

**DIPARTIMENTO DI INGEGNERIA NAVALE, ELETTRICA, ELETTRONICA E DELLE TELECOMUNICAZIONI**

**CORSO DI STUDIO IN INGEGNERIA ELETTRONICA E TECNOLOGIE DELL’INFORMAZIONE**

Tesi di Laurea Triennale

Ottobre 2020

**Progetto e implementazione di un sistema embedded per il controllo di presenza da remoto**

Candidato: Laura Triglia

Sebastiano Gioia

Relatore: Prof. Riccardo Berta

***A Nonno Lillo, Nonna Rina e Nonna Gra***

**Sommario**

Lo scopo di questo lavoro di tesi è quello di illustrare il sistema progettato per monitorare una stanza e notificare eventuali nuove presenze nella stanza stessa. I rilevamenti di presenza collezionati vengono mandati all’API Measurify, tramite la quale uno smartphone, ad essa opportunamente associato, viene notificato della nuova presenza nella stanza. Il progetto include anche la realizzazione di un’applicazione smartphone per la visione delle stesse notifiche. L’applicazione è stata creata tramite Android Studio con il tool Flutter.

Per la realizzazione del progetto è stata utilizzata una scheda Arduino, un sensore PIR per la rilevazione di presenze nella stanza, la libreria EdgeEngine che aiuta ad interagire la scheda Arduino con il cloud di Measurify. È stata scelta una piattaforma Arduino perché è un buon sistema embedded per la realizzazione di questo sistema.

1. **Introduzione**

La realizzazione di questo progetto nasce dalla necessità di monitorare e notificare la presenza in una stanza tramite un semplice sistema di allarme. L’idea di base era quella di avere un sistema in grado di notificare in tempo reale una nuova presenza nella stanza designata. Da qui è nata l’idea dell’implementazione di un modulo capace di trasmettere le informazioni dal nostro sistema hardware al telefono cellulare associato. Il tutto è gestito dall’API Measurify, il cuore del progetto attorno al quale sono state costruite tutte le implementazioni che verranno mostrate. Un’API (application program interface) è un insieme di funzionalità atte all’espletamento di dato compito. Con l’utilizzo di un’API la finalità è quella di ottenere un’astrazione di più alto livello, di solito tra l’hardware e il programmatore o tra il software basso e quello ad alto livello semplificando così il lavoro di programmazione. Proprio perché il compito di Measurify è quello di collezionare i campioni di un dispositivo di misura, posto in una stanza, è stato scelta questa API.

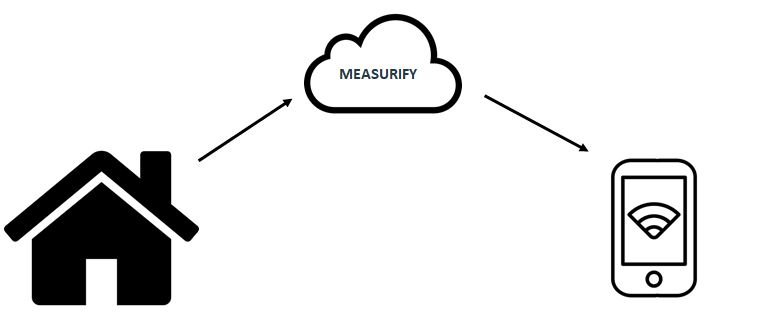
Il primo obiettivo del progetto era quello di avere una comunicazione tra il sistema hardware, formato dalla scheda Arduino e il sensore PIR, con Measurify tramite protocollo http. Per fare ciò è stato pensato di utilizzare la libreria EdgeEngine, nella programmazione del codice della scheda Arduino. EdgeEngine è una libreria frutto di un lavoro di tesi a cura di Luca Lazzaroni e Andrea Mazzara, da cui è nata una collaborazione. In tale collaborazione è stata sperimentata la funzionalità di EdgeEngine, come utilizzatori finali. Ulteriori dettagli verranno spiegati in seguito.

Altro punto cardine di questo progetto è stato l’utilizzo del tools Flutter combinato con la tecnologia di Google, “Firebase Cloud Messagging”. Flutter ci ha permesso di costruire l’applicazione, la quale, collegata con Measurify, crea il collegamento tra l’hardware impiegato nel rilevamento della presenza e la ricezione delle notifiche sul telefono associato. Nel concreto si è scelto quindi di implementare nell’applicazione, grazie a Firebase Cloud Messagging, le notifiche push, così da avere una simulazione il più possibile fedele a una situazione reale. La scelta delle notifiche push, ovvero notifiche che arrivano anche se l’applicazione è in background, è stata fatta proprio per rendere il sistema di allarme il più efficace possibile. È ovviamente di interesse avere un allarme che ci notifica immediatamente nel momento in cui è stata rilevata una presenza nella stanza. Sono due le parole chiave che hanno legato ogni aspetto di questo progetto:

* Semplicità
* Completezza

L’idea del progetto è nata con l’intento di creare un sistema in grado di avere molteplici funzionalità, tali per cui il sistema d’allarme potesse essere definito completo, ma che al contempo rimanesse semplice e accessibile. Anche le scelte riguardo la parte sensoristica, che verranno spiegate in seguito, hanno come idea comune quella di essere

Il sistema verrà ampliamente discusso e dettagliato nei prossimi capitoli.



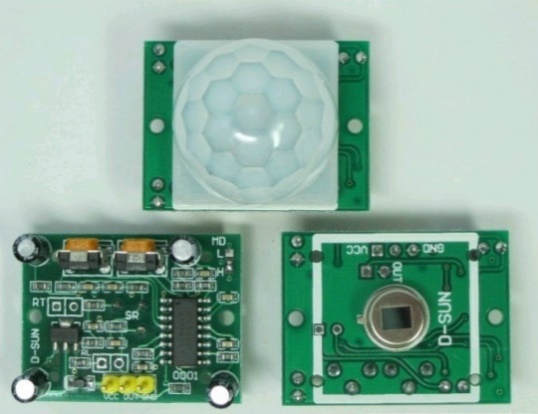
*Figura 1*: Schema riassuntivo dello schema

1. **Metodi e strumenti utilizzati**

Affinché questo progetto fosse realizzabile, è stato necessario avvalersi di differenti strumenti sia hardware, sia software.

**Sensore PIR**

Iniziando dai componenti elettronici il sistema si basa su un rivelatore di presenza PIR come nella figura 1. Un sensore PIR, o sensore ad infrarossi passivo è un sensore elettronico che misura i raggi infrarossi irradiati dagli oggetti nel suo campo di vista. Il suo funzionamento è piuttosto semplice: tutti gli oggetti con temperatura superiore allo zero assoluto emettono energia sotto forma di radiazioni luminose; la maggior parte delle volte queste radiazioni sono invisibili all'occhio umano, poiché aventi una frequenza inferiore al nostro spettro di frequenze luminose visibile, ma possono essere rilevate tramite specifici dispositivi elettronici progettati a tal scopo. Il termine passivo si riferisce al fatto che i PIR non emettono energia in nessuna forma ma lavorano esclusivamente rilevando l’energia sprigionata dagli oggetti. È dunque sembrato il componente elettronico più adatto al sistema che stavamo progettando.



