



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	6
3.2	MQTT	8
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	11
3.3.4	Altre funzioni	11
3.4	Node Red	11
3.4.1	Installazione	12
3.4.2	Dashboard	13
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:

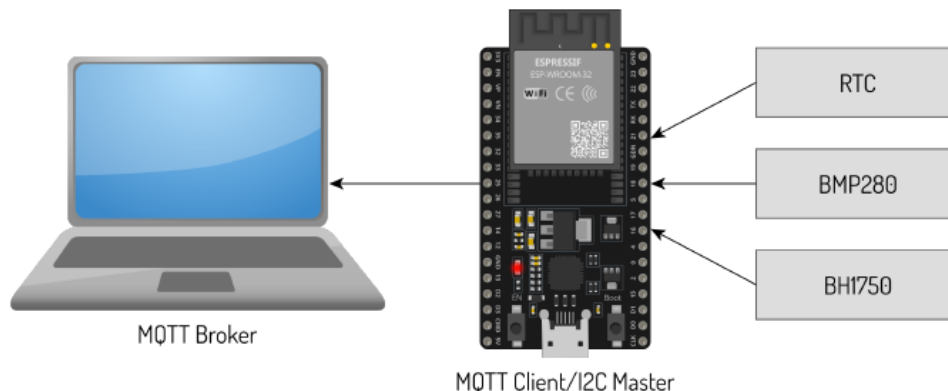


Figura 1: Schema a blocchi

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 sia possibile osservarli. 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 e altri dispositivi di Internet of Things (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.

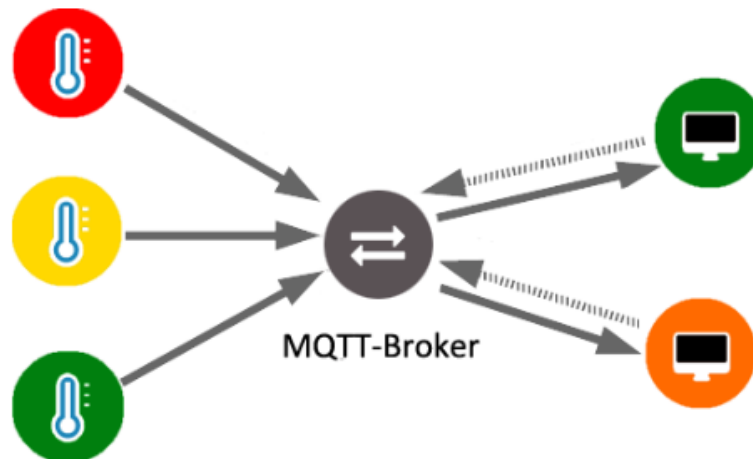


Figura 2: Schema Mqtt

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 of Things (IoT), poiché consente di creare dispositivi connessi in modo semplice e conveniente.

Nella figura 3 si può osservare l'ESP32.

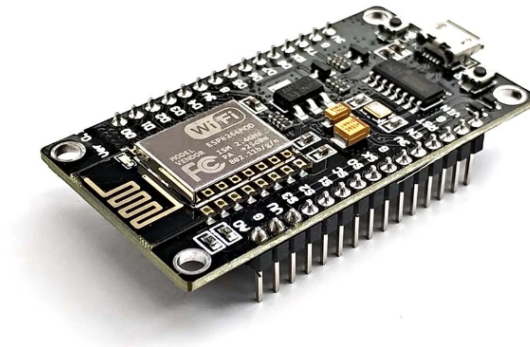


Figura 3: ESP32

2.3 Sensore temperatura e pressione BMP280

Il BMP280 è 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.

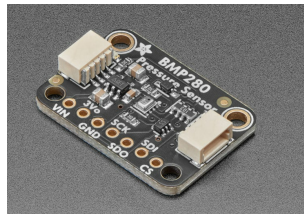


Figura 4: Sensore temperatura, umidità e pressione

2.4 Sensore di luminosità BH1750

Il BH1750 è 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 usato per misurare fino a 100.000 lux.

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

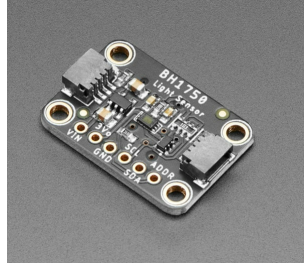


Figura 5: Sensore luminosità

2.5 Modulo RTC PCF8523

IL RTC PCF8523 è 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.

