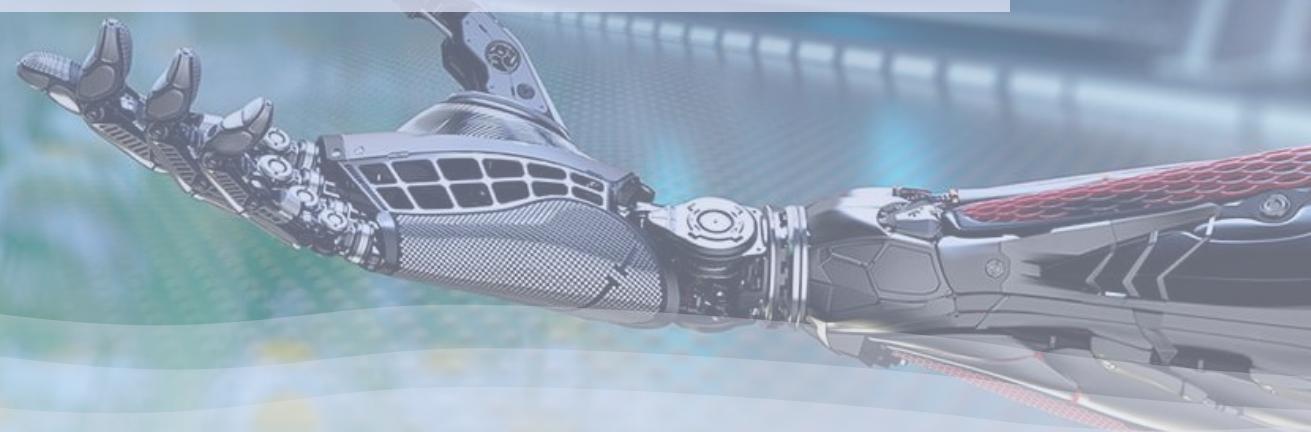


# Sviluppo di un dispositivo per un efficiente monitoraggio ambientale



Ad opera di:

Dalla Riva Alessandro      Scozzai Samuele  
Propedo Demien              Tavano Matteo

Progetto di Internet of Things aa. 2022-2023



UNIVERSITÀ  
DEGLI STUDI  
DI UDINE

# Nasce la necessità di adottare un protocollo di comunicazione affidabile



Monitorare i parametri

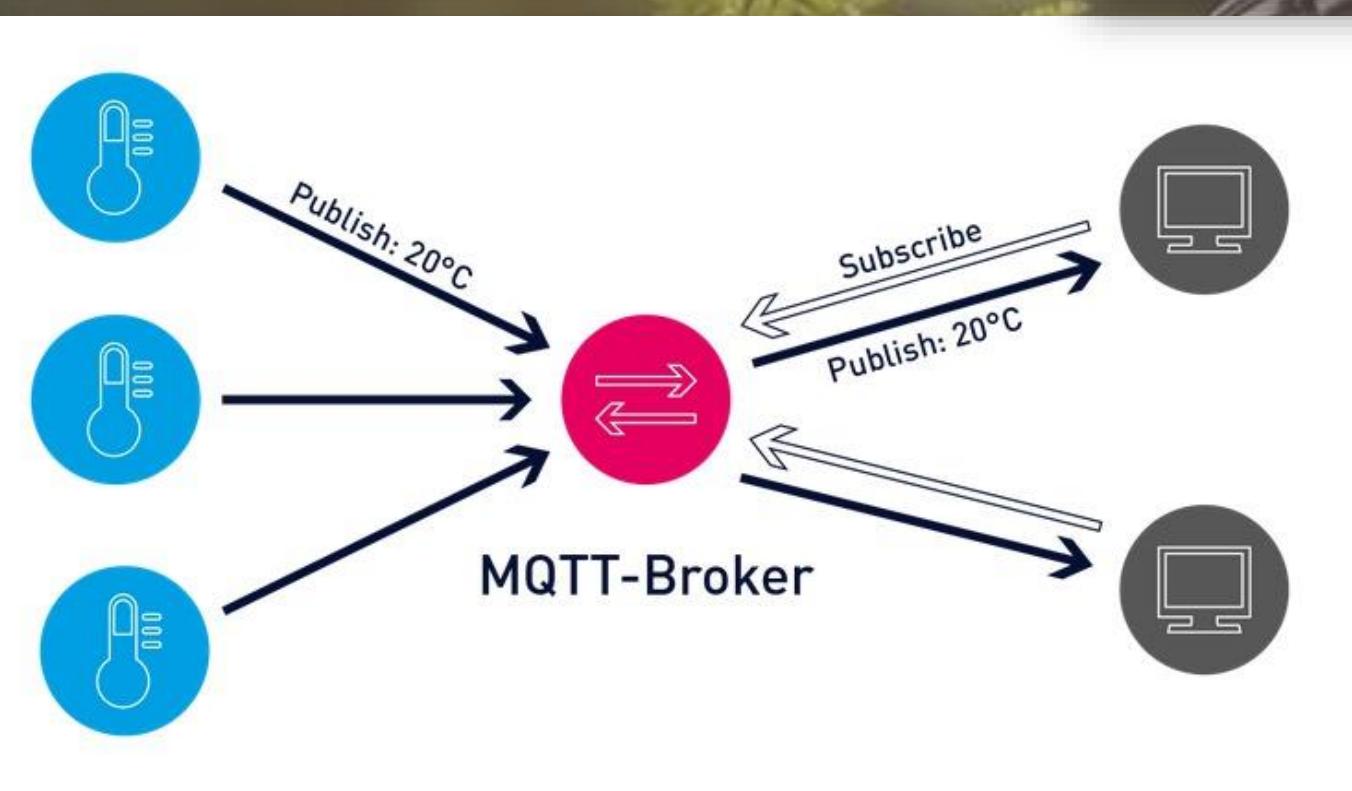
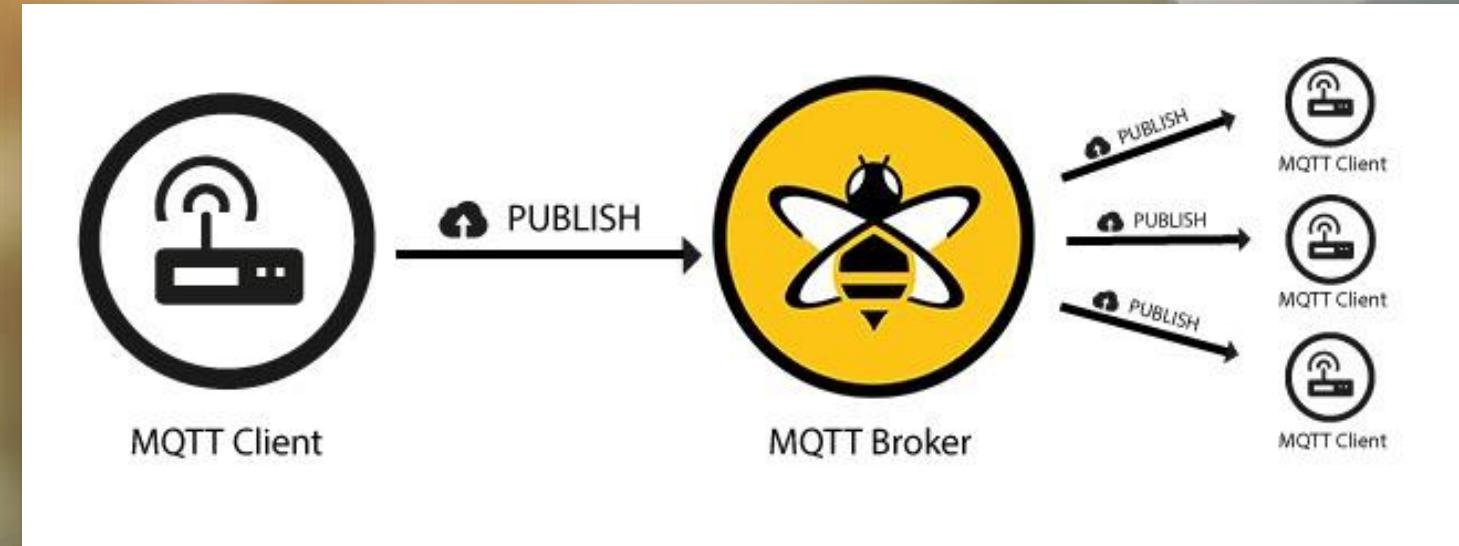