*Figura 2:* Sensore PIR

**Ambiente Arduino**

Il cuore hardware del progetto è basato sull’ambiente Arduino, il quale permette tramite le sue molteplici funzionalità di creare in modo relativamente semplice un sistema più complesso. Le schede compatibili con Arduino sono intuitive da utilizzare e il “Wiring”, il linguaggio di programmazione, si basa sul linguaggio C, (il codice viene programmato e installato sulla scheda utilizzando Arduino IDE, il quale è open-source), dunque la scelta di utilizzare questo ambiente risulta la più semplice da utilizzare, nonché la più completa.



*Figura 3*: Ambiente Arduino

**Scheda Wi-Fi**



*Figura 4:* Scheda Wi-Fi ESP32

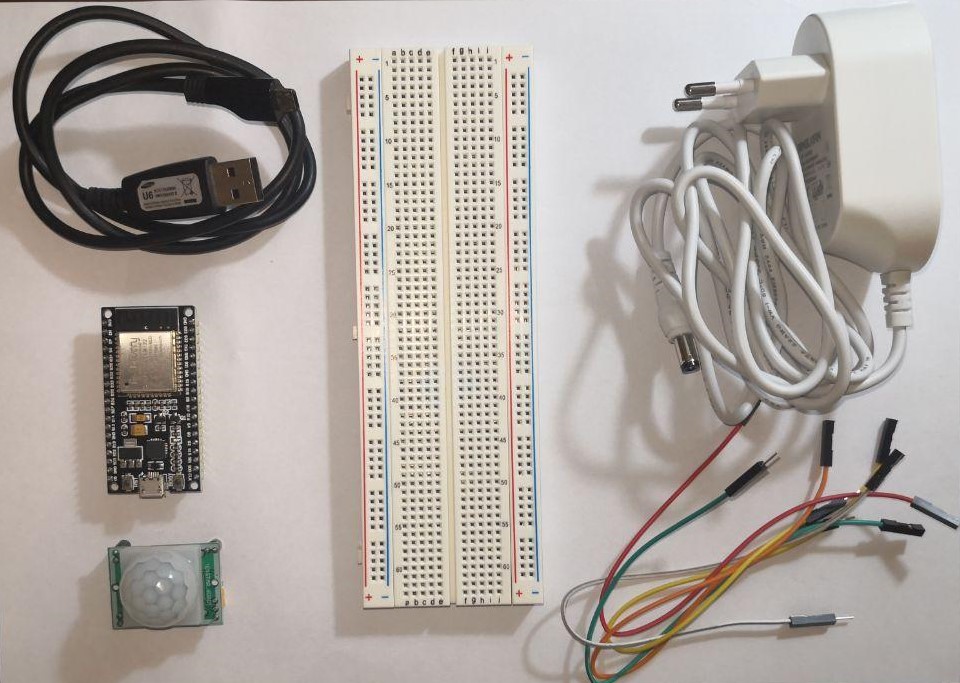
Inizialmente si era pensato di utilizzare la scheda Arduino Uno, che tuttavia essendo sprovvista di un modulo WI-Fi non soddisfa i requisiti per il sistema da progettare. La connessione internet è fondamentale per la comunicazione tra hardware e Measurify. Per la lettura del sensore è necessario avere una scheda avente dei pin digitali, sui quali effettuare la lettura. La scelta è dunque ricaduta su una scheda Wi-Fi basata su ESP32 che integra un modulo per comunicare con protocollo Wi-Fi 802.11 b/g/n, un modulo per comunicare via Bluetooth dual-mode (classico e BLE) e 34 pin digitali. Supporta una velocità di trasmissione di dati fino a 150 Mbps, ha una potenza di uscita sull’antenna pari a 20,5 dBm per garantire la massima portata. Dispone inoltre di interfacce per sensori di temperatura, touch sensor, SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR, connettore micro USB, pulsante di Boot e pulsante di reset.

**Componenti elettronici di base**

I componenti appena mostrati sono i principali componenti elettronici necessari allo sviluppo del progetto, ma non solo i soli. È stato ovviamente necessario l’utilizzo di ulteriori componenti quali

* Una breadboard, strumento utilizzato per creare prototipi di circuiti elettrici. A differenza della basetta millefori, che è base di rame con tanti fori (da qui il nome) su cui vengono saldati i componenti e i collegamenti che formano il prototipo, la breadboard non richiede saldature ed è completamente riutilizzabile poiché è strutturata in modo da avere delle linee di continuità che servono a collegare i componenti in modo “plug and play”.
* Un alimentatore per poter fornire l’energia necessaria al sistema
* Jumper maschio-femmina

Tutti gli strumenti hardware sopracitati sono mostrati in figura 5.



*Figura 5*: Componenti hardware del sistema

**Measurify**

Dal punto di vista software il punto focale del progetto è basato sull’uso dell’API Measurify. Tale API, creata in laboratorio dal Prof. Riccardo Berta, si basa su alcuni concetti fondamentali:

* Thing: si tratta di una “cosa” (una persona, un ambiente, un oggetto, ecc) per la quale si sta facendo una misurazione. Nel caso specifico di questo progetto la thing corrisponde alla stanza dove si trova il sensore.
* Feature: riguarda ciò che viene misurato (la temperatura, l’umidità, il peso, ecc). Nel caso specifico del nostro progetto si tratta dell’ingresso nella stanza, caratterizzato dal
* Device: si tratta di un dispositivo (hardware/software) che può misurare una certa dimensione (feature) su una determinata cosa (thing). Nel caso specifico di questo progetto il device è il sensore PIR che rileva la presenza nella stanza
* Measurement: si tratta di una misura effettuata da un dispositivo (device) per una certa dimensione (feature) su una determinata cosa (thing).

**Struttura del codice Arduino**

Ogni programma Arduino deve implementare due metodi principali, quello di setup() e quello di loop():

* Nella funzione di setup() viene implementato tutto il codice inerente alla dichiarazione dei pin e il settaggio dei parametri richiesti dalle librerie per il loro funzionamento.
* Nella funzione di loop() viene implementato il codice inerente al compito che deve adempiere il microcontrollore

**Implementazione della libreria EdgeEngine**

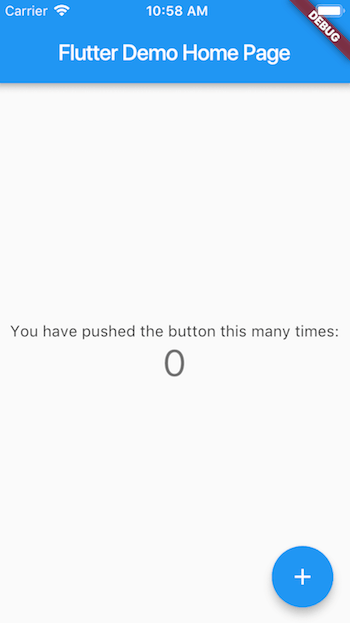
La connessione tra il cloud di Measurify e l’hardware è stato facilitato dall’implementazione nel codice Arduino della libreria EdgeEngine. Rispetto a quanto detto sopra:

* Nella funzione di setup() vengono settati i parametri per le funzionalità di librerie, come il login, la connessione internet, ecc.
* Nella funzione di loop() viene programmata la procedura per la connessione a Measurify, nonché l’invio dei campioni.