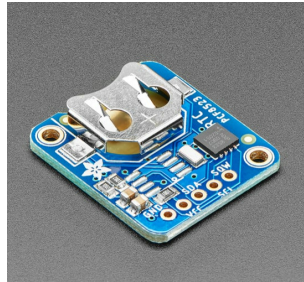


Figura 6: Modulo RTC

3 Implementazione

La realizzazione dell'intero progetto è stato suddiviso principalmente in due parti: una prima parte relativa alla parte hardware ed una seconda parte relativa alla parte software. Nel dettaglio tutta la parte dell'implementazione del progetto è stata suddivisa nelle seguenti fasi:

- Una prima fase in cui è stato realizzato l'assemblaggio mettendo insieme i vari componenti del progetto.
- Una seconda fase in cui viene realizzato il broker MQTT.
- Una terza fase dove viene programmato il codice attraverso l'Ide di Arduino.
- Una quarta fase dove è stato utilizzato Node-Red per andare a progettare una dashboard.
- Una quinta fase dove avviene l'utilizzo di Xampp per creare un database e memorizzare i dati all'interno di 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 (cavo dotato di connettori JST SH a 4 pin su entrambe le estremità) dove:

- Il colore nero viene utilizzato per il GND (Ground).
- Il colore rosso viene utilizzato per l'alimentazione 3.3V.
- Il colore blu per l'SDA (Serial Data Line) utilizzato per l'implementazione del protocollo I2C.
- Il colore giallo per SCL (Serial Clock Line) utilizzato per l'implementazione del protocollo I2C.

Se i collegamenti tra i pin dell'ESP32 e i pin dei sensori vengono scambiati, il funzionamento del sistema viene compromesso e si verifica un riscaldamento dei vari dispositivi che potrebbe danneggiarli. In figura 7 è possibile osservare il cavo per collegare in serie i due sensori e anche i suoi colori descritti in precedenza.

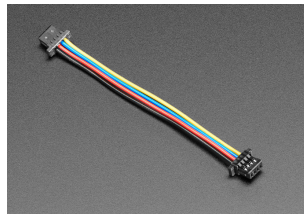


Figura 7: Cavo 4 pin

Per quanto riguarda tutto lo schema elettrico e le connessioni dei sensori ai pin dell'ESP32, sono presenti 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.

Nelle figura 8 si può osservare come i collegamenti sono stati realizzati tramite il software **Fritzing** (software di progettazione elettronica open-source che consente agli utenti di creare schemi elettrici) in modo tale da avere uno schema complessivo facilmente osservabile (i collegamenti sono stati fatti in modo sequenziale sensore-breadboard in tal modo da facilitare la lettura dello schema), andando a visualizzare i vari collegamenti sopra elencati.

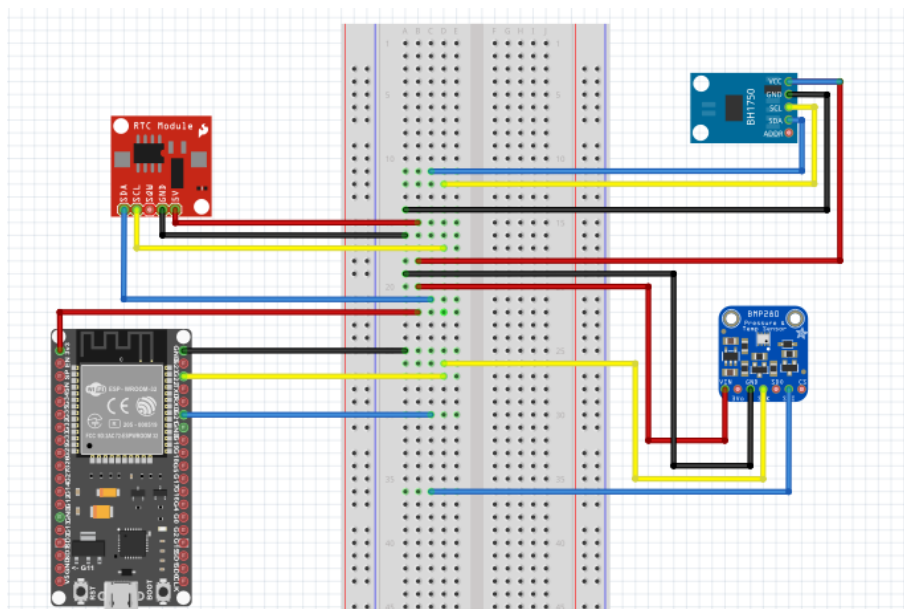


Figura 8: Schema elettrico

Nella figura 9 osserviamo l'intero collegamento fisico tra i vari componenti.