Raccogliere ed analizzare dati in  
sicurezza



Prevenire i guasti ed  
ottimizzare le risorse

# Perché scegliere Message Queuing Telemetry Transport?



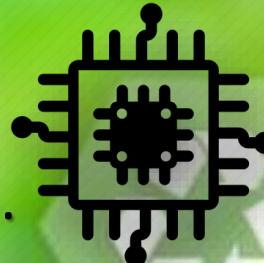
## Analisi dei requisiti che il dispositivo deve soddisfare:

- Rilevamento dei Parametri Ambientali
- Comunicazione e Trasmissione Dati
- Elaborazione e Analisi dei Dati
- Affidabilità
- Sicurezza
- Performance



# Progettazione

Microcontrollore **ESP8266**: integra un processore a bassa potenza e un modulo Wi-Fi.



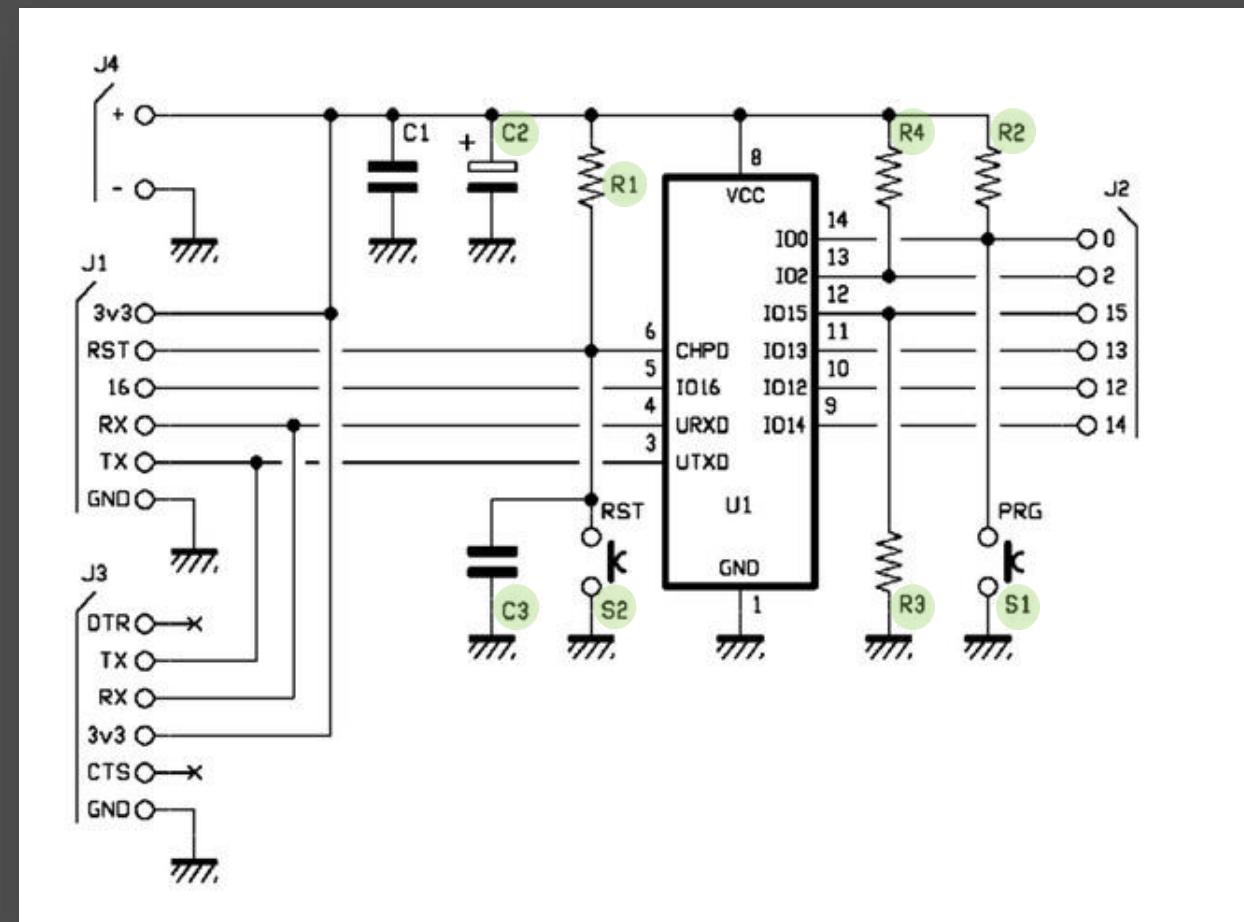
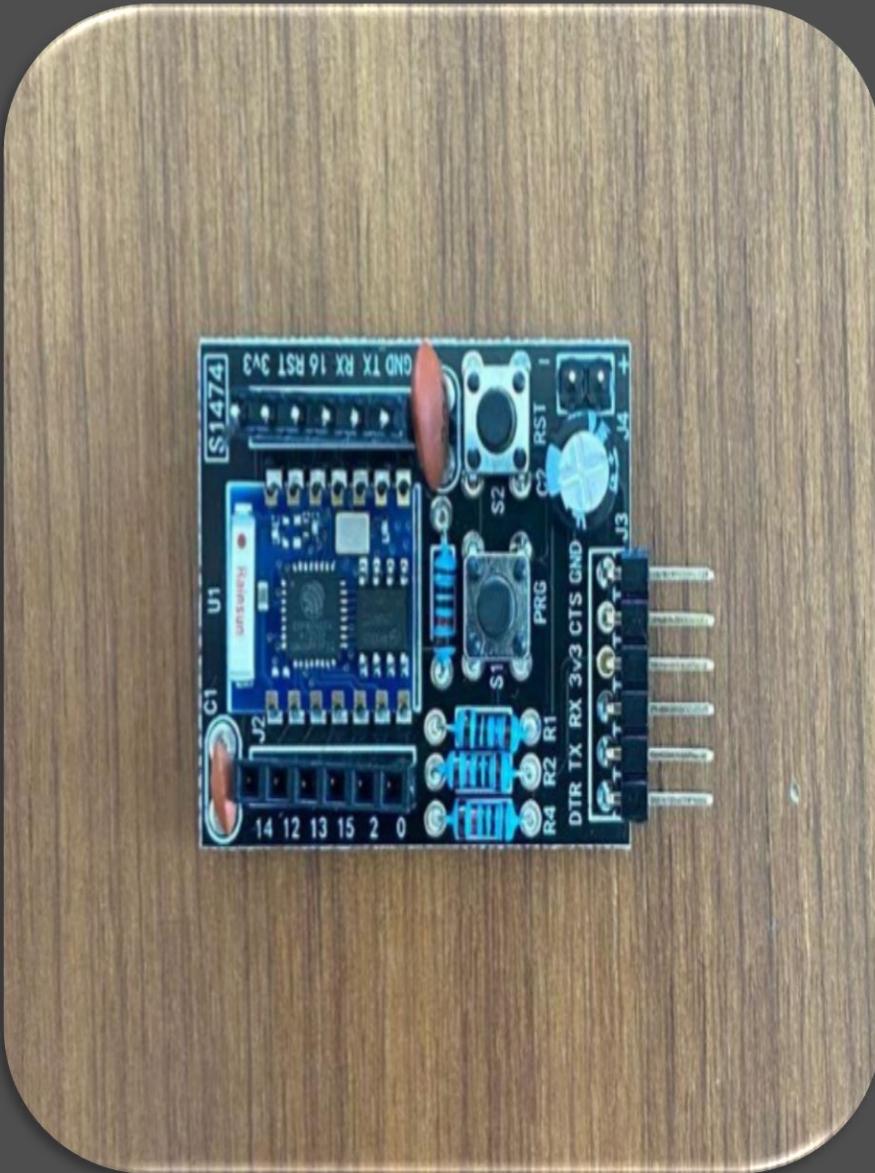
Grazie alla presenza di componenti base integrate è possibile ottimizzare le dimensioni del dispositivo soddisfando uno dei requisiti preposti.

