



Università Politecnica delle Marche

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

Acquisizioni e trasmissioni di misure ottenute tramite sensori utilizzando l'ESP32 come client e comunicazione con il broker MQTT

Corso di Sistemi Operativi Dedicati

Professore

Prof. Daniele Marcozzi

Gruppo 8

Eris Prifti
Wafa Mohammad

Anno accademico 2022-2023

Indice

1	Introduzione	2
2	Componenti	2
2.1	Broker MQTT	3
2.2	ESP32	3
2.3	Sensore temperatura e pressione BMP280	4
2.4	Sensore di luminosità BH1750	4
2.5	Modulo RTC PCF8523	5
3	Implementazione	5
3.1	Implementazione schema elettrico	5
3.2	MQTT	7
3.3	Arduino	8
3.3.1	Librerie e inizializzazione iniziale	9
3.3.2	Task per la lettura dei sensori	10
3.3.3	Task per l'invio dei dati al broker	10
3.3.4	Altre funzioni	11
3.4	Node Red	11
3.4.1	Installazione	11
3.4.2	Dashboard	12
3.5	MySQL	17
4	Problemi durante la realizzazione e soluzioni	20
5	Implementazioni future	20
6	Conclusioni	21

1 Introduzione

Il progetto consiste nell'andare ad utilizzare il sistema operativo Ubuntu versione 22.04 basato su Linux, visto a lezione, per andare a sviluppare attraverso un microcontrollore ESP32 un meccanismo di comunicazione tra quest'ultimo e il sistema operativo installato in una macchina virtuale. Praticamente non verrà utilizzato solo l'ESP32 come componente fisico ma ci saranno anche altri elementi come un sensore per la misura della luminosità, un sensore per la misura della temperatura e pressione ed inoltre sarà disponibile un modulo RTC utilizzato per tenere traccia dell'ora e della data.

Quello appena descritto si può riassumere nel seguente schema in figura 1:

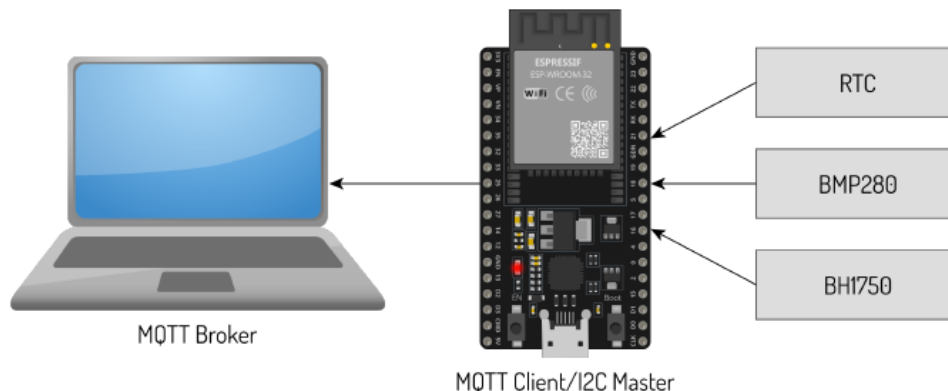


Figura 1: Schema a blocchi

La scheda ESP32 avrà il compito di acquisire le misure dai vari sensori e comunicarle mediante protocollo MQTT al broker. Il coordinamento dei vari task che l'ESP32 dovrà svolgere sarà affidato a FreeRTOS. Il broker MQTT può essere implementato all'interno di un qualsiasi laptop/macchina virtuale con sistema operativo Linux. I dati registrati all'interno del server MQTT dovranno essere visualizzabili mediante browser WEB. Inoltre i dati verranno salvati su un database in modo che dal browser WEB è possibile andare a vederli. Nello specifico l'ESP32 dovrà periodicamente leggere i dati dai sensori a sua disposizione e associare ad essi un timestamp ricavato mediante interrogazione del modulo RTC. I dati così ricavati dovranno essere inviati mediante protocollo MQTT al broker.

2 Componenti

Il sistema utilizzato è composto dai seguenti componenti:

- broker MQTT
- ESP32
- Sensore di temperatura e di pressione BMP280
- Sensore di luminosità BH1750
- Modulo RTC PCF8523 per il timestamp

I vari sensori utilizzano per la comunicazione il protocollo I2C. Il protocollo permette la comunicazione di dati tra due o più dispositivi I2C utilizzando un bus(canale di comunicazione) a due fili, più uno per il riferimento comune di tensione; in tale protocollo le informazioni sono inviate serialmente usando una linea

per i dati (SDA: Serial Data line) ed una per il Clock (SCL: Serial Clock line) e inoltre deve essere presente una terza linea: la massa, comune a tutti i dispositivi.

2.1 Broker MQTT

MQTT è un protocollo di messaggistica basato su standard, o un insieme di regole, utilizzato per la comunicazione tra macchine. I sensori intelligenti, i dispositivi indossabili e altri dispositivi di Internet delle cose (IoT) devono in genere trasmettere e ricevere dati su una rete con risorse limitate e larghezza di banda limitata. Questi dispositivi IoT utilizzano MQTT per la trasmissione dei dati, in quanto è facile da implementare e può comunicare i dati IoT in modo efficiente. MQTT supporta la messaggistica tra dispositivi e cloud e tra cloud e dispositivo. Il broker MQTT è il sistema backend che coordina i messaggi tra i diversi client. Le responsabilità del broker comprendono la ricezione e il filtraggio dei messaggi, l'identificazione dei client sottoscritti a ciascun messaggio e l'invio dei messaggi a questi. E inoltre responsabile di altri compiti quali:

- Autorizzare e autenticare i client MQTT
- Passare i messaggi ad altri sistemi per ulteriori analisi
- Gestire i messaggi persi e le sessioni dei client

Nella figura 2 si può vedere il funzionamento in maniera schematica del MQTT-Broker.

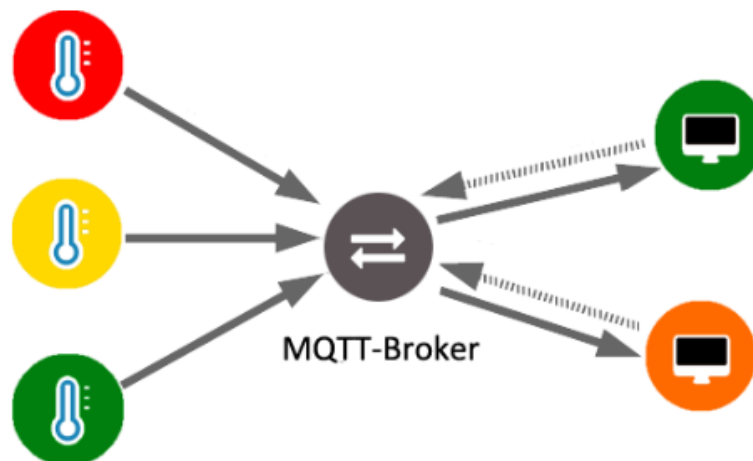


Figura 2: Schema Mqtt

2.2 ESP32

L'ESP32 è un microcontrollore, una scheda versatile e potente per la progettazione di dispositivi connessi e di sistemi IoT. Con le sue funzionalità avanzate e le periferiche integrate, consente di creare soluzioni innovative e personalizzate per soddisfare le esigenze specifiche dei progetti. Una delle principali caratteristiche dell'ESP32 è la sua capacità di elaborazione parallela, che gli consente di gestire più processi contemporaneamente. Questo significa che è possibile eseguire diverse operazioni, come la connessione WiFi, l'elaborazione di dati, il controllo di dispositivi esterni, tutti allo stesso tempo. Un'altra caratteristica interessante dell'ESP32 è la

sua capacità di funzionare come un dispositivo dual-mode, ovvero può essere utilizzato sia come un dispositivo autonomo che come un modulo slave in un sistema più grande. Ciò lo rende particolarmente utile per l'Internet delle cose (IoT), poiché consente di creare dispositivi connessi in modo semplice e conveniente.

