Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

CO2 Monitor:

Progettazione e sviluppo di un sistema IoT per monitoraggio della CO2 e prevenzione Covid in ambienti indoor

Relatore:
Chiar.mo Prof.
Marco Di Felice

Presentata da:
Matteo Sacco

Sessione IV
Anno Accademico 2021/2022

Abstract

Rimasti segnati dalla pandemia globale scatenatasi all'inizio del 2020, medici e comunitá scientifiche di tutto il mondo hanno posto l'accento su tematiche giá da tempo conosciute, quali l'importanza della qualitá dell'aria che tutti i giorni respiriamo.

L'introduzione di misure anti contagio ha infatti spinto gli studiosi ad effettuare ricerche ulteriori circa la IAQ con lo scopo di prevenire i contagi all'interno di ambienti chiusi quotidianamente frequentati, quali scuole, uffici, ed abitazioni private.

Basandosi su tali premesse, la tesi ha investigato la progettazione e realizzazione prototipale di un dispositivo IoT per rilevare i livelli di CO2 nell'aria e inviare notifiche quando il valore supera la soglia di sicurezza. É stata sviluppata un'applicazione mobile connessa al sistema per monitorare le misurazioni in tempo reale e storicizzarle. Assieme a questi due componenti, sono stati implementati altri moduli software al fine di permettere un funzionamento agevole del sistema: un server di raccolta dati, uno strato di API e due database. Il sistema é stato validato considerando diversi scenari d'uso.

La realizzazione del dispositivo e dell'intero sistema software correlato si inserisce nel contesto dell'innovazione tecnologica per il monitoraggio della qualitá dell'aria in spazi indoor.

Indice

Elenco delle figure

Listings

Capitolo 1

Introduzione

Negli ultimi anni, il monitoraggio della qualitá dell'aria é diventato sempre piú importante, sia per ragioni ambientali sia sanitarie, in particolare, a seguito dell'avvento del virus COVID-19, la necessitá di controllare quotidianamente la IAQ (qualitá dell'aria interna) é diventata sempre piú prioritaria.

Tra i principali fattori che possono influire sulla qualitá dell'aria interna si trova la concentrazione di anidride carbonica (CO2) nell'ambiente, i cui livelli, se superiori a 1000 ppm, possono aumentare il rischio di trasmissione di virus e batteri in ambienti chiusi.

Per questo motivo, é stato scelto di sviluppare un dispositivo di rilevamento di CO2 e di implementare un sistema software ad esso correlato, composto da un server di raccolta dati, una API per la comunicazione con il database e un'applicazione per smartphone.

L'obiettivo principale di questa tesi é stato quello di realizzare un dispositivo IoT capace di rilevare i livelli di CO2 nell'aria e di collegarlo a un'applicazione mobile in grado di inviare notifiche quando il valore rilevato supera la soglia di sicurezza. Inoltre, l'applicazione consente di monitorare le misurazioni del sensore in tempo reale oppure con uno storico, fornendo cosí una panoramica sempre aggiornata dell'evoluzione del sistema.

Il sensore di CO2 costruito fa parte del mondo dell'Internet of Things (IoT), un ecosistema di dispositivi interconnessi che comunicano tra loro e con il mondo esterno. L'IoT sta rivoluzionando molti settori, tra cui appunto quello del monitoraggio, consentendo di raccogliere e analizzare dati in tempo reale per migliorare la salute e il benessere delle persone. La realizzazione di questo dispositivo di rilevamento di CO2 e del sistema software ad esso correlato si inserisce in questo contesto di innovazione tecnologica e di ricerca di soluzioni intelligenti per il monitoraggio della qualitá dell'aria interna.

La tesi presenta nei cinque capitoli successivi i diversi aspetti legati alla realizzazione dell'intero sistema di monitoraggio del livello di CO2.

Nel secondo capitolo "**Stato dell'arte**", viene mostrata una panoramica sul mondo dell'IoT e sullo stato attuale dei sensori di rilevamento di CO2, nonché una loro applicazione reale in ambiente scolastico. Nel capitolo terzo "**Progettazione**" vengono invece discusse le fasi progettuali del sistema suddiviso per componenti, e viene mostrato da quali necessitá sono derivate le scelte progettuali effettuate. Successivamente nel capitolo quarto "**Implementazione**", verrá riproposta la suddivisione e la struttura del capitolo precedente, discutendo peró le scelte implementative e le tecnologie utilizzate per ogni componente del sistema. Saranno in seguito, nel capitolo quinto "**Validazione**", mostrati i dati rilevati ed il processo di validazione effettuato, ovvero come é stato impiegato il dispositivo costruito al fine di raccogliere i dati. Infine, nell'ultimo capitolo "**Conclusioni e sviluppi futuri**", verrá spiegato il processo di validazione e saranno discussi gli sviluppi futuri per il progetto.

Capitolo 2

Stato dell'arte

2.1 Paradigma IoT

L'aumentare dei dispositivi connessi nel mondo e la necessitá del loro impiego nella quotidianitá sono alla base del paradigma dell'IoT. IoT significa infatti internet delle cose, ovvero una rete di dispositivi connessi tra loro che comunicano e condividono dati. I dispositivi facenti parte del mondo IoT ad oggi sono sempre di piú e in continua crescita, basti pensare agli oggetti di uso comune come televisori, elettrodomestici, smartphone, smartwatch, sensori e in generale i dispositivi che hanno la possibilitá di connettersi ad internet per comunicare dati.

2.1.1 Cos'é

L'Internet of Things (IoT) é un paradigma di computing che si basa sull'idea di estendere la connessione ad Internet a una vasta gamma di dispositivi e oggetti di uso quotidiano, rendendoli in grado di raccogliere e scambiare dati. Gli oggetti che fanno parte dell'IoT sono spesso chiamati "oggetti connessi" o "dispositivi intelligenti".[18][19][20]

Esistono oggetti connessi di ogni tipo, dai dispositivi indossabili come smartwatch e fitness tracker ai sistemi di automazione domestica come ter-

mostati e prese intelligenti, o ancora dispositivi di sicurezza, automobili o altro. La particolaritá di questi dispositivi é che tutti quanti raccolgono dati sull'ambiente circostante, su una particolare informazione o sull'utilizzo che ne viene fatto e li inviano a un server o a un'altra piattaforma per l'analisi e il trattamento degli stessi.

Un altro aspetto molto importante, e sempre piú utilizzato, dell'IoT nella vita di tutti i giorni é la possibilitá non solo di leggere i dati raccolti da questi dispositivi, ma anche di controllarli via rete. Gli esempi piú facili a cui pensare sono l'automatizzazione e la programmazione da remoto: si pensi ad un termostato intelligente che puó impostare automaticamente la temperatura in un edificio in base a una serie di parametri impostati, reagendo quindi ad eventi misurati da sensori che comunicano con esso.

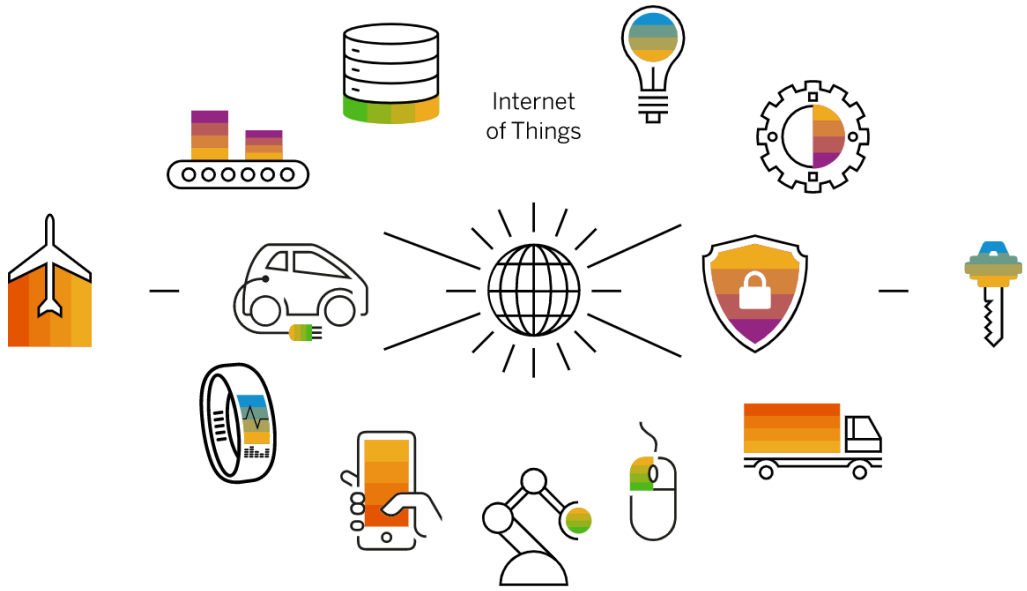

Figura 2.1: Paradigma IoT[18]

2.1.2 Dove viene impiegato

Questo paradigma oggi viene utilizzato, grazie anche ai modelli di intelligenza artificiale, nella maggior parte dei dispositivi embedded di impiego quotidiano: lavatrici, cellulari, termostati, auto, prese elettriche ...

Tutti i dispositivi facenti parte del mondo IoT, stanno diventando sempre piú efficienti ed accurati, in grado di catturare trend e preferenze ed adeguare il proprio comportamento al fine di migliorare aspetti della vita quotidiana o permettere un risparmio. Un esempio proprio di risparmio, forse il piú evidente, é quello energetico (e di conseguenza economico), ad esempio aria condizionata o riscaldamento che possono essere abbassati o addirittura spenti mentre in casa non ci sono persone. Un altro molto evidente e dal quale spesso la maggior parte delle persone é dipendente, consiste nell'analisi del traffico al fine di risparmiare tempo e carburante partendo all'ora giusta o facendo il percorso migliore: piú veloce o piú economico.

Tra le svariate applicazioni se ne possono trovare anche in ambito medico, ad esempio nel controllo dei contagi da COVID-19, come avvenuto durante la pandemia del 2020 con l'applicazione **Immuni**[17] sviluppata dalla "Presidenza del Consiglio dei Ministri". Grazie all'utilizzo di protocolli di comunicazione come Wi-Fi e Bluetooth, presenti all'interno dei telefoni cellulari, é stato possibile tracciare le interazioni fra diverse persone per capire qualora fossero lunghe abbastanza da permettere un contagio, e monitorare laddove un contagiato fosse entrato in contatto con persone sane.

Infine non puó non essere citato l'Industrial IoT (IIoT)[18][19][20], esso consiste nel settore che genera la maggior quantitá di dati al mondo ed é continuamente in forte crescita. La maggior parte dei dati IIoT proviene da videocamere di sorveglianza, automobili connesse ed applicazioni di produzione industriale e di trasporto. Questi dati sono utilizzati in tutti i settori, dalla gestione della supply chain alla sanitá (come visto con l'app Immuni).

Nell'industria manifatturiera e nelle catene di approvvigionamento, i sensori sono utilizzati per riconoscere e prevedere i problemi meccanici o per individuare i processi piú efficienti, i quali successivamente possono essere automatizzati. Nelle supply chain, le soluzioni IoT semplificano inoltre la maggior parte delle attivitá, dal tracciamento delle forniture alla logistica dei carichi e delle spedizioni, fornendo aggiornamenti in tempo reale agli utenti finali: basti pensare al servizio offerto da Amazon[16] per il tracciamento in tempo reale delle spedizioni.

Di seguito, verrá posta l'attenzione su applicazioni IoT relative al monitoraggio della qualitá dell'aria.

2.2 I sensori di rilevamento CO2

2.2.1 Come vengono usati

I sensori di rilevamento di CO2 sono utilizzati in diverse applicazioni per monitorare la concentrazione di anidride carbonica nell'aria. Posizionati in ambienti chiusi quali uffici, scuole e abitazioni, i sensori di CO2 possono aiutare a mantenere livelli ottimali di qualitá dell'aria, parametro molto importante per la salute e il benessere delle persone al loro interno.

Fattori di diverso tipo come la respirazione umana, la combustione di fonti di energia, l'utilizzo di apparecchiature elettriche e l'esposizione a prodotti chimici, influenzano infatti il livello di CO2 nell'aria all'interno di ambienti chiusi, in maniera inversamente proporzionale alle dimensioni del volume dell'ambiente (espresso in m^3). Una concentrazione troppo elevata di CO2, puó causare sintomi come mal di testa, stanchezza e difficoltá di concentrazione, e puó anche aumentare il rischio di malattie respiratorie. Pertanto, é importante mantenere livelli di CO2 nell'aria il piú bassi possibile, soprattutto negli ambienti chiusi dove le persone trascorrono la maggior parte del tempo durante il giorno.[3][21]

Uno degli impieghi dei sensori di CO2, messo in risalto dalla pandemia SARS-CoV2, é l'impiego per il controllo del traffico delle persone negli edifici: ad esempio per gestire il distanziamento sociale in tempo reale con informazioni sulla saturazione di gas nell'aria della stanza in esame. Da questo deriva un altro dei loro maggiori utilizzi, ovvero il controllo della qualitá dell'aria in ambienti chiusi, come uffici e scuole, per garantire che gli edifici siano sicuri e salubri per le persone.

L'aumentata concentrazione di CO2 nella stanza, infatti, puó portare piú facilmente alla trasmissione di virus e batteri, nello specifico secondo quanto citato dall'articolo "Covid-19 Guide on Ventilation and CO2 monitoring"[1], una concentrazione di gas superiore alle 1000 ppm rende l'ambiente non sicuro per le persone all'interno.

La concentrazione di CO2, come giá spiegato, puó dipendere da diversi fattori, ma tra tutti spicca l'aumento di monossido di carbonio (CO) e di vapore acqueo, perció una stanza chiusa con molte persone satura molto facilmente rispetto alla CO2.

A questo proposito, i sensori di CO2 possono essere collegati a sistemi di controllo dell'aria RTS (real time system) e con essi a dispositivi di allarme per avvisare le persone quando la concentrazione di CO2 raggiunge livelli pericolosi o non ottimali. Tra questi sistemi é possibile identificare anche dispositivi di domotica quali motori per apertura automatica delle finestre o aeratori da interno, utilizzati per permettere il ricircolo dell'aria e quindi l'ossigenazione della stessa, con conseguente calo della concentrazione di CO2.

Per permettere ció molti sensori sono dotati di connettivitá Bluetooth o Wi-Fi, consentendo quindi l'accesso ai dati di rilevamento anche tramite applicazioni per smartphone o assistenti vocali: questo consente di identificare e notificare tutte le persone presenti nella stanza dove é posizionato il sensore qualora il livello di CO2 nell'aria diventasse critico o ci fosse bisogno di un intervento manuale per il ricircolo dell'aria.