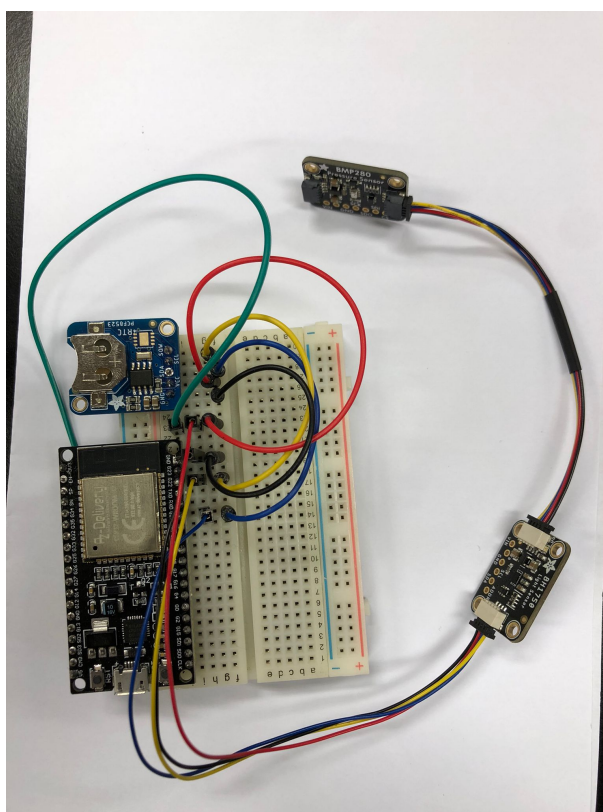


Figura 9: Schema fisico

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. Basato sul publish-subscribe, progettato per consentire la comunicazione tra dispositivi, come sensori e dispositivi IoT (Internet of Things). 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
```

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.

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 si renderebbe tale:

```
sudo systemctl enable mosquitto
```

3.3 Arduino

L'IDE Arduino (Integrated Development Environment) è un ambiente di sviluppo software utilizzato per programmare e caricare codice sugli Arduino boards. Una volta fatto il collegamento tra i vari sensori e l'ESP32, si è proceduto all'implementazione del 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. Dopo aver installato il software, sono state scaricate le librerie della board del microcontrollore ESP32 attraverso il seguente procedimento:

1. Dopo aver avviato il programma Arduino Ide sul proprio computer si va su file.
2. Nella barra dei menu in alto, cercare la voce "File", si aprirà un menù a tendina e selezionare "Preferences".
3. Selezionare "Additional board manager URLs e incollare il seguente link (ricavato da Github) https://espressif.github.io/arduino-esp32/package_esp32_index.json.

La scheda (board) sarà presente nella sezione "Strumenti" (Tools) e potrà essere selezionata. Inoltre, sarà necessario selezionare la porta seriale corretta per caricare il codice nell'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 vengono scaricate nella sezione **Tools**, aprendo il menù a tendina andando poi a selezionare **Manage libraries**; di seguito vengono elencate le librerie scaricate e che sono state anche utilizzate:

- 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 correttamente, e successivamente dopo aver installato il broker MQTT e creata una prima bozza di dashboard nel browser web è stato provato il funzionamento della libreria **PubSubClient**. Tutte queste prove hanno avuto risultati positivi.

Dopo aver verificato il corretto funzionamento, è stata effettuata una prima prova in cui i tre sensori sono stati utilizzati contemporaneamente, inviando i dati alla porta seriale. Tale prova ha avuto esito positivo.

Successivamente è stata fatta una prova dove si andava ad utilizzare il **void loop**, senza **task**, per controllare se si riuscisse 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 all' 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**, 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 delle misure",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){

        //Lettura dell'ora e della data
        DateTime adesso = rtc.now();
        tempo=adesso.timestamp(DateTime::TIMESTAMP_FULL);

        // Lettura della temperatura
        bmp_temp->getEvent(&temp_event);
        temperatura=temp_event.temperature;

        // Lettura della pressione
        bmp_pressure->getEvent(&pressure_event);
        pressione=pressure_event.pressure;

        // Lettura della luminosità
        lux=BH1750.getLux();
        BH1750.start();
```

```

        // Configurazione nuova chiamata del task
        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, dove 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**; le misure vengono convertite in char array e utilizzando il seguente codice i char array vengono uniti tutti insieme a generare un unico messaggio chiamato **payload** :

