

# UNIVERSITÀ DEGLI STUDI DI SALERNO



Dipartimento di Ingegneria dell'Informazione ed  
Elettrica e Matematica applicata

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

TESI DI LAUREA  
IN  
INTERNET OF THINGS

Titolo tesi

iTenda. Progettazione e ingegnerizzazione di una tenda a rullo smart.

Relatore  
Prof. RITROVATO PIERLUIGI

Candidato *Sica Ferdinando*  
Matr. 0612704522

*ANNO ACCADEMICO 2020/2021*

# INDICE

Introduzione	4
Struttura della tesi	4
1. Progettazione del sistema fisico	6
1.1.Funzionalità del sistema	7
1.2.Sstrumenti utilizzati (hardware e software)	7
1.3.Implementazione	10
1.3.1.Circuito	10
1.3.2.Codice	11
1.4.Risultato finale	13
2. Progettazione del sistema software	15
2.1.Funzionalità del sistema	15
2.2.Sstrumenti utilizzati	15
3. Integrazione, Visualizzazione e Descrizione del risultato finale	19
3.1.Splash Screen e Schermata Principale	19
3.1.1.Login	20
3.1.2.Sign In	21
3.1.3.Reset password	22
3.2.Homepage	23
3.2.1.Impostazioni	23
3.2.2.Statistiche	25
3.2.3.Recenti	26
3.2.4.Stato Tenda	28
3.3.ESP32	29
3.4.Database e PHP	32
Conclusioni	34

Bibliografia e Sitografia	35
Ringraziamenti	36

*Il nostro tempo è limitato, per cui non lo dobbiamo sprecare vivendo la vita di qualcun altro. Non facciamoci intrappolare dai dogmi, che vuol dire vivere seguendo i risultati del pensiero di altre persone. Non lasciamo che il rumore delle opinioni altrui offuschi la nostra voce interiore. E, cosa più importante di tutte, dobbiamo avere il coraggio di seguire il nostro cuore e la nostra intuizione. In qualche modo, essi sanno che cosa vogliamo realmente diventare. Tutto il resto è secondario.*

(Steve Jobs)

## **Introduzione**

Il seguente lavoro intende offrire al panorama della domotica un dispositivo innovativo e di facile utilizzo, in grado di semplificare, o meglio, di rimuovere determinate operazioni effettuate quotidianamente.

In particolare, il prodotto che si vuole realizzare ha come scopo quello di eliminare dal quotidiano l'operazione di apertura e chiusura di una tenda a rullo.

L'idea di tale prodotto è stata concepita durante il periodo di lockdown, in modo particolare durante una giornata di studio, e che tale operazione era in grado di interrompere il workflow, con conseguenza perdita di concentrazione. Anche il solo alzare o abbassare la tenda manualmente rallentava anzi bloccava il lavoro stesso.

La Domotica è una scienza interdisciplinare che si occupa dello studio delle tecnologie atte a migliorare la qualità della vita nella casa, realizzando case intelligenti, confortevoli, sicure e di semplice utilizzo.

Il prodotto da realizzare deve seguire questa linea di pensiero, deve rimuovere o semplificare un'operazione quotidiana cercando, come detto in precedenza, di migliorare la qualità della vita nella casa.

In particolare, si vuole realizzare un prodotto che possa offrire il più alto grado di personalizzazione, cercando di accontentare tutti i possibili utenti, attraverso diverse modalità di gestione dell'apertura o la chiusura della tenda.

Il termine Internet of Things fa riferimento alla tendenza di rendere interconnesso ogni oggetto della vita quotidiana. Se venti anni fa l'accesso alla rete era esclusivamente limitato a personal computer e sistemi informatici di medie-grandi dimensioni, oggi assistiamo all'avvento di dispositivi connessi di più disparata natura e dimensioni: orologi, tablet, semafori, robot da cucina, assistenti vocali, autoveicoli e navigatori satellitari.

Il prodotto da realizzare farà parte degli oggetti detti "smart", sarà infatti in grado di comunicare con l'ambiente esterno e, in particolare, con un database sul quale verranno memorizzati dati di interesse.

## **Struttura della tesi**

Nel capitolo I verrà descritta e analizzata la prima parte del progetto, che è parte integrante e fondamentale per l'implementazione finale, si discuterà quindi di ciò

che è stato realizzato nel corso di IoT analizzando tutti gli strumenti Hardware e Software utilizzati.

Nello specifico in questo capitolo si discuterà della realizzazione hardware e software del prodotto fisico.

Nel capitolo II verrà descritta e analizzata la seconda e ultima parte del progetto, analizzando dapprima i software utilizzati e in seguito come sono stati integrati tra di loro. In questo capitolo si andrà ad esaminare la parte riguardante la realizzazione dell'interfaccia grafica e le modifiche eseguite al codice del prodotto fisico, in quanto la parte fisica non è stata variata

Nel capitolo III verrà descritta e analizzata la realizzazione finale del progetto, di come sono stati integrati tra di loro tutti gli elementi, visualizzando l'interfaccia grafica realizzata, discutendo di essa e del suo funzionamento.

Si è deciso di inserire alcune parti di codice, con relativa spiegazione, per una miglior comprensione di quest'ultimo.

# **1. Progettazione del sistema fisico**

Il progetto prevede l'implementazione delle modalità di gestione della tenda.

In primis, l'apertura e la chiusura della tenda verrà attuata attraverso un servomotore e verrà comandata da sensore di movimento, assistente vocale Google, sensore di luminosità o attraverso la specifica fascia oraria; come sistemi di comunicazione utilizzeremo wifi e mqtt.

In questo capitolo verrà analizzata tutta la parte che riguarda l'implementazione hardware e software del prodotto fisico, il codice per il microcontrollore e i vari collegamenti tra i sensori.

L'obiettivo del corso IOT era quello di realizzare una tenda a rullo smart, in grado di capire le esigenze dell'utente e agire di conseguenza. Infatti, l'apertura e la chiusura di una tenda a rullo è una delle operazioni che si esegue più spesso durante la vita di tutti i giorni, basti pensare che questa operazione viene eseguita almeno due volte al giorno, all'alba e al tramonto. Senza contare le volte che quel fastidioso raggio di luce colpisce direttamente il monitor del nostro pc o della nostra tv, interrompendo quello che stavamo facendo e costringendoci ad abbassarla manualmente.

Il nostro progetto aveva quindi l'ambizione di rimuovere queste operazioni, automatizzando l'apertura e la chiusura di una tenda a rullo. L'automazione della stessa potrà essere gestita attraverso gesture, luminosità della stanza, comandi vocali e automazioni vere e proprie (orari prestabiliti).

Tuttavia, il dispositivo non era dotato di interfaccia grafica, essendo esso una scheda e alcuni sensori collegati tra loro tramite una "bread board", sprovvisto di uno schermo LCD, di conseguenza le impostazioni potevano essere modificate solo tramite il codice.

Il risultato finale vuole essere un sistema semplice e di facile comprensione che si interfacci con il dispositivo e permetta una modifica completa delle impostazioni, ovviamente diverse per ogni utente e per ogni dispositivo.

Per una più semplice comprensione del risultato finale, si è deciso di presentare, seppur in breve, la prima realizzazione del dispositivo, progetto già presentato all'esame di Internet of Things

## 1.1. Funzionalità del sistema

Le modalità di gestione verranno implementate da:

- **Sensore di movimento**, grazie al quale verranno riconosciute due tipi di gesture che identificano quale movimento attuare.
- **Sensore di luminosità**, che comanderà l'apertura e chiusura della tenda in base alla luminosità rilevata.
- **Comandi vocali**, attraverso l'interfacciamento con l'assistente Google.
- **Automazioni**, attraverso la comunicazione di una fascia oraria di chiusura/apertura.

In aggiunta il movimento della tenda sarà accompagnato da una lieve melodia realizzata tramite l'utilizzo di un buzzer.

## 1.2. Strumenti utilizzati (hardware e software)

Nel seguente capitolo vedremo nel dettaglio tutti gli strumenti utilizzati per implementare tutte le funzionalità elencate in precedenza, presentando dapprima i linguaggi di programmazione e gli IDE utilizzati e poi le componenti Hardware.

- **Python** è un linguaggio di programmazione di più "alto livello" rispetto alla maggior parte degli altri linguaggi, orientato a oggetti, adatto, tra gli altri usi, a sviluppare applicazioni distribuite, scripting, computazione numerica e system testing. Ideato da Guido van Rossum all'inizio degli anni Novanta, il nome fu scelto per la passione dello stesso inventore verso i Monty Python e per la loro serie televisiva Monty Python's Flying Circus ed è spesso paragonato a Ruby, Tcl, Perl, JavaScript, Visual Basic o Scheme.

Spesso viene anche studiato tra i primi linguaggi per la sua somiglianza a uno pseudo-codice e di frequente viene usato per simulare la creazione di software grazie alla flessibilità di sperimentazione consentita da Python, che permette al programmatore di organizzare le idee durante lo sviluppo, come per esempio il creare un gioco tramite Pygame oppure il back-end di un sito web tramite Flask o Django.

Python è stato utilizzato in Zerynth, un'implementazione software del linguaggio di programmazione che verrà analizzata nel dettaglio in seguito.

- **Zerynth** è un software per la programmazione di microcontrollori, esso si rivolge a piattaforme di microcontrollori a 32 bit ed è progettato per combinare Python con codice C, permette inoltre di collegare i microcontrollori al Cloud per lo sviluppo di prodotti Internet of Things (IoT).

Zerynth è composto da: una macchina virtuale, ovvero una macchina virtuale multithread basata su stack progettata per eseguire un bytecode Python personalizzato; un IDE open source (Zerynth Studio) potenziato da una toolchain a riga di comando per il provisioning di microcontrollori con una Zerynth VM e lo sviluppo di programmi Python per essa e un gestore di dispositivi avanzato che è un servizio di gestione di dispositivi e dati che semplifica la registrazione e l'organizzazione in modo sicuro , monitorare e gestire in remoto i dispositivi IoT su larga scala. Il codice è stato quindi scritto in python e tramite il sopracitato software caricato all'interno della board.