2.3 Come funzionano

Fino ad ora si é parlato di sensori di rilevamento di CO2 intendendo l'intero sistema che compie le azioni di: rilevare il livello di CO2 nell'aria, registrarlo, inviare le notifiche, far scattare allarmi, ecc... Ma ovviamente questa era solo una estrema semplificazione di tutto quello che serve al sistema per compiere tutte queste azioni. Da ora in avanti si parlerá infatti piú nello specifico di "dispositivi" o "sistemi" di rilevamento CO2, intendendo quindi l'assieme delle parti, ed i riferimenti al sensore (inteso come unitá che esegue le rilevazioni) verranno esplicitati.

2.3.1 Il dispositivo

I sensori di CO2 sono montati su microcontrollori, ovvero dispositivi che integrano un microprocessore, memoria e periferiche di I/O (input/output) in un unica scheda elettronica stampata (PCB), per permettere l'integrazione delle parti si rende infatti necessario anche progettare e stampare una PCB che permetta l'assemblaggio come board intermedia.

I microcontrollori sono in grado di eseguire programmi scritti attraverso un linguaggio di programmazione specifico, e di gestire l'acquisizione dei dati dai sensori di CO2 attraverso l'utilizzo di interfacce di comunicazione adeguate.

In quanto unitá centrale di tutto il dispositivo, il microcontrollore si occupa non solo della logica di registrazione dei dati, ma anche della trasmissione degli stessi ad altri dispositivi di controllo e monitoraggio, per essere poi elaborati e caricati in DB (database), inviati a dispositivi di segnalazione (allarmi, smartphone, ecc.). L'interfacciamento dei dispositivi di rilevamento di CO2 verso altri sistemi avviene tramite l'utilizzo di protocolli di comunicazione standard come Modbus, HTTP, IP, TCP, a seconda della struttura utilizzata.

2.3.2 Il sensore

La rilevazione del biossido di carbonio (CO2) puó essere fatta attraverso l'impiego di sensori di diverso tipo, i piú comuni possono essere classificati in quattro categorie in base alla tecnologia impiegata per effettuare la misurazione, ognuna con i propri vantaggi e svantaggi.

1. **Sensori ad infrarossi**[11][12]:
   Comunemente chiamati sensori NDIR (Non-Dispersive InfraRed), questi utilizzano l'emissione di luce infrarossa ad una specifica lunghezza d'onda della luce per misurare la quantitá di CO2 presente nell'aria. Il principio di funzionamento si basa sul fatto che ogni elemento sulla Terra assorbe solamente determinate lunghezze d'onda della luce, riflettendo tutte le altre.

   Nel caso del sensore NDIR, l'aria viene fatta passare attraverso la cella di rilevamento del sensore che al suo interno da un lato emette un fascio di luce a una determinata lunghezza d'onda, solitamente attorno ai quattro micron, e misura la quantitá di luce che arriva dall'altro lato della cella. La presenza di CO2 nell'aria campione fa sí che parte dei fotoni di luce vengano assorbiti, diminuendo quindi la quantitá di luce che raggiunge l'estremitá opposta della cella di rilevamento del sensore.

   Siccome la quantitá di luce assorbita é proporzionale alla quantitá di CO2 presente nell'aria, essa puó in questo modo essere misurata in base alla differenza tra la quantitá di luce emessa e quella rilevata.

   Pro:

   - Molto duraturi, con alcuni che durano piú di dieci anni
   - Altre sostanze non interferiranno con le letture
   - Funzionano bene con valori comuni di CO2 (intorno a circa 1000 ppm)

Contro:

- Potrebbero essere influenzati dall'umiditá e dalla temperatura a lungo termine

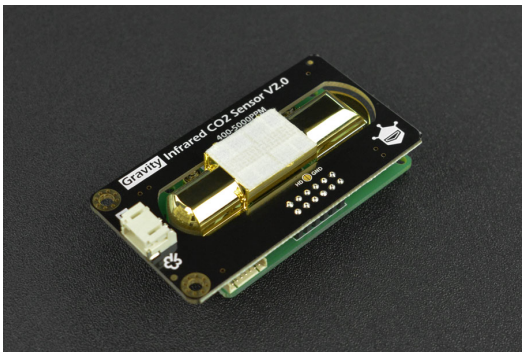

Figura 2.2: Sensore ad infrarossi[13]

2. **Sensori elettrochimici**[11][12]:
   I sensori elettrochimici rappresentano una tecnologia di rilevazione della concentrazione di anidride carbonica, che misura la corrente elettrica o la conducibilitá all'interno del sensore per determinare la quantitá di gas presente nell'aria.

   Tale tecnologia si basa sulla capacitá della CO2 di innescare una reazione chimica all'interno del sensore, determinando una variazione elettrica che, a seconda del tipo specifico di sensore, puó causare un aumento della corrente elettrica, una modifica di una corrente esistente o un cambiamento nella capacitá del sensore di condurre una corrente prestabilita con la quale é alimentato.

   Misurato il valore del cambiamento di capacitá o di tensione della corrente elettrica all'interno del sensore, questo verrá convertito in una misurazione del livello di CO2 poiché sará ad esso proporzionale.

   Pro:

- Meno suscettibili ai cambiamenti di umiditá e temperatura rispetto ai sensori NDIR o MOS

Contro:

- Altre sostanze possono influire sulla precisione delle letture
- Non durano tanto quanto i sensori NDIR
- Il sensore puó nel medio/lungo termine perdere precisione

Figura 2.3: Sensore elettrochimico[14]

3. **Sensori di capacitá**[11][12]:
   Chiamati sensori MOS (Metal Oxide Semiconductor), questi utilizzano la resistivitá di composti metallici per testare le quantitá di CO2 presenti nell'aria. La resistivitá puó essere riassunta come la facilitá con la quale l'elettricitá riesce a fluire attraverso un materiale. Ad esempio, il rame, spesso utilizzato nei circuiti elettrici, é meno resistente della gomma, che viene usata per bloccare i circuiti elettrici.

   Un sensore MOS ha una striscia o un film metallico che viene esposto all'aria facendo passare per questa una corrente elettrica costante attraverso la striscia stessa. Quando l'anidride carbonica entra in contatto con quest'ultima reagisce con il metallo e ne cambia la composizione chimica attraverso una reazione di riduzione o ossidazione. La reazione

scatenata altera quindi la resistivitá del metallo generando una variazione nella resistenza offerta al passaggio di corrente. L'entitá di questa variazione viene quindi misurata e ne viene valutato il rapporto con il livello di CO2 necessario a provocarla.

Pro:

- Il design molto semplice li rende facili da usare

Contro:

- Possono essere influenzati dalla temperatura e dall'umiditá
- Di solito vengono utilizzati a concentrazioni di CO2 piú elevate e meno comuni (maggiori 2000 ppm)
- Altre sostanze presenti nell'aria possono influire sulle letture

Figura 2.4: Sensore di capacitá

2.4 I dispositivi sul mercato

Per ognuna delle tipologie di sensore sopra descritte, in commercio oggi esistono differenti prodotti, caratterizzati dalla differente sensibilitá e qualitá costruttiva. Questo porta, naturalmente, ogni categoria di sensore ad avere un range di prezzo piuttosto ampio. Il singolo sensore potrebbe infatti costare dai 5€ a circa 300€ (IVA esclusa) nel caso di unitá per applicazioni industriali ad altissima precisione: é il caso di sensori INIR, ovvero sensori

Intergrated IR, che sfruttano una tecnologia IR avanzata.

Negli impieghi domestici o commerciali piú comuni, di fascia bassa fino a medio-alta, solitamente il prezzo delle singole unitá varia tra i 5€ e i 120€ circa: con i sensori IR tra i piú costosi, seguiti da quelli elettrochimici, di diffusione di massa ed infine dai sensori di capacitá.

Conseguentemente al prezzo del sensore che viene montato nel sistema, anche il dispositivo subisce un'oscillazione di prezzo in un range piuttosto ampio; si parla infatti di una variazione generale tra i 35€ circa ed i 500€. Ad influire sul prezzo, oltre all'unitá di misurazione montata, sono anche le caratteristiche del dispositivo stesso sia a livello fisico che funzionale. Infatti, tra i diversi sistemi é possibile trovare:

- Dispositivi portatili:
  Con un prezzo tra i 50€ e i 400€, esistono dispositivi per ogni applicazione e di conseguenza per ogni fascia di prezzo. Vengono usati per rilevazioni sul posto ed hanno la particolaritá di essere molto piccoli e maneggevoli, spesso collegati a delle applicazioni per smartphone.

Figura 2.5: Dispositivo di rilevamento CO2 portatile[23]

- Dispositivi da incasso:
  Con un prezzo tendenzialmente piú alto rispetto ai precedenti, che si aggira tra i 100€ e i 500€, vengono usati nelle abitazioni, negli uffici o

per applicazioni industriali, dove é necessario monitorare costantemente in tempo reale i livelli di CO2 nell'ambiente interessato. Hanno la particolaritá di leggere ed aggiornare dati 24/24h e spesso sono collegati a sistemi di allarme o notifica, inoltre quasi sempre utilizzano database per salvare i dati delle rilevazioni effettuate.

Figura 2.6: Dispositivo di rilevamento CO2 da incasso: Vemer Fiuto[24]

- Dispositivi da tavolo o da parete:
  Questi sono i dispositivi piú comuni per l'utente privato medio, ovvero quei dispositivi che generalmente si aggirano tra i 35€ e i 200€ (con picchi anche piú alti ma in rari casi), utilizzati in ambiente domestico. Nonostante non siano i piú precisi permettono di monitorare in tempo reale i livelli di CO2 mediante un'applicazione per smartphone ed eventualmente sfruttare l'Integrazione con allarmi o assistenti vocali per avvisare quando i livelli di CO2 sono troppo elevati.

Figura 2.7: Dispositivo di rilevamento CO2 da tavolo: Kaiterra Sensedge[22]

2.5 Progetto QAES

Il progetto QAES[3] (Qualitá dell'Aria negli Edifici Scolastici) si pone l'obiettivo di rendere le scuole ed i luoghi di lavoro sicuri contro l'aumento delle infezioni da virus COVID-19. É stato dimostrato che con semplici applicazioni dei dispositivi di rilevamento CO2 precedentemente descritti, é possibile intervenire per ridurre la probabilitá di contagio dovuta ad un elevato scambio di aerosol tra le persone che condividono una stanza poco ventilata per un lungo periodo.

Un semplice gesto, come aprire le finestre oppure lasciare libera la stanza per un lasso di tempo sufficiente a smaltire l'aria viziata accumulata, potrebbe essere sufficiente a permettere una condivisione sicura degli spazi comuni. Poiché la CO2 rappresenta il principale indicatore del livello di purezza dell'aria, é sufficiente misurarne il livello per capire quanto l'ambiente nel quale si sta effettuando la rilevazione sia saturo e, quindi, quanto questo ambiente vada lasciato arieggiare per tornare a livelli accettabili o ottimali di concentrazione di gas nell'aria.

La qualitá dell'aria influenza la salute e le performance dei bambini, che sono piú vulnerabili degli adulti poiché il loro sistema immunitario é ancora immaturo e quindi la stessa concentrazione di inquinanti che ad un adulto non darebbe problemi, risulta per loro piú dannosa. Non a caso, negli ultimi decenni, é stato registrato un aumento di asma bronchiale fra i bambini in etá scolare in tutte le scuole europee e, nello specifico in Italia l'asma e la rinite allergica risultano le patologie croniche piú diffuse nell'infanzia e nell'adolescenza.[3]

I risultati di studi condotti negli USA hanno evidenziato una correlazione fra esposizione a CO2, PM (particelle sospese) e muffe, e maggiori rischi di tosse secca notturna, rinite e tosse persistente. Inoltre, l'ipertensione nei bambini é stata associata a un'eccessiva esposizione alle PM, che possono addirittura causare uno squilibrio del sistema nervoso autonomo e disfunzioni del sistema vascolare arterioso tra cui vasocostrizione. I risultati del progetto

SEARCH (Sistema di Valutazione dell'Ambiente di Apprendimento e della Salute nelle Scuole) indicano che il sovraffollamento é a sua volta correlato a maggiori concentrazioni di PM e di CO2.

I fattori appena elencati concorrono a determinare il tasso di IAQ (qualitá dell'aria interna), legata alla bontá dell'aria all'interno e intorno agli edifici e alle strutture scolastiche. Comprendere e controllare gli inquinanti comuni negli ambienti chiusi puó aiutare a ridurre il rischio di problemi di salute, infatti secondo l'OMS (Organizzazione Mondiale della Sanitá), il 23% della mortalitá globale totale e il 26% dei decessi nei bambini piccoli sono causati da ambienti malsani, ovvero inquinati con fattori di rischio fisici, chimici o biologici esterni che possono influire sulla salute delle persone.

Gli effetti degli inquinanti dell'aria indoor possono essere immediati o manifestarsi anni dopo l'esposizione. Per determinarne l'impatto sulla salute dei bambini e degli insegnanti, oltre a misurare la concentrazione di agenti inquinanti, é necessario definire la durata dell'esposizione, ovvero "short-term" (inferiore a 24 ore) o "long-term" (di durata superiore a settimane).

Questo perché nell'aria sono presenti piú di 200 inquinanti quali monossido di carbonio, anidride carbonica, biossido e triossido di azoto, idrocarburi policiclici aromatici, composti organici volatili, allergeni, formaldeide, radon, contaminanti biologici e ozono. Secondo alcuni studi, le principali fonti di inquinamento indoor negli edifici scolastici sono il fumo di tabacco, il monossido di carbonio, i composti organici volatili, gli allergeni e le muffe.

Gli edifici scolastici sono caratterizzati da spazi chiusi (aule e palestre) in cui docenti e studenti trascorrono in media tra le sei e le otto ore al giorno. La qualitá dell'aria e la concentrazione di agenti inquinanti all'interno degli edifici scolastici dipendono da una serie di fattori complessi che interagiscono tra loro, come l'interazione tra l'edificio e l'ambiente esterno, le sorgenti interne di inquinanti, i sistemi di ventilazione, le attivitá degli occupanti e l'arredamento.

Capitolo 3

Progettazione

Il progetto di tesi consiste in un sistema di rilevamento di CO2 per applicazioni interne ad uso domestico, al fine di consentire agli utenti di monitorare una o piú stanze in base al numero di sensori presenti. Questi ultimi, collegati ad un server che invia i dati ad un database, permettono di avere persistenza delle informazioni registrate e quindi, attraverso una applicazione per smartphone, di essere monitorati in tempo reale ed inviare notifiche qualora i livelli di CO2 rilevati fossero troppo elevati.