**Flutter application**

Un ulteriore strumento utilizzato è stato Flutter, con lo scopo di creare un’applicazione in grado di ricevere le notifiche in tempo reale e connettersi a Measurify. Flutter è un framework open-source creato da Google per la creazione di interfacce native per iOS e Android. Flutter ha quattro componenti principali, che costituiscono la sua struttura:

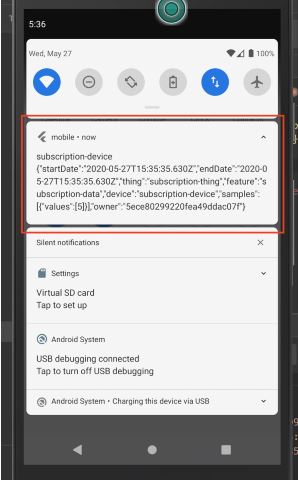
* Dart Platform: è un linguaggio di programmazione sviluppato da Google. Il compilatore Dart permette di scrivere programmi sia per il web sia per desktop e server
* Flutter Engine: è scritto principalmente in C++, fornisce supporto per il rendering a basso livello utilizzando la libreria grafica di Google, Skia Graphics. Inoltre, si interfaccia con SDK della piattaforma specifica come quelli di Android o iOS. Una particolarità molto apprezzata del Flutter engine, grazie al codice scritto in Dart, è quella di poter effettuare un "hot-reload" dell'applicazione dove la modifica del codice viene iniettata immediatamente all'interno dell'applicazione così da visualizzare all'istante le modifiche effettuate senza un riavvio completo o un cambio di stato.
* Foundation Library: scritta in Dart, fornisce classi e funzioni di base utilizzate per costruire applicazioni che utilizzano Flutter, come le API per comunicare con l'engine.
* Widgets: la progettazione dell'interfaccia utente in Flutter prevede l'assemblaggio e/o la creazione di vari widget. Un widget in Flutter rappresenta una descrizione immutabile dell'interfaccia utente; grafici, testo, forme e animazioni vengono creati utilizzando i widget. Inoltre il framework di Flutter contiene due set di widget conformi a specifici linguaggi di programmazione. I widget in stile Material Design implementano il design di Google, mentre i widget in Cupertino imitano il design iOS di Apple.



*Figura 6*: Esempio Flutter App

**Firebase Cloud Messagging**

Infine l’ultimo strumento utilizzato, non per importanza, è stato il tool di Google Firebase Cloud Messaging. Tale tool ha permesso di implementare le notifiche push, necessarie per avere un riscontro in tempo reale delle presenze collezionate. Firebase Cloud Messaging è una soluzione di messaggistica multipiattaforma che consente di inviare messaggi in modo affidabile senza alcun costo. Utilizzando Firebase Cloud Messagging, è possibile notificare a un'app client che sono disponibili nuovi messaggi di posta elettronica o altri dati per la sincronizzazione. Nel caso specifico di questo progetto è stato necessario utilizzare questo servizio per avere in tempo reale, tramite messaggi inviati da Measurify, la notifica che era stata registrata una nuova presenza nella stanza.

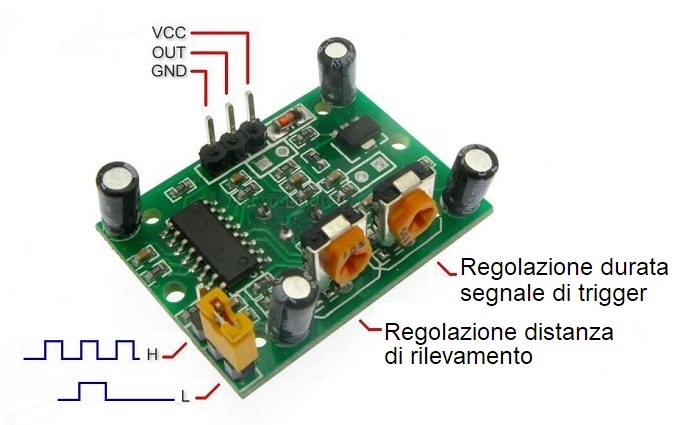


*Figura 7*: Esempio ricezione notifiche

1. **Sperimentazione e risultati**

Il lavoro di sperimentazione è partito dal progetto hardware. Difatti il primo punto da sperimentare è stato proprio la comunicazione tra il sensore e Measurify. Partendo dall’aspetto sensoristico è stato scelto di settare alcune variabili del sensore PIR. Difatti il sensore stesso può essere settato in diversi modi grazie ai due trimmer integrati. Tramite il primo è possibile modificare la sensibilità alla distanza che varia da 3 metri fino a 5 metri, mentre con il secondo è possibile modificare il ritardo del segnale, ovvero decidere per quanto il segnale rimanga alto anche dopo che il movimento è terminato: in questo caso il range va da pochi secondi fino a 5 minuti. Il ritardo del segnale può essere ulteriormente personalizzato, modificando il ponticello che è presente insieme ai tre connettori. Sono presenti due posizione:

* Nella posizione “L” il segnale parte immediatamente dopo il primo rilevamento e non viene più aggiornato fino a quando il timer non scade, portando il segnale a 0V. Se successivamente vengono rivelate altre presenze, il timer riparte.
* Nella posizione “H” re-inizializza il timer ogni volta che viene rilevato un movimento, quindi se viene eseguito un movimento continuo davanti al sensore, il timer continuerà a ripartire ed unicamente alla fine quando non ci sono più movimenti manterrà il segnale alto per il tempo impostato



*Figura 8*: Settaggi sensore PIR

Per tale sperimentazione è stato scelto il livello low, considerando una situazione reale in cui una persona fa più di un movimento dinanzi a un sensore. Se fosse stata scelta la posizione high, sarebbe stato ipotizzato di utilizzare il sensore unicamente in una situazione dove la persona nella stanza fa un movimento nel raggio di azione del sensore una sola volta. È quindi sembrato più utile scegliere la posizione low, in modo da simulare nel modo più reale possibile una nuova presenza nella stanza. Una volta utilizzato un sensore PIR funzionante e settato secondo i parametri richiesti, il sistema di allarme agiva come era richiesto.

