



Relazione progetto IoT

Corso di Internet of Things

Università degli studi di Udine, A.A. 2024-2025

Daniele De Martin [162521@spes.uniud.it]

Enrico Peressin [163503@spes.uniud.it]

Massimiliano Di Marco [144714@spes.uniud.it]

Aurora Marzinotto [162556@spes.uniud.it]

SISTEMA DI IRRIGAZIONE AUTONOMO TRAMITE ARDUINO E RASPBERRY PI

Indice

1. Introduzione	3
1.1. Sommario	3
1.2. Descrizione generale e obiettivi del lavoro	3
2. Componenti hardware e software	3
2.1. Strumenti Utilizzati	3
3. Implementazione hardware	5
3.1. Cablaggio pompa - alimentazione - relè	5
3.2. Montaggio sensori di temperatura/umidità e umidità del suolo	6
3.3. Montaggio pompa, sensore di livello e led	6
3.4. Il progetto completo	7
4. Implementazione software	7
4.1. Calibrazione dei sensori	7
4.2. Codice arduino.ino	7
4.3. Codice raspberry-serial.py	9
4.4. Installazione InfluxDB	10
4.5. Installazione Grafana	10
4.6. Scelta dei grafici e interfaccia di Grafana	11
5. Implementazione visione da remoto	12
5.1. Esposizione di una porta su internet e DNS	12
5.2. Servizio di VPN Tailscale	12
5.3. Visualizzazione dei grafici senza autenticazione	13
6. Osservazioni conclusive	13

1. Introduzione

1.1. Sommario

L'obiettivo di questo progetto è sviluppare un sistema di irrigazione automatico per una pianta di basilico. Il sistema deve essere in grado di monitorare l'umidità del terreno e mantenerla entro una soglia regolabile, in base alle esigenze della specie vegetale (nel nostro caso il basilico), attivando l'irrigazione quando necessario. Inoltre, i dati raccolti devono poter essere visualizzabili in tempo reale.

1.2. Descrizione generale e obiettivi del lavoro

Le funzionalità che il sistema deve rispettare sono riportate di seguito:

1. **Controllo di parametri** come temperatura, umidità dell'aria, umidità del terreno.
2. **Controllo del livello dell'acqua** disponibile per l'irrigazione in un apposito recipiente.
3. **Irrigazione automatica** tramite pompa di irrigazione predisposta.
4. **Caricamento su InfluxDb** dei dati raccolti in serie temporale.
5. **Uso di Grafana** per visualizzazione dei dati.
6. **Visualizzazione** da remoto.

2. Componenti hardware e software

2.1. Strumenti Utilizzati

- **Raspberry Pi 5:**

L'ultima versione del single-board computer sviluppato da Raspberry. Viene usato come supporto per il caricamento dei dati, la creazione di grafici e per esporre i servizi rendendoli disponibili da remoto.



- **Arduino Uno R3:**

Piattaforma open-source di prototipazione elettronica programmata tramite l'apposito IDE. Usato nella gestione dei sensori e nella traduzione da segnale analogico a digitale per il sensore di umidità del suolo e quello di livello dell'acqua.



- **Sensore di umidità dell'aria e temperatura (DHT11):**

È stato collegato ad Arduino per controllare la temperatura e l'umidità dell'ambiente.



- **Sensore di umidità del suolo:**

È stato inserito all'interno del terriccio in modo da poter monitorarne l'umidità per gestire l'irrigazione.



- **Sensore di livello dell'acqua:**

Il sensore è stato inserito nel recipiente dell'acqua per poterne controllare il livello e di conseguenza sapere quando ha bisogno di essere riempito.



- **Pompa d'acqua sommergibile:**

Immersa all'interno del recipiente, essa eroga l'acqua pompano attraverso l'apposito tubo.



- **Modulo relè ad un canale:**

Viene utilizzato per azionare la pompa tramite Arduino.