Le librerie di Zerynth utilizzate sono le seguenti:

- **MQTT** (MQ Telemetry Transport o Message Queue Telemetry Transport) è un protocollo standard ISO (ISO/IEC PRF 20922) di messaggistica leggero di tipo publish-subscribe posizionato in cima a TCP/IP. È stato progettato per le situazioni in cui è richiesto un basso impatto e dove la banda è limitata. Il pattern publish-subscribe richiede un broker di messaggi. Il broker è responsabile della distribuzione dei messaggi ai destinatari.

MQTT viene utilizzato dal microcontrollore per comunicare con l'esterno, in particolare per ricevere operazioni di apertura e chiusura e inviare messaggi di log.

- **IFTTT** è un servizio che permette ad un utente di programmare una risposta ad eventi del mondo di vario genere, c'è una lunga lista di tipi di eventi a cui IFTTT può rispondere, rilevabili via Internet. IFTTT ha partnership con centinaia di fornitori di servizi, i quali forniscono notifiche di eventi ad esso ed eseguono comandi che implementano le risposte. I programmi, chiamati applet, sono semplici e creati graficamente da parte dell'utente.

Abbiamo utilizzato IFTTT per creare degli applets per Google Assistant; i nostri applets reagiscono a due frasi, o loro alias, che vanno comunicate in maniera vocale all'assistente Google, il quale darà una risposta personalizzata scelta al momento di creazione dell'applets e, successivamente, IFTTT provvederà all'invio

di dati ad Adafruit, dal quale verranno recuperati dalla scheda tramite protocollo MQTT.

- **PWM** (acronimo del corrispettivo inglese pulse-width modulation). In elettronica e telecomunicazioni la modulazione di larghezza d'impulso è un tipo di modulazione digitale che permette di ottenere una tensione media variabile dipendente dal rapporto tra la durata dell'impulso positivo e dell'intero periodo (ciclo di lavoro). Grazie ai moderni microcontrollori, è possibile attivare o disattivare un interruttore ad alta frequenza e allo stesso modo rilevare lo stato e il periodo di un impulso.

Questa tecnica è stata utilizzata per far ruotare in senso orario/antiorario il servomotore e per far suonare al buzzer le diverse note.

- Un **thread** o thread di esecuzione, in informatica, è una suddivisione di un processo in due o più filoni (istanze) o sottoprocessi che vengono eseguiti concorrentemente da un sistema di elaborazione monoprocesso (monothreading) o multiprocessore (multithreading) o multicore.  
L'utilizzo di tale tecnica è risultata essenziale durante lo sviluppo, questo perché grazie a questa tecnica, il dispositivo è in grado di eseguire delle operazioni in contemporanea come ad esempio avviare la chiusura della tenda, continuare a controllare l'arrivo di altri comandi ed eventualmente interrompere tale operazione.

Verranno ora presentati gli elementi Hardware utilizzati:

- **ESP32** è una serie di microcontrollori system on a chip a basso costo e a bassa potenza con Wi-Fi integrato e Bluetooth dual-mode . ESP32 è creato e sviluppato da Espressif Systems , un'azienda cinese con sede a Shanghai, ed è prodotto da TSMC utilizzando il loro processo a 40 nm. È un successore del microcontrollore ESP8266 .

Risulta quindi essere il “cervello” del progetto, su di essa verrà eseguito il codice e ad essa verranno collegate le varie componenti, quindi sensori e attuatori.

In particolare la board utilizzata è la “JZK ESP32 DEVKIT V1”.

- Il **DC motor** è uno qualsiasi di una classe di motori elettrici rotanti che converte l'energia elettrica a corrente continua in energia meccanica. I tipi più comuni si basano sulle forze prodotte dai campi magnetici. Quasi tutti i tipi di motori DC hanno un meccanismo interno, elettromeccanico o elettronico, per cambiare

periodicamente la direzione della corrente in una parte del motore, verrà utilizzato per chiudere/aprire la tenda grazie al suo movimento rotatorio.

Si sarebbe potuto utilizzare anche uno stepper motor poiché compie la stessa azione, anche in modo più preciso, ma si è ritenuto inutile sfruttare tale precisione.

- **Sensore di prossimità** sono dei sensori in grado di rilevare la presenza di oggetti nelle immediate vicinanze, senza che vi sia un effettivo contatto con l'ADC (Analog to Digital Converter).

La distanza entro cui questi sensori rilevano oggetti è definita portata vedente. Alcuni modelli dispongono di un sistema di regolazione per poter calibrare la lunghezza di veduta. L'assenza di meccanismi d'attuazione meccanica, e di un contatto fisico tra sensore e oggetto, fa sì che questi sensori presentino un'affidabilità molto elevata. In particolare il componente utilizzato è l'hcsr04, grazie ad esso verranno rilevate le "gesture" per poter azionare la tenda.

- Un **fotorivelatore** è un dispositivo in grado di rivelare la radiazione elettromagnetica, fornendo in uscita un segnale avente un'intensità di corrente o una differenza di potenziale proporzionale all'intensità della radiazione rilevata. Questo sensore verrà utilizzato per azionare la chiusura della tenda in caso di eccessiva luce.
- Un **buzzer** o beeper è un dispositivo di segnalazione audio, che può essere meccanico, elettromeccanico, o piezoelettrico (abbreviato anche come piezo). I tipici utilizzi del buzzer includono dispositivi di allarme, timer, e PC speaker per i feedback sugli input dell'utente, come pressione dei tasti o click del mouse, nei vecchi personal computer. Il suono del buzzer verrà utilizzato per accompagnare il movimento della tenda, sia in fase di chiusura che di apertura, riproducendo un suono diverso in base all'operazione.

## 1.3. Implementazione

In questo capitolo verrà analizzata l'implementazione della prima parte del progetto, si descriverà quindi come gli elementi precedentemente elencati verranno sfruttati e uniti tra di loro.

### 1.3.1. Circuito

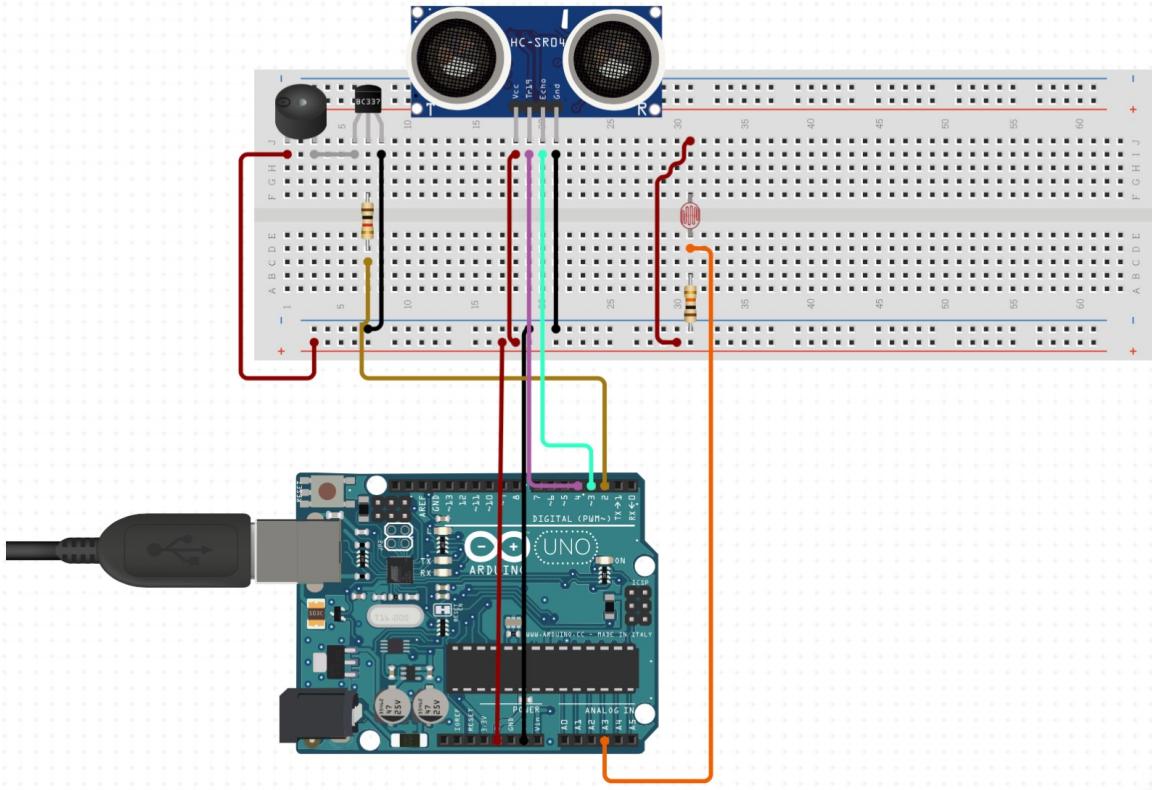


Figura 1. Rappresentazione circuitale del progetto

La prima cosa da realizzare è il circuito fisico per collegare la board con tutte le vari componenti necessarie alla realizzazione del sistema finale. In particolare, il circuito che è stato realizzato è quello che possiamo vedere in foto anche se con elementi diversi. Nello specifico, viene sfruttata una “bread-board”, uno strumento utilizzato per creare prototipi di circuiti elettrici che non richiede saldature ed è completamente riutilizzabile, che permette di aumentare i pin della board in modo da utilizzare più sensori.

### 1.3.2. Codice

Prima di iniziare a scrivere il codice finale, si sono testati singolarmente gli elementi del progetto, questo per capirne le specifiche e verificare le funzionalità, per poi unire il tutto.