Il punto successivo da sperimentare era sicuramente la comunicazione tra sensore e API: era necessario trovare il modo di verificare che, tramite la scheda Wi-Fi, ogni presenza collezionata fosse registrata da Measurify e messa a disposizione tra le “measurement”. Per verificare che le presenze venissero effettivamente registrate e memorizzate si è rivelato utile l’uso Postman. Postman è una piattaforma di collaborazione per lo sviluppo di API. Le funzionalità di Postman semplificano ogni passaggio della creazione di un'API e ottimizzano la collaborazione in modo da poter creare API migliori, più velocemente. Inoltre tramite Postman era possibile fare tutte le chiamate a Measurify e così verificare tutte le informazioni necessarie alla realizzazione del sistema. Una volta realizzato il device, le chiamate effettuate a Measurify sono tutte di tipo POST, ovvero creano delle measurement. Per poter effettuare le chiamate è tuttavia necessario disporre di un token, che ha una durata di trenta minuti, dopodiché è necessario richiedere nuovamente il token. Tale token si ottiene tramite un POST particolare quale

POST {{url}}/v1/login

che ha il seguente corpo:

{

"username" : “presence-username",

"password" : "presence-password”

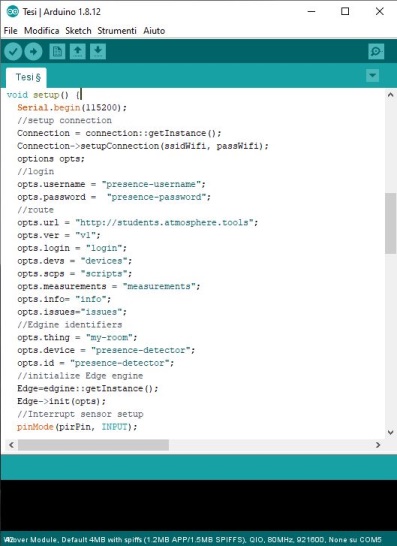
}

Il body è composto dalle credenziali che il professore Berta ha predisposto per l’utilizzo. Tale POST è stata inserita nel codice Arduino in modo che in maniera autonoma il dispositivo possa richiedere ogni volta un nuovo token, quando questo dovesse scadere. Ai diversi tipi di chiamata le API rispondono in JSON, linguaggio di programmazione dedicato al all’interscambio di dati fra applicazioni client/server. Nel caso della misura collezionata, rispondono con il JSON di tale misura o con un messaggio di errore se la chiamata non è stata formulata nel corretto modo. Un esempio concreto per visualizzare le misurazione precedenti tramite Postman è la seguente

GET {{url}}/v1/measurements?filter={"thing":”my-room"}&limit=10&page=1

che restituisce una pagina contenente dieci misure, effettuate sulla thing “my-room”. Questa chiamata è stata usata per verificare se il sistema hardware avesse correttamente inviato la misura, facendo quindi aggiornare la root delle “measurements”.

A questo punto l’utilizzo della libreria EdgeEngine è stato fondamentale, poiché ha permesso di concentrare il lavoro solo sulla scrittura del codice relativo al processo di collezione dei dati e alla verifica del suo funzionamento, piuttosto che sulla verifica della scadenza del token o sull’inserimento manuale delle presenze nelle “measurement”. Nel concreto tramite la libreria EdgeEngine è stato possibile settare le credenziali per l'accesso a Measurify; sarà la libreria stessa a svolgere il compito dell’aggiornamento del token. Come si può vedere nella figura 9, le credenziali vengono settate nel metodo di setup().



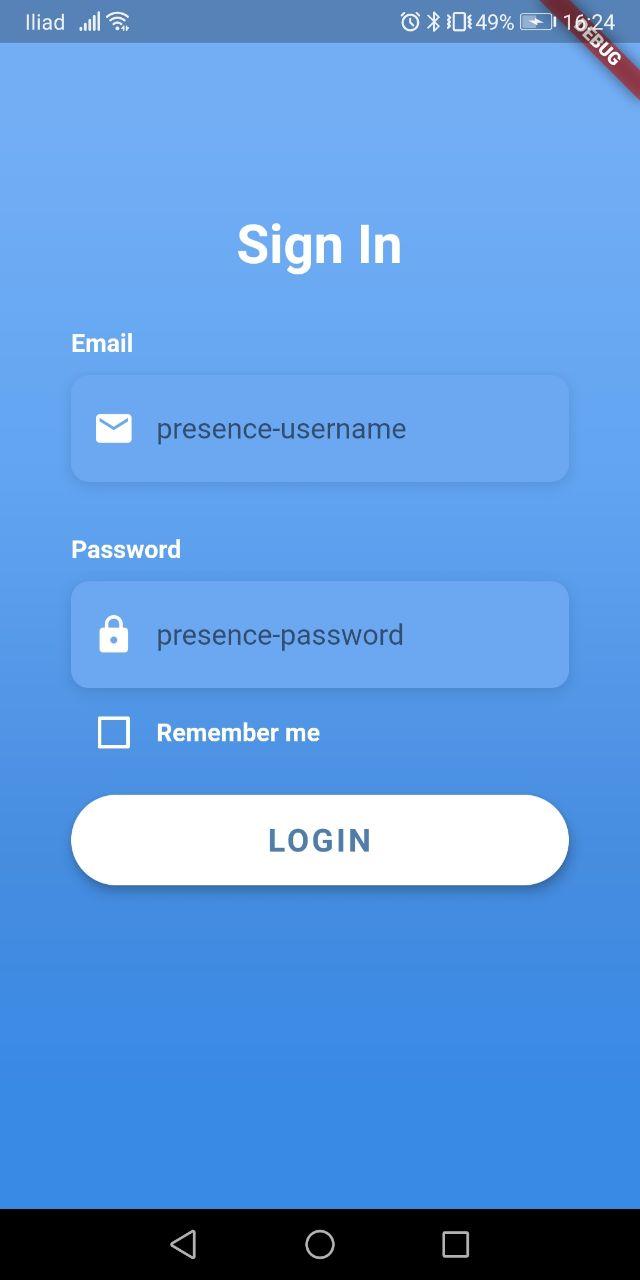
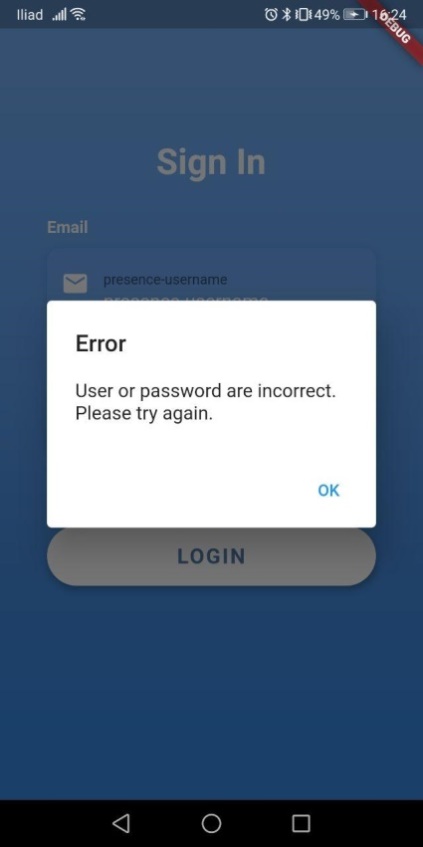
*Figura 9*: Settaggio Setup()

Una volta terminato il lavoro per quanto riguarda la parte sensoristica, è stato necessario concentrarsi sullo sviluppo di un’applicazione che permettesse di gestire il sensore a distanza tramite uno smartphone associato.

