

DHT11

Il DHT11 è un sensore digitale che fornisce due valori in virgola mobile: la temperatura in °C e l'umidità relativa in %. Poiché questo dispositivo ha un tempo minimo di campionamento di circa 2 secondi, le letture vengono effettuate rispettando tale intervallo, al fine di evitare valori errati. Qualora il sensore restituisca un valore non valido (`NaN`), la misura viene scartata, in modo da non introdurre rumore nei dati acquisiti.

Listing 4.1: Lettura da sensore DHT11

```
float temperaturaCorrente() {
  float t = dht.readTemperature();
  if (isnan(t)) return NAN;
  return t;
}

float umiditaCorrente() {
  float h = dht.readHumidity();
  if (isnan(h)) return NAN;
  return h;
}
```

LM393

Il modulo LM393 è utilizzato come rilevatore di rumori ambientali sopra soglia. Esso è collegato a un pin digitale dell'ESP32 configurato come interrupt hardware sul fronte di discesa (falling edge). Ogni volta che il sensore rileva un evento, viene eseguita una routine ISR che aggiorna un contatore dedicato.

Per evitare conteggi multipli indesiderati dovuti a rimbalzi elettrici o microvariazioni del segnale, è stato introdotto un meccanismo di debounce software. Il debounce, in questo contesto, consiste nell'ignorare eventi che si verificano troppo ravvicinati nel tempo

(al di sotto di una soglia minima), così da considerare valido un solo impulso per evento reale.

Listing 4.2: Interrupt per eventi acustici LM393 con debounce

```
volatile unsigned long lastNoiseEvent = 0;
volatile unsigned long noiseCounter = 0;

void IRAM_ATTR ISR_noise() {
  unsigned long now = millis();
  if (now - lastNoiseEvent > DEBOUNCE_MS) {
    noiseCounter++;
    lastNoiseEvent = now;
  }
}
```

Integrazione sensori con Zabbix e BLE

I dati acquisiti dai sensori vengono poi resi disponibili su entrambi i canali di comunicazione. Verso Zabbix, il contatore di eventi acustici è esposto come metrica differenziale, ossia viene calcolata la variazione rispetto alla lettura precedente. Sul canale BLE, invece, viene notificato direttamente il numero di eventi registrati nell'ultimo intervallo di invio. In questo modo, entrambe le piattaforme (locale e remota) dispongono di dati consistenti e adatti al proprio contesto di utilizzo.

4.2.4 Gestione della rete Ethernet

La configurazione della rete è un aspetto fondamentale per garantire la comunicazione stabile del dispositivo. I parametri principali (indirizzo IP, gateway, subnet e DNS) vengono salvati in memoria non volatile (Preferences) e caricati all'avvio del firmware. Durante l'inizializzazione, tali valori sono applicati tramite la funzione `ETH.config()`, che permette di configurare l'interfaccia Ethernet in maniera statica, evitando così la dipendenza dal DHCP.

Listing 4.3: Configurazione Ethernet all'avvio

```
IPAddress ip, gateway, subnet, dns;
ip.fromString(ipStr);
gateway.fromString(gatewayStr);
subnet.fromString(subnetStr);
dns.fromString(dnsStr);

// Avvio interfaccia Ethernet
```

```
ETH.begin();
ETH.config(ip, gateway, subnet, dns);
```

Una volta avviata, l'interfaccia di rete viene continuamente monitorata durante l'esecuzione del programma. Il firmware utilizza la funzione `ETH.linkUp()` per verificare lo stato del collegamento fisico. In caso di disconnessione, viene attivata una logica di retry che tenta la riconnessione dopo un intervallo prestabilito. La procedura prevede la ri-inizializzazione completa dell'interfaccia e la riapplicazione dei parametri salvati, così da riportare il dispositivo a uno stato operativo senza intervento manuale.

Listing 4.4: Supervisione e riconnessione rete

```
if (!ETH.linkUp() && millis() - lastRetry > ETH_RETRY_MS) {
  Serial.println("Ethernet non connessa, tentativo di retry...");
  ETH.begin();
  ETH.config(ip, gateway, subnet, dns);
  lastRetry = millis();
}
```

Grazie a questo meccanismo, il sistema risulta resiliente a temporanei flap di rete o a riavvii dell'infrastruttura sottostante, garantendo la continuità della raccolta dati e l'integrazione con la piattaforma Zabbix.

4.2.5 Comunicazione con Zabbix

Il firmware implementa un agente passive checks sulla porta TCP 10050, compatibile sia con Zabbix 6 sia con Zabbix 7. Il protocollo viene riconosciuto automaticamente in base al formato della richiesta ricevuta.

Modalità di funzionamento

- **Zabbix 6 (plain)**: il server invia una richiesta testuale contenente una chiave per riga; la risposta è un valore numerico in chiaro (es. `23.5`) oppure `Unknown key` se la chiave non è supportata.
- **Zabbix 7 (JSON)**: la richiesta è incapsulata in un pacchetto con header `ZBXD 0x01` e payload JSON. La risposta restituisce un oggetto JSON con `response`, `host` e l'array `data`, che contiene coppie {`key`, `value`} o {`key`, `error`}.

Chiavi implementate

Le chiavi disponibili sono:

- `esp.temp` – temperatura in °C

- `esp.humidity` – umidità relativa in %
- `esp.noise_events` – eventi acustici (differenziale)
- `agent.ping` – controllo di raggiungibilità

Esempi di interazione

Zabbix 6 (plain):

```
// Richiesta (server -> dispositivo)
esp.temp

// Risposta (dispositivo -> server)
23.5
```

Zabbix 7 (JSON, contenuto del payload):

```
// Richiesta
{
  "request": "passive checks",
  "host": "ZBX-ESP32",
  "data": [
    { "key": "esp.temp" },
    { "key": "esp.humidity" },
    { "key": "esp.noise_events" }
  ]
}

// Risposta
{
  "response": "success",
  "host": "ZBX-ESP32",
  "data": [
    { "key": "esp.temp", "value": 23.5 },
    { "key": "esp.humidity", "value": 41.0 },
    { "key": "esp.noise_events", "value": 3.0  }
  ]
}
```

4.2.6 Bluetooth Low Energy (BLE)

Il modulo BLE ha una duplice funzione: da un lato invia notifiche periodiche contenenti i valori dei sensori, dall'altro offre una console di configurazione che permette la gestione remota dei parametri del dispositivo direttamente da smartphone.

Servizio e caratteristiche

Il firmware espone un servizio BLE identificato da un UUID personalizzato. Al suo interno sono definite quattro caratteristiche principali. Tre di esse (`Temp`, `Hum` e `Noise`) hanno la modalità notify e inviano rispettivamente i valori di temperatura, umidità e numero di eventi acustici registrati. I dati sono trasmessi in forma testuale semplice, ad esempio `"23.5"` o `"41.0"`. La quarta caratteristica, denominata `Serial`, supporta sia write sia notify e implementa un canale testuale bidirezionale che simula una porta seriale UART. Attraverso di essa vengono inviati comandi di configurazione e ricevute risposte dal firmware. Per maggiore sicurezza, i messaggi sono offuscati mediante un'operazione XOR.

Notifiche periodiche

Le notifiche dei sensori vengono inviate a intervalli regolari, configurabili dall'utente e memorizzati in Preferences (con valore di default pari a 10 secondi). Ogni notifica contiene un singolo valore numerico formattato con una cifra decimale, senza ricorrere a strutture JSON più complesse. Questo approccio riduce la dimensione dei pacchetti BLE, migliorando l'efficienza della trasmissione.