Si è poi sfruttata una libreria esterna per il sensore “HC-SR04”, non ancora inclusa in Zerynth, scritta in C poiché Zerynth supporta anche questo linguaggio di programmazione.

Si analizzeranno ora solo le parti più interessanti del codice o comunque quelle più complesse che richiedono una piccola analisi.

In primis, il sistema subito dopo l'accensione, eseguirà il programma ad esso dedicato.

In seguito viene effettuato l'accesso al Wi-Fi necessario per poter comunicare con l'esterno, attraverso alcune librerie di Zerynth scritte ad-hoc per la board utilizzata.

In caso di corretta connessione al Wi-Fi, si procede a stabilire una connessione MQTT con mosquitto e Adafruit, due broker MQTT utilizzati per scambiare informazioni con l'esterno. Nello specifico mosquitto viene utilizzato per trasmettere informazioni di servizio infatti, ogni operazione eseguita dalla tenda o comunque tutto ciò che bisogna notificare, viene appunto comunicato attraverso questo canale, mentre Adafruit viene utilizzato per ricevere comandi dagli assistenti vocali, perché quando viene comandata l'apertura o la chiusura tramite questi ultimi, viene attivato un Applet su IFTT che procederà a pubblicare su un topic Adafruit il comando da eseguire, che verrà recepito ed eseguito dalla board.

All'accensione del dispositivo, viene eseguito il codice, impostando le variabili a valori di default (la tenda viene considerata completamente aperta) e vengono "startati" i vari thread con le relative funzioni.

In particolare si ha un thread per ognuna delle seguenti funzioni:

- **stepperMotor.movimento:** funzione che si occupa del movimento del motore della tenda, questa funzione è sempre in esecuzione in background e ha al suo interno una flag che attiva o meno il movimento, inoltre viene verificato quale movimento attuare, tramite una variabile di direzione interna, per aprire o chiudere la tenda. Quindi l'unione tra flag di attivazione e variabile di direzione permette il movimento del motore. Il movimento viene attivato utilizzando la tecnica del PWM già analizzata in precedenza.
- **buzzerSong.suoneria:** funzione che si occupa di far suonare il buzzer, questa funzione è sempre in esecuzione in background e ha al suo interno una flag che attiva o meno il suono, inoltre troviamo una variabile interna che seleziona quale melodia riprodurre; il suono viene riprodotto con la tecnica del PWM.
- **action:** controlla ogni secondo lo stato della variabile "operation" che definisce appunto l'operazione da eseguire. All'interno di questa, in base all'operazione da effettuare andiamo a modificare le variabili all'interno delle funzioni finalizzate a

far suonare la melodia e per far ruotare lo stepper motor. Al termine di ogni operazione viene mandato un messaggio MQTT contenente il valore dell'altezza della tenda, opportunamente calcolato. L'operazione di apertura/chiusura tenda in totale impiega circa 30 secondi, tempo modificabile tramite la variabile "max\_high\_tenda". Inoltre viene istanziato un timer per verificare quanto tempo impiega per completare l'operazione, in questo modo possiamo ricavare l'altezza della tenda, basandoci, appunto, sul tempo trascorso.

- **check\_brightness:** funzione che si occupa di verificare se c'è luce all'interno della stanza o dove si trova il fotoresistore. In particolare, ogni 30 secondi, viene verificato il valore riportato dal sensore e, se necessario, comanda la chiusura della tenda.

Dopo aver eseguito questi thread, la parte che viene eseguita dal thread principale verifica le gesture tramite il sensore ad infrarossi. Poiché le gesture da riconoscere sono relativamente veloci e difficili da rilevare, far girare questo codice sul "thread principale" gli attribuisce priorità, permettendo una maggior velocità di esecuzione e, di conseguenza, un miglior riconoscimento delle gesture.

## 1.4. Risultato finale

Ciò che si è ottenuto dopo questa prima parte del progetto è un sistema "stand-alone", preimpostato,

modificabile solo da codice  
ma perfettamente  
funzionante.

Come da prodotto finale in  
foto.

Si evidenzia come il  
microcontrollore EPS32 sia  
collegato tramite la  
"breadboard", a tutti gli altri  
elementi utilizzati;

tralasciando lo stepper  
motor che ha bisogno di un  
proprio driver per funzionare e quest'ultimo viene collegato al microcontrollore.

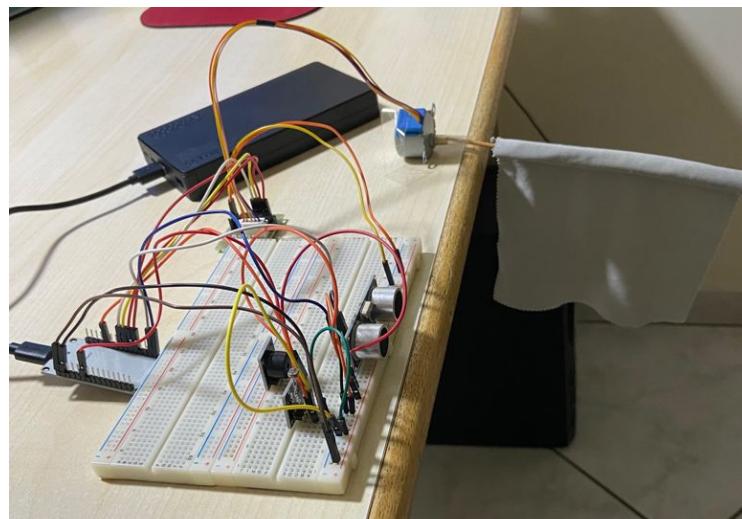


Figura 2. Apparato hardware

Dopo aver realizzato il prodotto fisico e dopo aver scritto il codice, si è passato ad una attenta e lunga fase di test e analisi. Ci si è soffermati molto sul riconoscimento delle gesture, poiché molto difficili da riconoscere con un sensore che fondamentalmente misura la distanza lungo una dimensione, infatti uno dei problemi da risolvere era il riconoscimento di movimenti che non erano gesture desiderate, come ad esempio il semplice passare la mano davanti al sensore. Dopo aver ottenuto un accettabile livello di riconoscimento delle gesture, si è passato al “calcolo” del tempo necessario per chiudere/aprire la tenda. Si parla di calcolo perché non si avevano gli strumenti per verificare l’effettiva apertura della tenda, sarebbero serviti altri sensori per risolvere questa problematica, ma per ovviare a ciò si è deciso di inserire un timer all’interno della funzione di movimento; il timer parte quando inizia il movimento e termina alla fine, ottenendo così una stima del tempo di attivazione dello stepper motor da poter utilizzare per calcolare l’altezza della tenda, ovviamente conoscendo la velocità di rotazione dello stepper motor. Dopo vari test si è giunti alla conclusione che il tempo totale necessario per compiere un intero movimento di apertura/chiusura era di 30 secondi, si è quindi impostato un “altezza massima” della tenda ad un valore pari a “30000”, questo perché il timer misurava i millisecondi quindi effettuando l’opportuna operazione (somma o differenza) tra tempo trascorso e altezza precedente si riusciva ad ottenere una stima abbastanza precisa dell’altezza della tenda.

Si è proceduto poi alla verifica della chiusura tramite sensore di luminosità e infine tramite assistente vocale Google; l’assistente riconoscerà le seguenti frasi:

- Per la chiusura: “chiude tenda”, “ho il sole negli occhi” e “non vedo il monitor” al quale risponderà con “certo, provvedo a chiudere iTenda”
- Per l’apertura: “apri tenda”, “voglio vedere le nuvole” e “sono sveglio” al quale risponderà con “certo, provvedo ad aprire iTenda”

Concludiamo dicendo che il progetto ha rispettato in pieno gli obiettivi che ci eravamo prefissati, risulta essere un prodotto di semplice utilizzo, pratico e plug and play, risulta infatti utilizzabile pochi secondi dopo l’attivazione, avendo comunque delle impostazioni di default che permettono l’utilizzo senza dover modificare impostazioni manualmente.

## **2. Progettazione del sistema software**

In questo capitolo analizziamo la seconda parte del progetto, in particolare la parte che riguarda l'interfaccia web, il database e alcune aggiunte al progetto di base.

Come per il capitolo precedente verranno prima analizzate le funzionalità che si vogliono implementare, in seguito tutti gli strumenti utilizzati. L'analisi delle singole funzionalità verrà affrontata nel capitolo successivo insieme all'interfaccia grafica ottenuta.

### **2.1.Funzionalità del sistema**

Il progetto sviluppato in precedenza, sebbene sia funzionante, non permette una completa modifica delle impostazioni se non modificando direttamente il codice, quindi non si ha modo di modificare il funzionamento della tenda durante il suo funzionamento. Risulta pertanto necessaria proprio un'interfaccia semplice e pratica che permetta all'utente finale di modificare a proprio piacimento il funzionamento della tenda.

Tale interfaccia permette di attivare, disattivare o modificare tutte le modalità di gestione di apertura/chiusura della tenda precedentemente analizzate personalizzando il più possibile l'esperienza d'uso del singolo utente.

È stata poi implementata una base di dati contenente informazioni di interesse come le impostazioni dell'utente, lo stato della tenda e le operazioni effettuate (data, ora e causa).

### **2.2. Strumenti utilizzati**

Verranno adesso descritti tutti gli strumenti utilizzati per implementare le funzionalità descritte sopra, in particolare:

- **Ionic** è un SDK open source completo per lo sviluppo di app mobili ibride creato da Max Lynch, Ben Sperry e Adam Bradley di Drifty Co. nel 2013. La versione originale è stata rilasciata nel 2013 e costruita su AngularJS e Apache Cordova. Tuttavia, l'ultima versione è stata ricostruita come un insieme di componenti Web, consentendo all'utente di scegliere qualsiasi framework di interfaccia utente, come Angular, React o Vue.js. Consente inoltre l'uso di componenti ionici senza alcun framework di interfaccia utente. Ionic fornisce strumenti e servizi per lo sviluppo di app Web ibride mobili, desktop e progressive basate su moderne

tecnologie e pratiche di sviluppo Web, utilizzando tecnologie Web come CSS, HTML5 e Sass. Le app mobili possono essere realizzate con queste tecnologie Web e poi distribuite tramite app store nativi per essere installate sui dispositivi utilizzando Cordova o Capacitor.