Per la realizzazione del progetto sono stati sviluppati ed integrati diversi componenti, tra cui dispositivo fisico (composto da sensore, microcontrollore e PCB), applicazione per smartphone, bucket di un DB, server per la ricezione dei dati, API per la manipolazione e la lettura dei dati da parte dell'applicazione, ed un protocollo di comunicazione per il passaggio dei dati dal sensore al server di raccolta.

Nel capitolo attuale verranno discusse le diverse componenti nella loro fase progettuale e, verrá mostrato come esse comunicano tra loro.

3.1 Obiettivi

Come accennato sopra, il progetto ha lo scopo di garantire la sicurezza discussa nel capitolo precedente in contesti domestici attraverso il monito-

raggio della IAQ. Tra gli obiettivi principali era quindi presente la creazione di un sistema low-cost in grado di rilevare i cambiamenti del livello di CO2, che permettesse all'utente di monitorarli ed essere avvisato in caso di livelli critici. Inoltre si é reso fondamenale anche garantire un'esperienza "User friendly", ovvero fornire all'utente finale uno strumento molto semplice ed intuitivo da usare, nonché stabile e sicuro.

3.2 Requisiti

Nella fase progettuale dell'intero sistema si sono tenuti presenti i requisiti che ogni componente doveva avere. Si é quindi deciso di suddividere gli stessi in base al dispositivo che li avrebbe dovuti realizzare in seguito.

3.2.1 Requisiti del dispositivo

Il dispositivo, composto principalmente da tre componenti fisiche (microcontrollore, sensore e PCB), deve rispettare i seguenti requisiti:

- Essere compatto:
  le dimensioni del dispositivo, seppur non impediscano allo stesso di essere utilizzato, risultano importanti dal momento in cui deve essere posizionato in ambienti casalinghi

- Supportare connessioni wireless:
  per poter funzionare al meglio il dispositivo deve essere collocato in posizioni sopraelevate rispetto all'altezza di una persona media, ad esempio in cima ad un mobile, e poter essere disposto in tutti gli ambienti di casa. Per questo avere la possibilitá di connettere lo stesso al server di raccolta dati attraverso il Wi-Fi diventa indispensabile per evitare di dover cablare cavi di rete in ogni stanza.

3.2.2 Requisiti dell'applicazione

In quanto unico mezzo di interfacciamento tra utente finale e dati rilevati, all'applicazione per smartphone é richiesto di soddisfare i seguenti requisiti:

- Avere un'interfaccia semplice
- Permettere la registrazione di un nuovo utente
- Permettere il Login di un utente giá registrato
- Permettere all'utente di associare i propri dispositivi
- Permettere di visualizzare i dispositivi associati
- Permettere di monitorare in tempo reale i dati raccolti da ogni dispositivo
- Permettere di monitorare uno storico parziale dei dati raccolti da ogni dispositivo
- Notificare l'utente in caso i livelli di CO2 rilevati per un dispositivo siano troppo elevati

3.2.3 L'architettura del sistema

La struttura architetturale di questo sistema, seppur semplice mostra alcune insidie di carattere decisionale, infatti in fase di progettazione dello stesso si é dovuta immaginare una struttura capace di comunicare a livello capillare tra tutte le sue componenti.

L'idea principale é stata la suddivisione quasi netta dei compiti tra le diverse componenti, mediante un'architettura a micro-servizi. Infatti, si é cercato di affidare quanto piú possibile ad ogni micro-servizio ed ogni struttura il proprio compito: questo motivo giustifica la presenza di diversi database e della divisione netta presente fra la API che comunica col client ed il server che comunica con dispositivo.

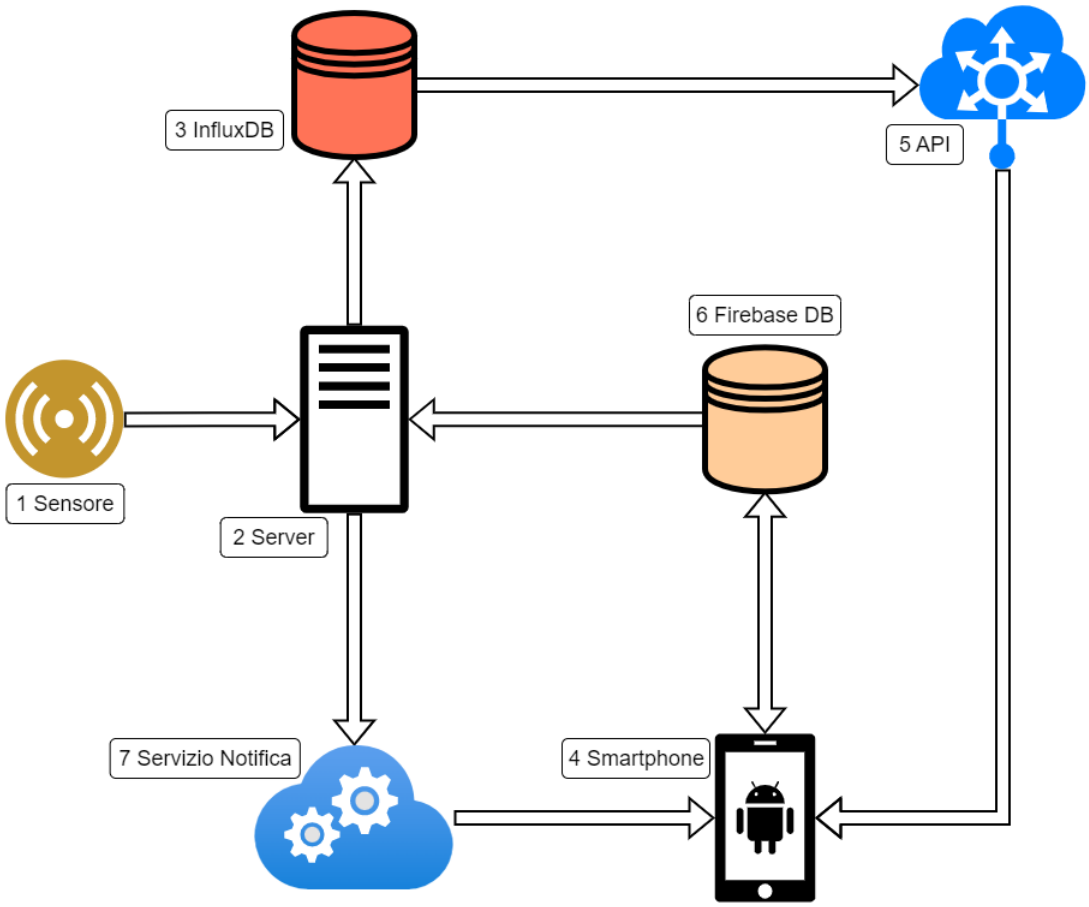

Figura 3.1: Architettura del sistema

Nel diagramma in Figura 3.1 viene mostrato come gli elementi sono tra loro in relazione. Il sensore **(1)** comunica i dati al server di raccolta **(2)**, il quale in primis esegue alcune operazioni di decomposizione della stringa ricevuta contenente il dato e salva il dato ricevuto e correttamente formattato sul primo database TSDB InfluxDB **(3)**. In seguito, dopo una veloce analisi del dato decide se inviarlo al servizio di notifica **(7)**: quest'ultimo ha l'unico compito di notificare l'utente qualora il dato rilevato sia troppo elevato.

Lo smartphone **(4)**, unico punto del sistema che viene operato dall'utente finale, ammette la registrazione dei dati relativi all'utente: email, password, sensori posseduti e token per la ricezione di notifiche. Per salvare questi dati

utilizza il secondo ed ultimo database **(6)**, Firebase. Quest'ultimo viene anche utilizzato dal server **(2)**, per essere in grado di recuperare le informazioni relative al token di notifica salvate su questo database.

Infine il client del sistema, ovvero lo smartphone, richiede alla API **(5)** i dati relativi ai sensori di suo interesse, i quali vengono forniti formattati nel modo desiderato a seconda di dove devono essere mostrati (in quale pagina dell'applicazione): questo avviene solamente mentre l'applicazione é aperta e viene eseguita. Per essere in grado di fornire questi dati, la API infine dovrá contattare il primo database **(3)** per riceverli.

3.3 I componenti software

Come é stato anticipato nella sezione precedente, l'intero sistema si suddivide in componenti hardware e software. La prima categoria di componenti, quella hardware verrá discussa di seguito nella sezione seguente: "2.4 Il dispositivo", siccome il dispositivo di rilevamento é l'unico della sua categoria.

Per quanto riguarda le componenti software possono essere suddivise a loro volta in due sottocategorie: Client e Server. Per il client l'unica presente é l'applicazione per smartphone, mentre per il server verranno discusse le scelte relative al server di raccolta dati ed al protocollo di comunicazione con il sensore, ai database ed infine alla API.

La scelta di dividere il software in cosí tante componenti é stata dettata dalla necessitá di suddividere i compiti assegnati ad ognuna di esse. Ragionando di un sistema software ideale, infatti, bisognerebbe immaginare di avere classi ed elementi che svolgono una sola azione ed una sola funzione. Soprattutto in situazioni analoghe a quella presentata, nelle quali é presente un sensore che invia dati con una frequenza molto alta, non si puó rischiare di avere un elemento del sistema che rimane bloccato a causa delle troppe

richieste o perché gli viene richiesto di svolgere compiti troppo lunghi.

Mentre al primo problema si puó solitamente porre rimedio utilizzando delle strutture adeguate a livello hardware, come per esempio l'impiego di un server con una potenza di calcolo adeguata, il secondo problema é una manifestazione coerente di una cattiva progettazione dell'intero sistema.

3.3.1 Il server ed il protocollo MQTT

Per comunicare e salvare i dati raccolti tra le diverse componenti del sistema, si é innanzitutto reso necessario creare uno standard interno al sistema per la formattazione del dato. Avendo deciso di salvare i dati raccolti dal sensore su un TSDB, ovvero un database specializzato nel salvataggio di grandi quantitá di dati come serie temporali, rispettando i protocolli standard utilizzati nel mondo dell'IoT, ogni dato richiede di avere una chiave temporale come parametro di ingresso nel database.

Inoltre, é stato necessario definire una nomenclatura per i sensori poiché ognuno di questi potesse avere un id univoco per essere successivamente identificato nelle operazioni di lettura dei dati dal database: si é quindi scelto di definire l'id come una serie di caratteri e numeri aventi la seguente forma "An_CO2_Sm".

La nomenclatura appena vista é stata immaginata e pensata apposta per essere capaci in futuro di integrare il dispositivo progettato in un sistema piú ampio di sensori, che possa includere anche funzionalitá differenti dalla lettura del livello di CO2.

É infatti possibile scomporre il formato scelto per il nome come segue:

- An → carattere identificativo del sensore composto da una lettera ed un numero, puó essere utile per tracciare le generazioni oppure fungere da estensione del seriale

- CO2 → indica la funzione del sensore o il tipo di dato raccolto dallo stesso

- Sm → seriale del sensore, composto da lettera e numero (in questo caso la lettera parte da S, questa scelta é stata fatta per avere una maggiore comprensibilitá)

Essendo un nome univoco, quello appena descritto é da immaginarsi come un seriale del sensore. Graie alla nomenclatura scelta, in un futuro nel quale si vorranno includere nel sistema sensori con funzionalitá ulteriori rispetto alla lettura del livello di CO2, sará sufficiente cambiare il parametro relativo alla funzione del sensore per indicare cosa questo sta leggendo (ad esempio DEG per indicare la temperatura in gradi, ...). Inoltre anche la gesitone del topic sará alquanto semplice poiché ricalcando quanto fatto per la nomenclatura del sensore si potrá generalizzare il topic sul quale vengono comunicati i dati raccolti dai sensori: un'idea potrebbe essere "An_FNC_Sm", dove FNC indica esattamente Function (funzione in inglese).

MQTT[26] é un protocollo di comunicazione che basa il suo funzionamento su una logica "publisher - subscriber", ovvero consiste in un server che rende disponibile un indirizzo ai client che possono registrarsi come publisher o subscriber relativamente ad un topic. Il topic a sua volta é da immaginarsi come un "argomento", ovvero una sezione all'interno della quale vengono scambiati messaggi tra tutti gli utenti registrati. Nello scambio di informazioni con un protocollo MQTT solamente i client sottoscritti come publisher possono scrivere, mentre quelli sottoscritti come subscribers ricevono i messaggi pubblicati e vengono notificati: un client puó essere registrato contemporaneamente in entrambe i ruoli.[27]

Per comunicare il dato appena raccolto, il dispositivo utilizza la comunicazione Wi-Fi per registrarsi con il ruolo di "publisher" ad un broker MQTT e pubblicare il dato al topic generico avente il nome descritto so-

pra "An_CO2_Sm".

Il server di raccolta dati, a sua volta, si sottoscrive come subscriber allo stesso topic e riceve i messaggi contenenti i dati raccolti dal sensore formattati "valore&seriale" (ad esempio 983.56&A0_CO2_01). Il dato viene quindi scomposto e poi salvato sul TSDB ospitato da InfluxDB.

Anche nella scelta del bucket nel quale salvare i dati, ovvero la collezione interna al database dove vengono scritte le informazioni, é stato tenuto conto di un possibile futuro ampliamento delle famiglie di sensori facenti parte del sistema. Questa scelta verrá approfondita nella sezione relativa al database.

3.3.2 Il Database InfluxDB

Il database scelto per l'archiviazione dei dati raccolti dal sensore é un Time Series Database (TSDB), nello specifico InfluxDB. Quest'ultimo utilizza un linguaggio di querying proprietario "FLUX": questo é il linguaggio utilizzato per formattare le richieste inviate alla API del database per ottenere i dati interessati. La particolaritá di questo database é la velocitá in risposta pur offrendo la possibilitá di archiviare quantitá di dati enormi e permettendo il caricamento dei dati con una frequenza molto elevata.

La gestione interna delle collezioni del Database prevede la suddivisione in Bucket annidati, all'interno dei quali sará possibile effettuare query dei dati suddivise per chiave di riferimento: nello specifico per questo progetto si é scelto di creare un bucket piú esterno chiamato "***home_sensors***" all'interno del quale salvare i valori con le etichette di riferimento annidate in base alla tipologia di dato rilevata, quindi in questo caso "***co2_measurements***". Infine il valore della misurazione, suddiviso in base al sensore che l'ha rilevato, verrá salvato con l'etichetta "***co2_value***". Come anticipato nella sezione precedente, anche la scelta della nomenclatura relativa alle collection tiene conto di possibili sviluppi futuri. Infatti, sará possibile salvare all'interno di

"home_sensors" i dati relativi a tutti i sensori, cosí da renderli piú accessibili poiché non necessarie connessioni ad altri bucket che non siano quello principale. Allo stesso tempo nelle collection annidate basterá sostituire il parametro "co2" con la funzionalitá scelta del sensore, esattamente come per la nomenclatura ed aggiornare le letture al database senza dover modificare interamente il codice delle varie parti.