Listing 4.5: Invio di un valore numerico come notifica BLE

```
void notificaFloat(BLECharacteristic* ch, float v) {
  char buf[8];
  sprintf(buf, "%.1f", v); // Formatta con una cifra decimale
  ch->setValue(buf);
  ch->notify();
}
```

Console di configurazione

L'accesso alla console richiede un'autenticazione basata su PIN numerico a sei cifre, anch'esso salvato in Preferences. Il canale di comunicazione, oltre ad essere offuscato con XOR, è protetto da ulteriori meccanismi anti brute force: una finestra temporale limita la frequenza dei tentativi, mentre un backoff progressivo e la disconnessione forzata intervengono in caso di abusi.

I comandi disponibili coprono tutte le principali esigenze operative: è possibile aggiornare l'hostname del dispositivo (`set host <nome>`), configurare i parametri di rete (`set ip`, `set gateway`, `set subnet`, `set dns`), modificare il PIN di sicurezza (`set pin <123456>`), oppure cambiare l'intervallo di notifica BLE (`set interval <s>`). Sono inoltre presenti comandi diagnostici come `info` e `ip`, che restituiscono informazioni di stato e indirizzo corrente, insieme a comandi di controllo come `restart` e `stop`.

Listing 4.6: Gestione di un comando di configurazione

```
if (cmd.startsWith("set ip ")) {
  salvaParametroRete("ip", cmd.substring(7));
  inviaMessaggioCifrato("IP aggiornato");
}
```

Persistenza delle configurazioni

Tutti i parametri modificati tramite la console BLE vengono salvati in memoria non volatile, così da essere applicati in modo automatico al successivo riavvio del dispositivo o in caso di interruzione dell'alimentazione.

4.3 Implementazione del monitoraggio in Zabbix

L'integrazione del sistema con la piattaforma Zabbix è stata testata e validata su entrambe le versioni 6 e 7, con il server di produzione basato su Zabbix 6. Per facilitare la configurazione e garantire uniformità tra i diversi ambienti, è stato sviluppato un template dedicato contenente item, trigger e una dashboard di monitoraggio completa. Questo approccio consente di importare e replicare rapidamente la configurazione su più host, riducendo al minimo gli interventi manuali.

4.3.1 Item del template

Gli item definiti nel template corrispondono alle chiavi esposte dal firmware ESP32. In particolare, vengono raccolti i valori di temperatura, umidità relativa ed eventi acustici, oltre a un controllo di raggiungibilità del dispositivo. La temperatura (`esp.temp`) e l'umidità (`esp.humidity`) sono acquisite come valori numerici in virgola mobile, con intervallo di campionamento pari a 60 s. Gli eventi acustici (`esp.noise_events`) sono invece registrati come metrica differenziale, con frequenza di acquisizione di 60 s, espressa in eventi/minuto. Infine, l'item `agent.ping` fornisce un'indicazione costante sulla disponibilità dell'host, atteso con valore pari a 1.0.

4.3.2 Trigger

Sulla base di questi item sono stati implementati diversi trigger, che permettono di generare allarmi in caso di condizioni anomale. Per la temperatura, il sistema segnala un avviso (Warning) quando la media a 5 minuti supera i 27°C, mentre genera un allarme critico (Disaster) se la media a 10 minuti oltrepassa i 35°C. Analogamente, per l'umidità relativa, viene generato un Warning se il valore medio a 10 minuti scende sotto il 30%, mentre un allarme di livello High viene sollevato se lo stesso parametro supera il 65%. Infine, per garantire la continuità del monitoraggio, è stato definito un trigger di disponibilità: viene segnalato un Error quando non sono ricevuti dati per oltre 10 minuti o quando l'item `agent.ping` restituisce un valore diverso da 1 per più di 5 minuti consecutivi.

4.3.3 Dashboard

Il template include una dashboard che offre una vista immediata dello stato del sistema: un grafico combinato di temperatura e umidità, un grafico degli eventi acustici/min e un widget a semaforo per la disponibilità dell'host. Indicatori numerici mostrano i valori istantanei delle principali metriche. Una sezione dedicata ai trigger ne riporta lo stato in tempo reale: il colore verde indica condizioni regolari, mentre il cambiamento di colore segnala situazioni critiche che richiedono intervento.

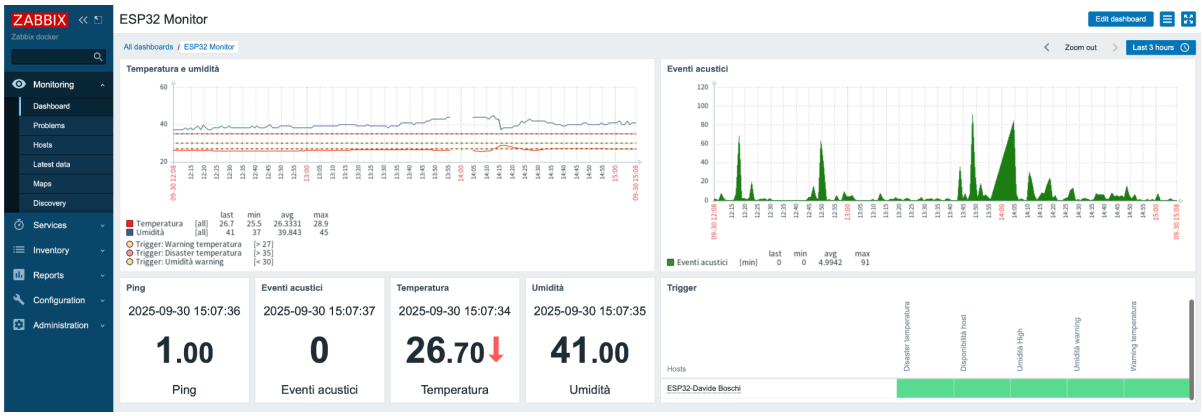

Figura 4.2: Dashboard Zabbix: valori correnti, grafici e stato dei trigger.

Grazie a questa configurazione, Zabbix non si limita a raccogliere dati ambientali, ma è in grado di generare allarmi tempestivi e di fornire un supporto proattivo al monitoraggio della sala server, garantendo affidabilità e continuità operativa.

4.4 Implementazione dell'app Android

L'applicazione mobile BLEMonitorApp costituisce il complemento software pensato per l'interazione locale con i dispositivi embedded basati su ESP32. È stata sviluppata interamente in Kotlin all'interno di Android Studio, adottando Jetpack Compose per la realizzazione dell'interfaccia utente. Questa scelta ha permesso di progettare una UI dichiarativa e modulare, capace di adattarsi in maniera dinamica allo stato applicativo. L'architettura segue il paradigma Model–View–ViewModel (MVVM), che garantisce una netta separazione tra logica di business, gestione dello stato e presentazione, migliorando la manutenibilità del codice e rendendo più semplice l'estensione futura del progetto. La persistenza locale dei dati è stata affidata a Room, la libreria ORM di Android che offre un accesso sicuro e reattivo al database SQLite integrato, semplificando così la gestione delle informazioni.

4.4.1 Funzionalità principali

L'applicazione è stata progettata con l'obiettivo di fornire al personale tecnico uno strumento semplice, immediato e al tempo stesso potente. All'avvio, l'utente può accedere a una schermata di scansione, che consente di rilevare in tempo reale i dispositivi presenti nelle vicinanze. Grazie a un filtro sul manufacturer data, vengono mostrati esclusivamente i nodi compatibili, contrassegnati dal prefisso `ZBX-`, riducendo così il rumore e facilitando l'individuazione dei dispositivi rilevanti.