- **Tubo flessibile in PVC:**

Al tubo, posizionato al centro del vaso, sono stati fatti dei fori sul lato destro e sinistro in modo da bagnare uniformemente il terreno della pianta.



- **Alimentazione da 3V:**

Per alimentare la pompa dell'acqua viene utilizzata un'alimentazione da 3V realizzata ponendo due batterie da 1,5 V in serie.



- **Led:**

Viene usato un led rosso per avvisare quando il livello dell'acqua è sotto una soglia limite e necessita di rifornimento.

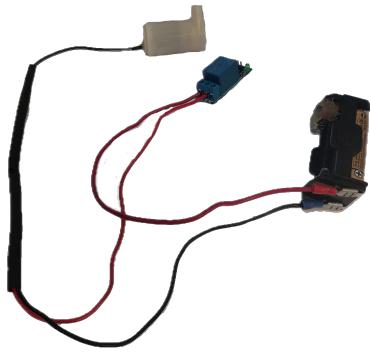


3. Implementazione hardware

Il progetto è stato costruito su una base di due tavolette di legno per rendere il tutto trasportabile e più ordinato. Nella prima tavoletta è stato posto un recipiente per l'acqua e di fianco la piantina di basilico. Nella seconda tavoletta è stata predisposta l'elettronica del progetto: Arduino con relativa breadboard e tutti i sensori sopraccitati, case con le batterie per l'alimentazione del motorino, relè e il Raspberry alimentato via USB-C.

3.1. Cablaggio pompa - alimentazione - relè

Il motorino per pompare l'acqua necessita di una alimentazione di 3V. È stato quindi preso un porta batterie e inserite due batterie, posizionate in serie, di tipo AA ciascuna da 1.5V. Il polo negativo della batteria è stato collegato direttamente al cavo che identifica il polo negativo del motorino con apposito incastro e dopo aver stagnato i cavi per garantire una miglior tenuta. Il polo positivo della batteria è stato collegato all'ingresso «normalmente aperto» del relè. Il circuito è stato poi chiuso mettendo il polo positivo del motorino nell'ingresso di «controllo» del relè. Il motorino rimane spento in condizioni normali e viene attivato in base alle condizioni rilevate dai sensori, chiudendo il circuito tramite l'attivazione del relè



Il pin di input del relè è stato collegato direttamente al pin digitale 13 di Arduino (identificato dal cavo arancione), mentre gli altri pin di ground e tensione sono stati collegati nelle apposite posizioni della breadboard.

3.2. Montaggio sensori di temperatura/umidità e umidità del suolo

Il sensore DHT11, per la rilevazione di temperatura e umidità dell'aria, è stato inserito nella breadboard rivolto verso l'esterno. Il pin di input è stato collegato (con un cavo blu) sulla porta digitale 8 di Arduino.