Nella figura 3 si può osservare l'ESP32.

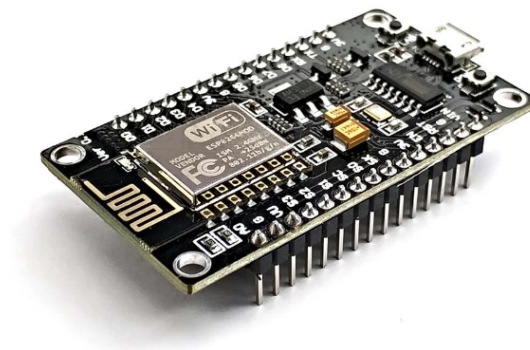


Figura 3: ESP32

2.3 Sensore temperatura e pressione BMP280

Un sensore ambientale con temperatura e pressione barometrica, ideale per tutti i tipi di rilevamento meteorologico e può essere utilizzato sia in I2C (per un cablaggio semplice e facile) che in SPI (per poter collegare un gruppo di sensori portando a zero il rischio delle collisioni di indirizzi I2C). Tale sensore è in grado di misurare la pressione barometrica con una precisione assoluta di ± 1 hPa e la temperatura con una precisione di $\pm 1,0^{\circ}\text{C}$. Poiché la pressione cambia con l'altitudine e le misure di pressione sono così buone, è possibile utilizzarlo anche come altimetro con una precisione di ± 1 metro.

Nella figura 4 si può osservare il sensore utilizzato per la misura della temperatura e della pressione.

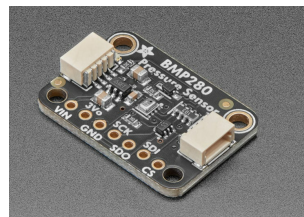


Figura 4: Sensore temperatura, umidità e pressione

2.4 Sensore di luminosità BH1750

Un sensore di luce ambientale in grado di rilevare la quantità di luce in un ambiente. Il BH1750 fornisce misurazioni della luce a 16 bit in lux, l'unità SI per la misurazione della luce, facilitando il confronto con altri valori, come i riferimenti e le misurazioni di altri sensori. Il BH1750 è in grado di misurare da 0 a 65K+ lux, ma con una certa calibrazione e una regolazione avanzata del tempo di misurazione, può essere convinto di misurare fino a 100.000 lux.

Nella figura 5 si può osservare il sensore utilizzato per la misura della luminosità.

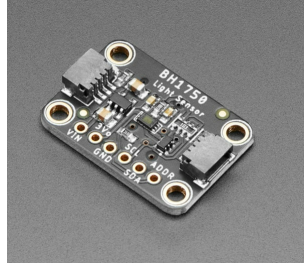


Figura 5: Sensore luminosità

2.5 Modulo RTC PCF8523

Si tratta di un ottimo orologio in tempo reale (RTC) a batteria che consente al vostro microcontrollore di tenere traccia dell'ora anche se viene riprogrammato o se viene a mancare l'alimentazione. Perfetto per il datalogging, la costruzione di orologi, la marcatura temporale, i timer e gli allarmi, ecc. RTC PCF8523 può funzionare con alimentazione a 3,3V o 5V.

Nella figura 6 si può osservare il modulo RTC utilizzato per il timestamp.

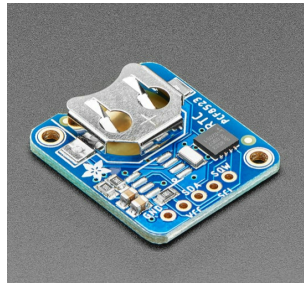


Figura 6: Modulo RTC

3 Implementazione

La realizzazione del progetto è stata divisa nelle seguenti fasi:

- Nella prima fase sono stati messi insieme i vari componenti del progetto.
- Nella seconda fase è stato realizzato il broker MQTT.
- Nella terza fase si ha la realizzazione del codice attraverso l'Ide di Arduino.
- Nella quarta fase è stato utilizzato Node-Red per andare a costruire una dashboard.
- Nella quinta fase è stato utilizzato Xampp per quanto riguarda la creazione di un database e il salvataggio dei dati in una tabella.

Queste fasi verranno viste in modo approfondito nei seguenti sottocapitoli.

3.1 Implementazione schema elettrico

Lo schema elettrico è stato molto semplice da realizzare attraverso l'utilizzo del sito di Adafruit, ente che ha prodotto questi sensori e continua a produrli. Infatti nei datasheet di Adafruit è possibile trovare come collegare i vari pin dell'ESP32 ai vari sensori.

I due sensori, il BMP280 e il BH1750, vengono collegati in serie attraverso il cavo Qwiic JST SH 4-Pin dove:

- Il colore nero viene utilizzato per il GND.
- Il colore rosso viene utilizzato per l'alimentazione 3.3V.
- Il colore blu per l'SDA utilizzato per l'implementazione del protocollo I2C.
- Il colore giallo per SCL utilizzato per l'implementazione del protocollo I2C.

Se vengono invertiti i collegamenti con i colori ai pin dell'ESP32, naturalmente non funziona il tutto ed inoltre si ha un riscaldamento dei dispositivi che in certi casi potrebbe arrivare anche a bruciarli. Nella figura 7 si può osservare il cavo descritto sopra:

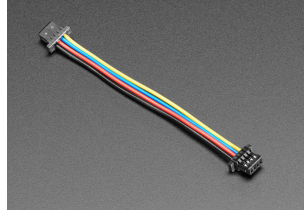


Figura 7: Cavo 4 pin

Per quanto riguarda tutto lo schema elettrico e i collegamenti dei sensori ai pin dell'ESP32 si hanno i seguenti collegamenti:

- Il pin 3.3V dell'ESP32 viene collegato alla Vin del sensore BMP280 e HB1750 e anche alla VCC del sensore RTC-PCF8523.
- Il pin GND dell'ESP32 viene collegato ai pin GND dei vari sensori.
- Il pin G22 dell'ESP32 viene collegato ai pin SCL dei vari sensori.
- Il pin G21 dell'ESP32 viene collegato ai pin SDA dei vari sensori.

Nella figura 8 si può osservare come i collegamenti sono stati fatti attraverso il software **Fritzing** per avere una maggiore idea del quadro complessivo e poi nella figura 9 si può notare come i collegamenti, derivati dall'implementazione attraverso il software, sono stati realizzati nei dispositivi fisici.