```

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);

```

Attraverso **client.publish** le informazioni vengono trasmesse al broker MQTT nella **topic dati** e il messaggio che viene trasmesso, **payload**, 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 fallimento della 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 suddivisa in due sezioni distinte. La prima sezione è riservata al developer, che ha il compito di programmare e configurare la dashboard secondo le esigenze del cliente. La seconda sezione è destinata all'utente finale, che visualizza e interagisce con la dashboard. La parte programmata è composta da nodi dove ogni nodo indica un'azione; ciò si può osservare nella figura 11:

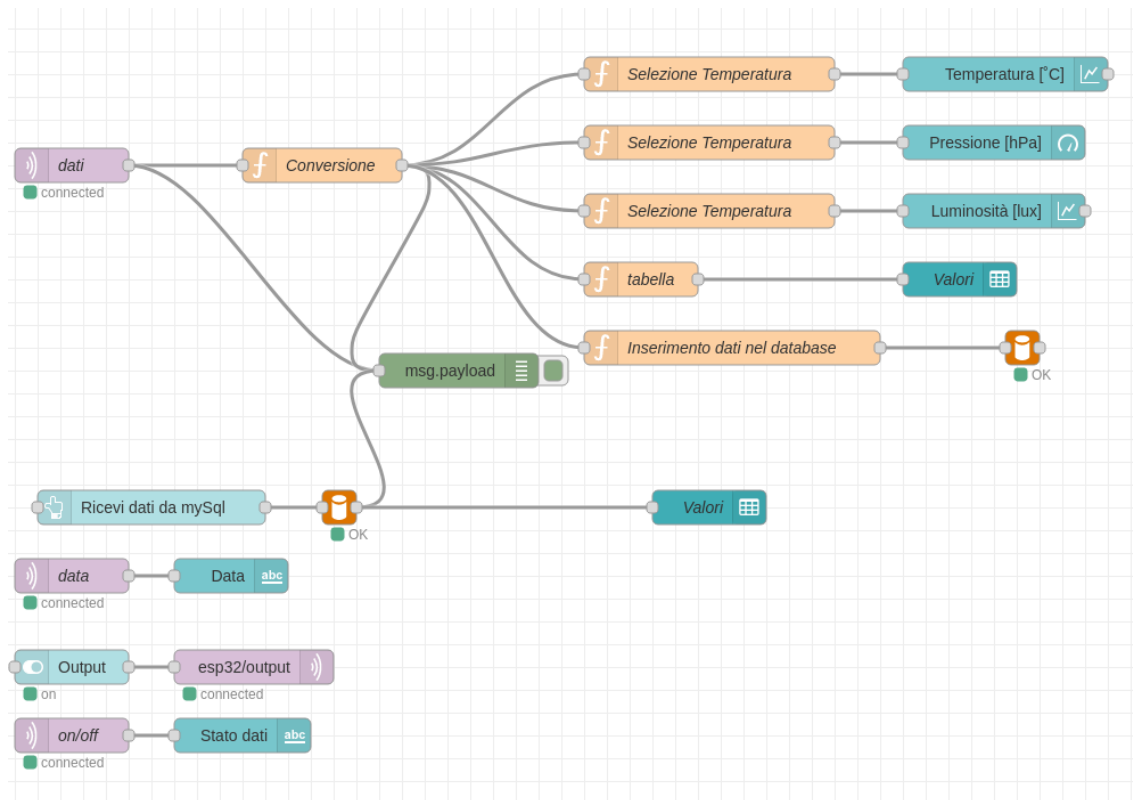
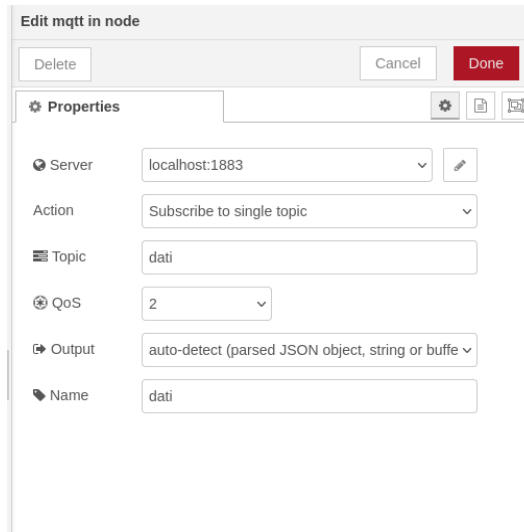


Figura 11: Dashboard composta da nodi

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.



Edit mqtt in node

Delete Cancel Done

Properties

Server localhost:1883

Action Subscribe to single topic

Topic dati