Inoltre, offre una vasta comunità di sviluppatori e un ampio supporto di risorse online

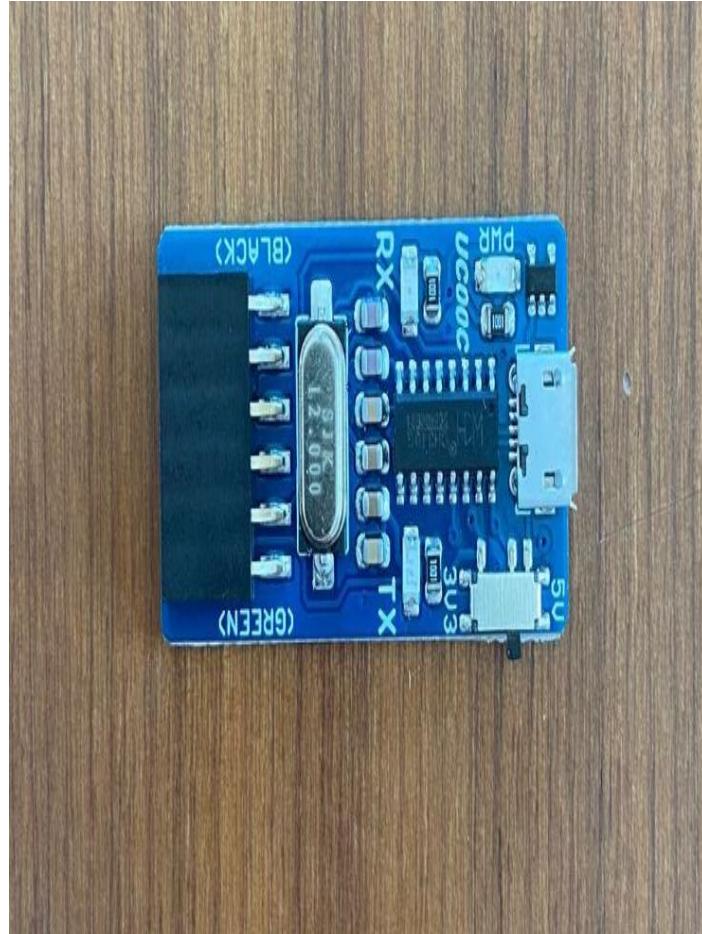
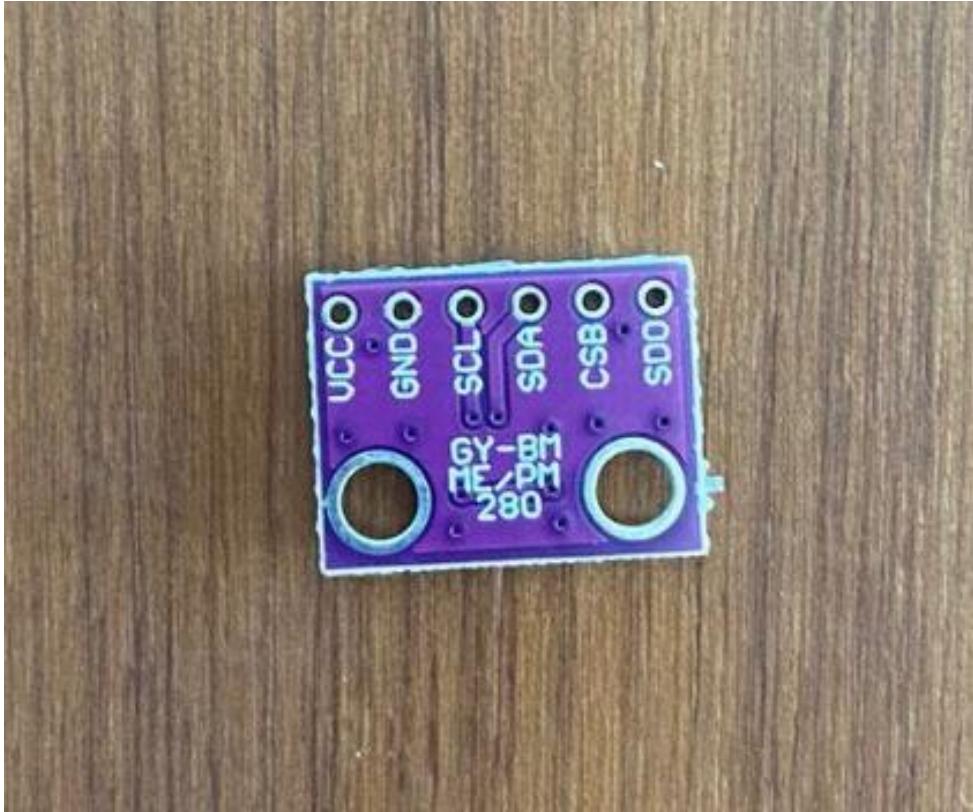


*Per questo progetto abbiamo pensato di realizzare una generica scheda basata su **ESP8266** con i soli **componenti essenziali***