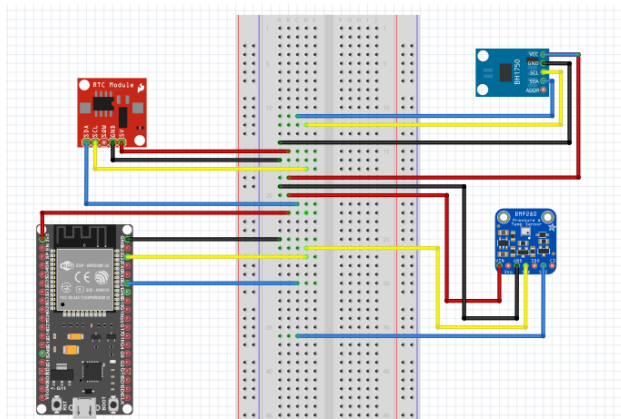


Figura 8: Schema dei collegamenti tramite Fritzing

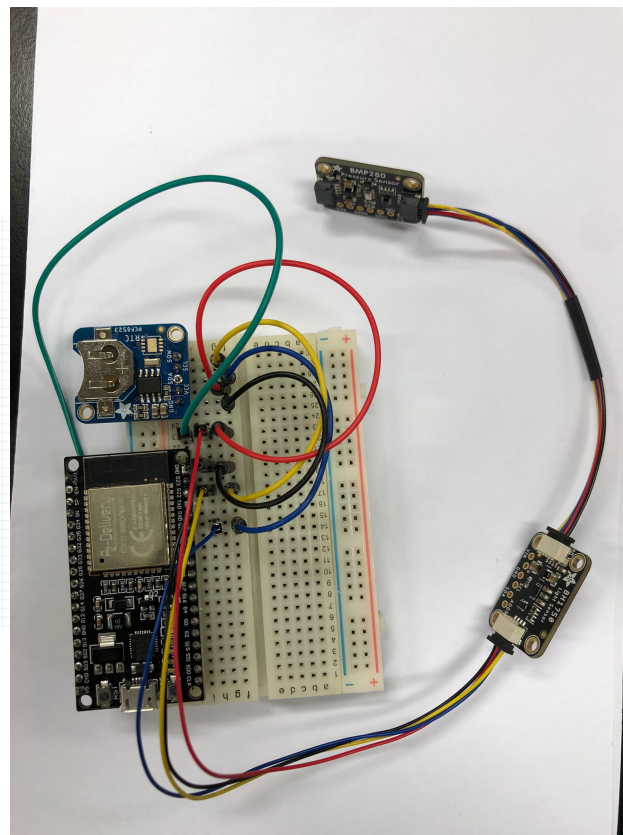


Figura 9: Schema fisico dei collegamenti

3.2 MQTT

MQTT (Message Queuing Telemetry Transport) è un protocollo di messaggistica leggero e di facile implementazione progettato per le comunicazioni tra dispositivi con risorse limitate e connessioni di rete instabili. E' stato deciso di utilizzare come broker MQTT **Mosquitto**. Di seguito viene riportata l'installazione di Mosquitto eseguito da terminale in Ubuntu. Per installare Mosquitto come broker MQTT si possono seguire i seguenti passaggi:

1. Aprire il terminale sul sistema Ubuntu.
2. Assicurarsi di avere accesso a Internet per poter scaricare il software necessario.
3. Utilizzare il gestore dei pacchetti del proprio sistema per installare il broker MQTT **mosquitto** e **mosquitto-clients** (nel nostro caso è stato utilizzato su Ubuntu).

```
sudo apt update -y && sudo apt install mosquitto mosquitto-clients -y
```

4. Durante l'installazione, potrebbe essere necessario la conferma dell'installazione e l'accettazione di eventuali dipendenze aggiuntive richieste dal broker Mosquitto.
5. Una volta completata l'installazione, il broker MQTT sarà in esecuzione sul nostro sistema Linux. Attraverso il seguente comando (sempre sul terminale) è possibile controllare lo stato del servizio:


```
sudo systemctl status mosquitto
```

6. Se non dovesse risultare attivo, attraverso il seguente comando comando si renderebbe tale:

```
sudo systemctl enable mosquitto
```

3.3 Arduino

Una volta fatto il collegamento tra i vari sensori e l'ESP32, si è proceduto andando ad implementare il codice che verrà caricato successivamente sul controllore attraverso l'Ide di Arduino. Per prima cosa è stato necessario installare Arduino Ide dal sito di Arduino. Una volta installato, un'operazione necessaria è quella di scaricare la libreria della board del microcontrollore ESP32 attraverso il seguente procedimento:

1. Una volta aperto il programma Arduino Ide si va su file.
2. Preferences.
3. Su additional board manager URLs si incolla il seguente link preso da Github **https://espressif.github.io/arduino-esp32/package_esp32_index.json**.

Una volta fatto ciò, troveremo la board nella sezione di **Tools** e può essere selezionata ed inoltre va selezionato la porta del seriale in modo da caricare il codice all'interno dell'ESP32.

In seguito, si devono scaricare le varie librerie utilizzate per l'implementazione totale del progetto sia per quanto riguarda i sensori che per quanto riguarda la parte dell'ESP32 utilizzato come client che comunica al broker MQTT. Le librerie utilizzate per tutto ciò vengono scaricate nella sezione **Tools**, dopo si va su **Manage libraries**; sono state scaricate le seguenti librerie:

- La libreria **Adafruit BMP280 library** per il sensore della temperatura e pressione.
- La libreria **hp_BH1750** per il sensore della luminosità.
- La libreria **RTCLib** per quanto riguarda il sensore dell'ora e della data.
- La libreria **PubSubClient** per la comunicazione del nostro client con il broker MQTT.
- Altre librerie che sono presenti nel codice e non vengono citate qui derivano dal download della scheda ESP32.

Queste librerie scaricate contengono degli esempi i quali sono stati utilizzati inizialmente per vedere se i sensori funzionavano e per vedere se l'indirizzo I2C (esempio: 0x23 indirizzo del BH1750, 0x77 indirizzo del BMP280) dei vari sensori funzionava e successivamente dopo aver installato il broker MQTT e creato una prima bozza di dashboard nel browser web è stato provato il funzionamento della libreria **PubSubClient**. Tutte queste prove hanno avuto risultati positivi.

Una volta visto che tutto funzionava si è fatta una prima prova dove i 3 sensori funzionavano insieme e stampavano nella porta seriale e ciò ha avuto esito positivo.

Successivamente è stata fatta una prova dove si andava ad utilizzare il **void loop**, senza **task**, per vedere se si riusciva ad avere una connessione con il broker e ad avere una visualizzazione dei dati utilizzando tutti i 3 sensori insieme alla libreria per il client.

Il codice implementato in Arduino è diviso in diverse parti le quali verranno viste in modo approfondito con le varie spiegazione nei vari sottocapitoli di seguito riportati.

3.3.1 Librerie e inizializzazione iniziale

Nella parte iniziale dello script vengono riportate tutte le librerie e anche le inizializzazioni per i parametri globali utilizzati. Un esempio di come includere una libreria è il seguente:

```
#include <freertos/FreeRTOS.h>
```

Una volta che sono state incluse tutte le librerie, si passa alla inizializzazione delle variabili globali come per esempio il payload che contiene il messaggio che deve essere mandato al broker:

```
char payload[100];
```

Oltre a variabili di questo tipo, ci sono variabili molto più importanti che sono state utilizzate per la comunicazione con il broker MQTT come :

```
const char* ssid = "You SSID";  
const char* password = "password";  
const char* mqtt_server = "your mqtt server";
```

Nel **SSID** si deve metter il nome della rete WiFi, mentre per quanto riguarda la **password**, naturalmente va messa la password della rete WiFi. Per quanto riguarda il **mqtt_server**, lì va messo l'indirizzo del vostro MQTT server o anche il suo hostname.