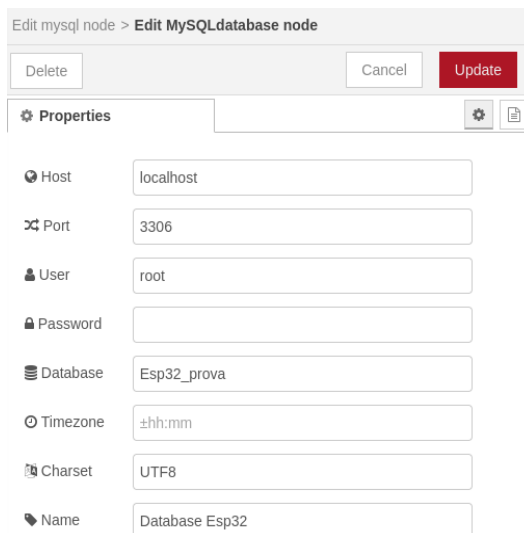
QoS 2

Output auto-detect (parsed JSON object, string or buffer)

Name dati

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.



Edit mysql node > Edit MySQLdatabase node

Delete Cancel Update

Properties

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:

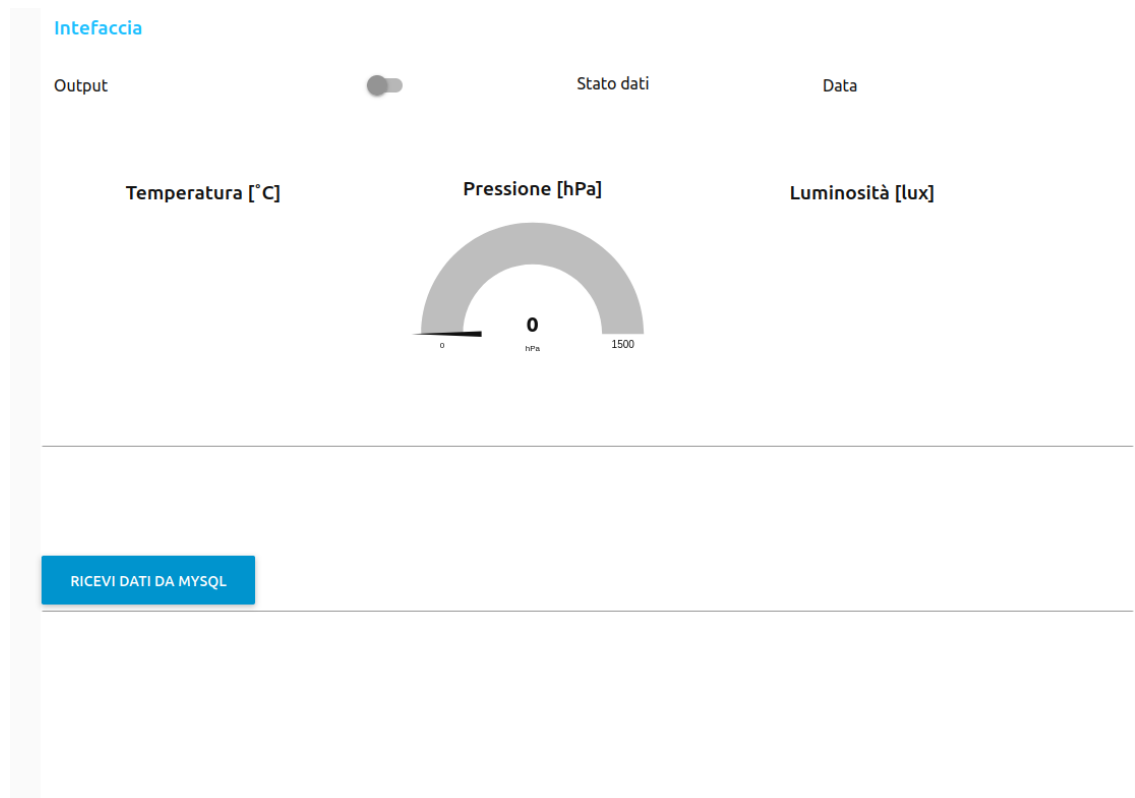


Figura 14: Interfaccia utente vuota

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.

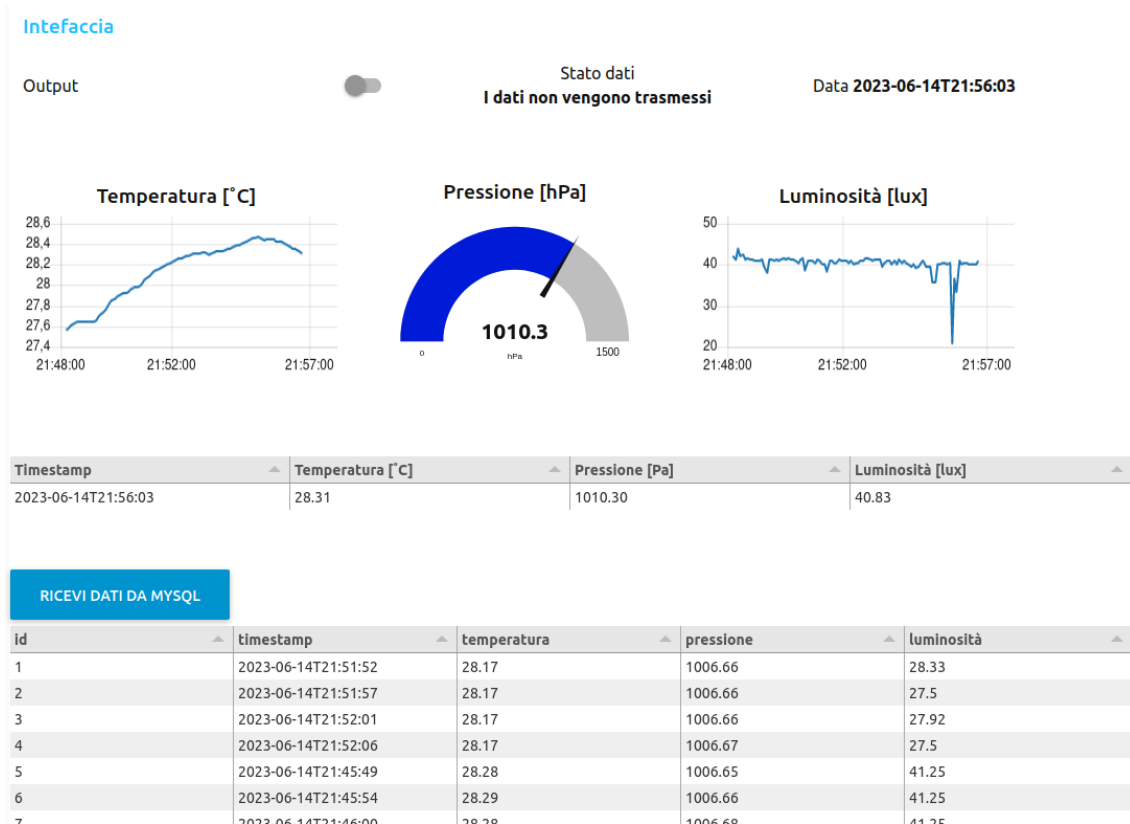


Figura 15: Interfaccia utente con dei dati

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

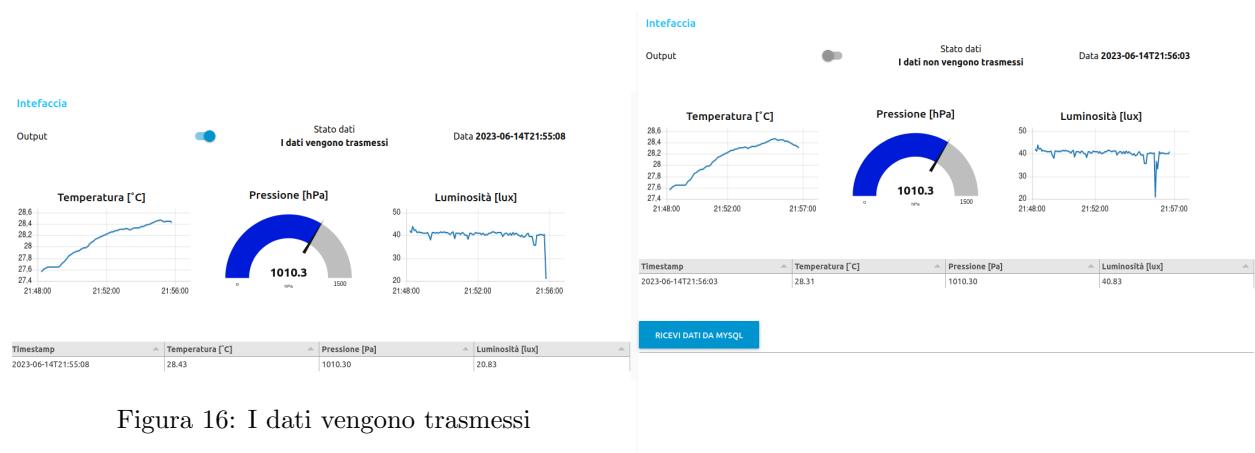


Figura 16: I dati vengono trasmessi

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. Uno volta scaricato il pacchetto corretto per linux, è possibile installarlo su Ubuntu con i seguenti comandi da terminale:

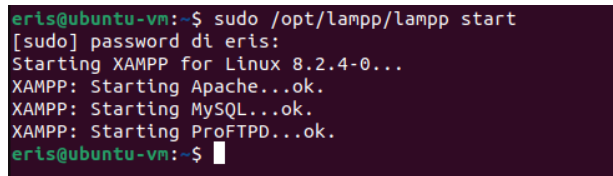
```
cd /home/eris/Scaricati
sudo ./xampp-linux-x64-8.2.4-0-installer.run
```

Il comando che inizia con **cd** serve per dire dove sta la posizione del elemento che con il secondo comando vogliamo installare.