Gli utenti possono creare le proprie app e personalizzarle per Android, iOS, Windows, desktop (con Electron) o browser moderni. Ionic consente la creazione e l'implementazione di app avvolgendo lo strumento di compilazione Cordova o Capacitor con uno strumento da riga di comando "ionico" semplificato.

Utilizzando quindi questo framework si ha avuto la possibilità di scrivere un unico "codice" trasportabile e comparabile per Android e iOS, personalizzando, anche se di poco, l'esperienza per singolo sistema operativo.

- **HTML e CSS.** In informatica l'HyperText Markup Language (traduzione letterale: linguaggio a marcatori per ipertesti), comunemente noto con l'acronimo HTML, è un linguaggio di markup. Nato per la formattazione e impaginazione di documenti ipertestuali disponibili nel web 1.0, oggi è utilizzato principalmente per il disaccoppiamento della struttura logica di una pagina web (definita appunto dal markup) e la sua rappresentazione, gestita tramite gli stili CSS per le nuove esigenze di comunicazione e pubblicazione all'interno di Internet.

L'HTML è un linguaggio di pubblico dominio, la cui sintassi è stabilita dal World Wide Web Consortium (W3C). È derivato dall'SGML, un metalinguaggio finalizzato alla definizione di linguaggi utilizzabili per la stesura di documenti destinati alla trasmissione in formato elettronico. La versione attuale, la quinta, è stata rilasciata dal W3C nell'ottobre 2014.

- **Angular.** Angular 2+ (o semplicemente Angular) è un framework open source per lo sviluppo di applicazioni web con licenza MIT, evoluzione di AngularJS. Sviluppato principalmente da Google, la sua prima release è avvenuta il 14 settembre 2016.

Angular è stato completamente riscritto rispetto a AngularJS e le due versioni non sono compatibili. Il linguaggio di programmazione usato per AngularJS è JavaScript mentre quello di Angular è TypeScript.

Le applicazioni basate in Angular vengono eseguite dal web browser dopo essere state scaricate dal web server (elaborazione lato client). Questo comporta il risparmio di dover spedire indietro la pagina web al web-server ogni volta che c'è

una richiesta di azione da parte dell'utente. Il codice generato da Angular gira su tutti i principali web browser moderni quali ad esempio Chrome, Microsoft Edge, Opera, Firefox, Safari ed altri.

- **HTTP.** In telecomunicazioni e informatica HTTP (protocollo di trasferimento di un ipertesto) è un protocollo a livello applicativo usato come principale sistema per la trasmissione d'informazioni sul web ovvero in un'architettura tipica client-server. Le specifiche del protocollo sono gestite dal World Wide Web Consortium (W3C). Un server HTTP generalmente resta in ascolto delle richieste dei client sulla porta 80 usando il protocollo TCP a livello di trasporto.

L'HTTP è un protocollo che lavora con un'architettura di tipo client/server: il client esegue una richiesta e il server restituisce la risposta mandata da un altro host. Nell'uso comune il client corrisponde al browser ed il server la macchina su cui risiede il sito web. Vi sono quindi due tipi di messaggi HTTP: messaggi richiesta e messaggi risposta.

- **PHP** (acronimo ricorsivo di "PHP: Hypertext Preprocessor", preprocessore di ipertesti; originariamente acronimo di "Personal Home Page") è un linguaggio di scripting interpretato, originariamente concepito per la programmazione di pagine dinamiche web. L'interprete PHP è un software libero distribuito sotto la licenza PHP.

Attualmente è utilizzato graficamente per applicazioni web lato server, ma può essere usato anche per scrivere script a riga di comando o applicazioni stand-alone con interfaccia. Un esempio di software scritto in PHP è MediaWiki, su cui si basano i progetti wiki della Wikimedia Foundation come Wikipedia.

Nel nostro progetto il **PHP** è stato fondamentale per far interfacciare l'applicazione con il database, inviando una richiesta HTTP al server contenente il file PHP, il server comunica con il database per poi procedere con il percorso inverso.

- **Database** o base di dati (o banca dati, a volte abbreviato con la sigla DB). In informatica si indica un insieme di dati strutturati ovvero omogeneo per contenuti e formato, memorizzati in un computer, rappresentando di fatto la versione digitale di un archivio dati o schedario. In particolare è stato utilizzato ElephantSQL, un servizio di hosting di database PostgreSQL fornito da 84codes AB. ElephantSQL gestisce attività amministrative di PostgreSQL, quali

installazione, aggiornamenti all'ultima versione e gestione del backup; inoltre è anche integrato in diverse piattaforme applicative cloud (dette Paas, PostgreSQL as a Service). ElephantSQL offre vari strumenti per semplificare l'utilizzo del database e che permettono il suo monitoraggio.

### **3. Integrazione, Visualizzazione e Descrizione del risultato finale**

Nella fase conclusiva ci siamo impegnati nell'integrazione dei vari moduli, quindi il riuscire a far comunicare applicazione con dispositivo fisico.

È stato poi necessario verificare il corretto funzionamento del prodotto ottenuto, effettuando diversi test in diverse condizioni.

Si andrà ad analizzare la sua interfaccia grafica, e in parte come è stata realizzata, sia ciò che riguarda il codice e le varie funzionalità. Verranno poi analizzate anche le modifiche effettuate sul codice per l'ESP32, siccome alcune funzionalità non erano state ancora implementate o comunque richiedevano alcune modifiche per poter funzionare al meglio. Analizziamo ora l'applicazione realizzata.

#### **3.1. Splash Screen e Schermata Principale**

L'app si apre mostrando una "splash screen" contenente il logo dell'applicazione che compie un'animazione che durerà qualche secondo, tempo necessario per permettere (nella maggior parte dei casi) una corretta connessione MQTT a mosquitto, terminando poi indipendentemente dalla riuscita della connessione al broker scelto. È stato inoltre inserita una barra di caricamento nel margine superiore dello schermo, non invadente, che scomparirà una volta effettuato correttamente il collegamento MQTT. In seguito, poiché questa è un'applicazione che comanderà uno specifico dispositivo, si è deciso di utilizzare come schermata principale una schermata di Login che, come è giusto aspettarsi, chiede di inserire username e password per autenticarsi e procedere alle schermate successive. Inoltre troviamo i seguenti buttoni:

- **LOGIN**, che verificherà le credenziali e, se corrette, permetterà l'accesso all'utente e caricherà le informazioni relative al suo dispositivo.
- **SIGN IN**, che porterà l'utente ad una schermata dedicata alla registrazione.
- **RESET PASSWORD**, che porterà l'utente ad una schermata dedicata al reset della password.

Analizziamo nello specifico le diverse schermate.

### 3.1.1. Login

Com'è facile aspettarsi, una volta "cliccato" o "tappato" sul bottone di login, l'applicazione provvederà ad effettuare una richiesta HTTP al server contenente il codice PHP che effettuerà una query al database; la richiesta HTTP contiene il nome della funzione del PHP da richiamare, l'username digitato e la password cifrata attraverso l'algoritmo "sha-256".

In base alla risposta ottenuta dal database, l'applicazione agirà di conseguenza, consentendo o meno l'accesso. La corretta autenticazione verrà segnalata dal database rispondendo solo e unicamente con l'username continuando poi con un'ulteriore richiesta verso il database richiedendo le impostazioni della board memorizzate su quest'ultimo che verranno poi utilizzate nelle schermate successive.

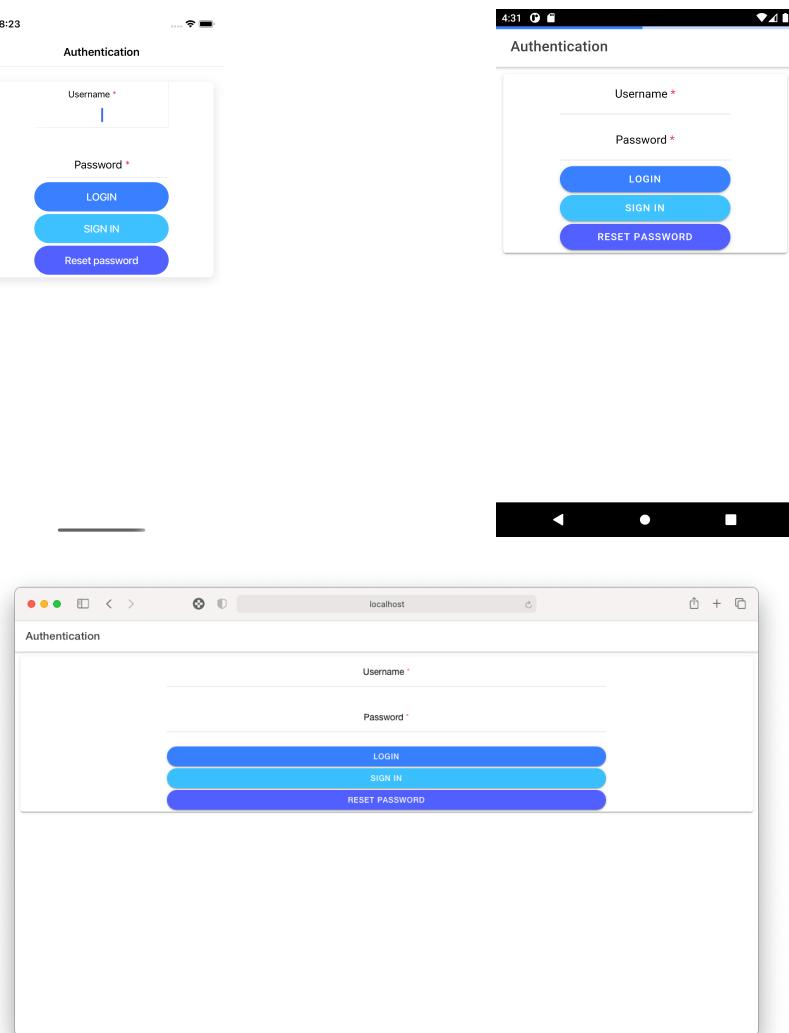


Figura 3. Schermata di Login

In caso di accesso non riuscito, il database risponderà con un messaggio di errore e verrà mostrato un alert, il pulsante “LOGIN” eseguirà una breve animazione di “shake” e verrà eliminato il valore contenuto nel campo password in modo da facilitare l’utente nella reimmissione dei dati.