Seguendo il percorso specificato da queste tre etichette sequenziali che identificano il dato, é possibile archiviare i dati con un ulteriore identificativo, rispettivamente il nome proprio (o id) del sensore che li sta raccogliendo.

Esisterá quindi un bucket specifico esclusivamente per ogni sensore { A0_CO2_01, A0_CO2_02, A1_CO2_01, ... }, di conseguenza la API offerta da InfluxDB permetterá di effettuare la ricerca dei dati per intervallo temporale, gap temporale tra le rilevazioni e limite dei dati direttamente sulla base del sensore interessato.

Per effettuare quindi la query dei dati del sensore A0_CO2_01 bisognerá scrivere una query che segua il perscorso:

\home_sensors\co2_measurements\co2_value\A0_CO2_01

e definire i parametri di query interessati, "limit" ed "intervallo di tempo". Nella query seguente viene mostrato come prendere i dati relativi alle rilevazioni effettuate dal sensore A0_CO2_01 nell'intervallo di tempo [$v.timeRangeStart \rightarrow v.timeRangeStop$]

Listing 3.1: Query di esempio da InfluxDB

```
from(bucket: "home_sensors")
 |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
 |> filter(fn: (r) => r["_measurement"] == "co2_measurement")
 |> filter(fn: (r) => r["_field"] == "co2_value")
 |> filter(fn: (r) => r["device"] == "A0_CO2_01")
 |> aggregateWindow(every: v.windowPeriod, fn: mean,
    createEmpty: false)
 |> yield(name: "mean")
```

Figura 3.2: Query al sensore A0_CO2_01 col tool visuale di InfluxDB

3.3.3 Il Database Firebase

In questo caso, a differenza del database precedente, le scelte progettuali sono state alquanto semplici poiché é bastato decidere di salvare in primis le credenziali di accesso dell'utente, garantendo la sicurezza dei dati grazie al sistema di autenticazione protetta offerto direttamente da Firebase, per poi utilizzare lo uid (ovvero l'id univoco) dell'utente per salvare gli altri dati non sensibili in chiaro sul database, senza bisogno di crittografia.

Il secondo database utilizzato dal sistema é, a differenza del primo, un DB NoSQL offerto da Google con la suite Firebase; questo pacchetto permette di creare all'interno di una console dell'utente diversi progetti e, di disporre di differenti tabelle all'interno dello stesso database. La particolaritá dei database NoSQL come Firebase é la possibilitá di inserire i dati desiderati

senza il bisogno di aver definito a priori la struttura interna del database.

I database NoSQL, infatti, utilizzano modelli di dati non relazionali, come ad esempio il modello a documento, il modello a grafo o il modello a chiave-valore. Questo significa che i dati non vengono organizzati in tabelle con righe e colonne, ma in strutture dati piú flessibili, adatte ad accogliere diversi tipi di dati. Inoltre, questi sono spesso utilizzati in applicazioni web e mobile, in cui la scalabilitá, la flessibilitá e le prestazioni sono fondamentali per mantenere un funzionamento altamente dinamico dell'applicazione stessa.

In questo progetto, il database ospitato da Firebase, viene utilizzato per salvare i dati utili al funzionamento dell'applicazione, ovvero dati di login dell'utente, sensori posseduti dall'utente ed infine il token utilizzato dal server di raccolta dati per inviare notifiche all'utente.

Viene quindi sfruttata la peculiaritá offerta dai database non relazionali di restituire i dati in formato JSON, standard utilizzato per formattare in testo semplice strutture dati organizzati in un albero di oggetti annidati. Grazie all'impiego di questo standard é facilmente possibile recuperare i dati ricevuti maneggiando l'oggetto in formato JSON, ecco un esempio di questa struttura di seguito:

Listing 3.2: Dati dal db NoSQL in formato JSON

```
{
  "notificationTokens": {
    "kM9iMnrZmfPCMEcpBQFrEiMlJX83": {
      "-NOUsbB-FwRGBCy1VDPs": "{\"notificationToken\":\"
          ExponentPushToken[neb7r1NzGriudx8KEiPGHY]\"}"
    }
  },
  "sensors": {
    "kM9iMnrZmfPCMEcpBQFrEiMlJX83": {
      "A0_CO2_01": {
```

```
        "-NOUjGU_EAghHgi-NC00": "{\"name\":\"Bedroom\",\"
            description\":\"The first sensor made\"}"
      },
      "A0_CO2_02": {
        "-NOUp_nXNc8NOphkdAJT": "{\"name\":\"Livingroom\",\"
            description\":\"\"}"
      }
    }
  }
}
```

Nell'esempio visto nel frammento di codice "Listing 3.2", la chiave utilizzata per salvare i dati relativi ai singoli "**notificationToken**" e "**sensor**" é lo uid spiegato precedentemente, per questo motivo all'interno delle due collezioni "**notificationTokens**" e "**sensors**" si trova la stessa chiave: in questo caso il sensore visualizzato ed il token di notifica appartengono allo stesso utente.

3.3.4 La API

L'ultimo tassello architetturale di questo progetto consiste in una molto semplice API serverless. Fra i diversi tasselli del sistema presentato, la API sviluppata ha l'unico compito di consentire all'applicazione per smartphone di contattare il database principale per richiedere dei dati specifici quando necessario.

Un'API (Application Programming Interface) consiste in un insieme di definizioni, protocolli e strumenti per lo sviluppo software, e viene usata per fornire un modo a diverse applicazioni o componenti di un sistema di interagire tra loro scambiando dati e informazioni.

Le API piú comuni sono le cosiddette API REST (Representational State Transfer), le quali dispongono di un'architettura di rete che utilizza un insieme di regole atte a creare servizi web consultabili da terze parti (ad esempio applicazioni che usufruiscono di un servizio). Questo tipo di API, sfrutta per

la comunicazione il protocollo HTTP e, di conseguenza, ne implementa i metodi di interfacciamento standard piú comuni: GET, POST, PUT, DELETE.

Al contrario, le API serverless, come dice la parola stessa, non dispongono di una struttura server al di sotto. Per questo motivo devono appoggiarsi a strutture terze per leggere e scrivere dati di qualsiasi tipo, sono in poche parole dei servizi cloud di interfacciamento a strutture dati terze.

Per questo motivo le API serverless risultano essere molto piú scalabili, nonché utilizzabili come livello intermedio di comunicazione tra un client ed un servizio. Essendo comunque ospitate su un server permettono di interporre sistemi di manipolazione dei dati o addirittura di analisi degli stessi, nonché di fornire l'accesso al client a piú servizi contemporaneamente con una richiesta unica a questa API.

É bene precisare che sarebbe stato possibile far comunicare direttamente l'applicazione con il database, sfruttando la API direttamente offerta dallo stesso, ma la scelta é ricaduta su un intermediario per riuscire a togliere dal client del sistema (ovvero l'applicazione) quanta piú logica computazionale possibile.

Grazie all'utilizzo di questo intermediario é infatti stato possibile fornire al client i dati formattati esattamente come necessario per poter essere interpretati, mostrati e salvati senza bisogno di ulteriori manipolazioni. Questo consente di avere performance migliori rispetto a ricevere una mole di dati talvolta molto massiva, da dover successivamente manipolare ed interpretare per essere in grado di mostrarli all'utente.

Nello specifico questo accade nella pagina destinata alla visualizzazione dello storico, dove possono essere selezionati i dati relativi persino agli ultimi 30 giorni: qui la quantitá di singole rilevazioni ricevute supera le 2000, ed il vantaggio di averle in formato JSON giá pronte per essere inserite in un grafico rende l'applicazione facilmente utilizzabile.

Esistono due formattazioni del dato rese disponibili dalla API, che vengono sfruttate in base alla pagina dell'applicazione che richiede il dato. La prima, utilizzata nella pagina dell'applicazione contenente tutti i sensori, restituisce un'unica misurazione contenente il tempo in cui essa é avvenuta, il valore rilevato e l'id dei dispositivo (come mostrato in "Listing 3.3"). L'id del dispositivo sará usato dal client per aggiornare tutti i sensori dei quali ha ricevuto i dati dopo aver contattato per ognuno di essi la API.

Listing 3.3: Esempio di array grezzo restituito dalla API

```
[
    {"time":"2023-02-17T14:16:20Z","value":1022.43,"device":"
       A0_CO2_01"}
]
```

Al contrario la seconda formattazione restituisce una lista di punti aventi coordinate **{ x, y }**, pronti per essere visualizzati in un grafico. In questo caso non é necessario specificare il dispositivo poiché si chiede alla API di restituire una serie di valori appartenenti ad un unico dispositivo, ne segue un esempio di risposta nel "Listing 3.4".

Listing 3.4: Esempio di array rifinito per il grafico restituito dalla API

```
[
    {"x":"2023-02-17T17:39:40Z","y":995.84},
    {"x":"2023-02-17T17:40:00Z","y":1054.51},
    {"x":"2023-02-17T17:40:20Z","y":917.37},
]
```

3.3.5 L'applicazione mobile

Come anticipato nelle sezioni precedenti, all'applicazione é richiesto di permettere all'utente di interfacciarsi con il sistema creato. Piú precisamente attraverso l'applicazione l'utente gestisce i propri sensori e, ha la possibilitá di monitorarne le rilevazioni in tempo reale. Oltretutto il ruolo piú importante rivestito dall'applicazione é quello di notificare l'utente al raggiungimento

di livelli critici di CO2 in uno qualsiasi degli ambienti nei quali sono stati posizionati i sensori.

Con lo scopo di rendere "**CO2 Monitor**", l'applicazione, accessibile a chiunque e di facile utilizzo, nella progettazione dell'interfaccia é stato adottato un approccio minimale ed altamente visuale. Composta di poche schermate, infatti, non permette all'utente di trovarsi a navigare funzionalitá sconosciute. Dopo avere aperto l'applicazione sará possibile innanzitutto autenticarsi grazie alla apposita schermata "**Authentication**" visibile di seguito nell'immagine "Figura 3.3".

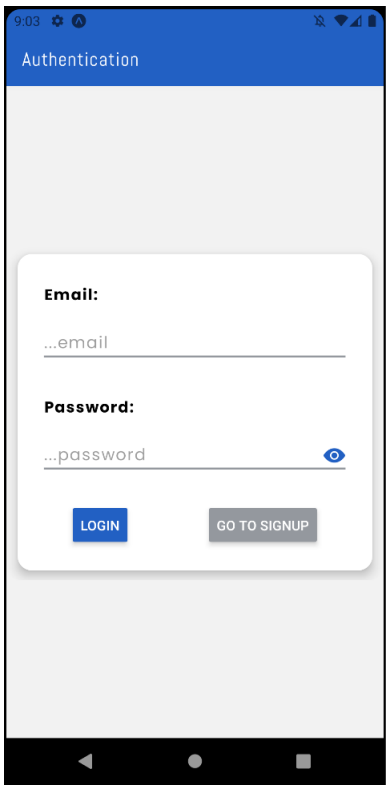

Figura 3.3: Schermata di autenticazione

Dopo aver eseguito la registrazione attraverso il pulsante "Go to signup", oppure essersi autenticati attraverso il pulsante "Login", sará possibile iniziare ad esplorare l'applicazione.

Questa offrirá accesso a tre schermate, la prima con il titolo di "**Sensors**" é la schemata di home, e permetterá di visualizzare, aggiungere e rimuovere i dispositivi che si possiedono e si desidera monitorare rendendo le modifiche persistenti sul database secondario visto in precedenza. Perció questa schermata sará un collegamento diretto con il database. Verranno caricati i sensori giá salvati sul DB e, all'aggiunta di nuovi verranno automaticamente salvati su di esso.

La schermata si compone di una lista di cards dove é possibile visualizzare nome e valore dell'ultima rilevazione di ogni sensore e, di un pannello a scomparsa per l'aggiunta dei nuovi come visibile nell'immagine "Figura 3.4".

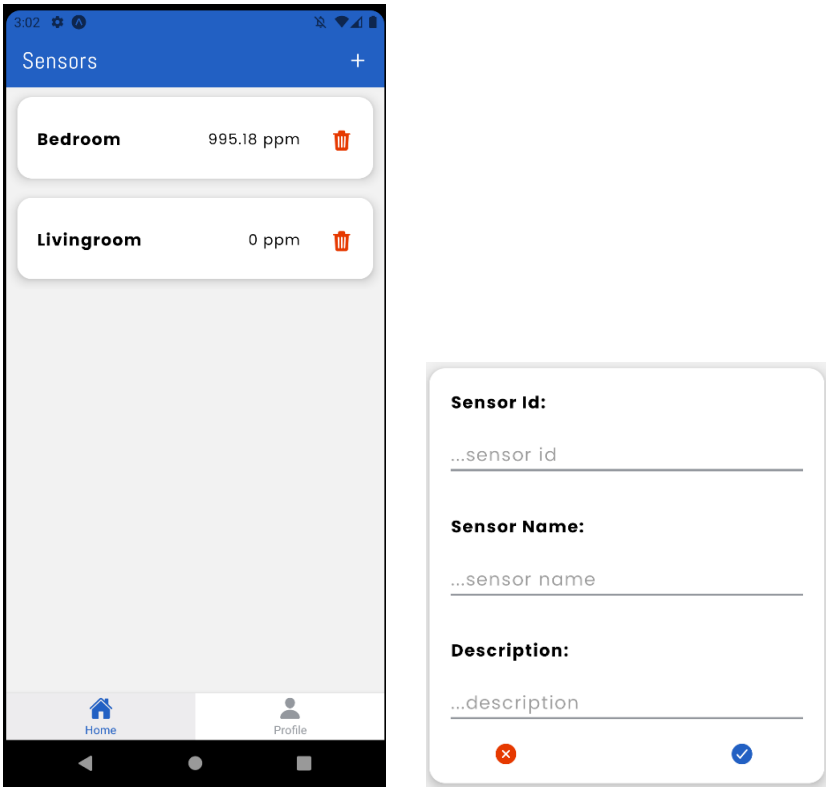

Figura 3.4: Schermata di Home e aggiunta sensori

Dalla schermata "Sensors" l'utente potrá toccare qualsiasi sensore visualizzato per accedere alla seconda schermata dell'applicazione, specifica

per ogni sensore e chiamata appunto "**Sensor**": essa riporterá infatti i dati principali relativi al sensore selezionato (nome, id e descrizione).

