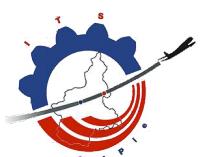
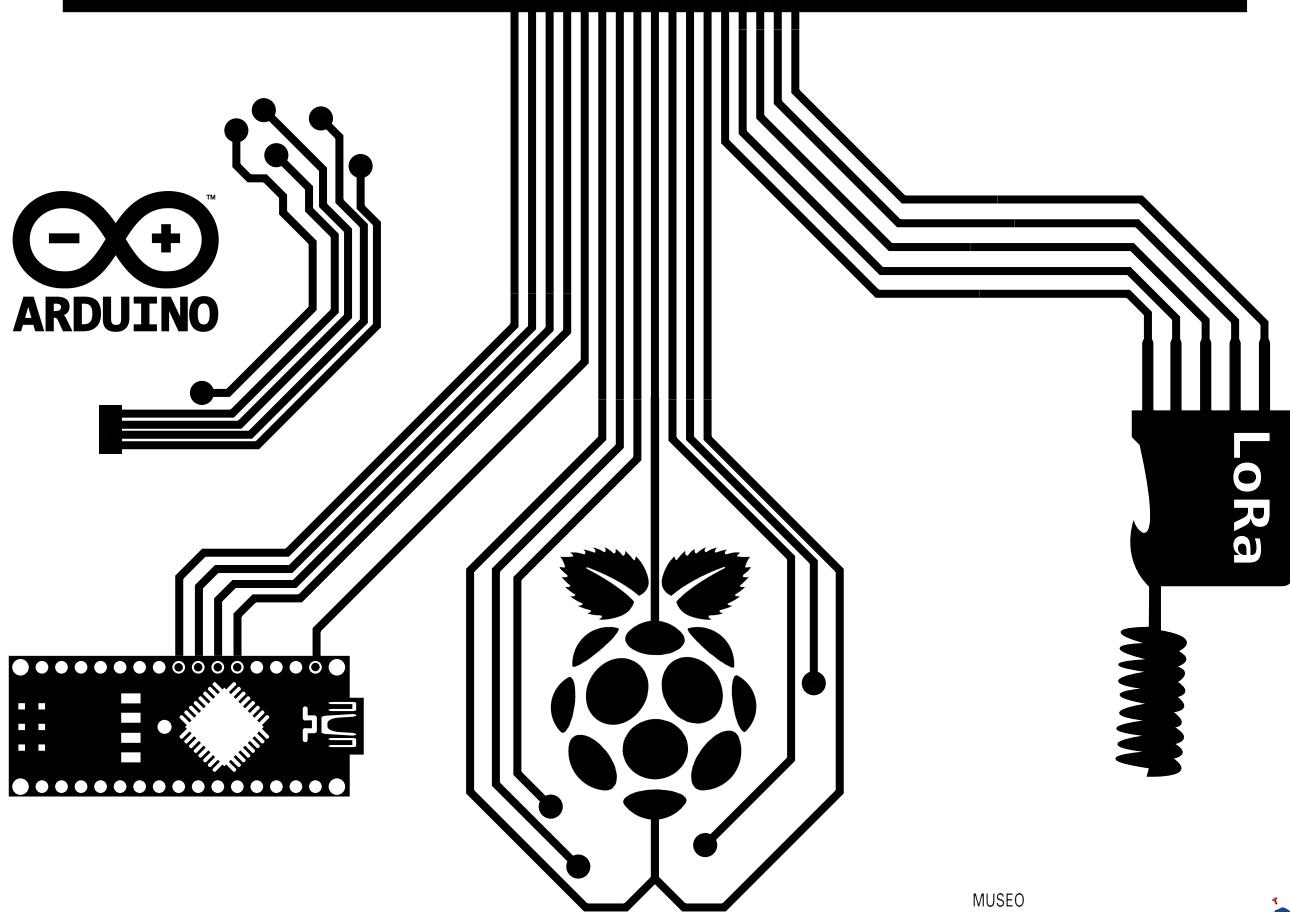