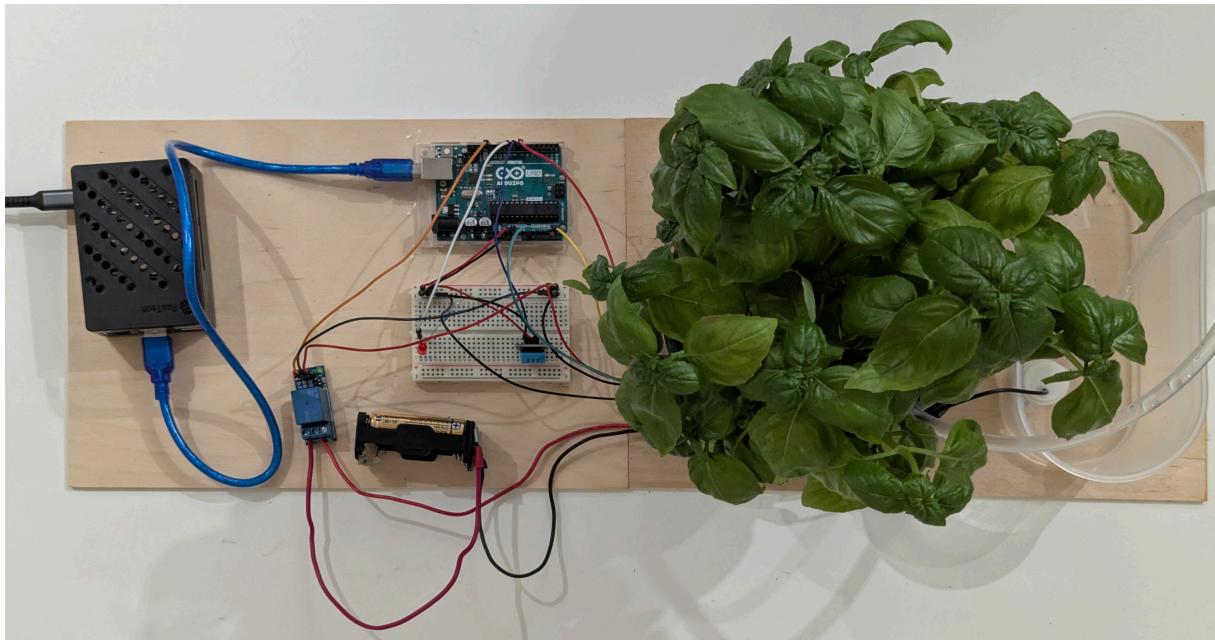
Il sensore per l'umidità del suolo è stato inserito in profondità, circa al centro della pianta, in modo da avere una rilevazione più accurata e precisa possibile. Il suo pin di trasmissione è stato inserito nell'ingresso analogico A5, identificabile dal cavo giallo.

3.3. Montaggio pompa, sensore di livello e led

La pompa è stata fissata alla base di un recipiente contenente dell'acqua, in modo tale da garantirne quasi il totale utilizzo. Pochi centimetri in parte è stato fissato allo stesso modo il sensore di livello in modo che poggiasse a terra per fornire un'indicazione più veritiera possibile del livello attuale dell'acqua. Il pin di segnale del sensore è stato collegato al pin analogico A4, identificabile dal cavo verde, mentre il canale di alimentazione al pin digitale 7 in modo da alimentarlo solo quando necessario per evitare corrosioni.

È stato poi posto un led rosso sulla breadboard, collegato con apposita resistenza di 220Ω , che verrà poi acceso quando il livello dell'acqua nella bacinella è critico. Il led è stato collegato sul pin digitale 9, identificabile dal cavo bianco.

3.4. Il progetto completo



4. Implementazione software

Sono stati sviluppati due file: `arduino.ino`, caricato nell'Arduino e `raspberry-serial.py`, in esecuzione sul Raspberry Pi.

4.1. Calibrazione dei sensori

Sono stati rilevati i valori registrati dal sensore di umidità del terreno nelle situazioni di totale aridità e saturazione del terriccio. Nel primo caso viene registrato un valore approssimativo di 350, nel secondo attorno ai 650.

L'umidità ideale del terreno per una pianta di basilico si aggira intorno al 60-70%, quindi è stato posto il valore di soglia a 450, che corrisponde al 40% di aridità (cioè il 60% di umidità), come valore sopra al quale la pompa inizia l'erogazione dell'acqua.

Il sensore di livello dell'acqua registra un valore intorno a 200 quando il contenitore è pieno e scende fino a circa 0 quando è vuoto. È stato scelto 15 come valore limite, che indica di riempire il contenitore.

4.2. Codice arduino.ino

Il programma nella fase `setup()` (eseguita una sola volta all'avvio) inizializza la comunicazione seriale (`Serial.begin(9600)`) per stampare i dati nel monitor seriale. Inoltre, viene avviato il sensore DHT11 con `dht.begin()`, vengono impostati i pin in modalità input/output a seconda del componente. La funzione `loop()`, che si ripete all'infinito, legge i dati dai sensori e stampa su monitor seriale tutti i valori letti nel formato:

TEMPERATURA: t UMID_ARIA: h UMID_SUOL0: m LIV_ACQUA: l, dove t , h , m , l rappresentano i valori rilevati dal sensore.