Oltre a ciò, durante la prima fase sono stati creati gli oggetti dei sensori derivanti dalle librerie installate attraverso il seguente comando:

```
RTC_PCF8523 rtc;  
hp_BH1750 BH1750;  
Adafruit_BMP280 bmp;
```

Successivamente sono stati dichiarati i due task con il seguente codice:

```
void task1_misure(void *param);  
void task2_invio_dati(void *param);
```

La creazione dei task avviene direttamente nel **void setup()** con il seguente comando:

```
xTaskCreate(task1_misure,"Task della data e ora",5000,NULL,2,NULL);  
xTaskCreate(task2_invio_dati,"Task stampa e invio dati",5000,NULL,1,NULL);
```

Dove ad ogni task che viene creato vengono passati i seguenti elementi:

- Nome del task utilizzato dalla macchina.
- Nome in stringa che potrebbe essere utile alla persona fisica per il riconoscimento del task.
- Dimensione dello stack.
- Parametro di input del task che è NULL in questo caso.
- Priorità del task.
- Si ha un handle che viene utilizzato per richiamare il task in parti del codice esterno al task, anche in questo caso è NULL.

3.3.2 Task per la lettura dei sensori

In questa sezione vediamo il task utilizzato per la misurazione dei sensori, questo task ha priorità maggiore rispetto al task per l'invio dei dati. Praticamente grazie al ciclo **while(true)** si riesce a ciclare infinitamente come avverrebbe con la funzione predefinita di Arduino **void loop**. All'interno del ciclo while viene messo il codice utilizzato per la misurazione dei sensori.

```
void task1_misure(void *param){
    sensors_event_t temp_event, pressure_event;
    while(true){
        DateTime adesso = rtc.now();
        tempo=adesso.timestamp(DateTime::TIMESTAMP_FULL);
        bmp_temp->getEvent(&temp_event);// sensore temperatura
        bmp_pressure->getEvent(&pressure_event);//sensore pressione
        temperatura=temp_event.temperature;
        pressione=pressure_event.pressure;
        lux=BH1750.getLux();
        BH1750.start();
        vTaskDelay(600/portTICK_PERIOD_MS);

    }
}
```

3.3.3 Task per l'invio dei dati al broker

Per l'invio dei dati è stato utilizzato un task che ha priorità minore del task che viene utilizzato per le misure. Come il task precedente si ha un ciclo **while** attraverso il quale il client comunica continuamente le misure ogni 5 secondi. Si fa un primo controllo, infatti se lo switch su Node-Red è **on** si trasmette un messaggio ogni 5 secondi, mentre se lo switch, sempre in Node-Red, è **off** si resta in attesa che diventi **on**; ciò si può notare nella porzione di codice riportato qui sotto:

```
if(stato==true){
    long now = millis();
    if (now - lastMsg > 5000) {
        client.publish("on/off","I dati vengono trasmessi");
    }
    else if(stato==false){
        client.publish("on/off","I dati non vengono trasmessi");
    }
}
```

Mentre per quanto riguarda l'invio vero e proprio dei dati avviene, sempre all'interno del ciclo **while**, utilizzando il seguente codice:

```
sprintf(payload, "\"%s\"_%s",VARIABLE_temperatura, tempoString );
sprintf(payload, "%s \"%s\"_%s", payload,VARIABLE_temperatura, tempString);
sprintf(payload, "%s \"%s\"_%s", payload,VARIABLE_pressione, preString);
sprintf(payload, "%s \"%s\"_%s", payload, VARIABLE_luminosita, lucString);
client.publish("dati", payload);
```

Prima di utilizzare il seguente codice le varie misure vengono convertite in un char array e dopo attraverso **client.publish** vengono trasmesse al broker MQTT nella **topic dati** e il messaggio che viene trasmesso è **payload** che contiene il timestamp, la misura della temperatura, la pressione e la luminosità in un dato istante di tempo.

3.3.4 Altre funzioni

Oltre a quello descritto sopra ci sono altre 3 funzioni che vengono implementate:

- **Funzione di callback**, utilizzata per vedere se lo switch è on o off nella dashboard Node-Red.
- **Funzione setup_wifi**, utilizzata per connettere l'ESP32 alla rete WiFi, per questo progetto la rete è la stessa del broker MQTT.
- **Funzione reconnect**, utilizzata per la connessione al broker MQTT e in caso di disconnessione vengono fatti vari tentativi fino a quando non viene stabilita una connessione e allo stesso tempo dice il tipo di errore per il quale è fallita la connessione.

3.4 Node Red

L'interfaccia utente, dove verranno visualizzati su schermo i dati acquisiti dai vari sensori, è stata realizzata tramite la creazione di una dashboard implementata su Node-Red. Node-Red è un ambiente di sviluppo visuale open-source basato su Node.js che facilita la creazione di applicazioni e automazioni IoT (Internet of Things) in modo intuitivo e veloce. È ampiamente utilizzato per creare flussi di dati, automatizzare processi e integrare dispositivi e servizi diversi.

3.4.1 Installazione

Per l'installazione di Node-Red su Ubuntu si devono seguire i seguenti passaggi:

1. Aprire il terminale sul proprio sistema Ubuntu.
2. Assicurarsi di avere accesso a Internet per poter scaricare il software necessario, aggiornando il sistema Ubuntu.

```
sudo apt update
```

3. Utilizzare il gestore dei pacchetti di Ubuntu per installare **Node.js**, che è un prerequisito per Node-Red, tramite il seguente comando:

```
sudo apt install nodejs
```

4. Per verificare che l'installazione sia andata a buon fine basta digitare i seguenti comandi:

```
node --version  
npm --version
```

5. tramite **npm** si installa Node-Red attraverso il comando:

```
sudo npm install -g --unsafe-perm node-red
```

6. Finita l'installazione, attraverso il seguente comando si avvia Node-Red:

```
node-red
```

Nella figura 10 si può osservare il risultato della riga di comando.

```
eris@ubuntu-vm:~$ node-red
19 Jun 21:07:04 - [info]

Welcome to Node-RED
=====

19 Jun 21:07:04 - [info] Node-RED version: v3.0.2
19 Jun 21:07:04 - [info] Node.js version: v20.2.0
19 Jun 21:07:04 - [info] Linux 5.19.0-45-generic x64 LE
19 Jun 21:07:04 - [info] Loading palette nodes
19 Jun 21:07:05 - [info] Dashboard version 3.5.0 started at /ui
19 Jun 21:07:05 - [info] Settings file : /home/eris/.node-red/settings.js
19 Jun 21:07:05 - [info] Context store : 'default' [module=memory]
19 Jun 21:07:05 - [info] User directory : /home/eris/.node-red
19 Jun 21:07:05 - [warn] Projects disabled : editorTheme.projects.enabled=false
19 Jun 21:07:05 - [info] Flows file : /home/eris/.node-red/flows.json
19 Jun 21:07:05 - [info] Server now running at http://127.0.0.1:1880/
19 Jun 21:07:05 - [warn]

-----
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----

19 Jun 21:07:05 - [info] Starting flows
19 Jun 21:07:05 - [info] Started flows
19 Jun 21:07:05 - [info] [mqtt-broker:10e78a89.5b4fd5] Connected to broker: mqtt://localhost:1883
```

Figura 10: Stato Node-Red

7. Per accedere all'interfaccia di Node-Red basterà aprire il proprio browser web ed andare all'indirizzo **http://localhost:1880** o anche digitando l'indirizzo ip della macchina e la porta che è sempre la 1880.

