

**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

***Alla mia famiglia***

**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.

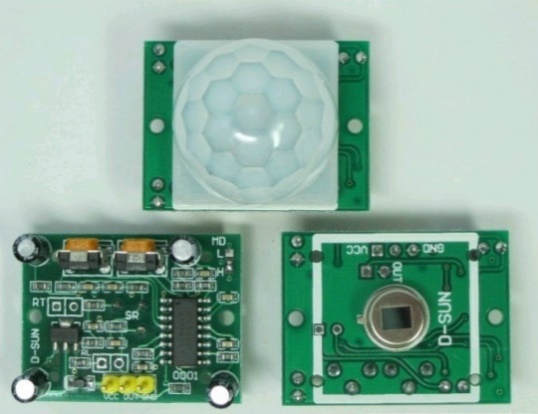
Il sistema verrà ampliamente discusso e dettagliato nei prossimi capitoli.

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 1*: 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*: Arduino IDE

**Scheda Wi-Fi**

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.



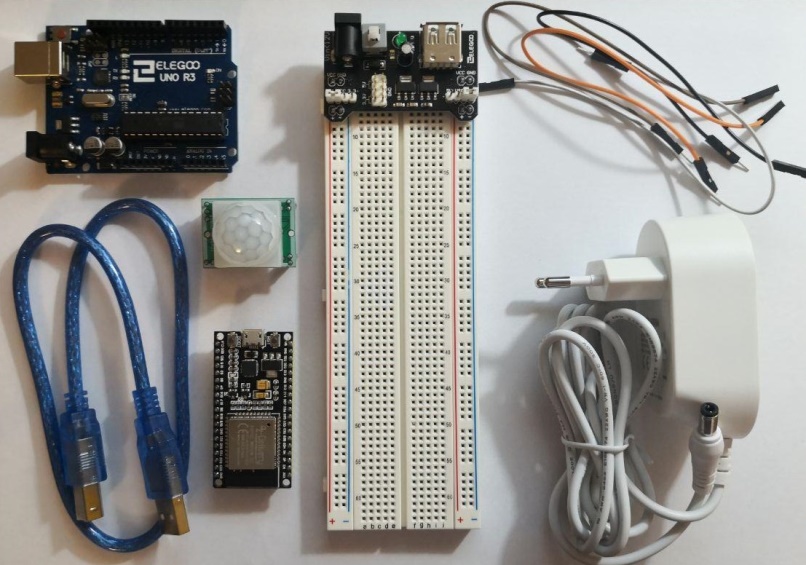
*Figura 4*: Scheda Wi-Fi Esp32

**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 progetto

**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**

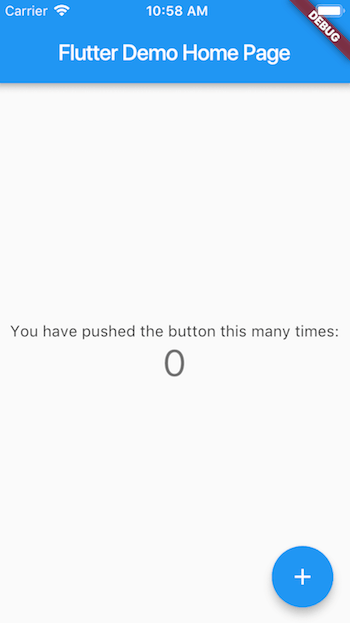
La connessione tra il cloud di Measurify e l’hardware è stato facilitato dall’implementazione nel codice Arduino della libreria EdgeEngine. Rispetto a quanto detto sopra:

* 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. 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:

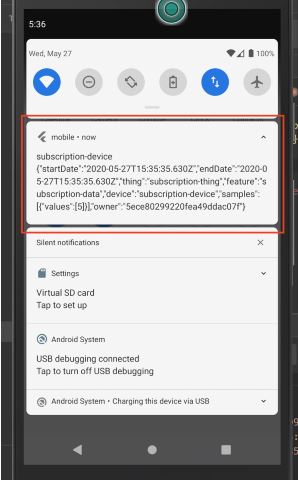
* Dart Platform: è un linguaggio di programmazione sviluppato da Google. Il compilatore Dart permette di scrivere programmi sia per il web sia per desktop e server
* Flutter Engine: è scritto principalmente in C++, fornisce supporto per il rendering a basso livello utilizzando la libreria grafica di 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 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 proprio la comunicazione tra il sensore e Measurify. A livello di specifiche hardware (foto del sistema hw) 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

Per tale sperimentazione è stato scelto il livello high (ricordati il perché)

Ogni volta che il sensore colleziona una presenza è necessario che tale campione venga memorizzato tra le “measurement” di Measurify e dunque reso disponibile dall’API stessa. Una volta realizzato il device, le chiamate effettuate a Measurify sono tutte di tipo POST, ovvero creano delle measurement. Per effettuare le chiamate è necessario disporre di un token, che si ottiene tramite

POST {{url}}/v1/login

con il seguente corpo:

{

"username" : “presence-username",

"password" : "presence-password”