Una volta selezionato il dispositivo, la connessione viene instaurata tramite protocollo GATT. Il processo è gestito automaticamente dall'app, che si occupa della discovery dei servizi e delle caratteristiche, offrendo una connessione stabile e sicura.

Oltre al monitoraggio, l'app mette a disposizione una vera e propria console di configurazione. Questa funzionalità, che emula il comportamento di una seriale UART, consente di inviare comandi testuali al dispositivo (ad esempio `set ip`, `set host` o `set pin`) e ricevere risposte immediate dal firmware. Per accedere a questa console è richiesto un PIN a sei cifre, che garantisce un livello base di autenticazione. La comunicazione dei comandi e delle risposte è resa più sicura dall'uso di una cifratura simmetrica di tipo XOR. Le configurazioni applicate non si perdono al riavvio, poiché vengono salvate direttamente sul microcontrollore.

Dal punto di vista della supervisione, i dati acquisiti dai sensori di temperatura, umidità e dal microfono ambientale vengono inviati al dispositivo mobile tramite notifiche BLE. Essi sono salvati automaticamente in locale e possono essere consultati in due modalità: come tabella cronologica, utile per analisi storiche, oppure attraverso grafici interattivi che si aggiornano in tempo reale. Nel momento in cui viene superata una soglia configurata, l'app genera un avviso locale, garantendo che il tecnico sia tempestivamente informato.

Un ulteriore livello di organizzazione è fornito dalla gestione multi-sala. Ogni dispositivo può infatti essere associato a un ambiente fisico, descritto da un nome, una breve descrizione e una posizione geografica. La visualizzazione avviene tramite Google Maps SDK, con marker che rappresentano i diversi ambienti monitorati. Grazie all'integrazione con le API di geofencing, quando l'utente entra all'interno di un'area registrata, l'app avvia automaticamente una scansione BLE e propone la connessione ai dispositivi presenti.

Infine, il sistema supporta la connessione simultanea a più dispositivi, ognuno gestito in maniera indipendente con la propria interfaccia di monitoraggio. Questa possibilità risulta particolarmente utile in scenari complessi, come il controllo di più sale server in parallelo, aumentando così l'efficienza operativa.

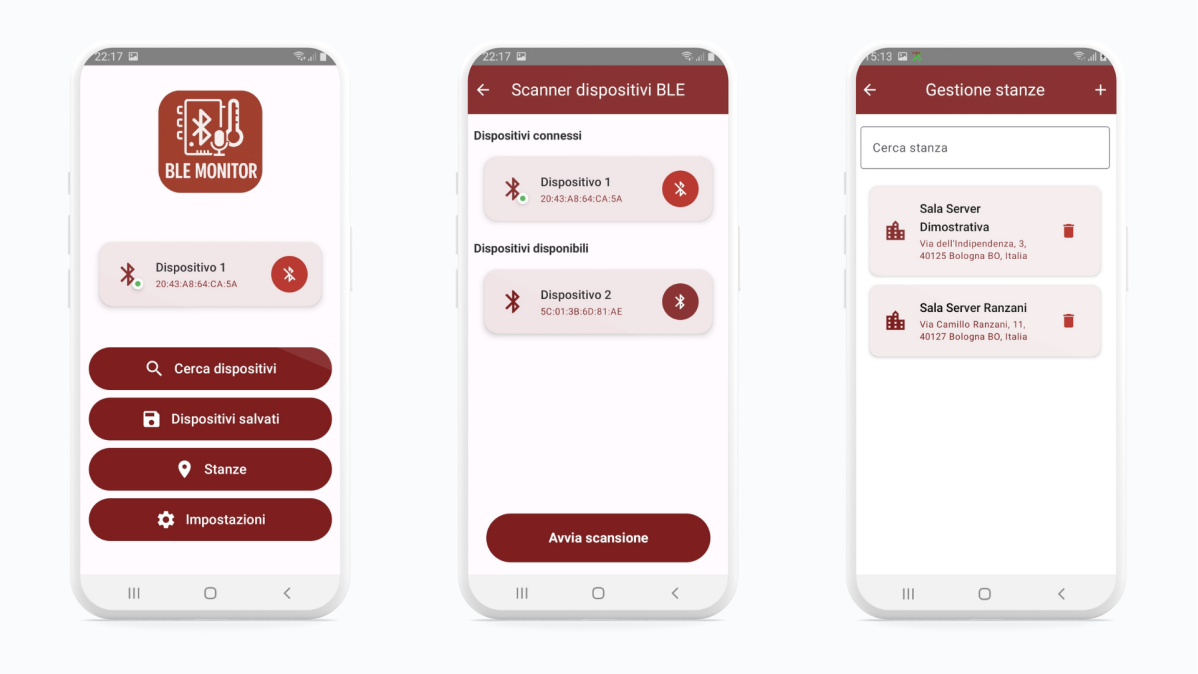

Figura 4.3: Schermate principali: homepage, scanner, sale salvate.

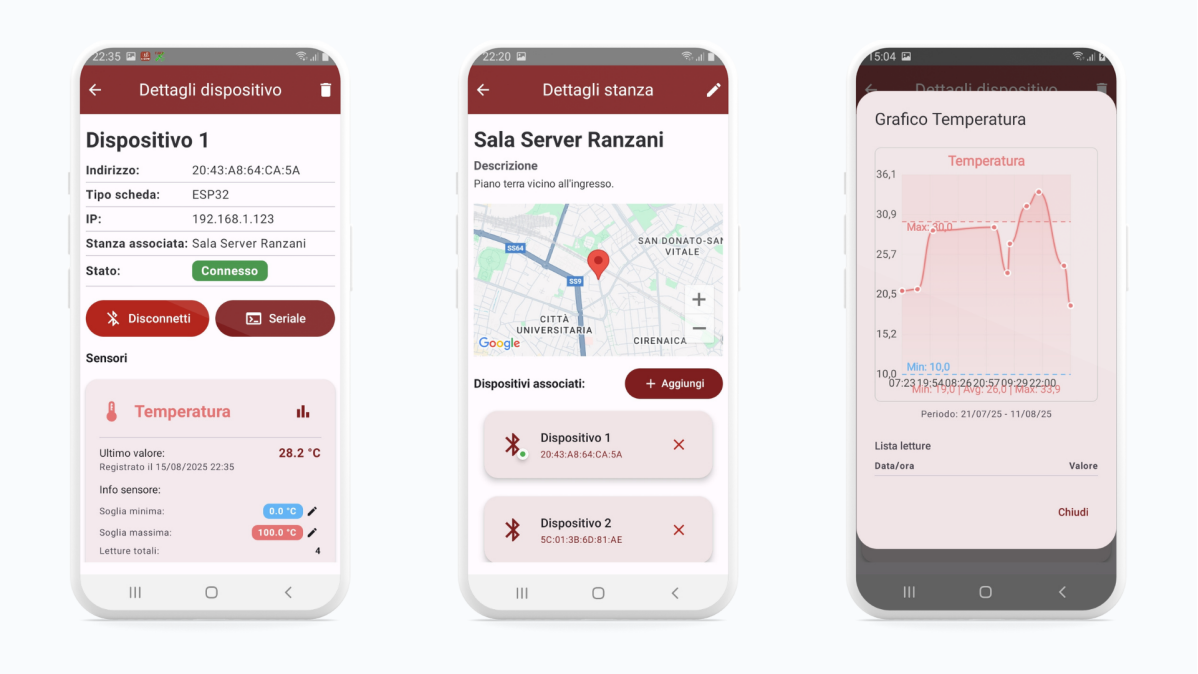

Figura 4.4: Schermate principali App: dettagli dispositivo, dettagli stanza, grafico temperatura.