3.4.2 Dashboard

La dashboard creata tramite Node-Red può essere divisa in due parti, una prima parte dove il developer programma e una seconda parte dove l'utente vede il risultato. La parte programmata è composta da nodi dove ogni nodo indica un'azione; ciò si può osservare nella figura 11:

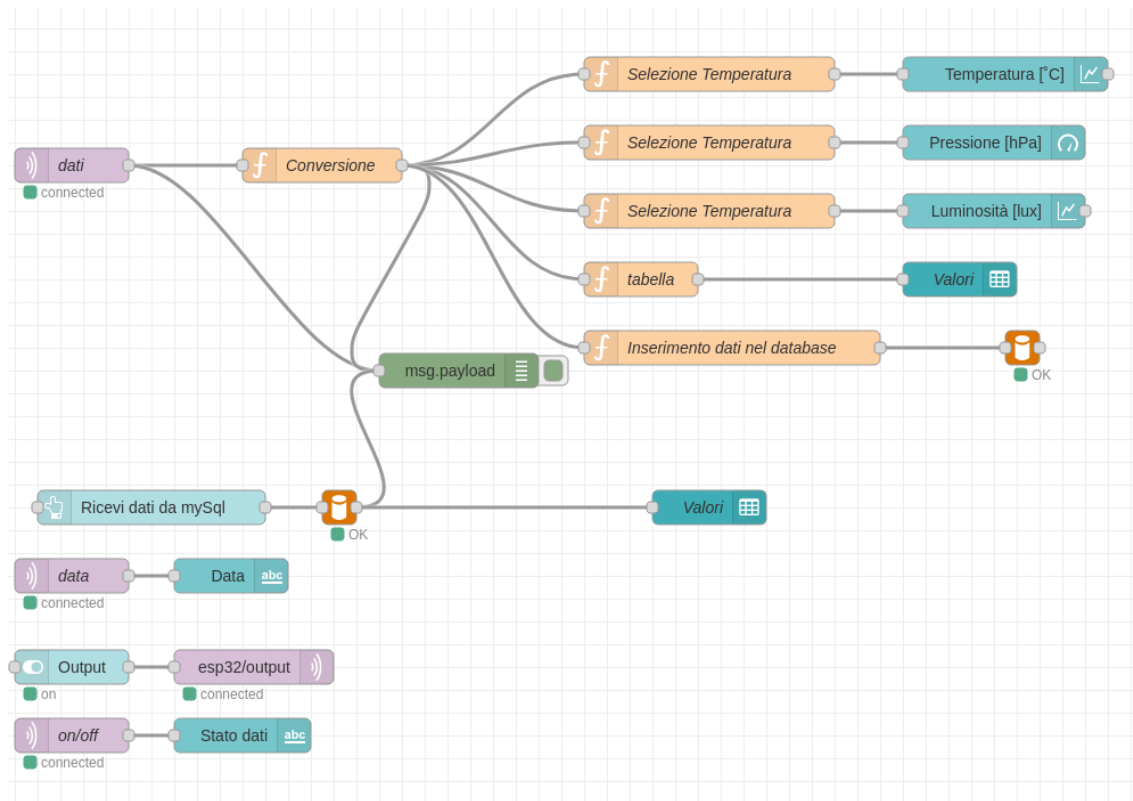


Figura 11: Dashboard composta da nodi

I nodi si possono dividere nelle seguenti categorie:

- Nodi di colore viola vengono utilizzati per la comunicazione MQTT sia in ricezione che in trasmissione e devono essere configurati come in figura 12.

Figura 12: Configurazione nodo MQTT

- Il nodo verde, è un nodo molto importante che ha la funzione di debug e attraverso il quale si può capire se ci sono errori nel flusso dei dati.
- I nodi di colore arancione chiaro sono delle funzioni attraverso le quali è stato possibile estrarre le informazioni per poi passarle ai nodi che vengono utilizzati per la rappresentazione grafica.
- I nodi di colore arancione scuro vengono utilizzati per interrogare il database e pure questi devono essere configurati come in figura 13.

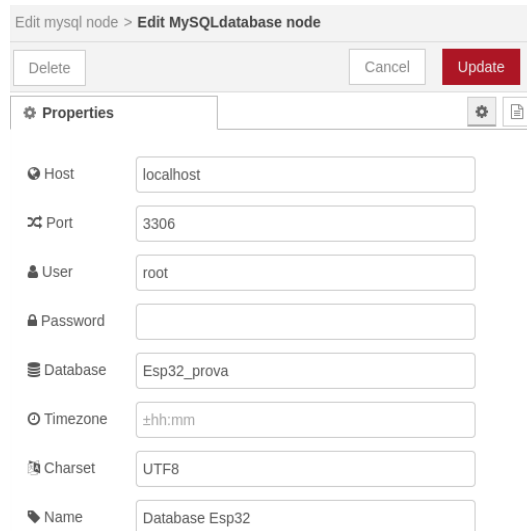


Figure 13 shows the configuration window for a MySQL database node. The window is titled "Edit mysql node > Edit MySQLdatabase node". It features three buttons at the top: "Delete", "Cancel", and "Update". Below these is a "Properties" section with a gear icon and a document icon. The properties are listed as follows:

Property	Value
Host	localhost
Port	3306
User	root
Password	
Database	Esp32_prova
Timezone	±hh:mm
Charset	UTF8
Name	Database Esp32

Figura 13: Configurazione nodo MySql

- I nodi restanti sono nodi utilizzati per la rappresentazione tabellare e grafica delle informazioni. Il nodo **output** è utilizzato come switch, mentre il nodo **Ricevi dati da mySql** è utilizzato come pulsante per stampare nella tabella i dati.

Mentre la seconda parte della dashboard rappresenta l'aspetto estetico come verrà vista dagli utenti, tutto ciò si può osservare in figura 14:

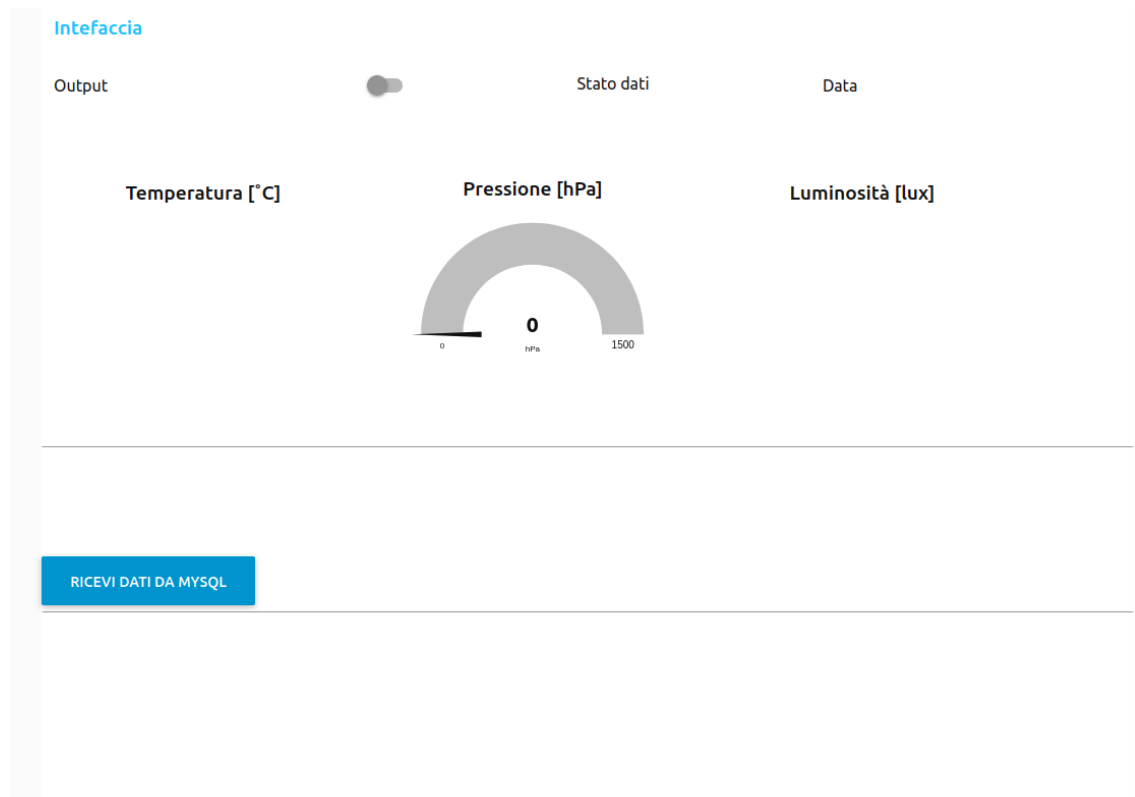


Figura 14: Interfaccia utente vuota

La figura 14 rappresenta l'interfaccia vuota, dove l'ESP32 è spento e di conseguenza non è connesso alla rete e nemmeno al broker MQTT, e dove anche il database, si è connesso, ma senza premere il pulsante per ricevere i dati non mostra nulla a video.

Mentre la figura 15 rappresenta l'interfaccia quando si ha interazione. Infatti l'interfaccia è composta:

- Nella parte superiore si ha uno switch che si mette a **on** per ricevere i dati oppure **off** per non ricevere i dati; si ha una box di testo che stampa a video **i dati vengono trasmessi** se i dati vengono trasmessi oppure stampa a video **i dati non vengono trasmessi** se i dati non vengono trasmessi ed ha una dipendenza con lo switch, ed inoltre si ha sempre una box di testo che stampa la data e l'ora di acquisizione dati.
- Nella parte centrale dell'interfaccia si hanno tre grafici utilizzati per vedere l'andamento nel tempo della temperatura, della pressione e della luminosità; sotto questi grafici si trova una tabella composta da una riga che stampa continuamente i valori del timestamp, della temperatura, della pressione e della luminosità ogni 5 secondi.
- Nella parte finale si ha una pulsante chiamato **Ricevi dati da MySql** utilizzato per stampare tutte le misure, prese fino a quel momento e salvate nel database, nella tabella.

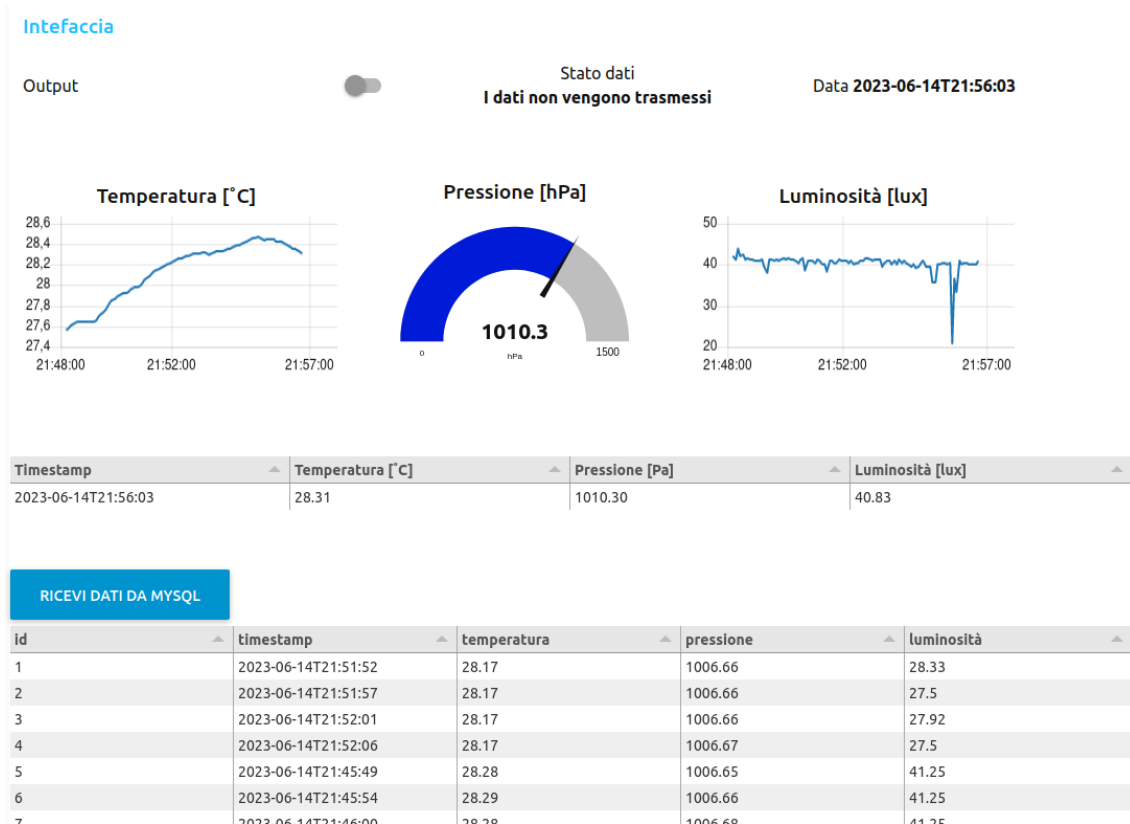


Figura 15: Interfaccia utente con dei dati

Nella figura 16 e nella figura 17 si può osservare come cambia la box di testo **Stato dati** e il colore dello switch:

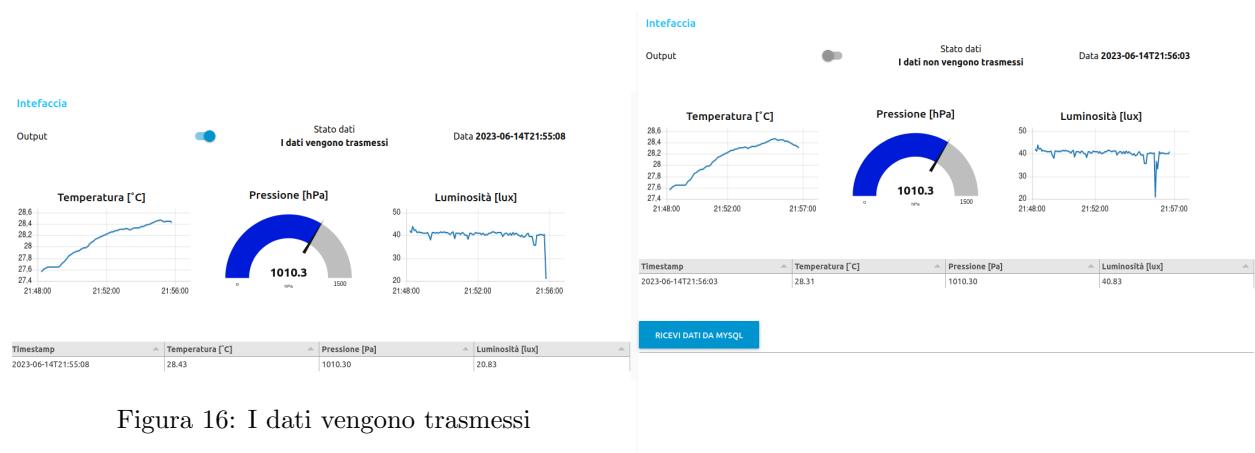


Figura 16: I dati vengono trasmessi

Figura 17: I dati non vengono trasmessi

3.5 MySQL

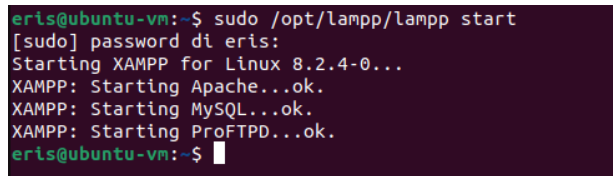
Per quanto riguarda l'implementazione del database è stato deciso di utilizzare MySql ed è stato possibile installarlo andando a scaricare dal sito di ApacheFriends il pacchetto Xampp che comprende sia Apache che è il nome di un server web libero sviluppato dalla Apache Software Foundation ed è la piattaforma server Web modulare più diffusa, in grado di operare su una grande varietà di sistemi operativi, tra cui UNIX/Linux, Microsoft Windows e OpenVMS; e comprende anche MySql che è un relational database management system composto da un client a riga di comando e un server. Una volta scaricato il pacchetto corretto per linux, è possibile installarlo su Ubuntu con i seguenti comandi da terminale:

```
cd /home/eris/Scaricati
sudo ./xampp-linux-x64-8.2.4-0-installer.run
```

Il comando che inizia con **cd** serve per dire dove sta la posizione del elemento che con il secondo comando vogliamo installare. Il comando **sudo** è l'abbreviazione dalla lingua inglese di super user do, *"esegui come super utente"*, in informatica, è un programma per i sistemi operativi Unix e Unix-like che, con dei vincoli, permette di eseguire altri programmi assumendo l'identità (e di conseguenza anche i privilegi) di altri utenti. Una volta installato il pacchetto è possibile startarlo in modo che vengono avviati i servizi MySql e Apache attraverso il comando:

```
sudo /opt/lampp/lampp start
```

Il risultato di questa riga di comando si può vedere nella figura 18:



```
eris@ubuntu-vm:~$ sudo /opt/lampp/lampp start
[sudo] password di eris:
Starting XAMPP for Linux 8.2.4-0...
XAMPP: Starting Apache...ok.
XAMPP: Starting MySQL...ok.
XAMPP: Starting ProFTPD...ok.
eris@ubuntu-vm:~$
```

Figura 18: Inizializzazione Xampp

Una volta fatto ciò si può andare tranquillamente nel browser FireFox e digitare **localhost/** e sarà aperta la schermata iniziale dove si trova la dashboard di Xampp come si può vedere in figura 19:



Figura 19: Dashboard Xampp

Una volta giunti qui si deve andare a cliccare in alto a destra **phpMyAdmin** per andare nella parte dove si trova il database e andare a creare un nuovo database che per il progetto è stato chiamato **Esp32** e creare una tabella per memorizzare i dati la quale è stata chiamata **esp32_tabella**; nella figura 20 si può osservare l'interfaccia **phpMyAdmin** con il database:

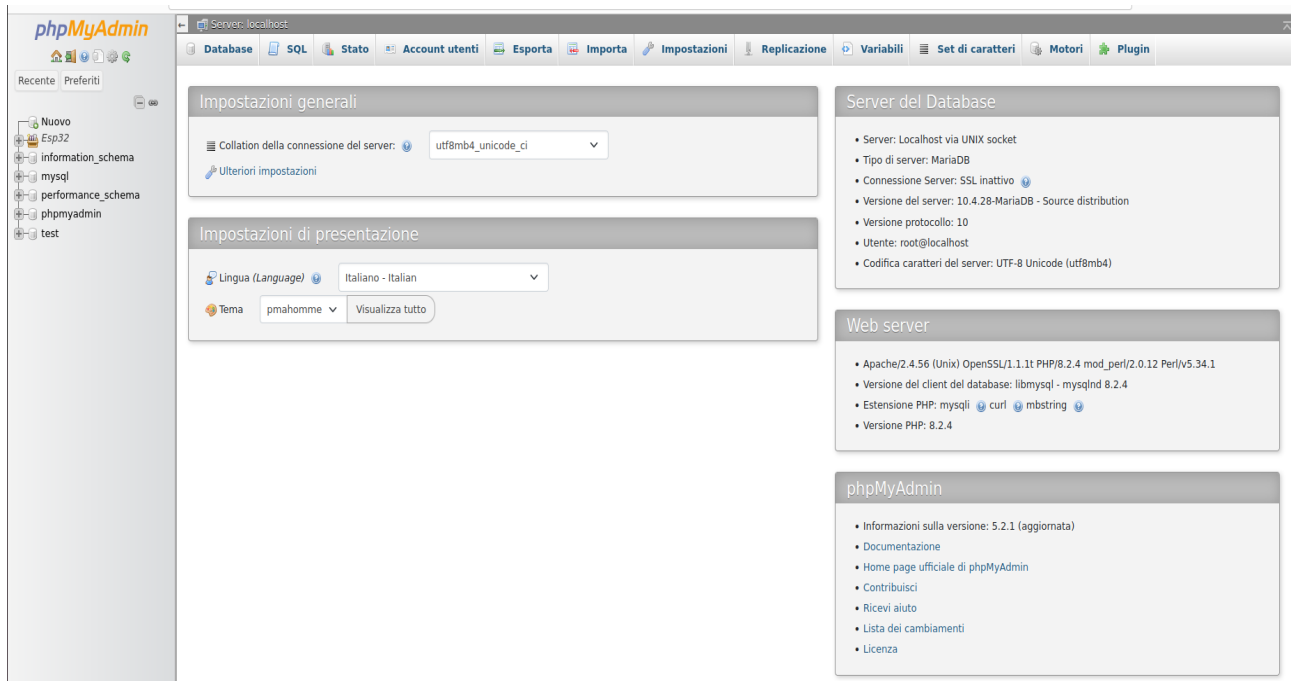


Figura 20: Interfaccia phpMyAdmin

Il risultato della creazione della tabella si può vedere nella figura 21:

id	timestamp	temperatura	pressione	luminosità
70	2023-06-14T21:52:12	28.31	1010.24	41.67
71	2023-06-14T21:52:17	28.31	1010.25	41.25
72	2023-06-14T21:52:22	28.31	1010.25	40.83
73	2023-06-14T21:52:27	28.32	1010.24	41.25
74	2023-06-14T21:52:32	28.32	1010.25	41.25
75	2023-06-14T21:52:37	28.31	1010.25	41.25
76	2023-06-14T21:52:42	28.3	1010.24	39.58
77	2023-06-14T21:52:47	28.31	1010.24	40.42
78	2023-06-14T21:52:52	28.32	1010.25	40.83
79	2023-06-14T21:52:57	28.33	1010.26	40.83
80	2023-06-14T21:53:02	28.33	1010.27	40
81	2023-06-14T21:53:07	28.33	1010.27	40.83
82	2023-06-14T21:53:12	28.33	1010.27	40
83	2023-06-14T21:53:17	28.34	1010.27	41.25
84	2023-06-14T21:53:23	28.35	1010.28	40.42
85	2023-06-14T21:53:27	28.36	1010.28	40.83
86	2023-06-14T21:53:32	28.37	1010.29	40.42
87	2023-06-14T21:53:37	28.38	1010.28	40
88	2023-06-14T21:53:42	28.39	1010.28	39.58
89	2023-06-14T21:53:47	28.39	1010.27	40
90	2023-06-14T21:53:52	28.4	1010.28	39.17
91	2023-06-14T21:53:58	28.41	1010.27	39.58
92	2023-06-14T21:54:02	28.43	1010.27	40.42
93	2023-06-14T21:54:07	28.44	1010.29	40.83
94	2023-06-14T21:54:12	28.45	1010.29	39.58
95	2023-06-14T21:54:17	28.46	1010.29	39.58
96	2023-06-14T21:54:22	28.46	1010.29	39.58

Figura 21: Tabella nel database dove vengono salvate le misure

Come si può vedere dalla figura 21 la tabella è composta da 5 colonne:

- La colonna **id** ed è un numero integer incrementale utilizzato come chiave primaria in modo da identificare univocamente ogni riga che viene salvata nel database.