Dopo aver effettuato il login verrà riprodotta un’animazione portando l’interfaccia verso l’alto.

### 3.1.2. Sign In

Con il tasto “Sign in” si viene portati sulla pagina di registrazione, dove vengono richiesti username, password, città, domanda di sicurezza e risposta a quest’ultima.

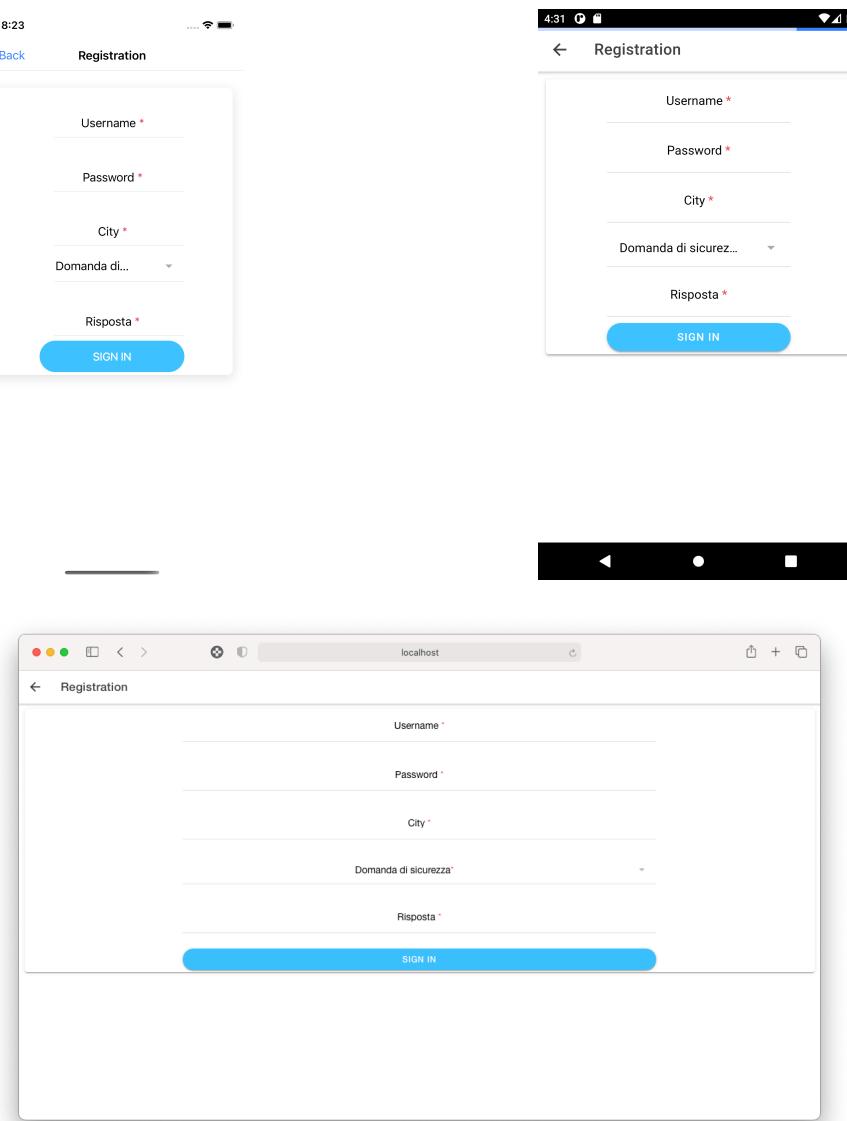


Figura 4. Schermata di Sign in

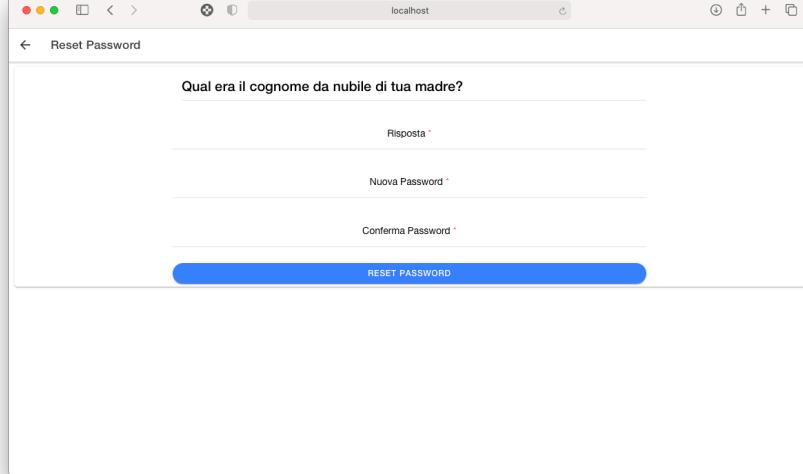
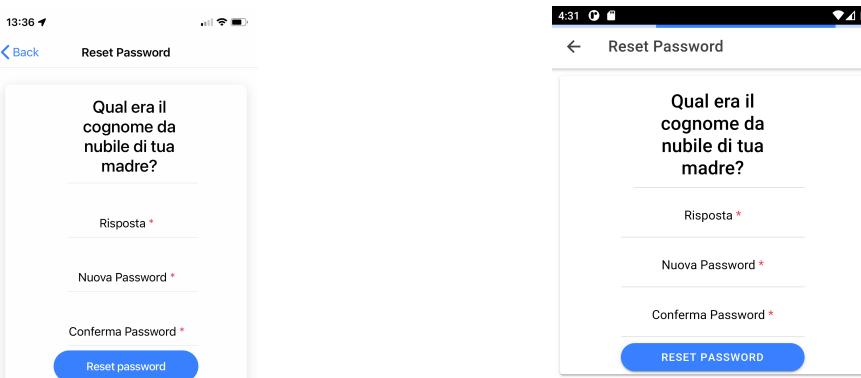
La password e la risposta alla domanda di sicurezza verranno criptate attraverso l'algoritmo "sha-256" prima di essere inviate tramite HTTP, questo per garantire una miglior sicurezza. Nella richiesta HTTP verranno inviate le informazioni inserite dall'utente e il nome della funzione contenuta nel file PHP.

La città viene richiesta poiché bisogna "geolocalizzare" il dispositivo, non essendo dotato di GPS, si è deciso di lasciare all'utente questo compito.

Siccome la password viene cifrata lato applicazione prima di essere inviata al database, si è deciso di permettere solo e unicamente il reset di quest'ultima se dimenticata. Il reset potrà essere effettuato rispondendo alla domanda di sicurezza.

### 3.1.3. Reset password

Questa procedura viene divisa in due schermate distinte, viene dapprima richiesto l'username che, se presente nel database, avrà una e una sola domanda di sicurezza



associata e relativa risposta, procedendo poi alla seconda schermata contenente la sopracitata domanda, uno spazio per inserire la risposta e una doppia immissione della nuova password, verrà verificato direttamente dall'applicazione se le due password inserite sono identiche mostrando, se necessario, un messaggio di alert segnalando l'errore. Ovviamente l'invio di queste informazioni avviene tramite richiesta HTTP contenente il nome della funzione del PHP, l'username, la password e la risposta di sicurezza.

In caso di risposta corretta viene modificata la password relativa a quel determinato account, in caso negativo viene invece riportato l'errore. Anche in questo caso la risposta e la password verranno cifrate prima di essere inoltrate al database, questo per garantire una sicurezza maggiore all'utente ma soprattutto perché la verifica della risposta di sicurezza viene effettuata tramite un confronto tra quella inviata e quella contenuta sul database anch'essa cifrata al momento della registrazione del profilo.

## 3.2. Homepage

Dopo aver effettuato il login verrà caricata l'homepage. Come possiamo vedere dall'immagine essa contiene una Tab superiore, contenente il titolo della schermata che stiamo visualizzando, la schermata di riferimento con tutte le informazioni del caso e una tab inferiore contenente le altre schermate che possiamo visitare.

Verranno ora analizzate le varie schermate disponibili.

### 3.2.1. Impostazioni

Siccome l'interfaccia web è stata ideata per permettere la modifica delle impostazioni della tenda, la prima schermata che viene visualizzata è proprio quella delle impostazioni. In particolare troviamo dei toggle che segnalano l'attivazione o meno di tale funzionalità, uno spazio dedicato alla modifica della città nel quale si trova il dispositivo, la modifica dell'orario di apertura e chiusura con relativo tasto di aggiornamento e infine tre button che permettono l'apertura / chiusura o blocco manuale della tenda.