# Architettura del sistema elettronico



## Sensore BME280



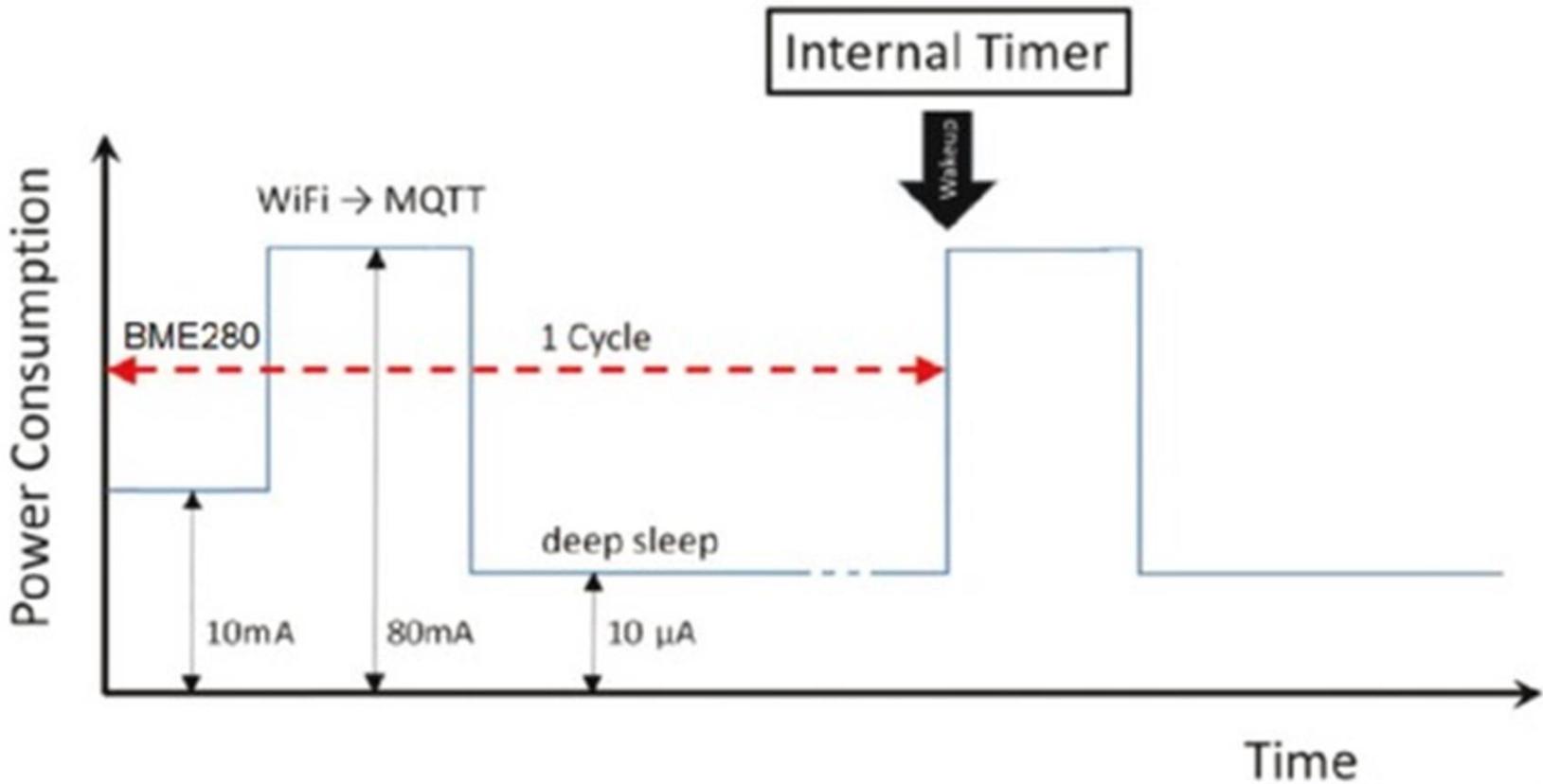
Convertitore da USB a UART 3.3V e 5V

# Scelta delle componenti Software e Cloud utilizzate

---

L'adozione della piattaforma cloud **ThingSpeak** e dell'**Arduino IDE** come ambiente di sviluppo per il progetto di monitoraggio ambientale ha dimostrato di offrire un connubio potente di flessibilità e semplicità operativa, unendo la *potenza del cloud computing alla facilità di sviluppo e programmazione offerta dall'ambiente Arduino*

## Logica di funzionamento



# REALIZZAZIONE DEL CIRCUITO



- ✓ Semplicità