Se il valore di umidità del suolo rilevato dal sensore supera 450 (cioè troppo secco), viene accesa la pompa per 2 secondi. Un'altra condizione è presente sul sensore di livello dell'acqua: quando il livello è minore di 15 viene acceso il led rosso, indicando all'utente di riempire il contenitore d'acqua.

Alla fine del `loop()` è presente un `delay(600000)` che sospende l'Arduino per 10 minuti.

cpp

```
arduino.ino
1 #include <DHT.h>
2
3 int MOTORE = 13;
4 int UM_SUOLO = A5;
5 int TEMP = 8;
6 int SENSOR_POWER = 7;
7 int WATER_LVL = A4;
8 int LED = 9;
9
10 int val = 0;
11 float liv_acqua = readSensor();
12
13 #define DHTTYPE DHT11
14 DHT dht(TEMP, DHTTYPE);
15
16 void setup() {
17   Serial.begin(9600);
18   dht.begin();
19   pinMode(LED, OUTPUT);
20   pinMode(MOTORE, OUTPUT);
21   pinMode(UM_SUOLO, INPUT);
22   pinMode(TEMP, INPUT);
23   pinMode(SENSOR_POWER, OUTPUT);
24   digitalWrite(SENSOR_POWER, LOW);
25   digitalWrite(MOTORE, HIGH);
26   delay(500);
27 }
28
29 void loop(){
30   int t = dht.readTemperature();
31   int h = dht.readHumidity();
32   float umidita_suolo = analogRead(UM_SUOLO);
33   Serial.print("TEMPERATURA:");
34   Serial.print(t);
35   Serial.print(" UMID_ARIA:");
36   Serial.print(h);
37   Serial.print(" UMID_SUOLO:");
38   Serial.print(umidita_suolo);
39   Serial.print(" LIV_ACQUA:");
40   Serial.println(liv_acqua);
41
42   if(umidita_suolo>450){ // soglia umidità terreno
43     digitalWrite(MOTORE, LOW);
44     delay(2000);
45     digitalWrite(MOTORE, HIGH);
46   } else{
47     digitalWrite(MOTORE, HIGH);
48     delay(2000);
49
50   }
51
52   if(WATER_LVL<15){ // soglia acqua
53     digitalWrite(LED, HIGH);
54     delay(1000);
55     digitalWrite(LED, LOW);
56   }
57
58   delay(600000); // sospensione per 10 minuti
59 }
```

```

49     }
50
51     if(liv_acqua<15){
52         digitalWrite(LED, HIGH);
53
54     } else {
55         digitalWrite(LED, LOW);
56     }
57
58     Serial.println();
59     delay(600000); // 10 minuti
60 }
61
62 int readSensor() {
63     digitalWrite(SENSOR_POWER, HIGH);
64     delay(10);
65     val = analogRead(WATER_LVL);
66     digitalWrite(SENSOR_POWER, LOW);
67     return val;
68 }

```

4.3. Codice raspberry-serial.py

Questo script Python viene eseguito sul Raspberry Pi e ha lo scopo di ricevere i dati tramite la porta seriale dall'Arduino e inviarli al database di *InfluxDB*.

Il programma è così strutturato: viene aperta una connessione sulla porta seriale corrispondente al cavo USB con cui Arduino è stato collegato.