In questo passaggio l’utilizzo di Flutter combinato con l’ambiente di sviluppo Android Studio è stato fondamentale. Il punto di partenza è stato lo scheletro grafico dell’applicazione. Era di interesse che l’applicazione rispondesse a determinate funzionalità, quali la possibilità di accedere alla propria area riservata di Measurify, la possibilità di gestire una o più stanze e sicuramente la possibilità di ricevere notifiche in tempo reale. Individuate le funzionalità base che l’applicazione doveva avere, si è scelto di basare lo schema grafico su due livelli: uno dedicato al login per accedere all’area di Measurify e uno dedicato alle ulteriori funzionalità dell’applicazione stessa, quali la conoscenza di quale stanze siano attive e l’implementazione delle notifiche push tramite Firebase Cloud Messaging.

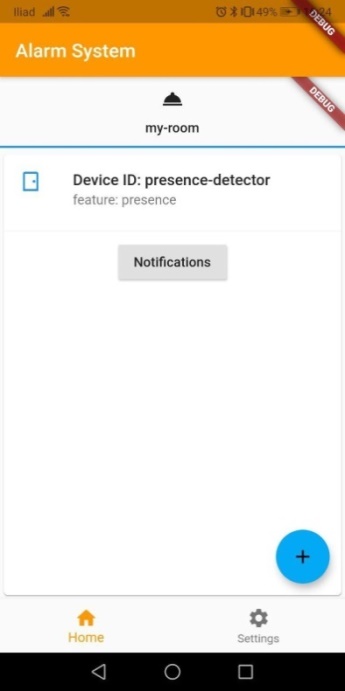
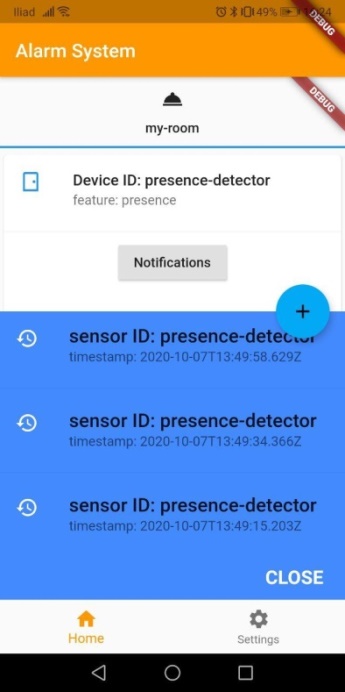
Il lavoro iniziale è stato quello di rendere l’applicazione funzionale da un punto di vista grafico e di riuscire a inquadrare lo schema generale per poter implementare tutte le funzionalità sopracitate. Si è scelto di utilizzare come stile grafico MaterialApp(), ovvero uno dei design che Flutter mette a disposizione e che somiglia a un applicazione in stile Android. Inoltre, l’applicazione è stata pensata su un punto di vista il più generale possibile, in modo che se nel futuro ci fossero delle modifiche al sistema, delle nuove implementazioni o delle nuove funzionalità, l’applicazione possa essere comunque utilizzata, a meno di piccole modifiche, ottenendo:

* Il primo livello, nonché la schermata di apertura, è composto da una schermata di login. Tale login permette l’accesso a Measurify, tramite le credenziali necessarie. Nel momento in cui vengono inserite credenziali errate l’applicazione mostra un messaggio di errore, per cui l’accesso a Measurify è negato. Nel caso in cui invece le credenziali inserite siano corrette, Measurify permette l’accesso. L’applicazione quindi, effettuando una chiamata http all’API Measurify con le credenziali che ha estrapolato dalle caselle di testo, ottiene in risposta il token in formato JSON. Il token viene poi convertito in stringa e salvato in una variabile globale, in modo da poter essere utilizzato per effettuare le chiamate successive. I motivi per i quali l’applicazione effettua le chiamate a Measurify sono due:
  + Un tipo di chiamata concerne la richiesta di conoscere le misurazioni effettuate.
  + L’altro tipo ha lo scopo di estrapolare le informazioni necessarie per creare il layout funzionale della pagina stessa, quali per esempio il numero di thing e il numero di device per ogni thing.

* *

*Figura 10*: Schermata Login *Figura 11*: Messaggio Errore Login

* Il secondo livello, essendo stato autorizzato l’accesso, mostra la schermata di Measurify con le stanze controllate da uno o più sensori. Nel caso specifico di questo progetto, la stanza è una come il sensore, ma nulla vieta di aggiungere ulteriori sensori di movimento o altri tipi di sensori per un progetto ancora più complete. Inoltre è possibile vedere in questo modo tutte le stanze attive, i sensori associati e le singole notifiche per differenti sensori. La schermata che si presenta è suddivisa in ulteriori due schermate tramite un widget di Flutter che permette al di navigare, con un’appbar posta nella parte bassa dell’applicazione, tra due schermate: rispettivamente sono una schermata Home e una schermata Settings. La schermata Home è a sua volta composta da un’appbar, questa volta posta in alto, che mostra tutte le stanze attive su Measurify. Anche questo genere di schermata grafico è stata possibile tramite un widget di Flutter. Infine se si preme sulla schermata di Settings è visibile l’elenco delle stanze, con la possibilità di attivare/disattivare le notifiche push di ogni singola stanza tramite un rispettivo pulsante di attivazione. Il layout delle schermate sopradescritte è mostrato in figura 10 e in figura 11.

*Figura 12*: Schermata Home *Figura 13*: History delle presence

*Figura 14*: Schermata Settings *Figura 15*: Attivazione subscription

Le figure sopra mostrate mostrano il risultato finale a livello, ma è importante osservare anche come è stato gestito l’ambiente di sviluppo per la creazione dell’applicazione. Il programma è stato suddiviso in più file di tipo Dart, poiché questo permette la separazione tra i vari livelli. Prima di mostrare il codice e tutti i file, è necessario dare qualche informazione sul linguaggio di programmazione di Flutter. Come già spiegato nei capitoli precedenti Flutter si basa su widget in particolare su due categorie di widget:

* Stateless Widget: sono widget immutabili, ovvero significa che le loro proprietà non possono cambiare nel tempo e che tutti I valoro sono quelli finali
* Stateful Widget: mantengono una stato che potrebbe cambiare durante il tempo di vita del widget. Per implementare tale widget sono necessarie:
  + una classe Steteful Widget che crea una instance di una classe State. Questa classe è immutabile e può essere scartata e succesivamente rigenerata
  + a classe state che persiste oltre il tempo di vita del widget.

Tramite queste due categorie sono stati creati tutti i widget presenti nell’applicazione.