Qui l'utente avrá la possibilitá di visualizzare uno storico delle rilevazioni sotto forma di grafico selezionando l'intervallo temporale interessato: quest'ultimo potrá variare da 1 minuto a 30 giorni, con un numero di dati nel grafico che varia da 3 ad oltre 4000 per la selezione "30 giorni".

L'intervallo di tempo potrá essere selezionato con due Picker che contengono gli intervalli di tempo predefiniti selezionabili, di seguito é possibile vedere nell'immagine "Figura 3.5" la schermata con un grafico rappresentante un lasso di tempo di cinque minuti.

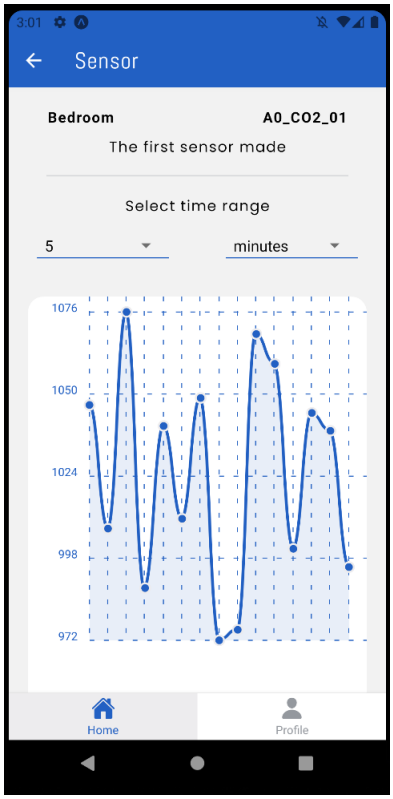

Figura 3.5: Schermata di dettaglio del sensore

Navigando l'applicazione l'ultima schermata visualizzabile sará quella relativa all'account, dove l'utente avrá la possibilitá di visualizzare i propri dati: indirizzo email e data di scadenza del token di accesso. L'applicazione

é stata studiata in modo tale da permettere all'utente di rimanere connesso ad ogni riavvio della stessa e, ma per garantire la maggiore usabilitá a lungo termine il token viene rigenerato ed aggiornato ad ogni accesso.

Quest'ultimo dipende infatti dalla politica adottata da Firebase, del quale viene sfruttato il servizio di autenticazione, quindi potrebbe verificarsi l'ipotesi in cui il token scada: grazie al meccanismo adottato in questa applicazione non sará necessario intervenire per l'utente poiché il token verrá rigenerato ed aggiornato in background ad ogni avvio dell'applicazione.

É possibile vedere come si presenta la schermata "**Profile**" nell'immagine "Figura 3.6" di seguito.

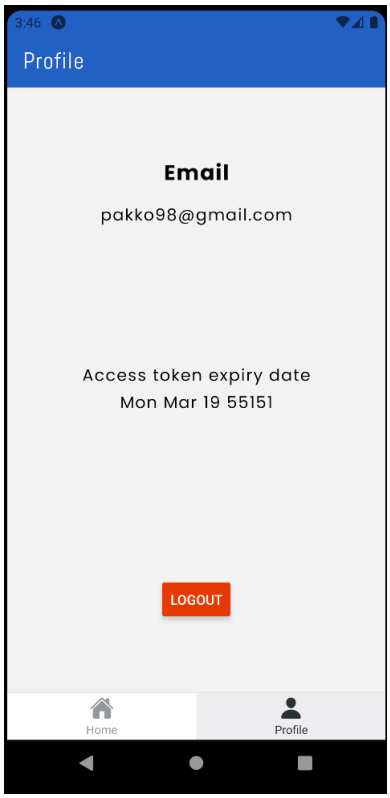

Figura 3.6: Schermata del profilo utente

Infine, sia con l'applicazione aperta che chiusa, sará possibile sfruttare la sua piú importante funzionalitá, ovvero quella di inviare notifiche al raggiungimento di una soglia critica di CO2. Seppur il sistema progettato non

consista in un dispositivo di sicurezza, il livello critico di CO2 non é stato reso impostabile dall'utente. Non conoscendo i livelli ottimali per un ambiente chiuso, quest'ultimo avrebbe potuto rendere completamente inefficace il dispositivo impostando un livello troppo elevato. Il livello scelto é quindi pari a **1000 ppm**, oltre il quale aumenta probabilitá di contagio per il virus Sars-Cov-2.

Le notifiche, come giá introdotto, sono create dal server di raccolta dati. Non appena viene trovato valore di CO2 superiore alla soglia descritta, il server contatta il database secondario per chiedere i token di tutti gli utenti che possiedono il sensore che ha appena comunicato il dato per generare le notifiche per i rispettivi dispositivi. Di seguito viene mostrato nell'immagine "Figura 3.7" come appaiono le notifiche all'utente sia con l'applicazione giá aperta che con l'applicazione chiusa.

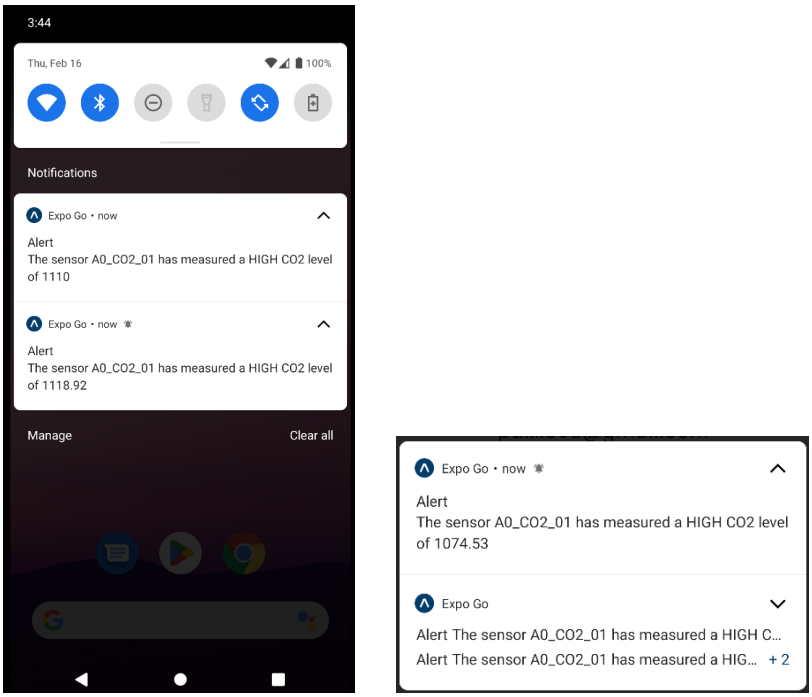

Figura 3.7: Notifiche push dell'applicazione

3.4 Il dispositivo

La componente hardware del sistema, ovvero il dispositivo per la raccolta dati, ha richiesto diversi step di progettazione e costruzione. Inizialmente si é reso necessario sviluppare la componente software che poi sarebbe stata caricata all'interno del microcontrollore per la gestione del funzionamento dello stesso. Quest'ultimo infatti ha consistito nella prima reale componente hardware dell'intero dispositivo, il quale é cosí stato preparato ad accogliere il sensore di rilevamento e gli elementi relativi all'interfaccia utente del dispositivo stesso.

A questa fase iniziale é seguita quella di progettazione dell'interfaccia del dispositivo, lo scopo infatti é quello di fornire un dispositivo semplice da utilizzare e che fornisse con semplicitá informazioni sul proprio status. Questa interfaccia presenta un bottone che consente di eseguire il reset e la procedura di taratura del sensore qualora fosse necessario, e due led di segnalazione dello stato di funzionamento, uno verde che lampeggia durante l'invio dei dati al server ed uno rosso che si accende fisso quando viene rilevato un errore di connessione al server o al router Wi-Fi, mentre durante la procedura di warm-up i led sono accesi entrambi.

Volendo rendere funzionale e pratico l'utilizzo del sensore, come giá spiegato, é inoltre stato necessario progettare una PCB (nome comune per indicare un circuito stampato), che ha il compito di fungere da livello intermedio sul quale assemblare tutte le componenti, che al contrario sarebbero state oggetti separati. Durante l'intero ciclo di progettazione e sviluppo, il dispositivo é stato infatti assemblato e testato su una breadboard (una scheda di prototipazione). La fase di creazione della PCB ha quindi consistito nella riproduzione degli schemi elettrici configurati sulla breadboard in un software apposito, Easy-EDA.

É possibile osservare nell'immagine "Figura 3.8" lo schema elettrico del-

le connessioni del dispositivo utilizzato nell'assemblaggio finale, mentre nell'immagine "Figura 3.9" é possibile osservare un modello 3D del risultato ottenuto con la creazione della PCB e dell'interfaccia per l'utente.

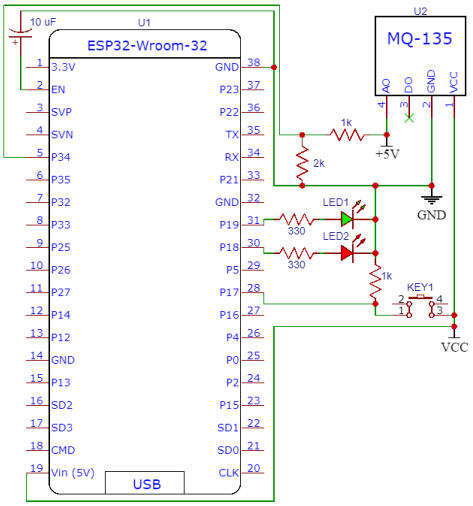

Figura 3.8: Schema di collegamento del dispositivo

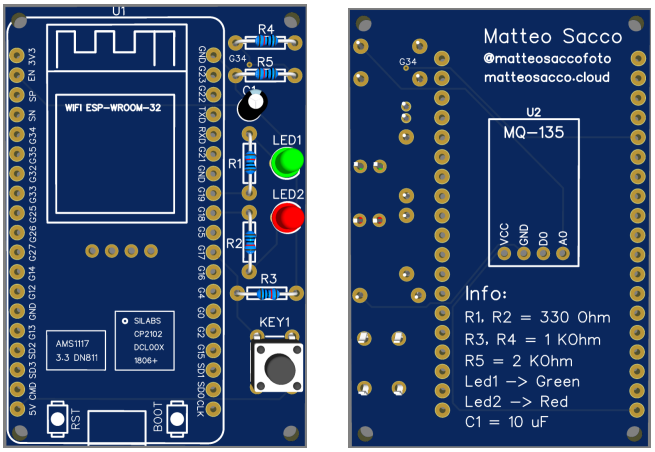

Figura 3.9: Modello 3D della PCB e dell'interfccia utente

Capitolo 4

Implementazione

Le scelte implementative per arrivare al sistema descritto sopra sono state spesso e volentieri dettate dalla possibilitá di far comunicare tra loro sistemi cosí differenti. Essendo per la maggior parte sviluppo software, ognuna di queste scelte ha dovuto trovare spazio in mezzo ad una vastitá di concorrenti equipollenti.

Ad ognuno dei tasselli descritti nel capitolo precedente, viene richiesto di rispettare una numerositá elevata di requisiti, ma soprattutto gli é esatto di essere facilmente utilizzabile all'interno del sistema.
Sará possibile quindi osservare quanto spesso le scelte implementative siano ricadute sullo strumento che piú facilmente avrebbe permesso al sistema di funzionare senza intoppi: é bene specificare che non per questo sono da considerarsi scelte al ribasso. Laddove necessario sono infatti stati interposti livelli di complessitá ulteriore alla minima richiesta per funzionare, ad esempio per migliorare le prestazioni o accertarsi della migliore efficacia di funzionamento possibile.

Per ottemperare questo obiettivo si é quindi reso necessario conoscere ed utilizzare diverse tecnologie: alcune basi di elettronica per poter progettare il dispositivo, ed avere una conoscenza ad ampio spettro nello sviluppo

software per la progettazione e l'implementazione dell'applicazione mobile, la realizzazione della API ed il server, infine per poter utilizzare il protocollo MQTT scelto per la comunicazione tra sensore e server di raccolta dati. Tra le altre competenze necessarie sono state essenziali basi di design e conoscenza di diversi programmi per la progettazione delle PCB ed IDE (Integrated Development Environment) per lo sviluppo del codice a livello software.

Nel capitolo seguente verranno discusse e mostrate tali scelte implementative assieme alle tecnologie che si é deciso di usare.

4.1 Il dispositivo

Per assemblare il dispositivo si é partiti dall'utilizzo di un microcontrollore ESP32 Wroom-32 prodotto da AZ-Delivery, poiché, seppur economico, é provvisto di moduli Bluetooth e Wi-Fi. Per la tipologia di dispositivo creato risulta infatti fondamentale avere accesso almeno ad un modulo Wi-Fi per poter comunicare i dati raccolti alle altre parti del sistema.

Per completare il dispositivo sono stati in seguito aggiunti diversi componenti tra cui, per la rilevazione del dato, un sensore MQ-135 prodotto dalla stessa azienda del microcontrollore, per il rilevamento di gas quali CO e CO2. Per rendere funzionante il sensore si sono rese necessarie diverse fasi, tra cui la progettazione di un circuito Voltage-Divider per abbassare l'output del sensore da una tensione di 5V ad una di 3V e permettere cosí al microcontrollore di leggere i valori correttamente.

Segue nell'immagine "Figura 4.1" un dettaglio del circuito voltage divider come utilizzato nella progettazione dello schema: é stato disegnato solamente un pin del microcontrollore (con riferimento al pin interessato) ed un unico pin anche per il sensore (il pin di uscita del segnale per la lettura da parte del microcontrollore).

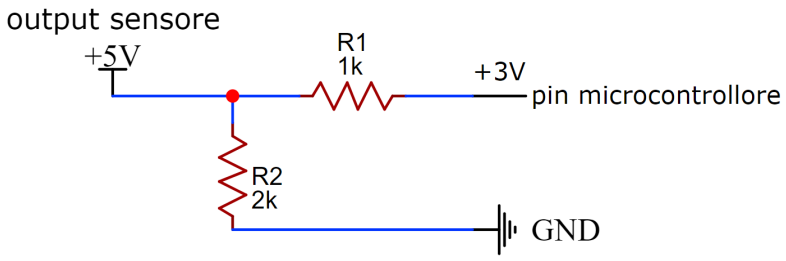

Figura 4.1: Circuito Voltage Divider

Per ottenere una lettura precisa dei dati del sensore, é inoltre stato necessario studiare un metodo di calibrazione e di warm-up del sensore. Questo ha portato alla progettazione ed allo sviluppo di uno script successivamente pubblicato su GitHub, il quale basandosi sulle informazioni rilasciate dal produttore esegue un warm-up del sensore di due minuti, e ne esegue subito dopo la calibrazione per un tempo di un'ora.