4.4.2 Architettura software

L'architettura di BLEMonitorApp riflette la divisione in livelli tipica del pattern MVVM e si articola in più pacchetti, ognuno con responsabilità ben definite. Il livello `data/` gestisce la persistenza locale con Room, includendo entità, DAO e repository; il livello `domain/` contiene i modelli e le interfacce astratte dei repository, garantendo indipendenza dalla piattaforma; `presentation/` comprende le schermate Compose, i componenti grafici riutilizzabili, la navigazione centralizzata tramite `NavGraph` e i ViewModel che orchestrano logica e stato; `services/` si occupa invece della comunicazione BLE, gestendo scansione, connessioni GATT, ricezione dei dati e console seriale. Sono presenti inoltre i pacchetti `geofence/`, che cura la registrazione delle aree di interesse, e `utils/`, che raccoglie funzioni di supporto, ad esempio per notifiche e permessi.

Listing 4.7: Entità Room per dispositivi BLE

```
@Entity(tableName = "ble_devices")
data class BleDeviceEntity(
    @PrimaryKey(autoGenerate = true) val id: Long = 0,
    val address: String,
    val name: String?,
```

```
    val roomId: Long?,
    val boardType: String?,
    val ip: String?,
    val lastConnected: Long?
)
```

4.4.3 Gestione della comunicazione BLE

La comunicazione è gestita da un servizio centralizzato denominato `BleService`, che opera in foreground per rimanere attivo anche in background. Questo servizio coordina componenti modulari e riutilizzabili, come lo scanner, il connection manager e il data processor, garantendo affidabilità e flessibilità. Attraverso questa architettura, l'applicazione è in grado di rilevare dispositivi compatibili, instaurare connessioni GATT, scoprire i servizi disponibili, interpretare i dati ricevuti e salvarli nel database, mantenendo al contempo attiva una console seriale protetta da PIN. In aggiunta, il sistema supporta la gestione di più connessioni contemporanee, senza conflitti.

Listing 4.8: Avvio scansione BLE con filtro dispositivi compatibili

```
val settings = ScanSettings.Builder()
    .setScanMode(ScanSettings.SCAN_MODE_LOW_LATENCY)
    .build()
bleScanner.startScan(null, settings, scanCallback)

val boardType = extractBoardType(result)
if (boardType.isNullOrEmpty()) {
    Log.i(TAG, "Device $name ($address) not compatible")
    return
}
```

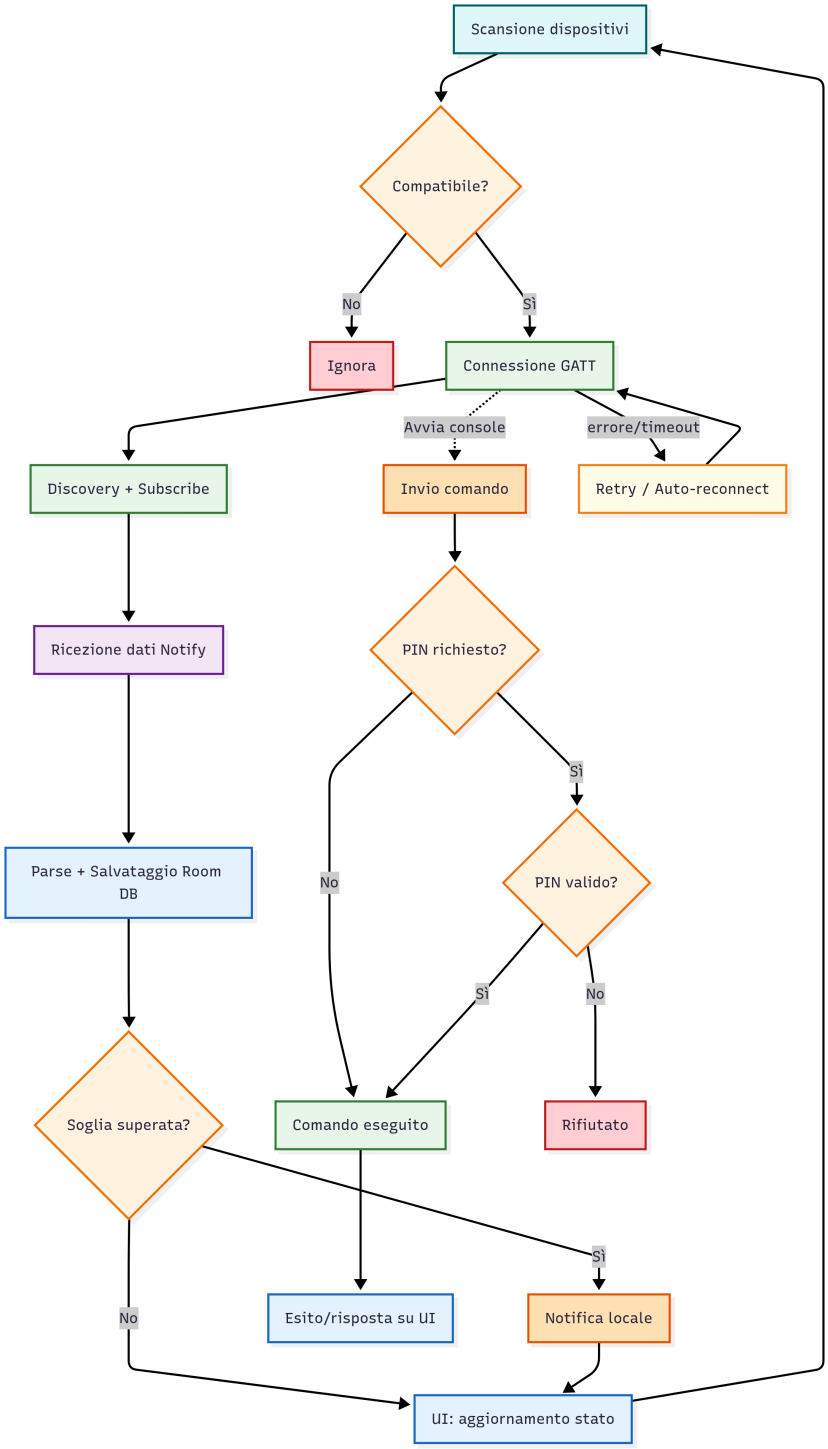

Figura 4.5: Flusso della comunicazione BLE.

4.4.4 Gestione delle sale e geofencing

La gestione degli ambienti è stata pensata per rendere l'applicazione più vicina alle esigenze pratiche di utilizzo. Ogni dispositivo può essere infatti associato a una sala fisica, caratterizzata da un nome e da una posizione geografica precisa. Le sale vengono mostrate su una mappa interattiva tramite Google Maps SDK, consentendo all'utente di avere una visione complessiva degli spazi monitorati. Il sistema integra inoltre le API di geofencing: quando l'utente entra nel raggio di una sala registrata, l'applicazione avvia in automatico una scansione BLE e notifica la presenza dei dispositivi disponibili, agevolando così il collegamento immediato.
Grazie a questo insieme di funzionalità, BLEMonitorApp non solo rende possibile la configurazione e l'interazione con i dispositivi anche in assenza di rete, ma consente anche di monitorare in tempo reale più ambienti in parallelo, offrendo al personale tecnico un supporto concreto per la gestione delle sale server e aumentando l'efficienza degli interventi.