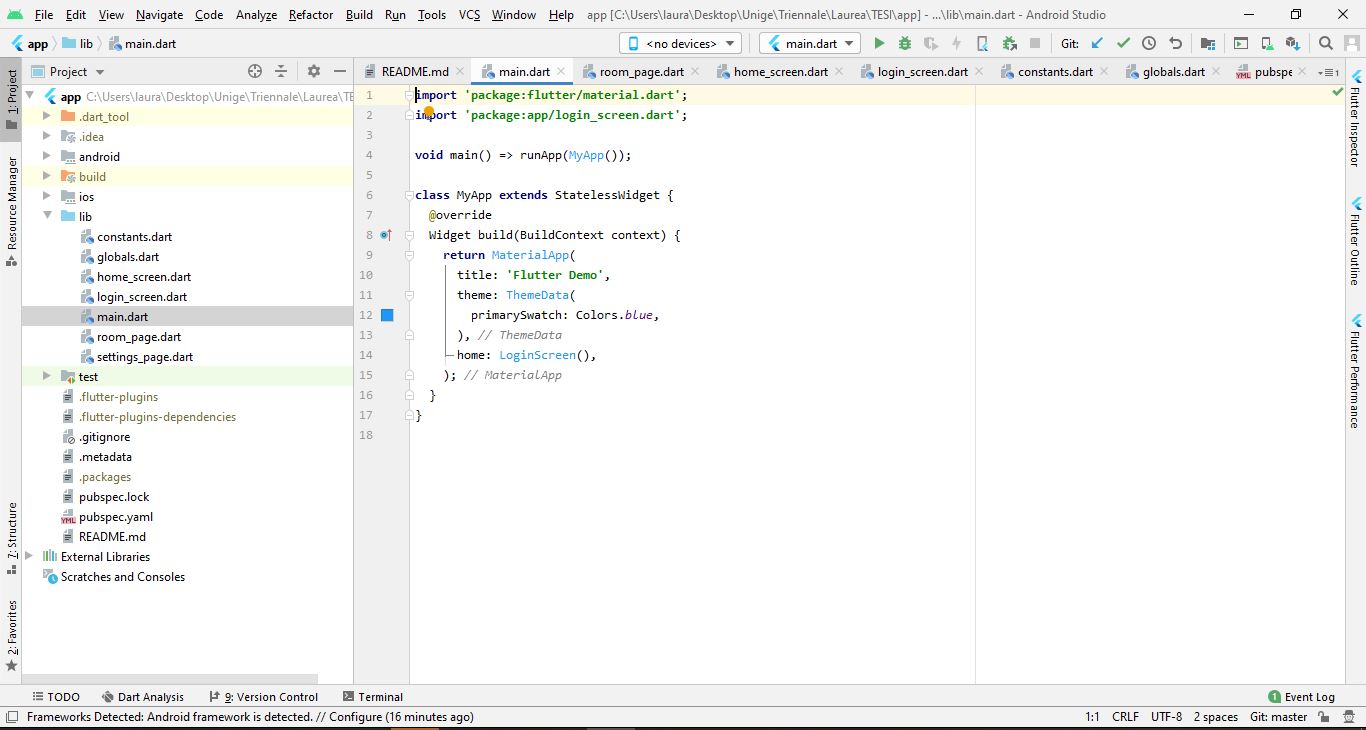
Inoltre è importante sottilinare che in flutter è possibile importare pacchetti, per permettere estensioni come firebase o altre funzionalità. Per fare ciò è sufficiente scrivere nel programma un comando come

import 'package:flutter/material.dart';

e specificare nel file pubspec.yaml tale dipendenza dal pacchetto. L’istruzione riportata come esempio è presente in tutti i programmi Flutter, essendo il pacchetto che permette di utilizzare il design MaterialApp().

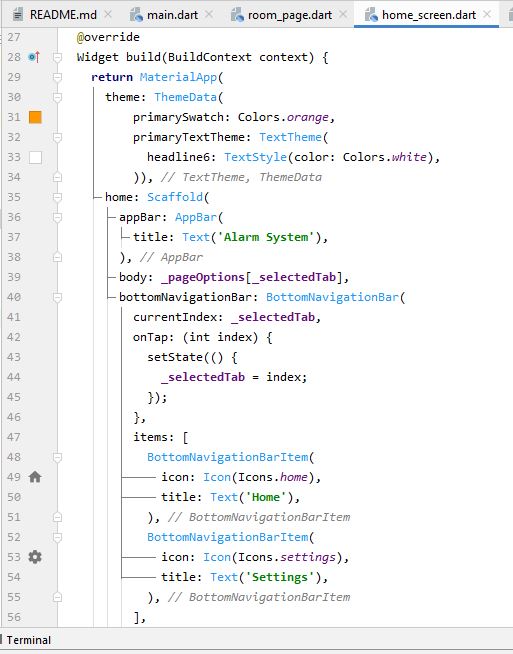
Di seguito saranno mostrati alcuni dei file di maggiore rilevanza che hanno contribuito alla costruzione dell’applicazione:

* MAIN.DART: ovviamente il programma è provvisto di un main, che costituisce il primo livello dell’applicazione. Ha unica funzione di creare il widget che conterrà la dipendenza dal LoginScreen.dart



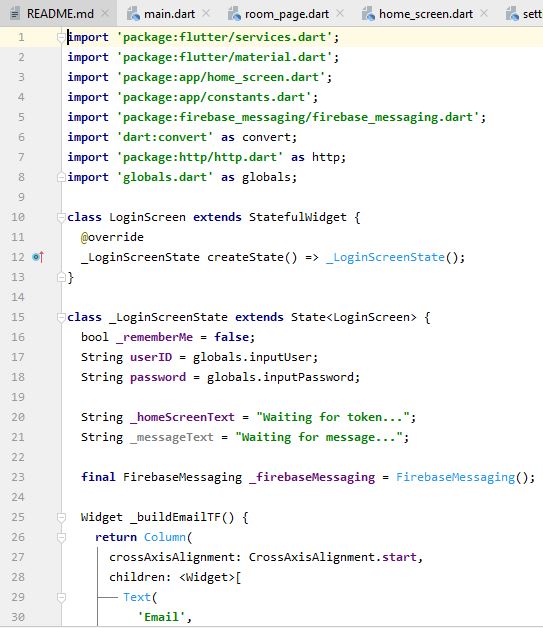
*Figura 16*: Codice main.dart

* HOME\_SCREEN.DART: in questo codice è stato scelto uno Stateful Widget, poiché si è utilizzato una tab bar, che ovviamente ha bisogno di essere dinamica e di poter cambiare a seconda della pagina selezionata. È stato inoltre utlizzato Scaffold(), un widget che flutter mette a disposizione con un design predefinito che aiuta nella creazione di appbar.