Uno 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

Dopo aver completato queste azioni, è possibile accedere al browser FireFox e inserire **localhost/**. In questo modo, si aprirà la schermata iniziale contenente la dashboard di Xampp, come illustrato nell'immagine allegata 19:



Figura 19: Dashboard Xampp

Dopo aver raggiunto questa fase, è necessario fare clic sull'opzione **phpMyAdmin** nell'angolo in alto a destra. In questo modo si accederà alla sezione del database, dove sarà possibile creare un nuovo database chiamato **Esp32** per il progetto, e successivamente creare una tabella denominata **esp32.tabella** per memorizzare i dati. Nella figura 20 si può osservare l'interfaccia di **phpMyAdmin** con il database.

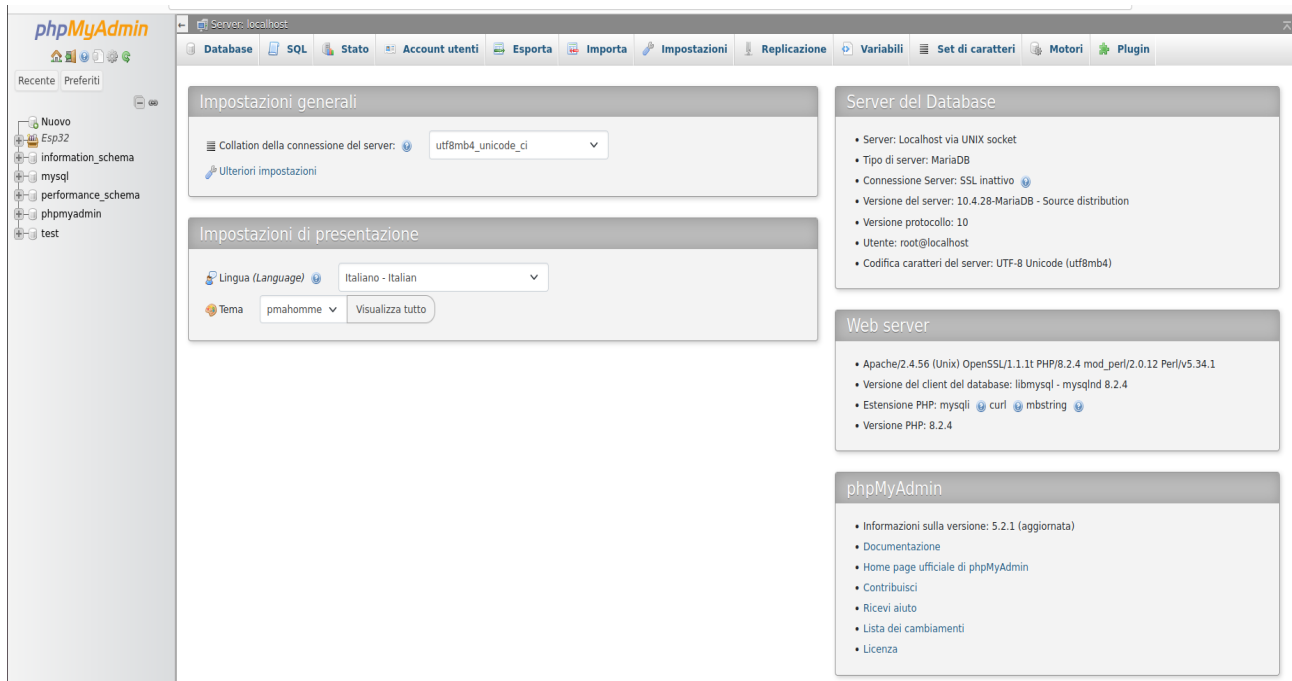


Figura 20: Interfaccia phpMyAdmin

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      : 0x3ffb8930
A2      : 0x00000000  A3      : 0x00000000  A4      : 0x3ffbc7e0  A5      : 0x3ffbc7e0