Arduino invia ogni 10 minuti una stringa contenente i valori dei sensori, che vengono estratti e convertiti. Si registra un timestamp e si crea un oggetto Point con tutti i valori, che viene poi inviato a InfluxDB tramite `write_api`. Ad ogni scrittura avvenuta correttamente, viene stampato l'output nel terminale per conferma. Nel codice vengono usate le seguenti librerie:

- `pyserial`: permette di comunicare tramite porte seriali, serve a leggere i dati da Arduino e inviarli a Raspberry Pi.
- `influxdb-client`: Interagisce con InfluxDB e serve per creare i punti dati e inviarli al server.

python

```

raspberry-serial.py
1 #!/usr/bin/env python3
2 import serial
3 import time
4 from serial.tools import list_ports
5 from influxdb_client import InfluxDBClient, Point, WriteOptions
6
7 # --- CONFIGURAZIONE SERIAL ---
8 # Impostare manualmente la porta
9 SERIAL_PORT = '/dev/ttyACM0'
10
11 BAUDRATE = 9600
12 TIMEOUT = 2 # secondi
13
14 # --- CONFIGURAZIONE INFLUXDB ---
15 INFLUX_URL = "http://localhost:8086"

```

```

16 INFLUX_TOKEN = "TOKEN"
17 INFLUX_ORG = "IoT"
18 INFLUX_BUCKET = "bucket_sensori"
19
20 def main():
21     port = SERIAL_PORT
22     if port is None:
23         print("Porta Arduino non trovata")
24         return
25
26     ser = serial.Serial(port, BAUDRATE, timeout=TIMEOUT)
27     print(f"Connesso ad Arduino su {port} @ {BAUDRATE}bps")
28
29     # Inizializzazione client InfluxDB
30     client = InfluxDBClient(url=INFLUX_URL, token=INFLUX_TOKEN, org=INFLUX_ORG)
31     write_api = client.write_api(write_options=WriteOptions(batch_size=1))
32     time.sleep(3)
33     line = ser.readline().decode()
34     while True:
35         line = ser.readline().decode()
36         if "TEMPERATURA" in line and "UMID_ARIA" in line and
37             "UMID_SUOLO" in line and "LIV_ACQUA" in line:
38             line = line.split(" ")
39             temp = int(line[0].split(":")[1])
40             hum = int(line[1].split(":")[1])
41             moisture = float(line[2].split(":")[1])
42             liv_acqua = float(line[3].split(":")[1])
43
44             # Creazione punto InfluxDB
45             timestamp = time.time_ns()
46             point = Point("sensori_pianta").tag("device",
47                 "arduino").field("Temperatura", temp).field
48                 ("Umidita' Aria", hum).field("Umidita' Terreno",
49                 moisture).field("Livello Acqua", liv_acqua).time(timestamp)
50
51             write_api.write(bucket=INFLUX_BUCKET, org=INFLUX_ORG, record=point)
52             print(f"Inviati: T={temp} C H={hum}% M={moisture} L={liv_acqua}")
53
54
55 if __name__ == "__main__":
56     main()

```

4.4. Installazione InfluxDB

InfluxDB V2 è stato installato sul Raspberry Pi direttamente dal repository ufficiale. Dopo l'installazione è stato abilitato il servizio con il comando `sudo systemctl enable influxdb`. La dashboard web di InfluxDB è disponibile alla porta 8086 del Raspberry Pi. Da qui è stato creato un utente admin, un'organizzazione e un bucket, sul quale verranno salvati i dati.

4.5. Installazione Grafana

Come per InfluxDB, Grafana è stato installato sul Raspberry Pi dal repository ufficiale e con `sudo systemctl start grafana-server` è stato avviato. L'interfaccia web è accessibile alla porta 3000.