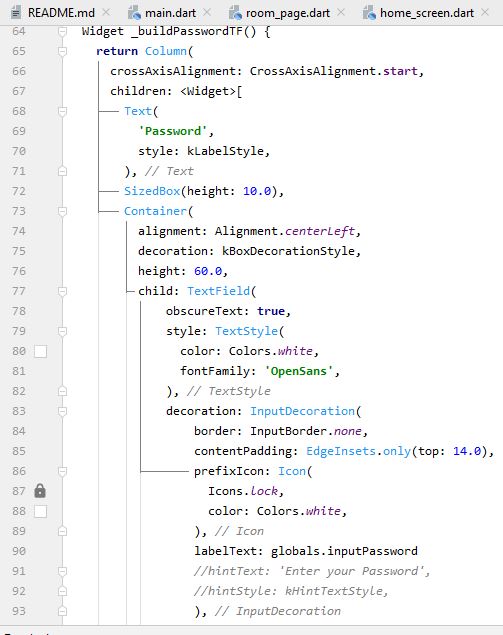
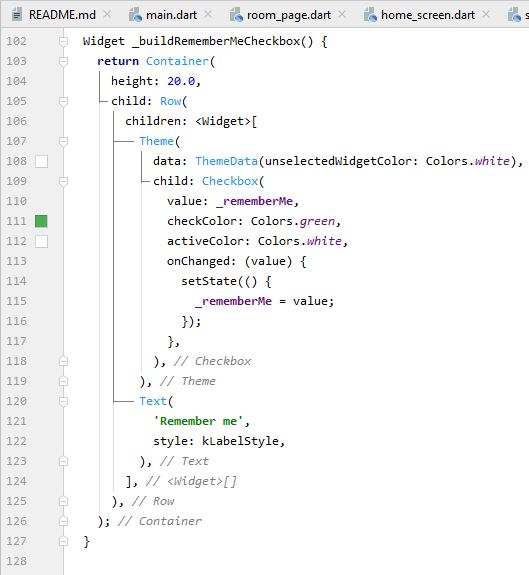
 

*Figura 17*: Codice login\_screen.dart (1)  *Figura 18*: Codice login\_screen.dart (2)

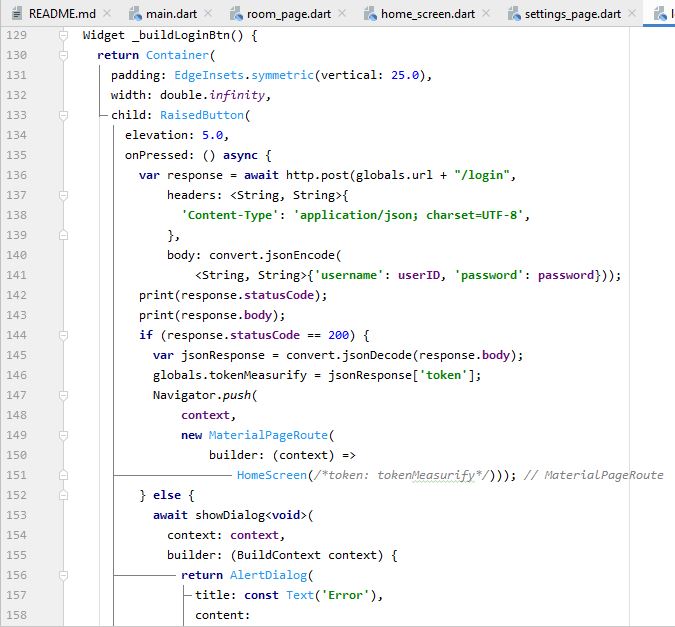
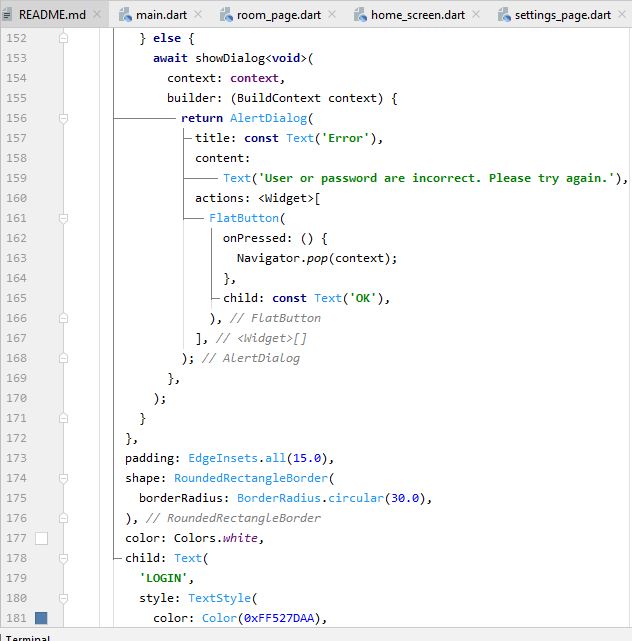
* LOGIN\_SCREEN.DART: anche questo file è basato su uno Stateful Widget. Inoltre con la schemata di login viene introdotta la problematica di richiedere l’accesso a Measurify. Con il login bisogna essere in grado di inserire le credenziali giuste e chiedere a Measurify il token di accesso.

*Figura 19*: Codice login\_screen.dart (3) *Figura 20*: Codice login\_screen.dart (4)

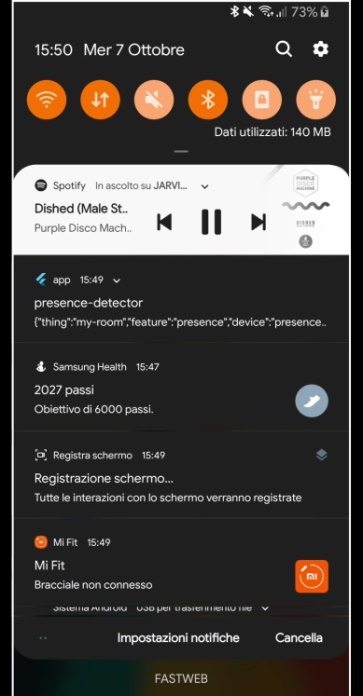
*Figura 21*: Codice login\_screen.dart (5) *Figura 22*: Codice login\_screen.dart (6)

*Figura 23*: Codice login\_screen.dart (7) *Figura 24*: Codice login\_screen.dart (8)

La sperimentazione ha portato alla luce la facilità dello strumento Flutter, in quanto Flutter stesso è basato sull’utilizzo di widget, ognuno dei quali può essere visto come una scatola contenente una differente funzionalità o un differente stile grafico che è stato implementato. Tramite i widget è stato possibile costruire un’applicazione a più livelli, in modo piuttosto semplice, senza la necessità di skills pregresse nel campo della programmazione di Flutter. Uno degli strumenti più utili di Flutter, dato grazie alle implementazioni di Dart, è quello di “Hot Reload”; tale strumento ci ha permesso di continuare a cambiare lo schema grafico e alcune funzionalità dell’applicazione senza la necessità di . L’applicazione è dunque completa sia da un punto di vista grafico sia da un punto di vista di funzionalità.

Infine con l’implementazione delle notifiche push, è stato necessario sperimentare l’intero sistema. Una volta che il sensore percepisce una presenza, il segnale viene inviato a Measurify, che essendo collegata all’applicazione sul telefono associato, invia la notifica tramite Firebase Cloud Messagging al telefono stesso. Il tutto avviene in un periodo di tempo relativamente breve. Il concetto base dietro Firebase Cloud Messagging è proprio quello di poter inviare messaggi, anche personalizzati, a tutti coloro che possono riceverli e hanno sottoscritto argomenti specifici, come nel caso di questo progetto per la notifica di un sistema d’allarme.



*Figura 16*: Esempio reale del funzionamento della notifica push

1. **Contributo personale e considerazioni conclusive**

Il progetto ha avuto il focus di testare le potenzialità di Measurify, con una attenzione particolare alla comunicazione tra hardware e software. Il contributo personale dato al progetto è stato totale per tutto quello che concerne la comunicazione tra i vari dispositivi utilizzati e l’idea del progetto stesso.