Ogni qualvolta si esegue un click su un toggle, viene visualizzato un alert che chiede la conferma di tale modifica, in caso di risposta affermativa modificherà il valore anche all'interno del database e notificherà alla board, tramite MQTT, che un'impostazione è cambiata. Il comportamento della board in questa situazione

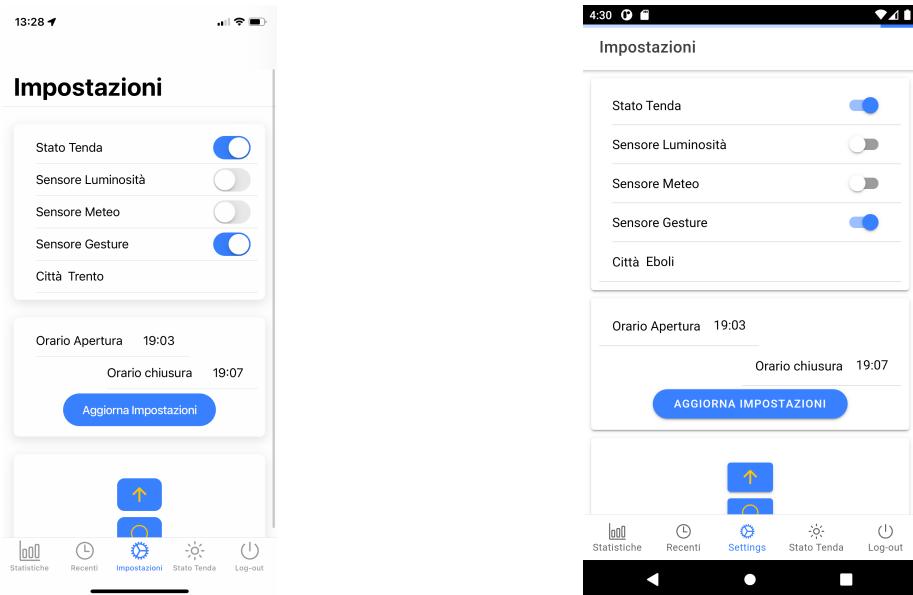


Figura 6. Schermata Impostazioni

verrà analizzato in seguito, in un capitolo dedicato alle nuove funzionalità della board.

Il bottone “Aggiorna impostazioni” ha fondamentalmente lo stesso funzionamento, se premuto mostra un alert (differente da quello mostrato per i toggle) che chiede la conferma alla modifica delle impostazioni, in caso affermativo notifica la modifica alla board e aggiorna tutti i valori all’interno del database.

I bottoni relativi al controllo manuale della tenda non hanno richiesto modifiche al codice della tenda, infatti, alla pressione di uno dei tasti, viene pubblicato un messaggio MQTT sul topic relativo alla tenda in questione contenente il valore

dell'operazione da effettuare. Essendo la tenda iscritta al proprio topic, riceverà il messaggio e effettuerà l'operazione richiesta.

### 3.2.2. Statistiche

Nella schermata statistiche troviamo invece un grafico contenente le operazioni effettuate negli ultimi sette giorni. Partendo dal nome utente, viene effettuata una richiesta HTTP al server contenente il codice PHP che comunicherà al database ottenendo tale informazione. In particolare nel database è stata realizzata una procedura e una View che hanno questo compito, infatti in seguito alla richiesta il database invia il conto delle operazioni per gli ultimi sette giorni; lato client vengono solo analizzati i dati, modificati opportunamente per essere interpretati e mostrato il grafico risultante. In questo modo si va ancora di più a limitare le operazioni che deve effettuare l'applicazione, rendendo tale processo più leggero.

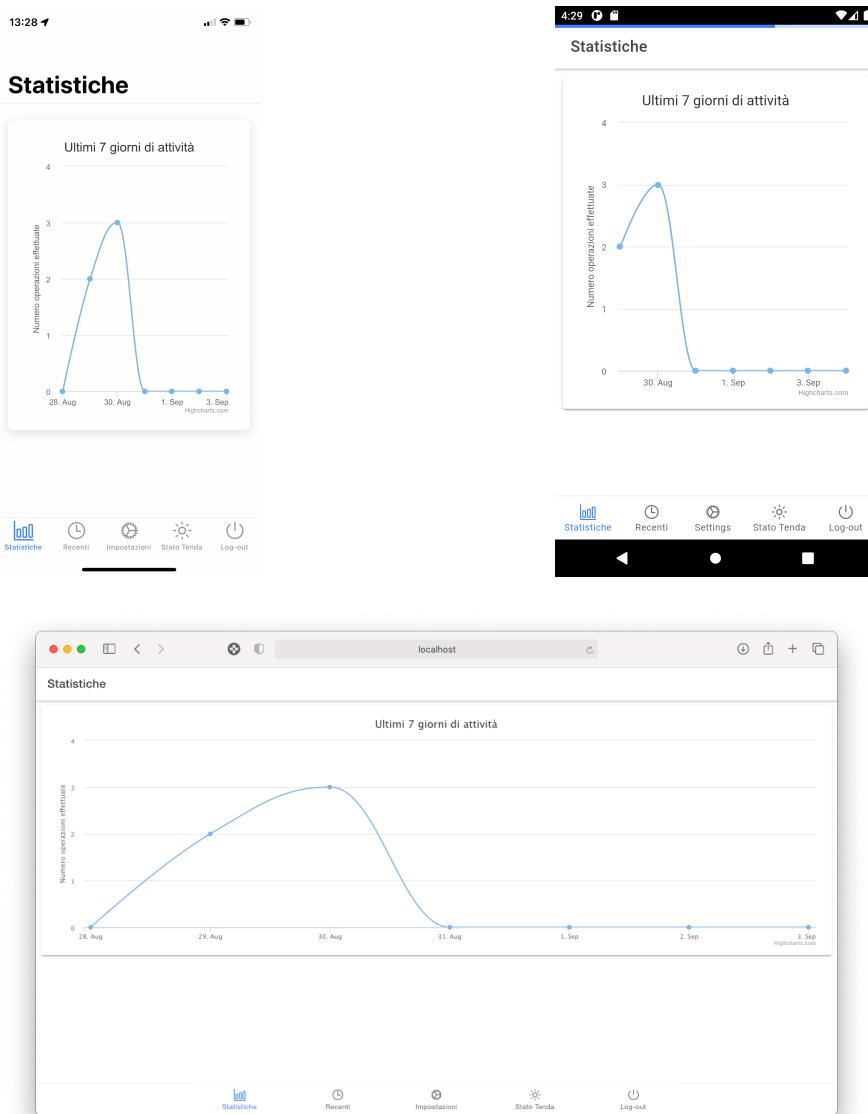


Figura 7. Schermata Statistiche

### 3.2.3. Recenti

Nella schermata “recenti” troviamo le ultime operazioni effettuate dalla tenda, quindi quelle di apertura e chiusura con relativa data, orario e causa.

Come possiamo vedere in foto, per identificare l’operazione effettuata troviamo sulla sinistra un’immagine che ne indica la direzione (freccia verde verso l’alto apertura, freccia rossa verso il basso chiusura). Inoltre nella tab superiore, contenente il nome della schermata, troviamo un counter di tale operazioni.

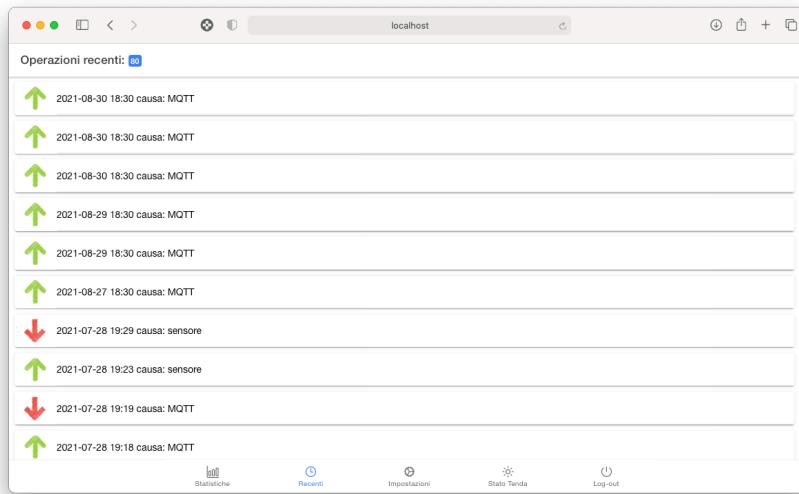
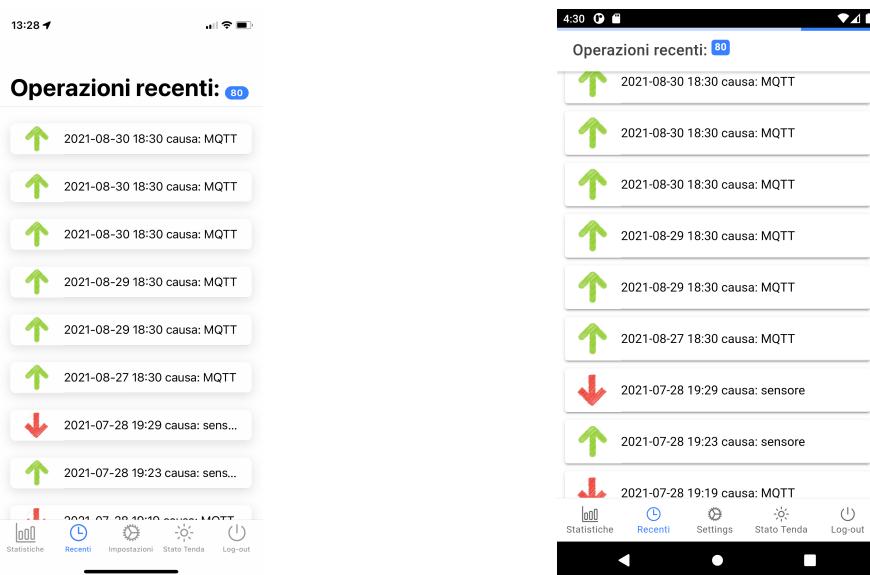