# Rivellino Sensors Manual



# Indice

<b>Moduli.....</b>	<b>5</b>
Modulo di elaborazione:.....	6
Cavo di alimentazione.....	6
Monitor Raspberry Pi.....	6
Modulo A (tirante):.....	7
Arduino Nano 33 BLE.....	7
Air quality 5 Click (MIKROE-3056).....	8
Regolatore LM2596 HW411.....	9
LORA RYLR498 (Condivisi   A, B, C).....	9
Modulo B (Galleria):.....	10
Arduino Pro Mini(Condivisi   B, C).....	10
High Sensitivity Water Sensor (HW-038).....	10
Pacco batterie 5xAA (Condivisi   B, C).....	10
DHT11 (Condivisi   B, C).....	10
Regolatore MCP1700-3302E (Condivisi   B, C).....	10
LORA RYLR498 (Condivisi   A, B, C).....	10
Modulo C (Polveriera):.....	11
Arduino Pro Mini (Condivisi   B, C).....	11
Pacco batterie 5xAA (Condivisi   B, C).....	11
DHT11 (Condivisi   B, C).....	11
Regolatore MCP1700-3302E (Condivisi   B, C).....	11
LORA RYLR498 (Condivisi   A, B, C).....	11
<b>Condivisi.....</b>	<b>12</b>
Comuni ai Moduli A, B e C.....	12
LORA RYLR498 Transceiver Module.....	12
Comuni ai moduli B e C.....	13
Arduino Pro Mini.....	13
DHT11 – Sensore di Temperatura e Umidità.....	14
Regolatore MCP1700-3302E.....	15
Pacco batterie 5xAA.....	15
<b>Software.....</b>	<b>16</b>
<b>    Software LoRa.....</b>	<b>16</b>
Spiegazione codice.....	16
Configurazione iniziale.....	16
Formato dei dati trasmessi.....	17
Gestione della ricezione (modulo A).....	17
Gestione degli errori.....	17
<b>    Software Modulo A.....</b>	<b>18</b>
Spiegazione codice.....	18
Ciclo operativo.....	18
Gestione dei sensori.....	19
Comunicazione con il Raspberry.....	19

Ottimizzazione energetica.....	20
Note.....	20
<b>Software Modulo B.....</b>	<b>20</b>
Spiegazione codice.....	20
Ciclo operativo.....	21
Gestione dei sensori.....	21
Ottimizzazione energetica.....	22
Comportamento in caso di errore.....	22
Note.....	22
<b>Software Modulo C.....</b>	<b>23</b>
Spiegazione codice.....	23
Ciclo operativo.....	23
Gestione dei sensori.....	24
Ottimizzazione energetica.....	24
Gestione degli errori.....	24
Note.....	25
<b>Linee generali di funzionamento del sistema.....</b>	<b>25</b>
Raccolta.....	25
Trasmissione.....	26
Elaborazione.....	26
Esportazione.....	27
Sincronizzazione temporale.....	27
Tolleranza a imprevisti.....	27
<b>Glossario.....</b>	<b>28</b>

# Premesse

Il progetto ha come obiettivo la realizzazione di un sistema di monitoraggio continuo di parametri ambientali nel sito del Rivellino degli Invalidi, del Museo Pietro Micca di Torino.

Sulla base delle informazioni disponibili ed in seguito ad una fase emotiva (per individuare le necessità degli interessati al progetto), di definizione (delle specifiche dei dispositivi successivamente sviluppati), Ideazione, prototipazione e test; si è scelto di realizzare un sistema centralizzato, ove un modulo di elaborazione (si veda in seguito) gestisce i dati inviati in modo asincrono da 3 sensori separati (moduli).

Uno di essi (modulo A, collegato fisicamente alla rete elettrica ed al modulo di elaborazione) ha l'ulteriore compito di ricevere via onde radio UHF i dati inviati dagli altri due moduli e successivamente di inviarli tramite seriale direttamente al modulo di elaborazione.

Nelle seguenti pagine si farà riferimento ai moduli secondo diverse convenzioni in base al caso:

- Modulo del tirante (A)
- Modulo della galleria (B)
- Modulo della polveriera (C)

Si ringraziano i professori Barra, Bertoni, Bogoni, Duroux per il materiale ed i consigli forniti.

Si ringrazia Fondazione ITS meccatronica e aerospazio per il materiale nonché per l'occasione di lavorare sul progetto in questione.

Si ringrazia il gruppo di lavoro del corso EMBT01 per il codice fornito.

Si ringrazia il gruppo di lavoro del corso EMBT02 per il materiale prodotto in parallelo a quello presentato e il supporto nello sviluppo elettrico, elettronico e meccanico.

Si ringraziano il Direttore Zaffaroni del Museo Pietro Micca di Torino e l'Archeologo Zannoni per il tempo investito nel progetto.

Questo manuale è a cura di: Barletta Gabriele, Dama Simone, Trombetta Francesco  
Il materiale prodotto e su cui si è basato questo manuale sono reperibili alla pagina GIT.