- ✓ Compattezza

Realizzato utilizzando connettori  
***pin-to-pin e Jumper***

Si tratta di un circuito che rimane attivo per **6 secondi ogni 3.600 secondi** (ogni ora...) con una rilevante diminuzione dell'energia assorbita

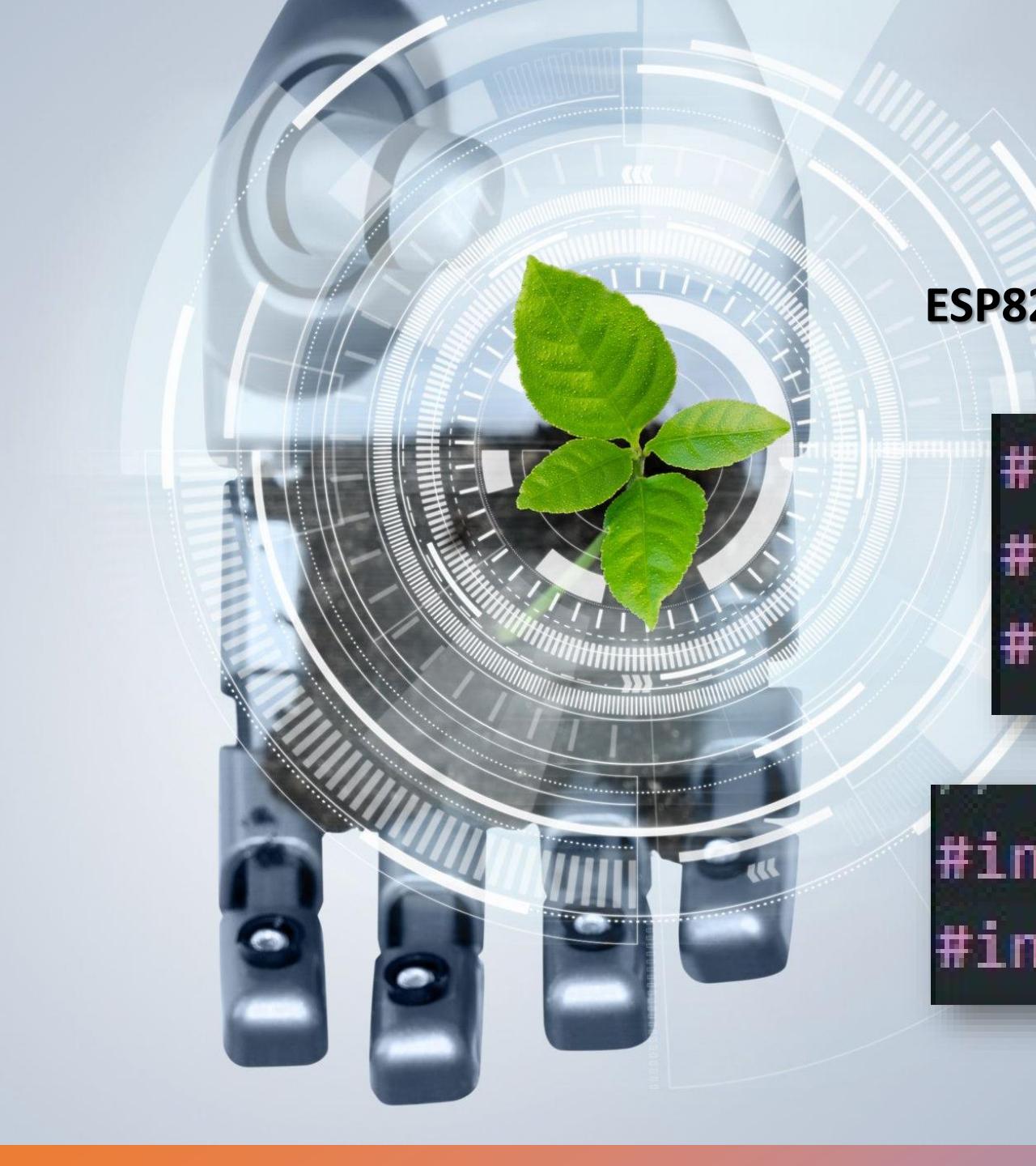
Sto caricando...

```
Lo sketch usa 257712 byte (51%) dello spazio disponibile  
Le variabili globali usano 26572 byte (32%) di memoria  
esptool.py v2.6  
2.6  
esptool.py v2.6  
Serial port COM8  
Connecting.....  
Chip is ESP8266EX  
Features: WiFi  
MAC: 5c:cf:7f:34:1a:86  
Uploading stub...  
Running stub...
```