4.5 Implementazione del case 3D

Il contenitore del dispositivo è stato realizzato a partire da un modello open–source in formato `.stl` [20], successivamente modificato per adattarsi alle esigenze specifiche del progetto. In una prima fase il case è stato stampato nella sua configurazione originale; tuttavia, i test iniziali hanno evidenziato una criticità legata alla scarsa circolazione dell'aria, che comprometteva l'accuratezza delle misure di temperatura e umidità fornite dal sensore DHT11. Per ovviare a questo problema è stata introdotta una modifica strutturale, consistente nell'aggiunta di un foro dedicato in prossimità del sensore, così da favorire un ricambio d'aria sufficiente e migliorare l'affidabilità delle rilevazioni ambientali.

La stampa del case è stata effettuata con una stampante BambuLab, utilizzando materiale PLA, scelto per il buon compromesso tra resistenza meccanica, stabilità dimensionale e semplicità di lavorazione. Una volta stampato, il contenitore è stato sottoposto a verniciatura, che oltre a migliorarne l'aspetto estetico ne ha incrementato la protezione superficiale contro graffi e usura. Per rafforzarne ulteriormente l'identità visiva, è stata realizzata un'incisione laser con una stampante Mecpow, riportando sulla parte superiore il logo dell'applicazione sviluppata: questo intervento ha conferito al prototipo un aspetto professionale, rendendolo al tempo stesso funzionale e riconoscibile.

Il risultato finale è un contenitore compatto, robusto ed esteticamente curato, in grado di ospitare in modo ordinato la scheda WT32-ETH01, i sensori ambientali e il modulo USB–C TTL. L'organizzazione interna garantisce sia la protezione dei componenti da urti, polvere e manipolazioni accidentali, sia la corretta esposizione dei sensori verso l'ambiente esterno, preservando così la precisione delle misure.

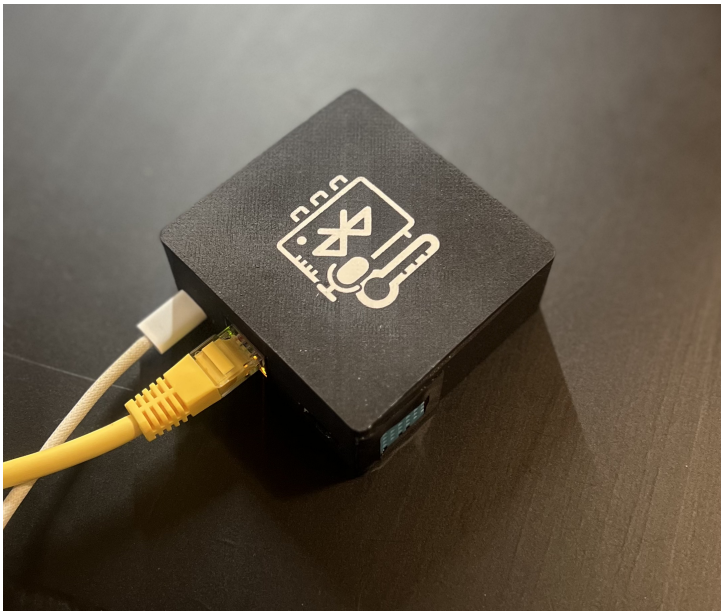

Figura 4.6: Case stampato in 3D in PLA, successivamente verniciato e inciso a laser con il logo del progetto.

Capitolo 5

Validazione

La validazione del sistema è stata condotta con l'obiettivo di dimostrarne l'efficacia, l'affidabilità e la stabilità in scenari realistici. Sono state analizzate tutte le componenti principali: il dispositivo embedded ESP32, la piattaforma di monitoraggio Zabbix e l'applicazione mobile BLEMonitorApp. Le prove hanno incluso sia test effettuati in un contesto sperimentale controllato, sia una validazione svolta direttamente presso la rete e i server del dipartimento, con il supporto del personale tecnico. Gli obiettivi principali erano verificare il corretto funzionamento tecnico, la continuità della connessione di rete e la capacità del sistema di generare notifiche tempestive, garantendo un'operatività priva di interruzioni indesiderate.

5.1 Metodologia

Le prove di validazione sono state articolate in più momenti e distribuite nell'arco di un'intera giornata. Il dispositivo è stato mantenuto acceso continuativamente per circa 24 ore in un ambiente domestico, così da verificarne la stabilità operativa e l'assenza di problemi prolungati. Durante questo periodo sono stati condotti test mirati: variazioni artificiali di temperatura, generazione di eventi acustici improvvisi e interruzioni della connettività Ethernet, per osservare la reazione del sistema a condizioni di stress controllate. Per la verifica della corretta integrazione con la piattaforma di monitoraggio è stato configurato un ambiente di test basato su Docker, contenente un'istanza Zabbix completa. In questo contesto il dispositivo ha inviato regolarmente i dati, consentendo di validare il funzionamento degli item e dei trigger in un ambiente isolato e facilmente replicabile. Infine, una sessione aggiuntiva è stata svolta presso la rete universitaria, con il supporto dei tecnici di dipartimento. Il dispositivo è stato collegato al server Zabbix istituzionale, dimostrando la piena compatibilità con infrastrutture reali e più complesse, oltre a confermare la semplicità di integrazione in contesti già operativi.

5.2 Connettività e disponibilità

La disponibilità del dispositivo è stata verificata tramite l'item `agent.ping`. Scollegando il cavo Ethernet per alcuni minuti, Zabbix ha rilevato correttamente l'assenza di dati, attivando il trigger di disponibilità host. Il grafico ha evidenziato in maniera chiara l'interruzione e il successivo ripristino automatico: l'ESP32 si è riconnesso autonomamente senza richiedere alcun intervento manuale, confermando la robustezza della gestione di rete anche in scenari di disconnessioni ripetute.

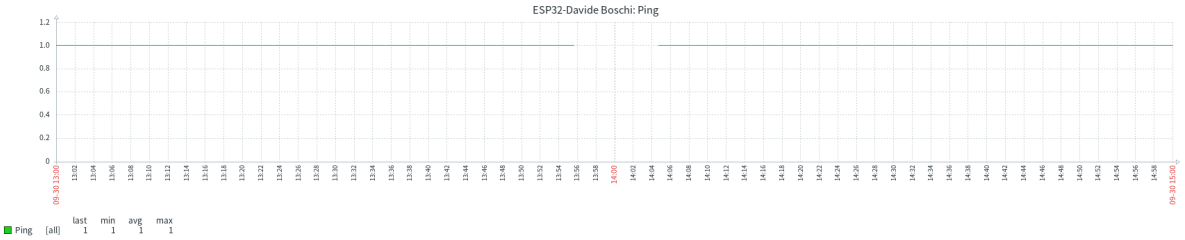

Figura 5.1: Grafico dell'item `agent.ping`: perdita e successivo ripristino della connettività.