Figura 8. Schermata Recenti

Continuando a scorrere nella schermata, al raggiungimento dell’ultima operazione, verrà mostrato un piccolo messaggio indicando il caricamento di ulteriori dati dal

database che verranno poi aggiunti a quelli già presenti. Nel caso in cui non dovessero essere disponibili altri dati oltre a quelli già caricati, verrà mostrato un alert che segnalera all'utente tale problematica. In particolare il codice che permette tale funzionalità è il seguente. Il “limite” viene inizializzato a 100 per poi aumentare di 50 ogni volta che l'utente raggiunge il fondo della pagina ricercando operazioni più vecchie. Viene richiamata la funzione `list_service` passando il valore del limite attuale e, tramite la parola chiave “await”, si aspetta la fine di tale procedura. Si procederà poi con lo “shift” del risultato ottenuto, che verrà poi concatenato con quello già presente.

```

loadMoreData(event) {
  setTimeout(async () => {
    this.limit += 50;
    await this.comp.list_service(this.limit);
    setTimeout(() => {
      for (let i = 0 ; i < this.data.length ; i++ )
        this.comp.result.shift();
      if (this.comp.result.length == 0 || this.limit >= 1000){
        event.target.disabled = true;
        this.comp.presentAlert("Ho caricato tutti i dati presenti ");
      }
      console.log(this.limit);
      this.data = this.data.concat(this.comp.result);
      event.target.complete();
    }, 1500);
  }, 500);
}

```

Figura 9. Funzione caricamento ulteriori operazioni della scheda

Come già detto in precedenza, le informazioni contenute in questa schermata vengono richieste al database, in particolare viene effettuata una richiesta HTTP al server contenente il codice PHP che si occuperà di richiedere tali informazioni al database, e verrà inserito il “limit” di valori da far restituire dal database in modo tale da non caricare subito tutti gli elementi disponibile ma solo un numero prestabilito per volta.

### 3.2.4. Stato Tenda

L'ultima schermata che andremo ad analizzare è la schermata "stato tenda".

In questa schermata possiamo trovare in ordine:

- Altezza tenda, contenente in percentuale lo stato di apertura o chiusura della tenda (0% tenda completamente abbassata, 100% completamente alzata) recuperata opportunamente dal database.
- Clima, contenente il clima relativo alla città inserita in precedenza.
- Temperatura, Temperatura min, Temperatura max, Tramonto e Alba facenti riferimento sempre alla città inserita in precedenza.

Queste ultime informazioni vengono recuperate effettuando una richiesta HTTP, ad "openweathermap", contenente la città del quale si vuole conoscere il meteo.

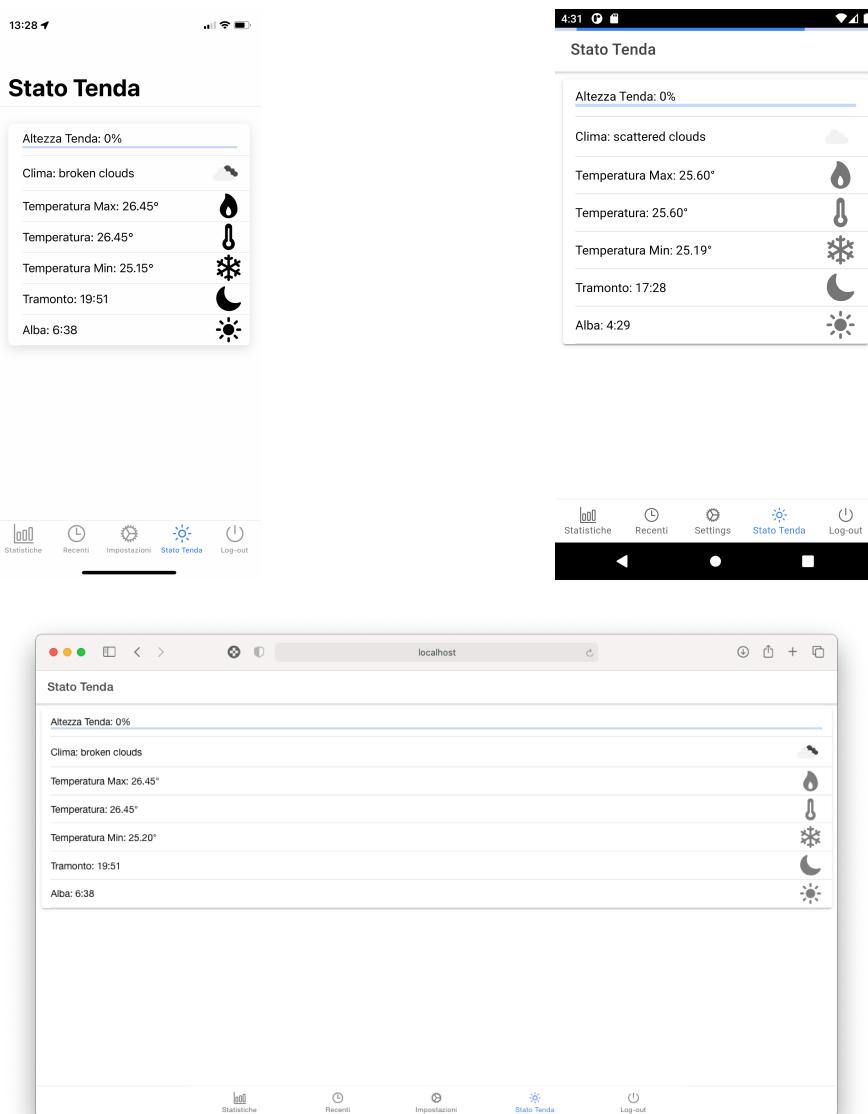


Figura 10. Schermata Stato Tenda

Inoltre a destra di ogni elemento sono state inserite delle immagini autoesplicative, solo quella relativa al clima modificherà in base a quest'ultimo.

### 3.3. ESP32

In questo capitolo andremo ad analizzare quelle che sono state le modifiche e le aggiunte nel codice della board per permettere a quest'ultima di funzionare nel modo corretto, riuscendo quindi ad aggiornare le proprie impostazioni e attivando le nuove funzionalità da implementare.

- **checkHours:** una funzione dedicata al controllo dell'orario, questo per verificare se l'orario attuale corrisponde con l'orario di chiusura o di apertura. Come possiamo apprezzare dal codice, viene effettuata una richiesta HTTP get su un determinato sito che, come suggerisce il nome, permette di recuperare l'orario. Si effettua innanzitutto una modifica sul valore riportato, essendo il fuso orario del sito diverso dal nostro, per poi essere confrontato con l'orario di apertura e chiusura scelto dall'utente.

```
def checkHours():  
    while True:  
        global operation  
        global causa  
        print("Verifico l'orario di chiusura e apertura")  
        response = requests.get("http://just-the-time.appspot.com/")  
        date = response.content  
        date = date[:16] + "+00:00"  
        date = (datetime.fromisoformat(str(date))).add(datetime.timedelta(hours=2)).tuple()  
        if (date[3] - int(orariochiusura[:2]) == 0) and (date[4] - int(orariochiusura[3:5]) == 0):  
            print("chiudo tenda causa orario")  
            causa = "Orario"  
            operation = -1  
        if (date[3] - int(orarioapertura[:2]) == 0) and (date[4] - int(orarioapertura[3:5]) == 0):  
            print("apro tenda causa orario")  
            causa = "Orario"  
            operation = 1  
        print("Ho verificato l'orario di chiusura e apertura")  
        sleep(50000)
```

Figura 11. Funzione checkHours

- **inizializzazioneTenda:** questa funzione recupera le impostazioni dal database, viene effettuata una richiesta HTTP di tipo post contenente l'username relativo alla board e la funzione del PHP da richiamare (in questo caso "settings"). Vengono verificati i valori riportati dal database, impostate le variabili locali in modo da

attivare o meno il funzionamento di determinate operazioni, memorizzata la città

```
def inizializzazioneTenda():
    print("inizializzazione")
    global flag_lux
    global flag_movimento
    global city
    global flag_weather
    global flag_sensore
    global orariochiusura
    global orarioapertura
    global high_tenda
    response = requests.post(ip, {"function": "settings", "username": username})
    js = json.loads(response.content)
    print(js)
    if js[0]['funzionamentotenda'] == 'f':
        flag_movimento = 0
        flag_lux = 0
        flag_weather = 0
        flag_sensore = 0
    else:
        flag_movimento = 1
        if js[0]['luminosita'] == 't':
            flag_lux = 1
        else:
            flag_lux = 0
        if js[0]['gesture'] == 't':
            flag_sensore = 1
        else:
            flag_sensore = 0
        if js[0]['meteo'] == 't':
            flag_weather = 1
        else:
            flag_weather = 0
    city = js[0]['citta']
    orarioapertura = js[0]['orarioapertura']
    orariochiusura = js[0]['orariochiusura']
    high_tenda = float(js[0]['altezzatenda']) * 30000
    print("inizializzazione.completata")
```

Figura 12. Funzione inizializzazioneTenda

nel quale si trova la board e l'orario di apertura e chiusura.

- **sendToDb:** funzione che si occupa esclusivamente di notificare al database che tipo di operazione è stata effettuata dalla tenda, apertura o chiusura, inviando, sempre tramite richiesta HTTP, l'username, l'operazione e la causa. La data e l'orario verranno poi aggiunti direttamente dal database.
- **updateStatus:** funzione che si occupa esclusivamente di aggiornare all'interno del database l'altezza della tenda, opportunamente modificata per poter essere facilmente interpretata dal database e dall'applicazione, inviata sempre tramite richiesta HTTP contenente l'username e la nuova altezza tenda.