}

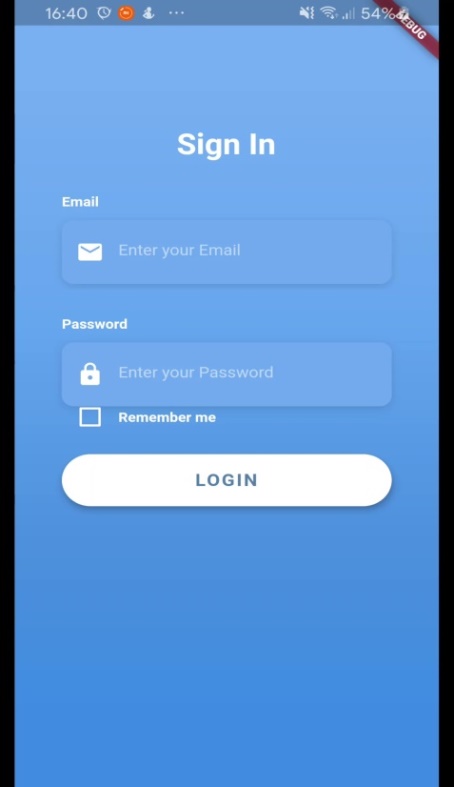
Tale POST è stata inserita nel codice Arduino in modo che in maniera autonoma il dispositivo possa richiedere ogni volta un nuovo token, quando questo dovesse scadere. Il body di questa POST costituisce le credenziali per accedere all’ambiente di Measurify. Per verificare che le presenze venissero effettivamente registrate e memorizzate è stato necessario l’utilizzo di 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. È stato di rilievo nello sviluppo del progetto proprio per controllare che tutti i passaggi funzionassero(?). Nel concreto, per verificare le misurazioni precedenti, era necessario utilizzare

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

Che restituiva tutte le misure, effettuate sulla thing “my-room”.

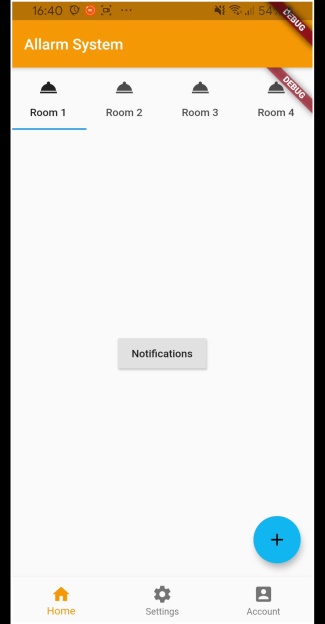
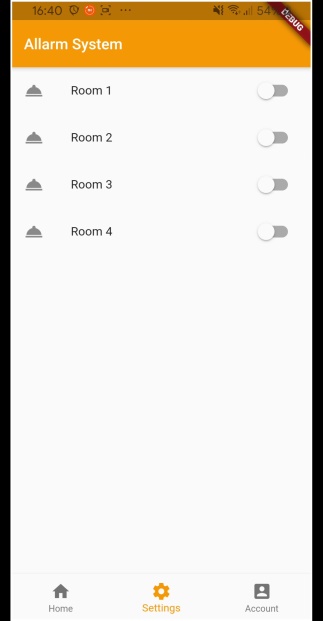
Una volta che la sezione hardware è stata ultimata, è stato necessario concentrarsi sul lavoro software, con una particolare attenzione alla creazione dell’applicazione. In questo passaggio l’utilizzo di Flutter combinato con l’ambiente di sviluppo Android Studio è stato fondamentale. L’applicazione si struttura su due livelli:

* Il primo, nonché la schermata di apertura, è composto da una schermata di login. Tale login permette l’accesso a Measurify, tramite le credenziali necessarie.



*Figura 8*: Login Applicazione

* Il secondo, essendo stato autorizzato l’accesso, permette di vedere le funzionalità di measurify. Inoltre è possibile vedere in questo modo tutte le stanze attive, I sensori associati e le singole notifiche per fdifferenti sensori. Nel caso specifico di questo progetto il sensore è unico e la stanza è singola, ma si è progettata l’applicazione con una finalità il più generale possibile. Le schermate che si presentano sono due, come mostrate in Figura 9 e figura 10. La prima mostra l’elenco delle “room” presenti e attive su Measurify; la seconda è una schermata di attivazione delle notifiche push, in modo da aver il controllo per le differenti stanze.

*Figura 9*: Schermata 1 *Figura 10*: Schermata 2

La sperimentazione ha portato alla luce la facilità dello strumento Flutter, in quando la parte grafica è costituita da widget. 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 (?). L’applicazione è dunque completa sia da un posto 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.

1. **Contributo personale e considerazioni conclusive**

Punti da toccare:

* Il contributo personale è stato totale. Il progetto è nato da noi e muore con noi
* Considerazioni conclusive: tutte quello che potremmo fare e ampliare con il sistema progettato.

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 soddisfacente il risultato a cui siamo arrivati, poiché il progetto ha dato i risultati voluti. Il sensore PIR registra la presenza nella stanza designata, invia il segnale attraverso la scheda Wi-Fi a Measurify; a quel punto Measurify è in grado di notificare il telefono cellulare associato. L’applicazione sviluppata in più livelli, come spiegato nel capitolo precedente, permette di accedere a Measurify e vedere la situazione.

1. **Riferimenti bibliografici**

Flutter

Firebase