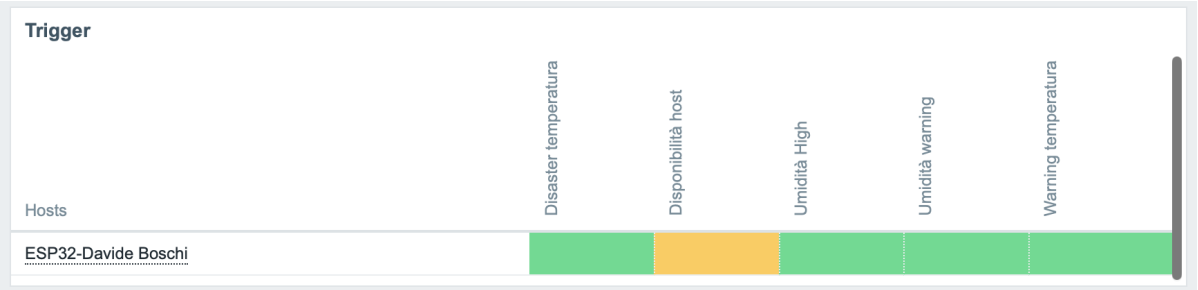

Figura 5.2: Dashboard dei trigger: attivazione dello stato PROBLEM in seguito a disconnessione di rete.

5.3 Monitoraggio della temperatura

Il sensore DHT11 è stato sottoposto a un riscaldamento rapido utilizzando una sorgente di calore localizzata (accendino tenuto a breve distanza). Questa sollecitazione ha provocato un aumento repentino della temperatura, che ha superato la soglia configurata di 27 °C, generando un allarme di tipo Warning. Il comportamento osservato ha confermato la corretta interazione tra dispositivo e piattaforma: il trigger è stato attivato tempestivamente e i grafici hanno mostrato l'andamento crescente fino al ritorno a valori stabili.

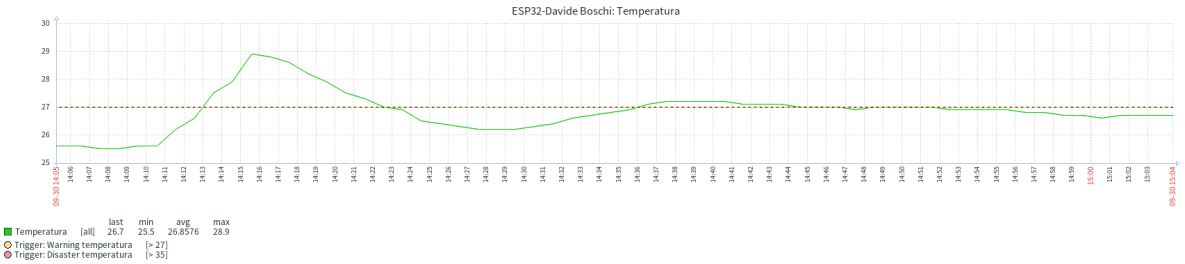

Figura 5.3: Andamento della temperatura e dell'umidità: evidenza del superamento della soglia di Warning.

Severity	Value	Name ▲
Disaster	OK	Disaster temperatura
Warning	OK	Disponibilità host
High	OK	Umidità High
Warning	OK	Umidità warning
Warning	PROBLEM	Warning temperatura

Figura 5.4: Attivazione del trigger di Warning per la temperatura al superamento della soglia.

5.4 Rilevamento eventi acustici

Il modulo LM393 è stato testato generando rumori improvvisi, quali battiti di mani e musica. Il sistema ha registrato incrementi netti nel numero di eventi al minuto, con picchi superiori a 100 al minuto, chiaramente visibili nei grafici. La presenza di una linea basale prossima allo zero, interrotta da valori elevati in corrispondenza dei rumori, ha confermato la sensibilità del sensore e l'assenza di falsi positivi.

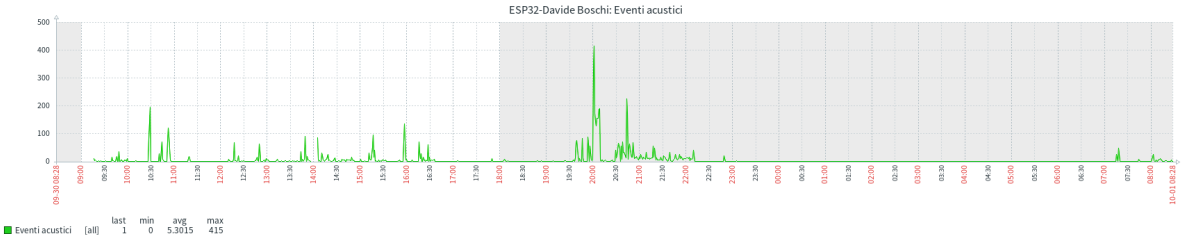

Figura 5.5: Andamento degli eventi acustici: picchi corrispondenti ai rumori prodotti durante i test.

5.5 Notifiche e interazione dall'app Android

In parallelo al monitoraggio centralizzato su Zabbix, l'app BLEMonitorApp ha fornito un canale di interazione diretto per l'utente. Durante i test sono state raccolte schermate significative che mostrano la connessione al dispositivo ESP32, la ricezione di notifiche locali al superamento delle soglie e l'utilizzo della console BLE, che riproduce il funzionamento di una chat seriale. L'app si è dimostrata stabile: connessioni e disconnessioni sono avvenute senza difficoltà, mentre le notifiche locali hanno garantito avvisi immediati in parallelo a quelli del sistema centralizzato.

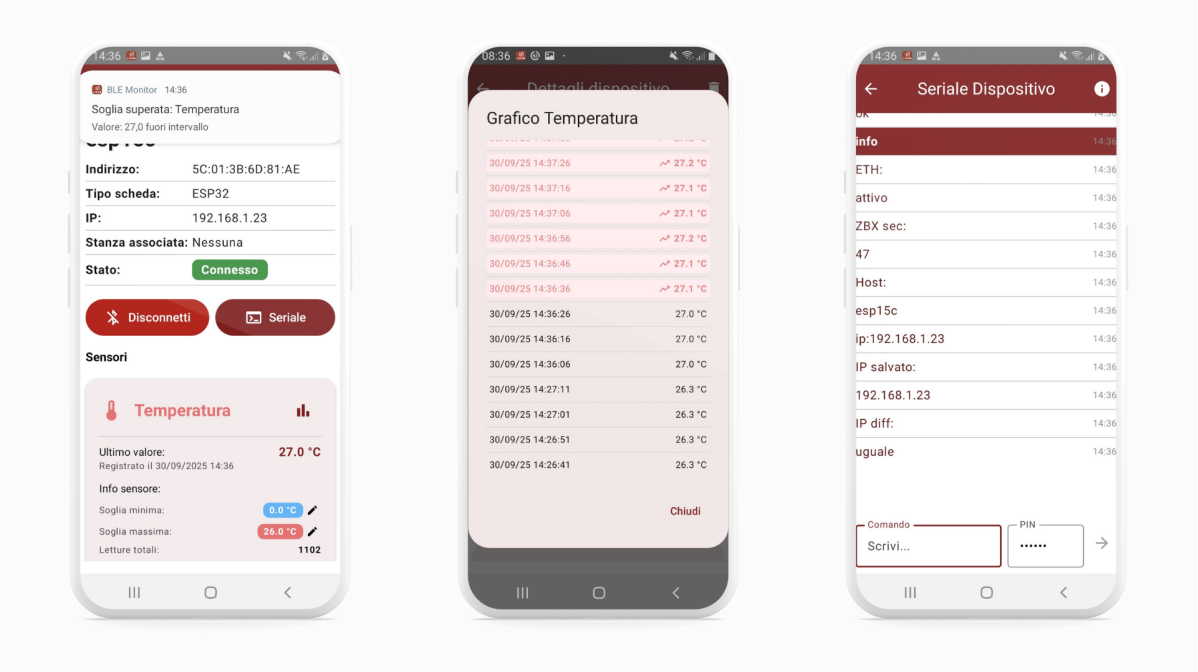

Figura 5.6: Schermate dell'app BLEMonitorApp: schermata dispositivo con notifica superamento soglia, storico dei dati, console BLE

5.6 Test di stabilità prolungato

Per validare ulteriormente la robustezza del sistema, il dispositivo è stato mantenuto in funzione continuativa per un'intera giornata all'interno di una stanza domestica. In questo scenario realistico, l'ESP32 ha garantito la continuità operativa senza interruzioni o malfunzionamenti. Zabbix ha registrato regolarmente i dati ambientali, senza falsi allarmi né irregolarità, mentre l'app BLEMonitorApp ha continuato a fornire in tempo reale la possibilità di consultare lo stato del dispositivo e ricevere notifiche.