Va tenuto in considerazione che questo processo é da eseguirsi soltanto una volta all'acquisto del sensore, ma solamente dopo averlo lasciato completare il warm-up dichiarato dalla casa produttrice, che varia da 24 a 72 ore in base al tempo di inutilizzo del sensore: solitamente appena nuovo é consigliato un tempo di warm-up di almeno 48 ore.

La procedura di warm-up eseguita dalla calibrazione dura appena due minuti, ed é la stessa che potrá eseguire l'utente finale premendo il pulsante presente sulla PCB. La procedura di calibrazione, come il warm-up per l'utente finale non possono infatti durare due giorni e, viene perció introdotta la possibilitá di fare il warm-up del sensore in versione ridotta. Si é visto dai test e l'utilizzo quotidiano che questo consente comunque all'utente di avere un dispositivo che restituisce letture valide, aspettando solamente due minuti dall'accensione per riportare in temperatura la bobina del sensore.

Per eseguire la calibrazione del sensore, é necessario averlo prima attaccato al microcontrollore, tuttavia non serve aver collegato l'intera PCB con

l'interfaccia utente. Dopo aver collegato il sensore occorre quindi caricare sul microcontrollore lo script di calibrazione sul microcontrollore ed aspettare che questa termini. Dopo che la calibrazione sará completata, sará possibile inserire il valore ottenuto nelle funzioni di lettura del dato dal sensore ed utilizzarlo per rilevare i dati corretti.

Nelle porzioni di codice "Listing 4.1" e "Listing 4.2" é possibile rispettivamente osservare l'output della calibrazione e l'utilizzo di questo valore nelle funzioni di lettura dei dati.

Listing 4.1: Output della calibrazione

```
Warming up sensor: (...this operation takes 2 minutes...)
_4%_8%_12%_16%_21%_25%_29%_33%_37%_42%_46%_50%_54%_58%_63%_67
   %_71%_75%_79%_84%_88%_92%_96%_100%
--- Sensor warmed up correctly
Calibrating sensor ...
MQ135 RZERO Calibration Value : 16.04
1 : 16.32
16.32
------------------------------
// ...
120 : 2874.03
27.04
------------------------------
Calculating the weighted mean of all the values read ...
The R_Zero found after calibration is : 26.64
```

Ad ogni step é mostrata la somma di tutti i valori "*r_zero*" trovati fino a quel momento ed il valore del singolo step, alla fine é possibile osservare che nonostante la media dei 120 valori trovati sarebbe equivalente a circa 23.95, il valore di r_zero finale risulta essere "**26.64**": questo accade grazie alla media pesata dei valori trovati.

Listing 4.2: Lettura del valore di CO2 con il parametro di calibrazione

```
// definisco il sensore con numero di pin e valore r_zero
   trovato mediante la calibrazione
```

```
MQ135 mq135_sensor(SENSOR, 26.64);

// eseguo la lettura del valore di CO2 misurato in ppm
float ppm = mq135_sensor.getPPM();
```

Nel frammento di codice appena visto é possibile osservare due punti cruciali del codice, il primo consiste nella definizione del sensore. Il secondo punto, inserito nello stesso "Listing", é in realtá ció che avviene alla lettura del valore, ovvero in un altro punto del codice, all'interno della funzione di lettura vera e propria.

Nella funzione appena citata, vengono eseguite anche altre operazioni, come la comunicazione del dato al broker MQTT, del quale il dispositivo é un publisher. Nel ciclo di funzionamento del microcontrollore viene infatti richiamata la funzione "**publishToBroker()**", la quale esegue i controlli necessari per la connessione con il broker MQTT, richiede la lettura dei dati del sensore ed infine pubblica il messaggio con il valore letto verso il broker.

Qui il valore letto viene convertito in una stringa ed assemblato con il nome del sensore, il quale é ovviamente unico per ogni dispositivo, per poi essere pubblicato sul broker. Nell'esempio che segue nel frammento di codice "Listing 4.3" é possibile osservare la funzione per il dispositivo "A0_CO2_01".

Listing 4.3: Funzione publishToBroker()

```
void publishToBroker() {
    String device_name = "&A0_CO2_01";
    checkMqttConnection();
    float ppm_msg = readCO2Level();

    String tmp = ppm_msg + device_name;
    const char *msg = tmp.c_str();

    mqttClient.publish(mqtt_topic, msg);
}
```

4.2 Il server di raccolta dati

Dopo aver raccolto i dati dal sensore, come spiegato assieme all'architettura del sistema, i dati vengono letti dal server di raccolta dati, il quale utilizza il broker MQTT come subscriber.

Per implementare questo server é stato scelto il linguaggio "**TypeScript**"[5], ovvero una variante del piú famoso "JavaScript"[6]. La differenza principale tra i due é la possibilitá di utilizzare i tipi delle variabili in TypeScript. Questo consiste sia in un vantaggio, sia in uno svantaggio a seconda dei casi.

In primis si rende evidente il vantaggio di utilizzare variabili tipizzate poiché conferiscono una struttura piú solida al server. Dopo aver letto la stringa inviata dal sensore sul broker, ne effettuerá il parsing, ovvero la scomposizione in parametri. Verranno ricercati due parametri nello specifico, il valore letto e l'id del dispositivo: il messaggio letto avrá una stringa strutturata come segue "**978.25&A0_CO2_01**".

La stringa come appena visualizzata dovrá essere scomposta ricercando i due parametri come "numero" e "stringa". Grazie alla tipizzazione offerta da TypeScript sará possibile accorgersi di eventuali errori direttamente in fase di parsing, ancora prima di cercare di salvare i dati sul database.

Dopo aver ottenuto i valori cercati, il server si connetterá con il database InfluxDB (quello principale) per salvare il dato ottenuto. Solamente in seguito verrá controllato il valore appena salvato poiché se superiore a **1000** é un indice di scarsa IAQ (qualitá dell'aria interna). Qualora venisse trovato un valore di CO2 superiore alle 1000 ppm, saranno notificati tutti gli utenti che monitorano il sensore che ha inviato il dato attraverso una funzione apposita: segue il codice eseguito dal server alla ricezione di un dato nel "Listing 4.4".

Listing 4.4: Server di raccolta dati alla ricezione di una nuova misurazione

```
mqtt_client.on('message', async (topic: string, message:
    Object) => {
```

```
    const msg_string: string[] = message.toString().split
        ('&');
    let value: number;
    try {
        value = parseFloat(msg_string[0]);
    } catch (error) { return; }
    const device: string = msg_string[1];

    const writeApi = influx_client.getWriteApi(influx_org,
        influx_bucket);
    writeApi.useDefaultTags({ device: device });
    const point = new Point('co2_measurement').floatField('
        co2_value', value);
    writeApi.writePoint(point);
    writeApi.close().then(() => { console.log('\tWritten: ' +
        point); })
            .catch((err) => { console.error(err); });
    if (value > 1000)
        await sendNotificationsAsync(device, value);
});
```

A sua volta la funzione "**sendNotificationsAsync**" sfrutta la API offerta da ExpoNotifications, un pacchetto per la gestione dell'invio di notifiche messo a disposizione direttamente da Expo, il quale consiste in un pacchetto di supporto allo sviluppo di applicazioni con il framework "React Native"[4].

L'invio di notifiche, seppur gestito attraverso la API appena citata, prevede una fase preliminare gestita dal server di raccolta dati. Questa consiste nella lettura di tutti i token di notifica degli utenti che possiedono o monitorano attraverso il proprio account il sensore che ha appena inviato il dato.

Per fare ció viene contattato il secondo database in due step: in primo luogo vengono richiesti gli id di tutti gli utenti che osservano il sensore, per poi essere in grado di ricevere come dato dal database il token di notifica di ognuno di essi. Una volta ottenuto il token, viene contattata la API appena

citata per notificare i dispositivi degli utenti.

4.3 La API serverless

Per implementare la API é stato scelto di sfruttare la scalabilitá e le potenzialitá delle API serverless, avendo cosí anche la possibilitá di implementare solamente i metodi necessari per il sistema. Nello specifico é stata creata solamente una funzione che si occupa di comunicare con il database principale su InfluxDB e, si occupa di restituire i dati formattati come é richiesto dall'utente in uno dei due formati visti precedentemente.

Per implementare la API si é utilizzato il linguaggio JavaScript nella sua piú semplice forma, altresí detta "Vanilla JavaScript", assieme al framework Netlify[8], creato appositamente per implementare Serverless API.

Questa API consiste infatti in una applicazione web che puó essere ospitata da un server e resa pubblica, esattamente come si fa con i siti web, ma quando viene contattato il suo indirizzo invece che mostrare una pagina web restituisce dei dati in base agli input immessi. Non é infatti stata implementata alcuna interfaccia, ma viene solamente eseguita una "fetch" all'indirizzo di questa API esplicitando alcuni valori nello URI (la stringa dell'indirizzo), per fornire le informazioni necessarie a leggere i dati corretti dal database.

Per testare questa API prima di utilizzarla dall'applicazione é stato utilizzato "Postman"[7], un programma che permette di effettuare chiamate a qualsiasi indirizzo registrando le risposte, pensato apposta per il testing delle API: ne segue un'immagine "Figura 4.2".

La struttura della API in sé risulta essere molto semplice in quanto implementa solamente uno dei metodi visti per le API Rest, ovvero il metodo "**GET**". Questo esegue la funzione richiesta nell'URL della richiesta, infatti, proprio attraverso l'URL sará possibile decidere quale funzione richiamare.

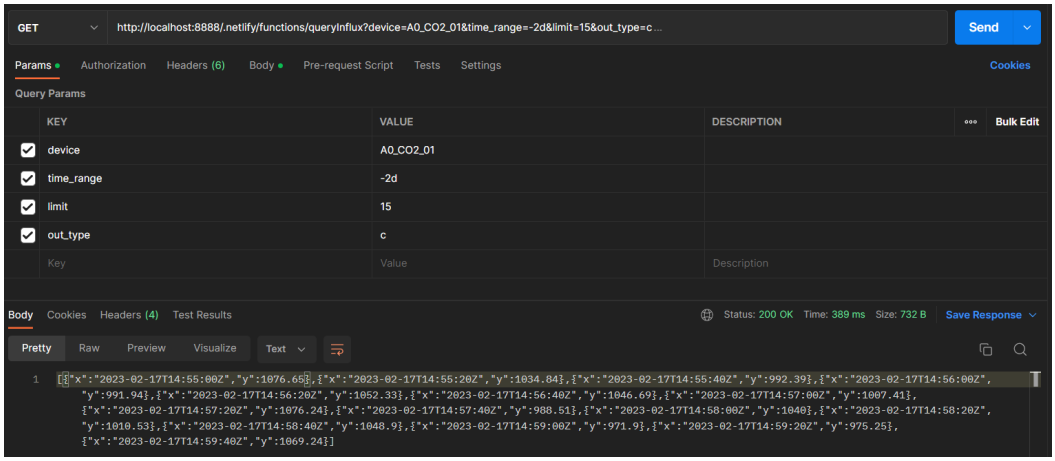

Figura 4.2: Interfaccia Postman

In questa API é presente solamente una funzione poiché essendo l'output l'unico a mutare nella sua formattazione, viene gestito il valore di preferenza dell'utente rispetto a quest'ultimo attraverso un parametro nell'URL. L'URL presenta quindi la seguente struttura: "indirizzo della API" + "funzione" + "parametri".

http://localhost:8888/.netlify/functions/queryInflux?
device=A0_CO2_01&time_range=-2d&limit=15&out_type=c

L'esempio di URL di cui sopra puó essere scomposto nel modo seguente:

- indirizzo: http://localhost:8888/.netlify/
- funzione: functions/queryInflux
- parametri:
  device: A0_CO2_01 (il dispositivo del quale interessano i dati)
  time_range: -2d (indica un intervallo da - 2 giorni ad ora)
  limit: 15 (quante misurazioni sono richieste in output)
  out_type: c (il tipo di formattazione dell'output)

Poiché la API non restituisce in alcun modo dati sensibili o dati che permettano di essere ricondotti ad un individuo, si é scelto di rendere le

letture pubbliche, é quindi contattabile da chiunque previa conoscenza del suo funzionamento e richiesta di informazioni coerenti (é ovviamente necessario conoscere i parametri di formattazione degli output ed i nomi dei dispositivi). Questo ha permesso di evitare di dover affidare al client le credenziali per l'accesso al database, che rimangono quindi solamente all'interno della API.

4.4 L'applicazione mobile

Per l'implementazione dell'applicazione si é scelto di lavorare con il framework "React Native", sviluppato dalla multinazionale "Meta", che permette di scrivere il codice in JavaScript e TypeScript. Nel processo di sviluppo ed implementazione questo é risultato essere molto efficiente per via della consistenza mantenuta a livello del codice fra le diverse componenti software realizzate.

Inoltre utilizzare questo framework ha consentito di accedere alla possibilitá di sviluppare contemporaneamente un'applicazione per smartphone per entrambi i piú popolari sistemi operativi mobile: Android ed IOS. Grazie a questa peculiaritá offerta da React Native é possibile implementare, manutenere e sviluppare attraverso un unico progetto due applicazioni che altrimenti avrebbero persino richiesto differenti architetture progettuali. É anche da considerarsi che grazie allo sviluppo in React Native sará possibile installare questa applicazione su circa il 95% degli smartphone presenti oggi sul mercato e nelle mani degli utenti finali.

All'interno del progetto sono state utilizzate anche altre librerie di supporto a funzionalitá specifiche, ad esempio la libreria "Redux"[9] per la gestione dello stato dell'applicazione. L'obiettivo di Redux é quello di separare la logica dello stato dal resto dell'applicazione, aiutando cosí a fornire un'interfaccia semplice e prevedibile alla quale accedere per aggiornare il suo stato con i nuovi dati stato. Inoltre Redux interagisce con un flusso di dati uni-

direzionale, ovvero che possono essere modificati solamente in un punto dell'applicazione e le quali modifiche vengono propagate in tutti i componenti che stanno usando i dati al momento del cambiamento.

Il pattern architetturale appena descritto mediante l'utilizzo di Redux, nonché quello che si é deciso di adottare per questa applicazione, é il pattern Flux[10]. Questo, sviluppato da Meta assieme a React Native, si prepone l'obiettivo di semplificare la gestione dei dati in applicazioni web e mobile di grandi dimensioni, dove altrimenti il dato rischierebbe di essere modificabile in diversi punti e da diversi componenti.

Grazie a questo pattern viene introdotto un componente chiamato "store" nell'applicazione, il quale ha l'unico compito di modificare lo stato dei dati utilizzati dall'applicazione quando ne vengono richiamate le "actions". Lo store é infatti suddiviso in "actions" e "reducers", le prime consistono in vere e proprie azioni, dalla lettura di dati da un db alla modifica di un dato internamente all'applicazione; mentre le seconde sono funzioni che rappresentano gli stati dei dati, ovvero quando vengono interrogati ne aggiornano il valore e lo restituiscono al componente chiamante.