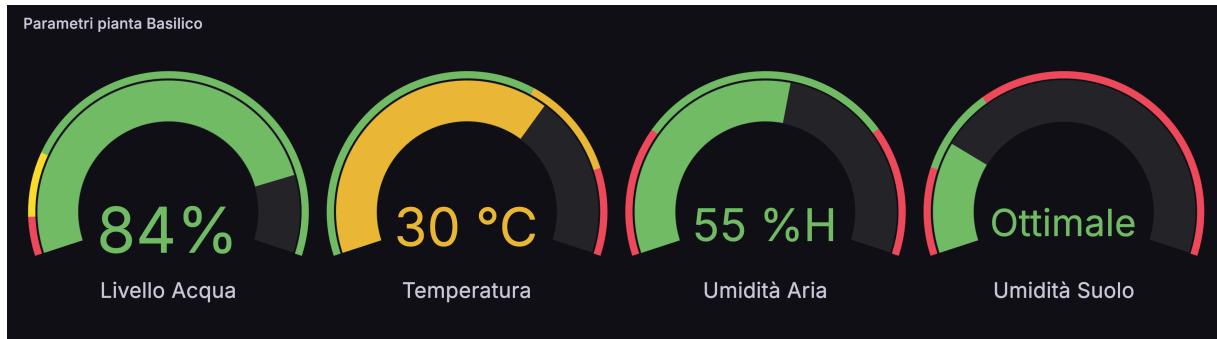
Nella sezione *Connections > Data source* è stato aggiunto InfluxDB, inserendo tutti i parametri impostati precedentemente (credenziali, nome organizzazione, bucket...). Nella sezione *Dashboard*, sono state create le due visualizzazioni, prendendo i dati da InfluxDB tramite la query scritta in Flux.

```
flux
Query
1 from(bucket: "bucket_sensori")
2 |> range(start: -24h)
3 |> map(fn: (r) => ({
4   r with
5     _value:
6       if r._field == "Livello Acqua"
7         then float(v: r._value) / 2
8         else float(v: r._value)
9 }))
```

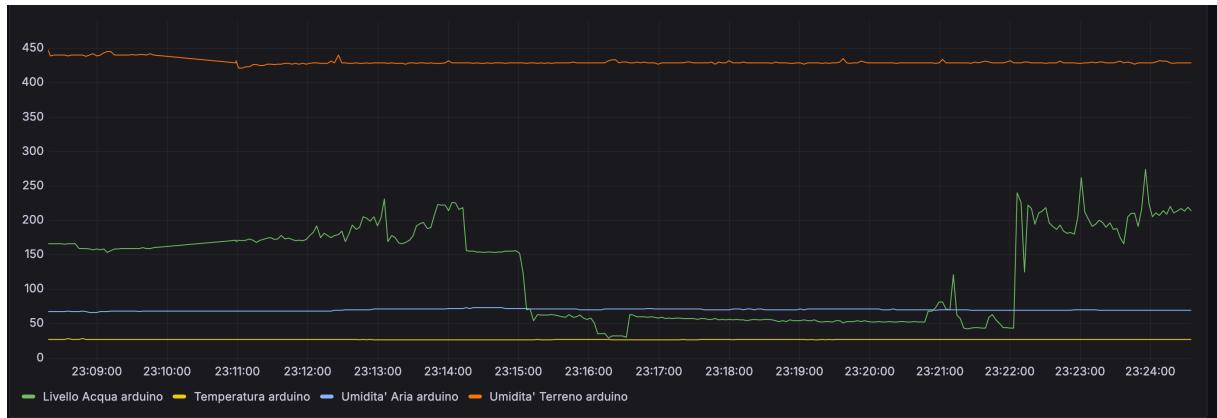
La query richiede le tuple provenienti dal bucket *bucket_sensori* e fa un mapping per rendere il valore del livello dell'acqua espresso in forma percentuale ed essere così meglio interpretabile.

4.6. Scelta dei grafici e interfaccia di Grafana

Dopo aver scritto l'opportuna query per estrarre i dati da *InfluxDB*, è stato scelto come grafico per l'esposizione quello di tipo *Gauge*. Esso permette la visualizzazione live dei quattro parametri di interesse: livello dell'acqua, temperatura, umidità dell'aria e del suolo, inoltre sono stati aggiunti dei parametri di threshold, unità di visualizzazione e colorazioni per interpretare al meglio i dati.



È stata anche creata un'altra dashboard del tipo *Time Series*, per visualizzare l'andamento dei vari parametri.