A6      : 0x3ffbc2900  A7      : 0x00000008  A8      : 0x00000001  A9      : 0x00000000
A10     : 0x3ffbc878  A11     : 0x80000001  A12     : 0x00000000  A13     : 0x3ffb8a20
A14     : 0x00000000  A15     : 0x00000000  SAR     : 0x00000000  EXCCAUSE: 0x0000001c
EXCVADDR: 0x00000040  LBEG    : 0x00000000  LEND    : 0x00000000  LCOUNT   : 0x00000000

Backtrace: 0x400d16ec:0x3ffb8930 0x400d1830:0x3ffb8950 0x400d1464:0x3ffb8970 0x400d148d:0x3ffb8990 0x400d149e:0x3ffb89b0 0x400d14b5:0x3ffb89d0 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.
- Ottimizzare l'interfaccia utente, dashboard, attraverso un numero maggiore di comandi per quanto riguarda la parte che si interfaccia al database, in particolare la possibilità di selezionare una fascia temporale o date precise per recuperare le informazioni. Inoltre si può rendere l'interfaccia più accattivante dal punto di vista estetico, utilizzando altri colori e aggiungendo loghi.

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

6 Conclusioni

Si possono personalizzare e ampliare queste idee in base alle proprie esigenze e agli obiettivi specifici del proprio progetto. È possibile adattare la dashboard di Node-Red per includere ulteriori funzionalità e visualizzazioni che siano pertinenti al contesto del progetto. Inoltre, si possono integrare altre tecnologie o servizi, come ad esempio l'utilizzo di un database specifico o l'implementazione di algoritmi personalizzati. In conclusione, lo sviluppo di questo progetto è stato molto interessante in quanto ha offerto un primo approccio all'implementazione di una soluzione personalizzata. Si ha la libertà di scegliere i mezzi e le tecnologie da utilizzare, purché si raggiunga il risultato desiderato. Il progetto copre diversi aspetti, come la creazione dell'interfaccia, l'utilizzo di una piattaforma come Node-Red, la programmazione dell'ESP32 e la selezione del database. Pertanto, anche nella sua forma ridotta, rappresenta un progetto completo che offre diverse opportunità di apprendimento e adattamento.

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/>