- La colonna **timestamp** è di tipo varchar di dimensione 25 dove viene salvata l'ora e la data della misura.
- La colonna **temperatura** è di tipo float e viene utilizzata per salvare la misura della temperatura.
- La colonna **pressione** è di tipo float e viene utilizzata per salvare la misura della pressione.
- La colonna **luminosità** è di tipo float e viene utilizzata per salvare la misura della luminosità.

4 Problemi durante la realizzazione e soluzioni

Durante tutto lo sviluppo del progetto ci sono stati diversi problemi e sono state trovate delle soluzioni per risolverli. Questi problemi vengono elencati qui di seguito:

- Il primo problema è stato quello di andare a cambiare le impostazioni per quanto riguardo l'accesso ad internet della macchina virtuale dove è stato installato Ubuntu 22.04; infatti all'inizio la scheda di rete aveva l'opzione **Rete con NAT** e questo attraverso le impostazioni andava cambiato in **Scheda con bridge** in modo da far riuscire a comunicare il client, che in questo caso è l'ESP32, e il broker MQTT.
- Un secondo problema è stato nella creazione dei task, infatti quando si creavano i task inizialmente veniva dato una dimensione dello stack di 1024 e questo era un problema perchè andava a resettare e riavviare ogni volta l'ESP32 come mostrato in figura 22:

```
ELF file SHA256: 71e3503411b82c88

Rebooting...
[REMOVED]
*****Guru Meditation Error: Core  0 panic'ed (LoadProhibited). Exception was unhandled.

Core  0 register dump:
PC      : 0x400d16ef  PS      : 0x00000030  A0      : 0x800d1833  A1      : 0x3ffb9930
A2      : 0x00000000  A3      : 0x00000000  A4      : 0x3ffbc7e0  A5      : 0x3ffbc7e0
A6      : 0x3ffcc2900  A7      : 0x00000008  A8      : 0x00000001  A9      : 0x00000000
A10     : 0x3ffbc878  A11     : 0x80000001  A12     : 0x00000000  A13     : 0x3ffb9a20
A14     : 0x00000000  A15     : 0x00000000  SAR     : 0x00000000  EXCCAUSE: 0x0000001c
EXCVADDR: 0x00000040  LBEG    : 0x00000000  LEND    : 0x00000000  LCOUNT   : 0x00000000

Backtrace: 0x400d16ec:0x3ffb9930 0x400d1830:0x3ffb9950 0x400d1464:0x3ffb9970 0x400d148d:0x3ffb9990 0x400d149e:0x3ffb99b0 0x400d14b5:0x3ffb99d0 0x400d13c4:0:
```

Figura 22: Problema stack del task

Questo problema è stato risolto andando ad aumentare la dimensione dello stack, infatti si dava una dimensione minore rispetto a quello che serviva.

- Il terzo ed ultimo problema che si è presentato è quello in cui le acquisizioni delle misure dei sensori andavano in conflitto perchè si utilizzavano 3 task, un task per ogni sensore, ed essendo che di canale I2C, l'ESP32 ne ha solo uno si andava a creare un conflitto tra i task. Per risolvere questo problema si è deciso di optare per un solo task di acquisizione misure.

5 Implementazioni future

Le implementazioni future che si possono fare sono innumerevoli, però tra tutte è stato deciso di portare queste che vengono riportate di seguito:

- Implementare tecniche di controllo di sicurezza maggiori tramite l'aggiunta di id e password per gli account in modo che non tutti possono interagire con il broker e trasmettere i dati al database.
- Implementare delle tecniche per aumentare la qualità del servizio quando i pacchetti vengono trasmessi sia alla dashboard che al database.

- Implementazione di un sistema di monitoraggio remoto attraverso applicazione. Sviluppare un'applicazione mobile per visualizzare e controllare i dati raccolti dai sensori e dal modulo RTC. In questo modo, si potrebbe accedere ai dati in tempo reale da qualsiasi luogo e ricevere notifiche o avvisi in base alle condizioni rilevate.
- Creazione di un sistema di allarme. Utilizzando i dati del sensore di luminosità, temperatura e pressione, si potrebbe creare un sistema di allarme personalizzato. Ad esempio, se la luminosità scende al di sotto di una determinata soglia o la temperatura supera un valore predefinito, si potrebbe inviare un avviso o attivare un allarme sonoro.
- Integrazione con servizi di notifica. Si potrebbe integrare il sistema con servizi di notifica come SMS, email o app di messaggistica. In questo modo, quando vengono rilevate determinate condizioni o allarmi, il sistema può inviare notifiche immediate agli utenti interessati. Ad esempio, se la temperatura supera una soglia critica, il sistema può inviare un messaggio di avviso ai dispositivi mobili degli utenti registrati.
- Creazione di un sistema di automazione domestica. Utilizzando i dati raccolti dai sensori e il modulo RTC per gestire l'ora, si potrebbe creare un sistema di automazione domestica. Ad esempio, si programmerebbe il sistema per accendere automaticamente le luci quando la luminosità scende al di sotto di una soglia predefinita o attivare il riscaldamento quando la temperatura scende al di sotto di un certo valore. Questo renderebbe la casa più intelligente e automatizzata.

Si possono personalizzare e ampliare queste idee in base alle proprie esigenze e agli obiettivi specifici del proprio progetto.

6 Conclusioni

Lo sviluppo di questo progetto è stato molto interessante perchè è un primo approccio, anche se in forma ridotta, di come un cliente potrebbe chiedere di eseguire un lavoro; infatti all'inizio della consegna del progetto non vengono specificati i mezzi che si devono usare, né il database né il linguaggio e tante altre cose, quindi si ha grande libertà di scelta basta che alla fine si arrivi al risultato desiderato.

Oltre a ciò è stato interessante perchè nel progetto si vanno a trattare tanti temi, dalla creazione dell'interfaccia, alla piattaforma utilizzata e il codice per programmare l'ESP32 fino alla selezione del database. Quindi si può concludere che è un progetto completo anche se nella sua forma ridotta.

Riferimenti bibliografici

- [1] Adafruit: <https://www.adafruit.com>
- [2] Arduino download: <https://support.arduino.cc/hc/en-us/articles/360019833020-Download-and-install-Arduino-IDE>
- [3] Mosquitto broker: <https://mosquitto.org>
- [4] Node-Red: <https://nodered.org>
- [5] Xampp: <https://www.apachefriends.org/it/index.html>
- [6] FreeRTOS: <https://www.freertos.org/>