## Programmazione dell'ESP

Per la programmazione serve il convertitore USB-Seriale esterno pertanto, possiamo utilizzare il **convertitore UART**



# LO SKETCH

reperibile nella cartella:

**ESP8266\_thingspeak\_MQTT\_BME280\_deepsleepEl**

```
#include "PubSubClient.h"
#include <ESP8266WiFi.h>
#include "secrets.h"
```

```
#include <Wire.h>
#include <Adafruit_BME280.h>|
```

```
16:02:59.180 ->
16:02:59.695 ->
16:02:59.695 -> Connecting to WiFi SSID ██████████....
16:03:04.572 -> WiFi connected with IP address: 192.168.1.182
16:03:04.572 -> Attempting MQTT connection...
16:03:05.868 -> Connected
16:03:05.868 -> T= 32.08°C  H= 45.13%  P=1015.22hPa  V=3.34V
16:03:05.868 -> Sending payload: field1=32.08&field2=45.13&field3=1015.22&field4=3.34&stat
16:03:05.868 -> Publish ok
16:03:06.540 -> Go to sleep!
16:03:06.540 ->
16:03:06.540 ->
```

La variabile **DEBUG** se impostata su true permette di abilitare i messaggi su Serial Monitor di Arduino

```
#define SECRET_SSID "MySSID" // replace MySSID with your WiFi network name
#define SECRET_PASS "MyPassword" // replace MyPassword with your WiFi password
#define channelID 123456 // Thingspeak channel number
#define writeAPIKey "ABCDEFGHILMNOPQR" // Thingspeak API Key
```

Per evitare di dover attendere tempi troppo lunghi, in fase di debug è possibile modificare anche il parametro **UPDATE\_INTERVAL\_SECONDS** per ridurre il tempo di invio a solo **un minuto**

## void() Setup

```
// Connect BME280 GND TO pin14 OR board's GND
pinMode(14, OUTPUT);
digitalWrite(14, LOW);

Serial.begin(115200);
delay(10);

// BME280 Initialise I2C communication as MASTER
Wire.begin(13, 12); //Wire.begin([SDA], [SCL])

BMEStatus = bme.begin();
if (!BMEStatus)
{
    if (DEBUG) { Serial.println("Could not find BME280!"); }
    //while (1);
}

bme.setSampling(Adafruit_BME280::MODE_FORCED,
                Adafruit_BME280::SAMPLING_X1, // temperature
                Adafruit_BME280::SAMPLING_X1, // pressure
                Adafruit_BME280::SAMPLING_X1, // humidity
                Adafruit_BME280::FILTER_OFF );

// read values from the sensor
float temperature, humidity, pressure;
```

# Setup del BME

```
if (BMEStatus)
{
    temperature = bme.readTemperature();
    humidity = bme.readHumidity();
    pressure = bme.readPressure() / 100.0F;
}
else
{
    if (DEBUG) Serial.println("Could not find BME280!");
    temperature=0;
    humidity=0;
    pressure=0;
}

float voltage = ESP.getVcc();
voltage = voltage/1024.0; //volt

if (DEBUG)
{
    Serial.println("T= " + String(temperature) + "°C  H= " + String(humidity) + "%  P=" + String(pressure) + "hPa  V=" + voltage + "V");
}
```

# Costruzione del payload MQTT

```
// Construct MQTT payload
payload="field1=";
payload+=temperature;
payload+="&field2=";
payload+=humidity;
payload+="&field3=";
payload+=pressure;
payload+="&field4=";
payload+=voltage;
payload+="&status=MQTTPUBLISH";
```