Viene prima verificata l'altezza della tenda ed impostata ad un valore specifico, poiché il valore viene calcolato attraverso un timer e non risulta perfettamente accurato, di conseguenza viene verificato il valore ottenuto e per poi essere impostato ad un valore standard di massimo o minimo, e in seguito viene effettuata una richiesta HTTP di tipo post contenente i dati necessari per andare ad aggiornare quello che è il valore dell'altezza della tenda, quindi tale valore, l'username e ovviamente il nome della funzione PHP da eseguire.

```
def updateStatus():
    global high_tenda
    if high_tenda > 30000:
        high_tenda = 30000
    if high_tenda < 0:
        high_tenda = 0
    print("aggiorno il database con un'altezza di ", high_tenda)
    high = round(high_tenda/30000,2)
    response = requests.post(ip, data={"function": "updateHigh", "username": username, "high": high})
    print("valore inviato al db: ", high)
    print(json.loads(response.content))
    sleep(1000)
```

Figura 13. Funzione updateStatus

- **Check\_weather:** come suggerisce il nome, tale funzione verifica il meteo nella città, viene pertanto effettuata una richiesta HTTP get al sito che possiamo vedere nella figura sottostante, restituendo poi il valore inviato come risposta dal sito. Questa funzione infatti viene richiamata da un'altra funzione “check\_lux\_weather” che vedremo in seguito.

```
def check_weather():
    global city
    try:
        print("Connessione all'url per le previsioni meteo...")
        params = {
            "APPID": api_key,
            "q": city #-----> your city
        }
        url = "http://api.openweathermap.org/data/2.5/weather"
        return requests.get(url, params=params)
        break
    except Exception as e:
        print(e)
```

Figura 14. Funzione Check\_weather

- **check\_lux\_weather**: tale funzione richiama la precedentemente citata “Check\_weather” e implementa anche la funzione “check\_lux”, presente all’interno del codice della prima parte del progetto. Si occupa quindi di verificare i dati restituiti dalla funzione check\_weather e verificare il valore restituito dal fotoresistore, nel caso in cui uno dei due controlli risultasse vero viene comandata la chiusura della tenda.

```

if check_lux_weather():
    global high_tenda
    global flag_weather
    global operation
    global flag_lux
    global causa
    try:
        while True:
            if high_tenda > 0 and flag_weather:
                response = check_weather()
                if response.status==200:
                    print("Richiesta http per le previsioni meteo andata a buon fine")
                    js = json.loads(response.content)
                    print("Weather:",js["weather"][0]["id"],js["main"]["temp"]-273,"degrees")
                    print("-----")
                    if(js["weather"][0]["id"] >= 200 and js["weather"][0]["id"] < 800):
                        operation = -1
                        causa = "meteo"
                        print("Abbassamento tenda a causa delle previsioni meteorologiche")
                        publish_to_self("It's raining, closing the roller blind",1)
                    if high_tenda > 0 and flag_lux:
                        sleep(2000)
                        if digitalRead(D4) == 0:
                            operation = -1
                            causa = "luminosita"
                            print("Luce rilevata, abbassamento della tenda")
                            publish_to_self("Light detected",1)
                            sleep(30000)
                        sleep(5000)
    except Exception as e:
        print("Problema durante il controllo delle previsioni meteo",e)

```

Figura 15. Funzione check\_lux\_weather

Le sopracitate funzioni sono le modifiche principali effettuate al codice del microcontrollore, le altre modifiche effettuate all’interno del codice risultano marginali e di poco interesse, infatti la maggior parte sono piccoli aggiustamenti ai nomi delle variabili, l’aggiunta di thread per richiamare le nuove funzioni implementate e altre chiamate a funzioni aggiunte nel codice precedente. È stato poi verificato il corretto funzionamento del prodotto e di tutte le sue funzionalità dopo le ultime modifiche effettuate.

## 3.4. Database e PHP

In questo capitolo verrà discussa e analizzata, brevemente, la parte implementativa del Database e del PHP, che è stata realizzata dal mio collega e sarà da lui

approfondita. Come già detto in precedenza, si è utilizzato ElephantSQL per gestire la base di dati in cloud dove evidenziamo quattro tabelle:

- **Utenti**: che ha il compito di salvare i dati relativi ad ogni utente registrato tramite l'app e i loro valori di impostazione della tenda.
- **Operazioni**: che ha il compito di salvare le operazioni effettuate da ogni utente e le relative informazioni.
- **Domande**: che viene utilizzata per il salvataggio delle domande di sicurezza degli utenti.
- **ViewConZero**: che propone un listato delle operazioni effettuate negli ultimi sette giorni per uno specifico utente. In particolare, per realizzare tale tabella è stata creata una procedura che elimina tutti i dati all'interno di quest'ultima, e la "ripopola" con i dati di interesse.

Per quanto riguarda invece il PHP è stato realizzato un file contenente tutte le funzioni necessarie per interfacciarsi con il database, ogni funzione corrisponde ad una particolare query. La scelta delle funzioni da eseguire viene effettuata verificando il valore “function” passato nella richiesta HTTP di tipo post fatta dall'applicazione o dalla board. Quindi per ogni funzione PHP, corrisponderà una post con uno specifico valore in “function” e poi tutti gli altri parametri necessari per eseguire la query (come ad esempio l'username, la password, la risposta alla domanda di sicurezza, l'altezza della tenda).

Vediamo ad esempio come è stata realizzata la funzione “register” all'interno del file PHP.

```
if($_POST['function']=='register'){
    $query= "INSERT into Utenti(username,password,citta) values ('".$_POST['username']."','".$_POST['password']."','".$_POST['city']."');";
    pg_send_query($con,$query);
    $res=pg_get_result($con);
    $state=pg_result_error_field($res, PGSQL_DIAG_SQLSTATE);
    if ($state==0){
        $query2="INSERT into Domande(username,domanda,risposta) values ('".$_POST['username']."','".$_POST['question']."','".$_POST['reply']."');";
        pg_send_query($con,$query2);
        $res2=pg_get_result($con);
        $state2=pg_result_error_field($res2, PGSQL_DIAG_SQLSTATE);
        if($state2==0)
            echo json_encode("Sign up confirmed");
        else
            echo json_encode("Error during registration");
    }
    else
        echo json_encode("Error during registration");
}
```

Figura 16. Funzione register

## **Conclusioni**

Il prodotto finale inquadrabile nel panorama della Domotica e dell'IoT consente una totale personalizzazione e automatizzazione di una tenda a rullo, proponendo un prodotto facile da utilizzare e semplice nel funzionamento. Il progetto condotto in team, ha previsto una verticalizzazione delle specifiche realizzati e ha accolto il mio contributo in relazione alla parte di interfaccia grafica, Web, iOS e Android, orientando ogni singola fase alla ricerca e alla realizzazione di un prodotto il più leggero possibile indipendentemente dalla piattaforma utilizzata, sfruttando la potenza di calcolo dei server contenenti il database e il file PHP. Il risultato di tale ricerca è un prodotto semplice e pratico, facile da comprendere anche ai meno esperti con un'interfaccia personalizzata in base alla piattaforma finale. L'aspetto più apprezzabile, sia in fase di processo che di realizzazione del prodotto, è risultato essere l'uso di tecnologie innovative, stimolando il desiderio di ricerca in un settore estremamente affascinante e denso di opportunità per ulteriori approfondimenti. Si è riusciti ad abbracciare numerosi aspetti dell'informatica odierna, passando dalla realizzazione di un prodotto fisico, all'implementazione di un'interfaccia grafica per fisso e mobile alla realizzazione di un database in cloud.

## **Bibliografia e Sitografia**

Paolo Atzeni, Stefano Ceri, Piero Fraternali, Stefano Paraboschi, Riccardo Torlone  
“Basi di Dati”, V Edizione.

Web Languages. URL: <https://www.w3schools.com/>.

PHP documentazione. URL: <https://www.php.net/docs.php>.

ElephantSQL documentation. URL: <https://www.elephantsql.com/docs/>.

Python documentation. URL: <https://docs.python.org/3/>.

Ionic documentation. URL: <https://ionicframework.com/docs>.

Zerynth documentation. URL: <https://docs.zerynth.com/latest/>.

## **Ringraziamenti**

Al termine di questo percorso esprimo la mia gratitudine al Prof. Pierluigi Ritrovato per la competenza e la professionalità che da sempre offre ai propri studenti, per il supplemento di disponibilità che mi ha riservato per la realizzazione di questo lavoro e per l'arricchimento della mia formazione attraverso un percorso di tirocinio che si è rivelato stimolante.

Ringrazio tutta la mia famiglia per essere riuscita ad aiutarmi in questo percorso senza condizionarmi mai, per avermi lasciato la giusta libertà per imparare.

Grazie ai miei nonni, a chi oggi avrei voluto fossero ancora qui, a festeggiare con orgoglio il mio traguardo.

Un ringraziamento particolare va anche al mio collega e concittadino Vito con il quale ho affrontato il progetto di tirocinio e la maggior parte di questo percorso formativo, formando un ottimo gruppo di lavoro dotato di una grande intesa ed efficienza.

Grazie agli amici e colleghi Gerardo, Renato e i due Marco, che mi hanno accompagnato ma soprattutto aiutato molto in questi anni ed è sicuramente grazie a loro se sono riuscito a raggiungere questo traguardo.

Ringrazio Mattia E Camilla, con i quali ho condiviso molti progetti nell'arco degli studi, anche in compagnia di Vito, si è formato un bellissimo gruppo di lavoro con il quale si sono riempite le giornate del LockDown, realizzando progetti e ottenendo ottime soddisfazioni.

Ringrazio tutti i componenti del gruppo "Friendz" con i quali ho condiviso gli anni del liceo e insieme ai quali ho affrontato le esperienze più belle di questi anni, in particolare vorrei ringraziare Piercarmine e Pierfrancesco, che sono forse quelli che mi hanno sopportato di più in questi anni.

Infine, un ringraziamento va agli amici di tutti i giorni che mi hanno supportato e sopportato durante questi tre anni, rendendo ogni giornata speciale ed indimenticabile.