L'esperimento ha dimostrato che il sistema è in grado di operare in modo stabile e senza problemi anche in condizioni d'uso quotidiane, senza la necessità di interventi manuali, confermandone l'idoneità per scenari reali e prolungati.

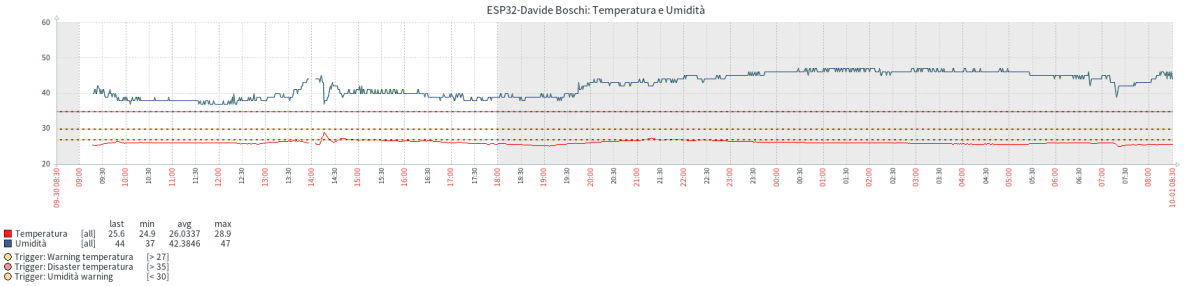

Figura 5.7: Grafico temperatura e umidità durante il test prolungato.

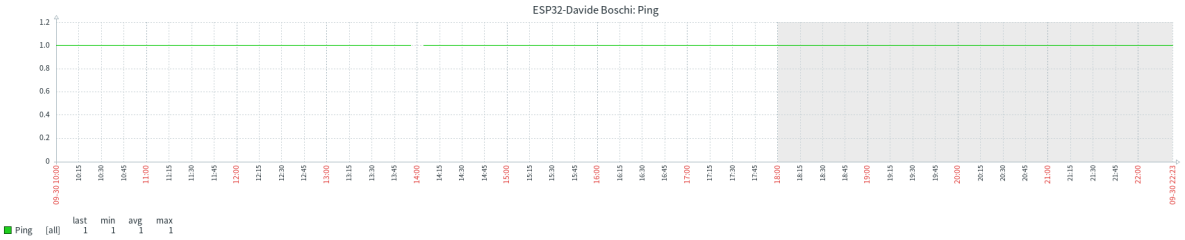

Figura 5.8: Grafico ping durante il test prolungato.

5.7 Risultati complessivi

L'insieme delle prove ha dimostrato che il sistema è pienamente funzionante sia dal punto di vista hardware che software. Il dispositivo ESP32 ha gestito in autonomia la riconnessone alla rete, mantenendo stabilità anche in seguito a interruzioni. Il sensore di temperatura ha permesso di rilevare variazioni critiche con precisione, mentre il modulo acustico ha mostrato elevata sensibilità a eventi sonori improvvisi. I valori raccolti si sono rivelati coerenti con le condizioni ambientali reali (26–29 °C per la temperatura, 37–45% per l'umidità e picchi acustici fino a 90 eventi/minuto). Infine, il test di durata ha confermato la stabilità complessiva del sistema in un contesto d'uso continuativo, rendendolo pronto per applicazioni concrete sia in ambito domestico che in infrastrutture complesse come quelle universitarie.

Capitolo 6

Conclusioni e sviluppi futuri

Il lavoro svolto ha portato alla realizzazione di una soluzione IoT embedded basata su microcontrollore ESP32, integrata con la piattaforma Zabbix e supportata da un'applicazione mobile Android. Il sistema sviluppato si è dimostrato capace di raccogliere e fornire in tempo reale informazioni sui principali parametri ambientali di una sala server, quali temperatura, umidità ed eventi acustici, garantendo al contempo affidabilità, semplicità di configurazione e possibilità di interazione locale tramite Bluetooth Low Energy (BLE).

Le attività di validazione hanno confermato la robustezza complessiva della soluzione: il dispositivo è risultato stabile anche in presenza di disconnessioni di rete, i sensori hanno fornito dati coerenti con le condizioni ambientali effettive e l'applicazione mobile ha offerto un'interfaccia efficace per la configurazione e il monitoraggio locale. La compatibilità con più versioni di Zabbix (6 e 7) ha inoltre dimostrato la flessibilità del sistema, rendendolo idoneo a essere integrato in contesti operativi eterogenei.

Nonostante i risultati raggiunti, il progetto presenta ampi margini di evoluzione. Un primo ambito di miglioramento riguarda l'estensione multipiattaforma: l'applicazione, attualmente sviluppata solo per Android, potrà essere affiancata da una versione per iOS, al fine di ampliare l'accessibilità e ridurre la dipendenza da un singolo ecosistema. Dal punto di vista della sensoristica, il prototipo potrà evolvere integrando sensori più avanzati, in grado di rilevare ulteriori parametri ambientali come la concentrazione di particolato, la presenza di gas nocivi o le vibrazioni meccaniche. Tali miglioramenti consentirebbero di arricchire le metriche disponibili e aumentare la qualità complessiva del monitoraggio.

Anche la componente acustica rappresenta un'area di sviluppo promettente: la sostituzione del modulo LM393 con un microfono digitale I^2S permetterebbe l'acquisizione di forme d'onda complete e l'analisi di spettrogrammi o pattern sonori, aprendo la strada a funzionalità di manutenzione predittiva. In prospettiva, l'elaborazione dei segnali acustici potrà essere ulteriormente approfondita attraverso l'impiego di tecniche di machine learning, con l'obiettivo di individuare correlazioni tra determinate firme sonore

e specifiche condizioni operative, come l'aumento del carico dei server, la presenza di ventole in accelerazione o altri comportamenti anomali. Tali analisi consentirebbero di implementare un sistema di rilevamento precoce delle anomalie basato su dati sonori, ampliando notevolmente le capacità diagnostiche della soluzione.

Parallelamente, l'introduzione di nuove chiavi in Zabbix (non più limitate a temperatura, umidità, rumore e ping) potrà estendere il monitoraggio anche a parametri di sistema, come uptime, utilizzo della memoria o stato dell'interfaccia Ethernet.

Dal punto di vista architetturale, un'evoluzione significativa potrà derivare dalla migrazione dell'attuale implementazione monolitica, sviluppata in ambiente Arduino IDE, verso una struttura basata su ESP-IDF e FreeRTOS. Ciò consentirebbe una gestione più modulare dei task dedicati ai sensori, alla rete e alla comunicazione BLE, migliorando manutenibilità, scalabilità e stabilità del firmware. L'adozione di librerie più evolute e l'integrazione con protocolli come MQTT o CoAP rappresenterebbero inoltre un ulteriore passo verso una soluzione professionale e industrialmente adottabile.