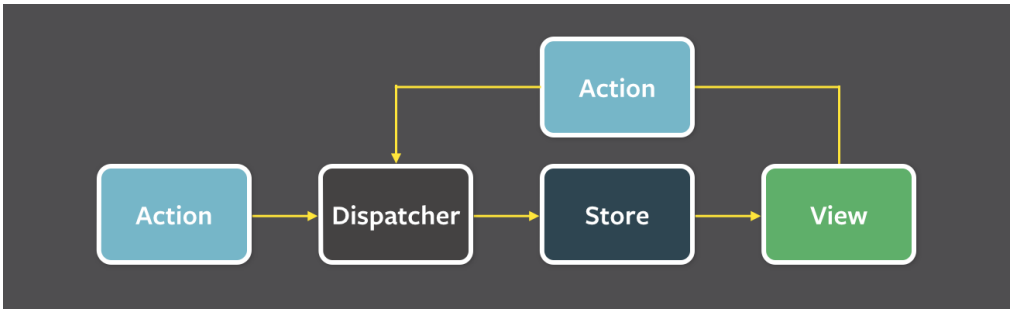

Figura 4.3: Flusso dell'architettura Flux[15]

Nell'immagine "Figura 4.3" viene illustrato lo schema relativo al flusso di funzionamento del pattern Flux: le azioni vengono create e gli vengono

passati dei parametri, i quali vengono elaborati e successivamente trasmessi attraverso i dispatcher agli store; questi ultimi aggiornano il proprio stato sulla base dei parametri ricevuti e le view aggiornano ció che viene mostrato. Le view stesse possono peró scaturire azioni e far ripartire il ciclo: nella realtá la maggior parte delle azioni sono spesso scaturite dalle view grazie alle interazioni dell'utente.

Un esempio di quanto appena descritto con il pattern Flux, si puó trovare nell'applicazione all'aggiunta di un nuovo sensore. Qui sará richiamata una action da parte della view, che solo dopo aver contattato il database secondario per salvare i dati aggiornerá attraverso un dispatcher lo stato della lista di sensori presenti. Infine la view, alla quale é affidato il compito di mostrare solamente la lista di sensori, potrá visualizzare il nuovo sensore senza che le venga direttamente passato come oggetto.

Segue nel frammento di codice "Listing 4.5" un esempio di quanto detto all'interno del quale sará possibile vedere chiamata la funzione "**dispatch**", la quale rappresenta il dispatcher spiegato poco fa.

Listing 4.5: Funzione create sensor

```
export const createSensor = (userId, id, name, description)
   => {
  return async (dispatch, getState) => {
    const sensors_list = getState().sensors.sensors_list;
    sensors_list.forEach((sns) => {
      if (sns.id === id)
        throw new Error('This Sensor already exists');
    });
    const data = JSON.stringify({name: name, description:
       description});
    await db.ref(`sensors/${userId}/${id}`).push(data);

    const sensor = new Sensor(id, name, description, 0);
    dispatch({ type: CREATE_SENSOR, sensor: sensor });
    saveDataToStorage(id, name, description);
};};
```

Di seguito invece, nel frammento di codice "Listing 4.6", viene mostrato come questa funzione venga fatta partire dalla view dopo la pressione del tasto di creazione del sensore.

Listing 4.6: Pressione del tasto di creazione sensore

```
const createNewSensorHandler = useCallback(async () => {
  setError(null);
  setIsLoading(true);
  if (!isSensorValid) {
    setError('Invalid Sensor ID');
  } else {
      try {
        const userData = await AsyncStorage.getItem('userData
           ');
        const transformedData = JSON.parse(userData);
        const { userId } = transformedData;
        await dispatch(createSensor(userId, newSensorId,
           newSensorName, newSensorDescription));
      } catch (err) {
          setError(err.message);
      }
      setIsAddingSensor(false);
  }
  setIsLoading(false);
}, [dispatch, newSensorId, newSensorName,
    newSensorDescription, isSensorValid]);
```

All'interno dell'applicazione si é successivamente reso necessario porre attenzione a due punti cardine, il primo nella schermata generale dei sensori, mentre il secondo nella schermata relativa al singolo sensore. Qui, per avere gli aggiornamenti in tempo reale dei dati rilevati si é reso necessario mandare in esecuzione continua una fetch dei dati verso la API discussa precedentemente. Non potendo eseguire una funzione a ciclo infinito per evitare di rallentare l'applicazione e di riempire di richieste la API, lo stratagemma adottato é stato temporizzare la funzione ogni 15 secondi.

Dal momento che React Native consente di creare applicazioni che vivono all'interno di una Activity unica, si é reso necessario controllare l'esecuzione della funzione in maniera subordinata all'evento "isFocused" della schermata che la esegue. Questo evento, come suggerisce il nome, consente di eseguire la funzione solamente quando la schermata é visualizzata e di non continuare ad inviare richieste mentre l'utente é in un'altra pagina.

Il contrario avrebbe portato problemi quando l'utente fosse passato dalla schermata generale dei sensori alla pagina del singolo, infatti sarebbero state continuamente inviate richieste simultanee dalle funzioni di entrambe le pagine, creando problemi di sovraccarico verso la API che avrebbe quindi avuto tempi di risposta piú lunghi.

Nel frammento di codice "Listing 4.7" segue il codice utilizzato nella pagina generale dei sensori per aggiornare lo stato del valore di rilevazione di ogni singolo sensore.

Listing 4.7: Funzione ciclica di aggiornamento delle rilevazioni dei sensori

```
useEffect(() => {
  if (isFocused) {
    const intervalId = setInterval(() => {
      updateSensorHandler();
    }, 15000);
    return () => clearInterval(intervalId);
  }
}, [isFocused]);
```

Capitolo 5

Validazione

Per la validazione dei dati raccolti dal sensore é stato scelto di utilizzare come campione un ambiente domestico che nell'arco della giornata é solitamente soggetto ad ospitare flussi di persone e tempi di permanenza differenti in base all'orario.

Un fattore chiave da tenere in considerazione per poter analizzare i dati in maniera corretta, é la morfologia dell'ambiente usato come campione. Questo, infatti, consiste in una stanza avente un'area di circa 12m^2, per il 30% circa ricoperta da mobilio. Inoltre la stanza scelta dispone di una finestra avente le dimensioni di circa 150x180 cm, dimensioni che offrono la possibilitá di sfruttare un ricircolo dell'aria interna molto veloce non appena la finestra viene aperta.

Per riuscire ad interpretare i dati in maniera corretta, si noti che i test effettuati per la validazione dei dati si svolgono in una giornata di inverno, dove quindi si é piuttosto restii a rimanere molto tempo con le finestre completamente spalancate per via delle temperature. Infine si tenga presente che il posizionamento del sensore in prossimitá della finestra ha influito nella raccolta dei dati.

Si discuteranno in questo capitolo quattro scenari relativi a differenti momenti della giornata nei quali il livello di CO2 ha avuto comportamenti diversi. Durante le fasi di monitoraggio, osservazione, formattazione ed analisi dei dati é stato utilizzato come strumento "**Grafana**"[25], un'applicazione web per la visualizzazione e l'analisi interattiva dei dati raccolti in un database.

5.0.1 Scenario 1: Il primo picco

Il primo scenario presentato ha luogo la mattina presto, tra le 6:30 e le 7:00. In questa frazione di tempo il livello di CO2 nella stanza raggiunge circa le 750 ppm, valore dovuto allo scarso ricircolo dell'aria durante la notte, momento in cui la finestra e la porta restano chiuse.

Non appena una persona entra nella stanza e vi resta per una mezz'ora circa, é possibile notare dal grafico "Figura 5.1" come il livello di CO2 aumenti vertiginosamente fino a sfiorare le 900 ppm. Prima di uscire dalla stanza, questa viene fatta arieggiare aprendo un'anta della finestra con la tapparella abbassata e gli spazi infra doghe aperti. É possibile notare che subito dopo aver lasciato la stanza ed aperto la finestra, il livello di anidride carbonica inizia a scendere lentamente. Nel grafico di seguito é possibile monitorarne l'andamento fino alle 9:00.

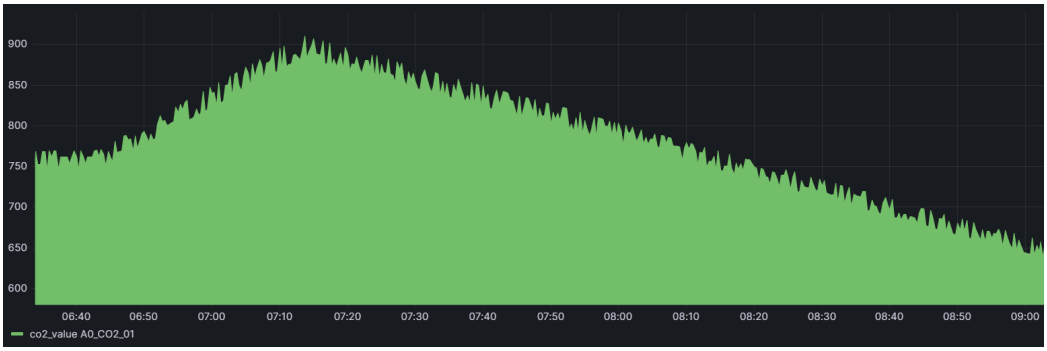

Figura 5.1: Rilevazioni di CO2: scenario 1

5.0.2 Scenario 2: Il secondo picco

Come secondo scenario si é scelto di analizzare il periodo di tempo subito successivo al precedente, dal quale é possibile osservare un distaccamento temporale di solamente mezz'ora.

In questa fase della mattina la finestra della stanza viene chiusa ed una persona resta nella stanza per un lasso di tempo di circa tre ore, nelle quali anche la porta della stanza rimane chiusa. É possibile notare come in questo momento della giornata il livello di anidride carbonica rilevato dal sensore sia di nuovo in aumento, fino a sfiorare un livello ben oltre la soglia di sicurezza di 1000 ppm.

Dopo aver ricevuto diverse notifiche dall'applicazione mobile, la persona nella stanza decide di aprire la finestra (questa volta con la tapparella alzata). Si puó notare la differenza manifesta nel livello di CO2 tra la ventilazione della stanza con tapparella abbassata e alzata. In questo scenario, infatti, il livello di anidride carbonica si riabbassa drasticamente in meno di mezz'ora, scendendo da circa 1100 ppm a circa 700 ppm.

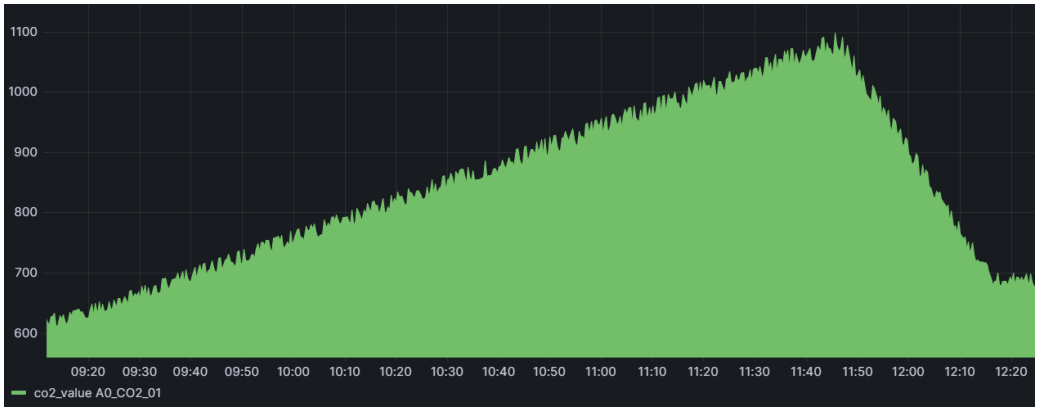

Figura 5.2: Rilevazioni di CO2: scenario 2

5.0.3 Scenario 3: La stanza vuota

Dopo aver fatto arieggiare efficacemente la stanza alla fine dello scenario precedente, riportando il livello di CO2 a valori intorno alle 650 ppm, in questa fase centrale della giornata l'intera abitazione rimane vuota, con alcune finestre leggermente aperte e tutte le porte spalancate.

Per analizzare il comportamento del livello di CO2 in questa fase del giorno, si é deciso di lasciare la porta aperta ma la finestra chiusa nella stanza in esame. Come é possibile notare dal grafico "Figura 5.3", il livello di anidride carbonica continua a scendere nelle quasi cinque ore successive allo scenario precedente, fino a raggiungere le 600 ppm circa.

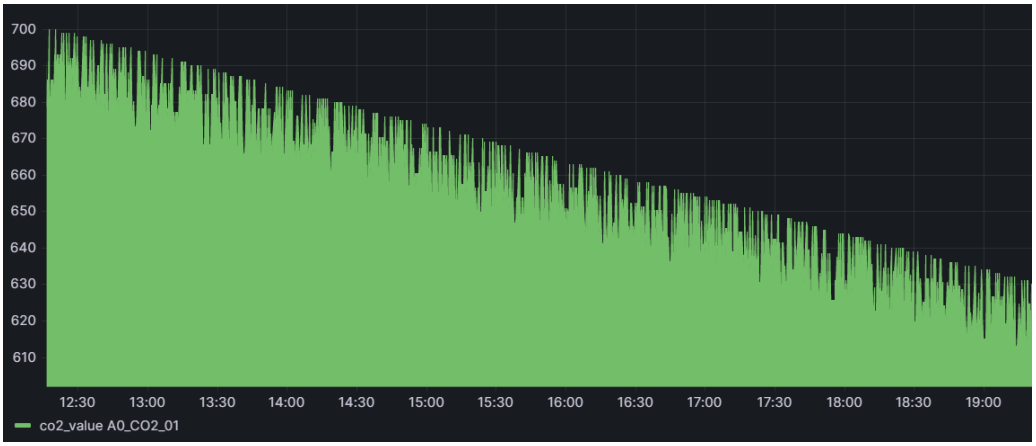

Figura 5.3: Rilevazioni di CO2: scenario 3

5.0.4 Scenario 4: La sera e la notte

Al rientro delle persone in casa, la stanza in esame viene utilizzata da una di esse per un lasso di tempo analogo a quello mattutino, non si assiste peró allo stesso innalzamento del livello di CO2 avuto nello scenario 1, infatti da 650 ppm circa verrá portato a 750 ppm circa. Le differenze presenti con la mattina sono da ricercarsi nel fatto che le finestre e le porte di casa sono

ancora aperte dal pomeriggio e sul momento viene aperta anche la finestra della stanza in esame per un lasso di tempo breve.