# Connessione Wifi

```
//Connect to Wifi
if (DEBUG)
{
    Serial.println();
    Serial.print("\nConnecting to WiFi SSID  ");
    Serial.print(SECRET_SSID);
}

WiFi.begin(SECRET_SSID, SECRET_PASS);

int timeOut=10; // Time out to connect is 10 seconds
while ((WiFi.status() != WL_CONNECTED) && timeOut>0)
{
    delay(1000);
    if (DEBUG) { Serial.print("."); }
    timeOut--;
}

if (timeOut==0) //No WiFi!
{
    if (DEBUG) Serial.println("\nTimeOut Connection, go to sleep!\n\n");
    ESP.deepSleep(1E6 * UPDATE_INTERVAL_SECONDS);
}

if (DEBUG) // Yes WiFi
{
    Serial.print("\nWiFi connected with IP address: ");
    Serial.println(WiFi.localIP());
}
```

## Riconnesion e al client MQTT

```
// Reconnect if MQTT client is not connected.  
if (!client.connected())  
{  
    reconnect();  
}  
  
mqttPublish();  
  
delay(200); // Waiting for transmission to complete!!! (ci vuole)  
WiFi.disconnect( true );  
delay( 1 );  
  
if (DEBUG) { Serial.println("Go to sleep!\n\n"); }  
// Sleep ESP and disable wifi at wakeup  
ESP.deepSleep( 1E6 * UPDATE_INTERVAL_SECONDS, WAKE_RF_DISABLED );
```

# MQTT Publish

```
void mqttpublish()
{
    // read values from the sensor
    float temperature, humidity, pressure;

    if (DEBUG)
    {
        Serial.print("Sending payload: ");
        Serial.println(payload);
    }

    // Create a topic string and publish data to ThingSpeak channel feed.
    String topicString = "channels/" + String( channelID ) + "/publish/" + String(writeAPIKey);
    unsigned int length=topicString.length();
    char topicBuffer[length];
    topicString.toCharArray(topicBuffer,length+1);

    if (client.publish(topicBuffer, (char*) payload.c_str()))
    {
        if (DEBUG) Serial.println("Publish ok");
    }
    else
    {
        if (DEBUG) Serial.println("Publish failed");
    }
}
```

# Riconnessione

```
void reconnect()
{
    String clientName="MY-ESP";
    // Loop until we're reconnected
    while (!client.connected())
    {
        if (DEBUG) Serial.println("Attempting MQTT connection...");
        // Try to connect to the MQTT broker
        if (client.connect((char*) clientName.c_str()))
        {
            if (DEBUG) Serial.println("Connected");
        }
        else
        {
            if (DEBUG)
            {
                Serial.print("failed, try again");
                // Print to know why the connection failed.
                // See http://pubsubclient.kolleary.net/api.html#state for the failure code explanation.
                Serial.print(client.state());
                Serial.println(" try again in 2 seconds");
            }
            delay(2000);
        }
    }
}
```

# Configurazione del Cloud

Creazione di un Nuovo Canale

Accesso al Dashboard

Creazione di un Account ThingSpeak

The screenshot shows the 'Channel Settings' page of a ThingSpeak channel. The top navigation bar includes 'Private View', 'Public View', 'Channel Settings', and 'Sharing'. The 'Sharing' tab is currently selected. The main section is titled 'Channel Settings' and displays the following information:

Setting	Value	
Percentage complete	50%	
ID Canale	██████████	
Nome	Meteohit	
Descrizione	Weather Station ESP8266	
Campo 1	Temperature (°C)	<input checked="" type="checkbox"/>
Campo 2	Humidity (%)	<input checked="" type="checkbox"/>
Campo 3	Pressure (hPa)	<input checked="" type="checkbox"/>
Campo 4	Battery (V)	<input checked="" type="checkbox"/>
Campo 5		<input type="checkbox"/>

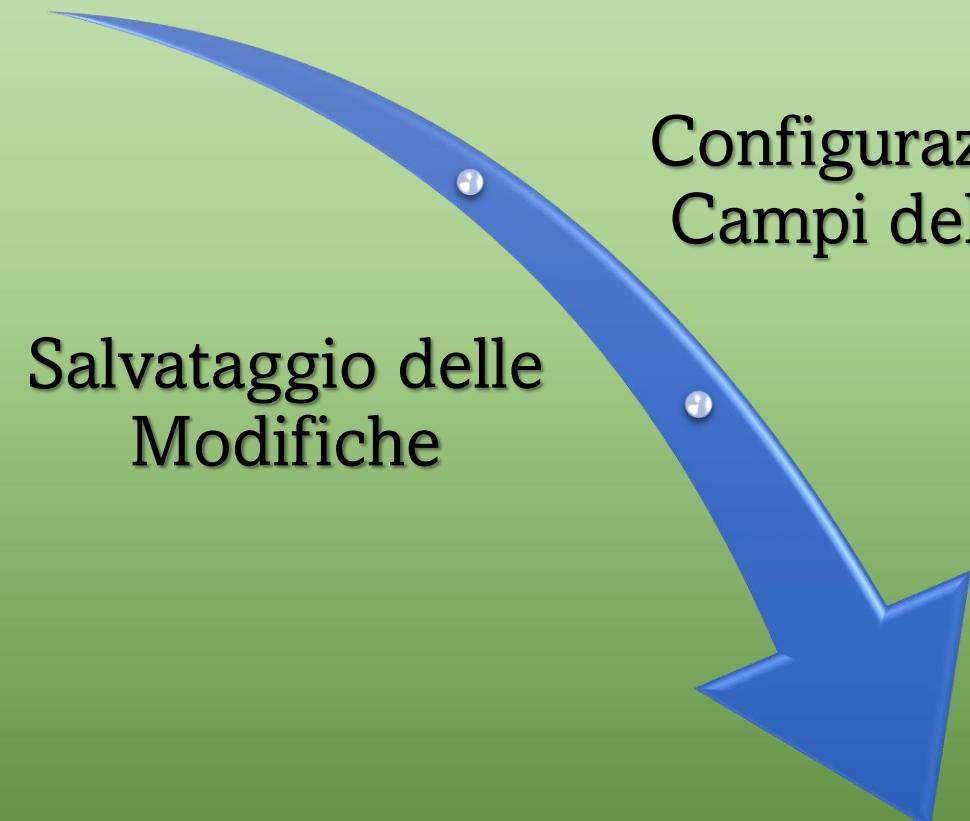
Thingspeak garantisce una maggiore personalizzazione dei dati raccolti offrendo la possibilità di una post analisi dei dati tramite appositi script