Anche l'hardware potrà essere oggetto di ottimizzazione. Il contenitore attuale, realizzato in stampa 3D a partire da un modello open–source, potrebbe essere sostituito da un case progettato ex novo in formato STL, ottimizzato per favorire i flussi d'aria, la disposizione dei sensori e la facilità di manutenzione. Questo permetterebbe di ottenere un dispositivo più robusto, funzionale e adatto a una produzione in piccola serie.

In conclusione, il progetto ha raggiunto pienamente gli obiettivi prefissati, offrendo un prototipo funzionante, affidabile e integrabile con infrastrutture reali. Le prospettive di sviluppo delineano un percorso concreto per l'evoluzione della soluzione verso un sistema maturo, scalabile e adatto a contesti produttivi complessi. Il passo successivo consisterà nell'ampliamento del numero di dispositivi installati e nella loro applicazione effettiva all'interno delle sale server dell'Università di Bologna, così da validare la soluzione in scenari reali e consolidarne ulteriormente l'affidabilità.

In prospettiva, il lavoro dimostra come l'integrazione di tecnologie IoT, piattaforme di monitoraggio e applicazioni mobili possa tradursi in strumenti realmente adottabili in ambienti critici, contribuendo al miglioramento dell'affidabilità, della sicurezza e della resilienza delle infrastrutture digitali. Tale risultato non rappresenta soltanto la conclusione di un percorso progettuale, ma anche il punto di partenza per future evoluzioni orientate alla professionalizzazione e all'impiego industriale del sistema.

Bibliografia

[1] F. Adelantado et al. «Understanding the Limits of LoRaWAN». In: *IEEE Communications Magazine* 55.9 (2017), pp. 34–40. DOI: 10.1109/MCOM.2017.1600613.

[2] M. Agiwal, A. Roy e N. Saxena. «Next Generation 5G Wireless Networks: A Comprehensive Survey». In: *IEEE Communications Surveys & Tutorials* 18.3 (2019), pp. 1617–1655. DOI: 10.1109/COMST.2016.2532458.

[3] A. Al-Fuqaha et al. «Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications». In: *IEEE Communications Surveys & Tutorials* 17.4 (2015), pp. 2347–2376. DOI: 10.1109/COMST.2015.2444095.

[4] Amazon. *Schede prodotto sensori elettronici (DHT22, MQ-135, PT100, SDS011, BME680, LM393)*. Ultimo accesso: Settembre 2025. 2025. URL: https://www.amazon.it.

[5] Arduino. *Arduino IDE*. Ultimo accesso: 29 settembre 2025. 2025. URL: https://www.arduino.cc/en/software/.

[6] arXiv. *Edge Computing for IoT*. Ultimo accesso: settembre 2025. 2024. URL: https://arxiv.org/html/2402.13056v1.

[7] ASHRAE TC 9.9. *Thermal Guidelines for Data Processing Environments*. Ultimo accesso: settembre 2025. 2021. URL: https://tpc.ashrae.org/?cmtKey=fd4a4ee6-96a3-4f61-8b85-43418dfa988d.

[8] Kevin Ashton. *The Internet of Things*. MIT Auto-ID Labs (pagina di riepilogo storica). Ultimo accesso: settembre 2025. 1999. URL: https://www.historyofinformation.com/detail.php?id=3411.

[9] Tencent Cloud. *Zabbix monitoring system architecture and principle analysis*. https://cloud.tencent.com/developer/article/1782273. Ultimo accesso: 28 settembre 2025. 2020.

[10] Com4. *Guida ai protocolli IoT*. Ultimo accesso: settembre 2025. 2023. URL: https://www.com4.no/it/blog/guida-protocolli-iot.

[11] Device Authority. *Understanding IoT Networks: A Beginner's Guide*. Ultimo accesso: settembre 2025. 2023. URL: https://deviceauthority.com/understanding-iot-networks-a-beginners-guide.

[12] Evonsys. *Effectively Using Zabbix Tool for Network Monitoring*. https://www.evonsys.com/blog/effectively-using-zabbix-tool-for-network-monitoring. Ultimo accesso: 28 settembre 2025. 2023.

[13] FinancesOnline. *Internet of Things (IoT) Market Statistics 2023–2030*. Ultimo accesso: settembre 2025. 2023. URL: https://financesonline.com/iot-statistics.

[14] Genetec. *L'importanza dell'IoT nella sicurezza fisica*. Ultimo accesso: settembre 2025. 2023. URL: https://www.genetec.com/it/blog/limportanza-delliot-nella-sicurezza-fisica.

[15] IBM. *What is the Internet of Things*. Ultimo accesso: settembre 2025. 2024. URL: https://www.ibm.com/think/topics/internet-of-things.

[16] IEEE Technology and Society. *Edge Computing and IoT Data Breaches: Security, Privacy, Trust and Regulation*. Ultimo accesso: settembre 2025. 2021. URL: https://technologyandsociety.org/edge-computing-and-iot-data-breaches-security-privacy-trust-and-regulation.

[17] Zabbix LLC. *Zabbix Documentation*. https://www.zabbix.com/documentation/current/en/manual. Ultimo accesso: settembre 2025. 2025.

[18] Math is in the Air. *Internet of Things illustration*. Ultimo accesso: settembre 2025. 2015. URL: https://www.mathisintheair.org/wp/wp-content/uploads/2015/01/IoT.png.

[19] Office of the Victorian Information Commissioner. *Internet of Things and Privacy – Issues and Challenges*. Ultimo accesso: settembre 2025. 2020. URL: https://ovic.vic.gov.au/privacy/resources-for-organisations/internet-of-things-and-privacy-issues-and-challenges.

[20] Printables. *RS485 ESP32 WT32-ETH01 USB Type-C Snap Fit Case*. Ultimo accesso il 28 settembre 2025. 2025. URL: https://www.printables.com/model/770951-rs485-esp32-wt32-eth01-usb-type-c-snap-fit-case.

[21] Emily Roberts. *Connectivity Redefined: Key Characteristics of IoT Technology*. Ultimo accesso: settembre 2025. 2023. URL: https://emilyroberts.hashnode.dev/connectivity-redefined-key-characteristics-of-iot-technology.

[22] SentiNet3. *Monitoraggio ambientale per data center*. Ultimo accesso: settembre 2025. 2025. URL: https://www.sentinet3.com/it/caratteristiche/datacenter-infrastructure-management/monitoraggio-ambientale.

[23] Espressif Systems. *ESP32 Series Datasheet*. https://www.espressif.com/en/products/socs/esp32. Ultimo accesso: settembre 2025. 2024.

[24] TD SYNNEX. *Il futuro di Internet of Things: dall'IIoT al Metaverso*. Ultimo accesso: settembre 2025. 2023. URL: https://blog.tdsynnex.it/il-futuro-di-internet-of-things-dalliot-al-metaverso.

[25] The Insight Partners. *Crescita del mercato IoT 2023–2030*. Ultimo accesso: settembre 2025. 2023. URL: https://www.theinsightpartners.com/reports/iot-market.

[26] J. Xu et al. «Narrowband Internet of Things: Evolutions, Technologies, and Open Issues». In: *IEEE Internet of Things Journal* 5.3 (2018), pp. 1449–1462. DOI: 10.1109/JIOT.2017.2786725.