È stato un lavoro stimolante, che ha permesso un approfondimento nel mondo dell’IOT. Grazie a questo progetto è possibile entrare nei meccanismi di progettazione moderna e avere una visione il più aperta possibile su come funzioni in questo momento la gestione di un sistema embedded. Si è rivelato essere non solo un lavoro stimolante, ma anche molto soddisfacente poiché tutti i risultati che erano stati prefissati sono stati raggiunti e tutti con esiti positivi. La parte più interessante è stata proprio sperimentare tutte le diverse funzionalità di Measurify, avvalersi di una libreria come Edge Engine che ci ha permesso un notevole risparmio di lavoro sul codice Arduino, rimanendo tuttavia un progetto basato sulla semplicità.

Difatti anche le scelte che sono state fatte rispetto alla parte sensoristica sono proprio state fatte nell’ottica di scegliere qualcosa di semplice, accessibile e comune. Il funzionamento del sistema stesso, come già più volte descritto si basa su pochi semplici concetti:

* Il sensore PIR registra la presenza nella stanza designate
* Il segnale viene inviato tramite la scheda Wi-Fi a Measurify;
* A quel punto Measurify è in grado di notificare il telefono cellulare associate
* Entra in gioco l’applicazione, che sviluppata su più livelli, permette di accedere alla propria area riservata di Measurify e ricevere le notifiche in tempo reale.

Come già ampiamente approfondito, l’utilizzo della libreria EdgeEngine ha avuto un ruolo fondamentale nella prima parte della progettazione. È stato possibile concentrarsi su aspetti differenti del progetto e rendere il progetto stesso più complesso e completo. Tramite tale libreria la comunicazione tra sensore e Measurify è stata molto semplificata. Inoltre aver potuto scambiare informazioni con i due tesisti Luca Lazzaroni e Andrea Mazzara, autori di Edge Engine, ha permesso di capire più a fondo come funziona un lavoro in team e come approcciare un prodotto da utilizzatori. Difatti per quanto riguarda Edge Engine il nostro contributo è stato quello di testare il lavoro fatto dai due tesisti, proprio come utilizzatori finali, e di conseguenza avere un feedback in entrambi le direzioni: per loro, questo progetto ha costituito il primo test con un cliente e un sistema fisico, per questo progetto ha permesso di avvalersi di tutte le funzionalità di Edge Engine usandola come utilizzatori finali.

La parte che ha sicuramente richiesto più lavoro è stato lo sviluppo dell’applicazione. Tuttavia poter applicare e approfondire nuove conoscenze in un campo che ancora era di nostra conoscenza è stato molto interessante. Flutter si è rivelato essere uno strumento più che adatto per la creazioni di applicazioni semplici, ma di impatto.

Il sistema può ovviamente essere esteso sia a livello hardware sia a livello software. Sarebbe possibile, visto che Measurify è in grado di analizzare anche dati stream, aggiungere una videocamera, sia utilizzando una scheda Raspberry Pie oppure un modulo Cam Arduino. L’implementazione di una videocamera permetterebbe di avere uno stream video della stanza; l’applicazione non solo notificherebbe la nuova presenza, ma permetterebbe anche di vedere cosa succede nella stanza, in questo modo il sistema sarebbe molto più completo e funzionale. Ci sono molte altre implementazioni possibili, come l’utilizzo di un cicalino che si attivi nel momento in cui la presenza viene rilevata, dando così anche un aspetto sonoro al sistema stesso e avvertendo l’intruso della presenza di un sistema di allarme nell’abitazione. Inoltre il sistema potrebbe essere comprensivo di più sensori di movimento, associati a diverse stanze. Le idee per migliorare il progetto sono tante, ma i risultati ottenuti fino a questo momento sono stati più che soddisfacenti. Il sistema risponde a tutte le caratteristiche di base che un sistema di allarme dovrebbe avere: un sensore che rilevi le presenza, un’ API in grado di collezionare tali presenze e infine un’applicazione in grado di notificare in tempo reale le presenze rilevate dal sensore stesso.

Si può dunque concludere che il sistema ha risposto brillantemente a tutte le caratteristiche e le funzionalità implementate; dunque è stato un successo.

1. **Riferimenti bibliografici**

www.flutter.dev

www.firebase.google.com

**Ringraziamenti**

Sono finalmente arrivata alla fine di questo turbolento percorso, dettato da tanta determinazione, tanta felicità e anche tanta sofferenza. Mi sembra più che giusto fermarmi qualche minuto e ringraziare tutte le persone che hanno fatto parte di questo viaggio insieme a me.

Voglio ringraziare mia mamma, mio papà e mia sorella per aver sempre creduto nelle mie capacità, per avermi supportato in ogni momento, per essere sempre stati fieri della persona che sono diventata. Grazie per tutte le volte che avete gioito insieme a me per un nuovo esame passato, per avermi consolata quando le cose andavano male e per avermi sopportata quando ero nervosa (cioè sempre).

Vorrei ringraziare tutta la mia famiglia, nonni, zii e cugini che sono sempre stati orgogliosi di tutto quello che ho fatto e che hanno sempre mostrato la loro solidarietà in qualsiasi momento, di gioia o di dolore, passato in questi anni.

Grazie a te, Nicolò, che mi hai fatto capire che nonostante tutte le avversità, ho la forza per rialzarmi e per avermi dato quella forza quando mi mancava. Grazie per essere stato al mio fianco in ogni momento, per avermi spronata a dare sempre il massimo, per avermi fatto capire che ogni tanto le cose non vanno come si vuole e per avermi appoggiata in ogni mia decisione. Grazie di avermi fatta ridere quando volevo solo piangere e di avermi sopportata quando diventavo la solita catastrofista.

Grazie a quegli amici di una vita, Gianmaria, Roberta e Giorgia perché ci siete stati, ci siete e so che ci sarete sempre.

Grazie ai miei bro, per aver reso questi anni più leggeri, per aver condiviso così tante esperienze insieme, per essere dei veri amici. Un grazie speciale a te Giorgia, che hai saputo esserci sempre, anche quando non te lo chiedevo.

Grazie a coloro che hanno condiviso questo pazzesco percorso con me, a quella banda di scalmanati ingegneri che siamo diventati tutti, ma più di tutti grazie a Sarvenaz e Carola. Grazie ragazze per essere state lì, per aver condiviso la disperazione di ogni esame, per tutte le risate insieme. Avete reso questo percorso speciale.

Ultimo ma non per importanza un grazie al mio co-tesista, nonché amico Seba. Ricorderò sempre con piacere le intere giornate (e serate) passate ad arrabbiarci per le cose che non funzionavano, gioire un minuto dopo e scoprire invece che comunque nulla era come doveva essere, per tutte le risate che ci siamo fatti quando ormai eravamo troppo stanchi e per la fantastica amicizia che è nata.

La soddisfazione più grande? Ora posso finalmente dire “Fidatevi ragazzi, sono un ingegnere”.