## Chiave API di Scrittura

Chiave

[REDACTED]

Genera Nuova Chiave di Scrittura



## Configurazione dei Campi del Canale

Salvataggio delle Modifiche

Ottenere le API Keys

Troveremo due tipi di chiavi: una per la lettura (**Read API Key**) e una per la scrittura (**Write API Key**)

# CONFIGURAZIONE DEVICE MQTT

## ThingSpeak

- Gestione *centralizzata* dei dati
- Offre la flessibilità di configurare un dispositivo MQTT *direttamente sulla piattaforma*
- Supporta sia la *pubblicazione* che la *sottoscrizione* di messaggi MQTT
- Consente l'*interazione bidirezionale* tra i dispositivi IoT e la piattaforma cloud
- Acquisire dati da sensori connessi al dispositivo e trasmetterli *in tempo reale*
- Elevata *versatilità* e un'*integrazione semplificata*



# Analisi dei risultati e problematiche affrontate

## Returns

- -4 : MQTT\_CONNECTION\_TIMEOUT - the server didn't respond within the keepalive time

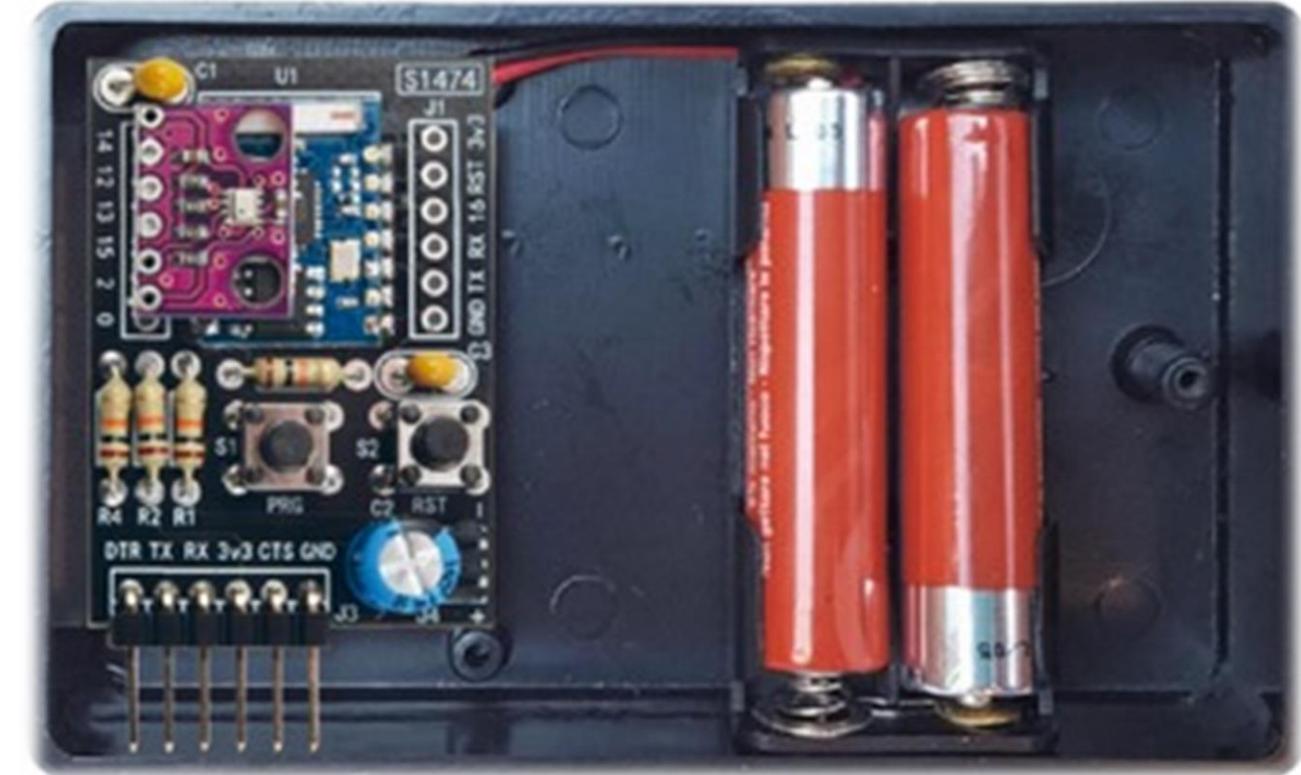
I dati acquisiti sono relativi  
al periodo 27-30 Dec





# Implementazioni future

ALIMENTAZIONE BASATA SU DUE BATTERIE AA



## **Conclusioni**

Il progetto è giunto a una fase quasi operativa in grado di monitorare i parametri ambientali e trasmettere i dati in modo affidabile al cloud. Consolidando l'efficacia del sistema nel contesto dell'Internet delle cose e del monitoraggio ambientale.





Grazie per l'attenzione