Quanto mostrato di seguito é invece il momento subito successivo a quello appena descritto, dopo cena infatti la stanza viene utilizzata con finestra socchiusa e porta chiusa da due persone per un lasso di tempo di circa due ore dalle 22:00 alle 00:00. É possibile notare nel grafico in "Figura 5.4" l'aumento del livello di CO2 durante il periodo di tempo descritto.

Seppur all'inizio della permanenza di queste due persone nella stanza il livello di CO2 sembra rimanere moderato per via della finestra socchiusa, passata circa mezz'ora comincia ad aumentare fin' oltre il limite di 1000 ppm, arrivando persino a circa 1200 ppm.

Rispetto alla velocitá di aumento del livello di anidride carbonica rilevata nello scenario 1, in questo caso si assiste ad un aumento meno vertiginoso nonostante la stanza ospiti il doppio delle persone. Il fenomeno osservato é imputabile con alta probabilitá alla finestra che permette comunque il passaggio dell'aria ed il ricircolo con l'esterno.

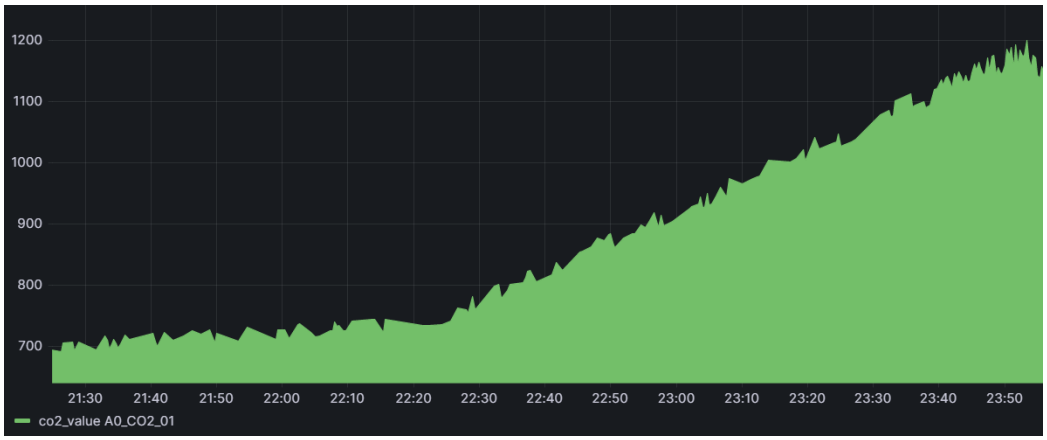

Figura 5.4: Rilevazioni di CO2: scenario 4 durante la serata

L'ultima azione compiuta prima di lasciare la stanza fino al mattino seguente consiste nell'aerazione della stanza stessa. Dopo aver raggiunto infatti un livello critico di anidride carbonica, si é deciso di spalancare la finestra per cercare di abbattere velocemente il livello di anidride carbonica raggiunto.

Lasciando la stanza libera con porta e finestra aperte per circa un'ora si é infatti riusciti a portare il livello di CO2 da circa 1200 ppm a quasi la metá, ovvero circa 700 ppm. In questa fase la finestra aveva la tapparella abbassata ma tutti gli spazi infra doghe aperti per consentire il passaggio dell'aria, questo ha consentito un discreto ricircolo di aria nonostante si sia reso necessario un tempo piuttosto lungo per ristabilire un livello di anidride carbonica accettabile.

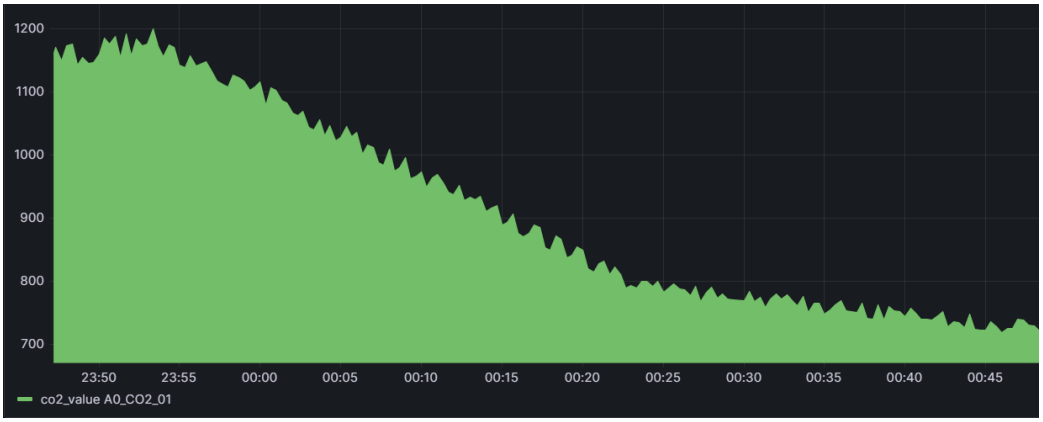

Figura 5.5: Rilevazioni di CO2: scenario 4 prima della notte

5.0.5 Note sulla validazione

É infine opportuno tenere presente che i dati mostrati precedentemente sono stati rilevati attraverso l'utilizzo di un sensore a basso costo, il quale risente molto di agenti esterni quali temperatura ed umiditá.

Inoltre, la stanza utilizzata come campione di test per la validazione dei dati ha un termosifone posto sotto la finestra, quindi nelle vicinanze del

sensore, provvisto inoltre di una vaschetta di acqua per mantenere umida l'aria all'interno della stanza. Questo potrebbe essere un motivo ulteriore a spiegazione del calo di precisione mostrato dal sensore.

Il dispositivo costruito é quindi stato testato in un ambiente non ottimale, questo ha dato origine alla raccolta di dati che pur fornendo un'idea dell'andamento della situazione reale, non riescono a descrivere con elevata precisione la realtá stessa.

Come é possibile notare dai grafici della sezione precedente, ogni settore presenta infatti un andamento quasi lineare durante le variazioni in calo e crescita del livello di CO2. Questo é imputabile alla piuttosto bassa sensibilitá del sensore, che oltre a mostrare una variabilitá notevole nelle misurazioni all'interno di intervalli di tempo molto piccoli, presenta criticitá di funzionamento dovute a surriscaldamento dopo ore di utilizzo continuo. É accaduto spesso che variazioni notevoli del livello di CO2, ad esempio dovute alla presenza di piú persone nella stanza, portassero il sensore a sovrastimare il livello toccando anche valori piú alti di quelli mostrati di circa 500 ppm, oppure che venissero mostrate reazioni tardive ai cambiamenti del livello di CO2 con il sensore surriscaldato.

Capitolo 6

Conclusioni e sviluppi futuri

La tesi ha presentato un sistema di monitoraggio della qualitá dell'aria interna, focalizzato sulla rilevazione dei livelli di anidride carbonica (CO2) nell'ambiente. Il dispositivo di rilevamento é stato progettato come parte dell'ecosistema dell'Internet of Things (IoT), un campo in rapida evoluzione che sta rivoluzionando molti settori, tra cui quello della sensoristica.

Il sistema presentato é composto da un dispositivo di rilevamento CO2, un server di raccolta dati, una API per la comunicazione con il database e un'applicazione per smartphone. L'applicazione mobile si occupa di inviare notifiche quando il valore di CO2 supera la soglia di sicurezza e fornisce la possibilitá di consultare i dati relativi ai dispositivi associati in tempo reale o uno storico di 30 giorni.

Nei capitoli precedenti, dopo una panoramica sull'attuale stato dell'arte nel mondo dell'IoT, dei sensori di rilevamento CO2 e dei progetti ad essi collegati, sono state descritte le fasi riguardanti progettazione e implementazione dell'intero sistema. In particolare, sono state discusse le scelte progettuali effettuate, nonché le tecnologie utilizzate per ogni componente del sistema. Sono infine stati presentati i dati rilevati dal sensore e il processo di validazione effettuato.

Il lavoro svolto per la creazione dell'intero sistema ha visto superati gli scogli maggiori all'inizio, nello specifico durante la fase di progettazione del dispositivo e dello standard utilizzato per la nomenclatura dei sensori. Queste fasi citate hanno richiesto infatti di avere chiaro in mente cosa si volesse realizzare sin dal principio al fine di garantire uniformitá tra le differenti fasi del processo di sviluppo. Non si poteva infatti correre il rischio di perdere dati a causa di una riformulazione della nomenclatura in corso d'opera, questo avrebbe impossibilitato il server nel riconoscere la stringa ricevuta contenente la misurazione effettuata dal dispositivo.

Il prodotto presentato ad oggi soffre di qualche carenza dovuta principalmente all'hardware scelto, é quindi da intendersi come un prototipo da sviluppare in futuro. Piú nel dettaglio sará necessario sostituire il sensore utilizzato con altri piú costosi ma molto piú affidabili. Inoltre allo stato attuale il dispositivo richiede una presa di corrente per essere utilizzato, ma questo non va incontro all'obiettivo che si prepone il progetto, ovvero di avere un dispositivo che sia di facile impiego. É in corso di sviluppo una prima evoluzione del dispositivo attuale che prevede il funzionamento attraverso una batteria, cosí da poter essere posizionato ovunque in casa.

Per quanto riguarda il software sono invece previste meno modifiche, una peró si rende necessaria per facilitare l'impiego del dispositivo e renderlo a tutti gli effetti utilizzabile da utenti qualsiasi. Ad oggi infatti il dispositivo puó essere impostato solamente attraverso il codice sorgente per accedere ad una rete Wi-Fi e svolgere le sue funzioni. Sfruttando la connessione Wi-Fi interna del microcontrollore si potrá implementare una funzionalitá aggiuntiva rispetto alle attuali per permettere l'associazione ad una rete Wi-Fi attraverso web browser: permettendo cosí ad ogni utente di poter impostare totalmente il proprio sensore per utilizzarlo in autonomia.

In conclusione si ritiene che il lavoro svolto possa permettere lo sviluppo

di un piú ampio ecosistema di sensori per la casa, come descritto nel capitolo terzo "Progettazione", integrando rilevazioni di diverso tipo ed offrendo la possibilitá di controllare in remoto lo stato dell'abitazione. Il lavoro svolto nelle fasi progettuali ed implementative apre infatti le porte a sistemi piú complessi senza necessitá di riprogettazione, poiché l'attenzione é stata fin da subito posta sulla possibilitá di ampliare il sistema in costruzione.

Bibliografia

[1] Covid-19: Guide on Ventilationand CO2monitoring - Unite the Union

[2] Indoor environment monitoring as a measure to reduce epidemic spreading - Jana Loosová, Miloš Hernych

[3] Qualitá dell'Aria negli Edifici Scolastici. WP3. Indagine sullo stato dell'arte e del mercato - Chiara Ugolini, Annamaria Belleri

[4] React Native
https://reactnative.dev/

[5] TypeScript
https://www.typescriptlang.org/

[6] JavaScript
https://eloquentjavascript.net/

[7] Postman
https://www.postman.com/

[8] Netlify functions
https://www.netlify.com/products/functions/

[9] Redux
https://redux.js.org/

[10] Flux pattern
https://facebook.github.io/flux/

[11] CO2 Sensors: Which Type Should You Be Looking For? - Dianna Smith
https://learn.kaiterra.com/en/air-academy/carbon-dioxide-sensors#:~:text=While%20the%20other%20kinds%20of,portable%2C%20accurate%20carbon%20dioxide%20monitoring.

[12] How to choose the right CO2 sensors
https://www.pressac.com/insights/how-to-choose-the-right-co2-sensors/

[13] Gravity: PWM Infrared Carbon Dioxide Sensor (400-5000 ppm)
https://www.dfrobot.com/product-1549.html

[14] Gravity: Analog Electrochemical Carbon Dioxide Sensor (0-10000 ppm)
https://www.dfrobot.com/product-1023.html

[15] Flux In-Depth Overview
https://facebook.github.io/flux/docs/in-depth-overview

[16] Amazon
https://www.amazon.it/

[17] Immuni
https://www.immuni.italia.it/

[18] Sap - Cos'é l'IoT e come funziona?
https://www.sap.com/italy/insights/what-is-iot-internet-of-things.html

[19] Oracle - Che cos'é l'IoT?
https://www.oracle.com/it/internet-of-things/what-is-iot/

[20] Amazon AWS - Cos'é l'IoT?
https://aws.amazon.com/it/what-is/iot/

[21] Everything You Need To Know About Carbon Dioxide (CO2) - Dianna Smith

https://learn.kaiterra.com/en/resources/carbon-dioxide-co2?hsCtaTracking=14be5fee-2dab-4872-8681-d41a98eccee6%7C3856dec9-e685-435b-9f08-9d19b3010b63

[22] Kaiterra Sensedge
https://www.kaiterra.com/sensedge

[23] Renkeer - CO2 Sensors: Definition, Types, And How To Choose?
https://www.renkeer.com/co2-sensors-types-and-choose/

[24] Vemer Fiuto
https://www.vemer.it/it/catalogo/ga_e_sicurezza/rivelatori_qualita_aria/da_incasso/fiuto_VE788100

[25] Grafana
https://grafana.com/

[26] MQTT
https://mqtt.org/

[27] What is MQTT and Why it is Important for the Internet of Things - Alexis Leibbrandt
https://akenza.io/blog/what-is-mqtt?utm_term=&utm_campaign=Dynamic+Search+Campaign+-+EUutm_source=adwords&utm_medium=ppc&hsa_acc=7184580909&hsa_cam=19248438136&hsa_grp=144426835677&hsa_ad=641301872801&hsa_src=g&hsa_tgt=dsa-19959388920&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gclid=EAIaIQobChMIg9Pe68S6_QIVDtUYCh07sgF0EAAYASAAEgKZGvD_BwE

Ringraziamenti

Ringrazio il professore Marco Di Felice per avermi accompagnato in questo percorso, dal tirocinio curricolare alla stesura della tesi. Non conoscevo il mondo della sensoristica, nessuna base di elettronica e nemmeno i microcontrollori prima che mi venissero proposti come argomento di tirocinio: oggi sono una mia grande passione, e mi hanno portato a trovare il mio primo lavoro.

Ringrazio anche i miei colleghi ed amici conosciuti in universitá, il gruppo della "prima fila", coi quali ho condiviso un lungo e bel percorso, imparando a fare squadra a lezione, nella preparazione degli esami e fuori dal contesto universitario.

Ringrazio i miei genitori, che mi hanno supportato e spronato a portare a termine il mio percorso di studi nonostante abbia diverse volte vacillato nell'intenzione di completarlo.