È stato attivato il sistema di alert che *Grafana* mette a disposizione impostando un bot su *Telegram* (@basilicoobot) che avvisa quando il livello della bacinella scende sotto il valore di soglia, inviando una notifica.



5. Implementazione visione da remoto

Per visualizzare i grafici su *Grafana*, se si è nella stessa rete del Raspberry, è sufficiente collegarsi alla porta 3000 di esso conoscendo il suo indirizzo IP privato (nel nostro caso 192.168.1.32). Per rendere questo servizio accessibile anche se si è fuori casa si presentano due soluzioni: esporre una porta del router di casa in modo da reindirizzare sulla porta corretta del Raspberry oppure sfruttare un servizio di VPN.

5.1. Esposizione di una porta su internet e DNS

Questa alternativa permette di rendere visibile il servizio su internet, pertanto chiunque disponga di una connessione a internet è libero di visualizzare i grafici su *Grafana*. Per iniziare sono state modificate le tabelle di NAT del router di casa, in questo modo tutte le richieste che arrivano al router nella porta 3000 vengono reindirizzate all'indirizzo 192.168.1.32 nella porta 3000 dove è esposto il servizio *Grafana*.

Per rendere l'interazione migliore e non doversi ricordare l'indirizzo IP del router a memoria è stato attivato un nome di dominio che va a sostituire l'indirizzo IP pubblico del router di casa.

L'accesso a *Grafana* è quindi disponibile al link <http://basilico.blog:3000> da qualsiasi dispositivo con connessione a internet.

5.2. Servizio di VPN Tailscale

Per rendere meno dipendente il Raspberry dal router di casa è stato adottato anche il servizio **Tailscale**, un servizio di mesh VPN costruita su WireGuard che crea una rete privata fra i dispositivi iscritti. In questo modo, anche se il Raspberry viene spostato in un'altra rete, per esempio collegato a un hotspot 4G, i dispositivi nella rete privata del Raspberry possono continuare a raggiungerlo liberamente. Grazie a **MagicDNS** si evita di ricordare gli indirizzi IP, quindi il servizio di *Grafana* è disponibile al link <http://raspberrypi:3000> da qualunque device autenticato nella rete privata.

5.3. Visualizzazione dei grafici senza autenticazione

Il grafico Gauge è disponibile liberamente al link <http://basilico.blog:3000/public-dashboards/93dd5dfa69884ef0a5264a5a2b108807> quando il Raspberry è connesso al router di casa.

In alternativa è sempre disponibile al link <http://raspberrypi:3000/public-dashboards/93dd5dfa69884ef0a5264a5a2b108807> se si è connessi tramite dispositivo registrato alla rete Tailscale.

6. Osservazioni conclusive

Questo progetto ci ha permesso di sperimentare con strumenti diffusi come Arduino e Raspberry Pi, con l'obiettivo di monitorare e regolare i parametri ambientali e vitali di una pianta di basilico.

L'implementazione non ha comportato costi elevati per quanto riguarda i sensori e la pompa dell'acqua (~15€), mentre più ingenti sono state quelle per il Raspberry Pi 5 (~100€) e l'Arduino Uno (~30€). È importante sottolineare che, pur potendo realizzare una versione più semplice del sistema anche senza Raspberry, questo ricopre il ruolo fondamentale di server locale per raccolta e visualizzazione remota dei dati.

I software utilizzati (*InfluxDB* e *Grafana*) sono risultati intuitivi e di facile gestione. Anche i codici .ino e .py di supporto ad'Arduino e a Raspberry non hanno richiesto conoscenze eccessivamente approfondite per essere implementati.

In conclusione, il progetto ci ha permesso di spaziare su più fronti sia nell'ambito software che nell'ambito hardware del mondo dell'Internet of Things e realizzare un sistema di controllo di irrigazione automatica che può essere adattato o ampliato con altre piante semplicemente modificando i parametri rilevati dai sensori.