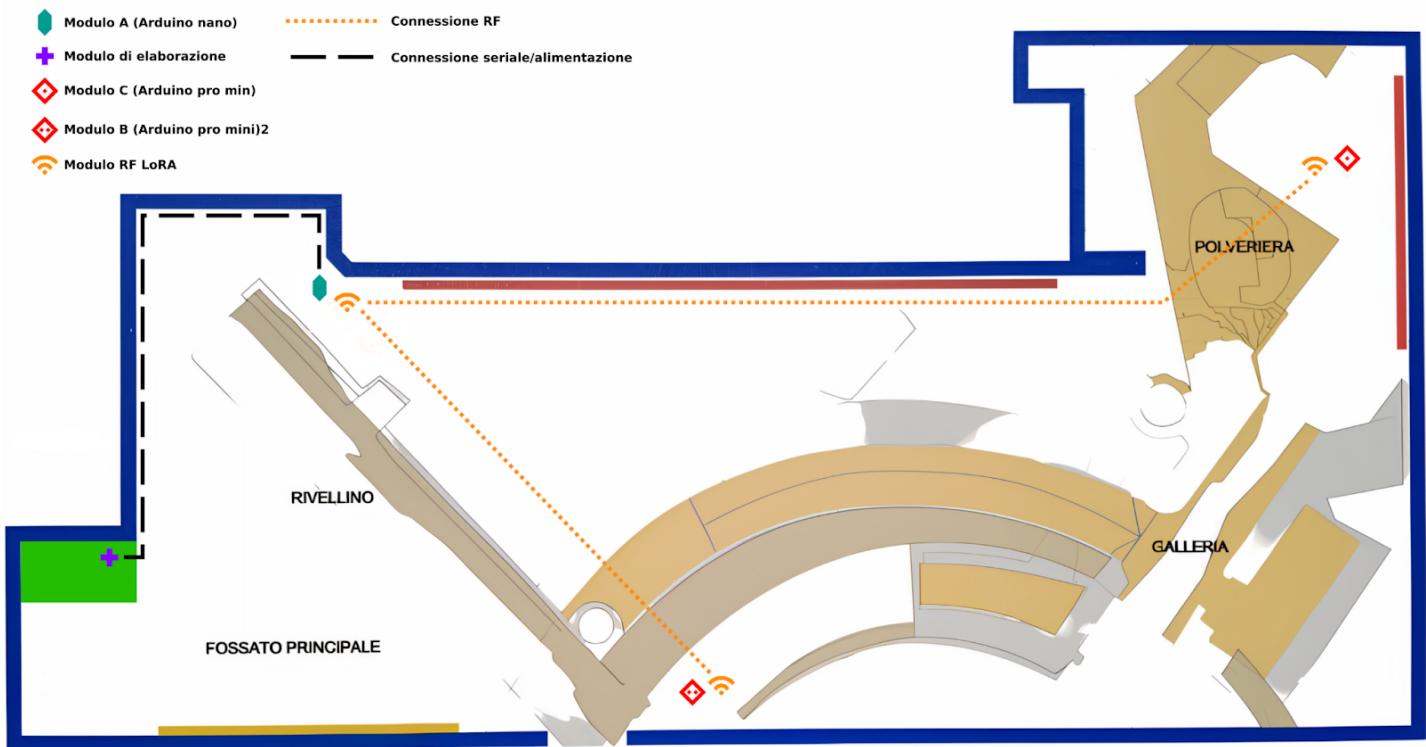
Le grafiche, tra cui la copertina sono a cura di Barletta Gabriele.

# Moduli

Questa sezione si dedica alle componenti di ogni modulo e le specifiche delle stesse.

I componenti presenti in più di un modulo sono stati trattati in dettaglio nella sezione successiva ("Condivisi") e si presentano come:

**Nome comp. (Nome sezione** in cui è trattato | **moduli** a cui è comune).



# **Modulo di elaborazione:**

## **Raspberry Pi 4 B**

### **Funzionamento**

Il modulo di elaborazione ha il compito di ricevere ed elaborare i dati prodotti dai moduli A, B e C, organizzarli in un formato leggibile e conservarli sotto forma di file CSV.

### **Processore:**

Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC  
1.8GHz

**RAM:** 2GB

**GPIO:** 40 Header GPIO

### **Alimentazione:**

- 5V CC tramite connettore USB-C (minimum 3A1 ) • 5V DC via GPIO header (minimo 3A1)
- 5V CC tramite GPIO

### **Connettori:**

- IEEE 802.11b/g/n/ac wireless LAN2.4 GHz e 5.0 GHz, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 Porte USB 3.0
- 2 Porte USB 2.0

### **Audio/Video:**

- 2 Porte micro HDMI ports (up to 4Kp60 supported)
- 2 Porte lane MIPI DSI display
- 2 Porte lane MIPI CSI camera
- 4 Porte audio stereo e video composito

Vano SD: Micro SD per immagazzinamento dati

Temperatura di esercizio 0 °C - +50 °C

### **Cavo di alimentazione**

### **Monitor Raspberry Pi**

# Modulo A (tirante):

## Arduino Nano 33 BLE

### Processore:

Nordic nRF52840 (via modulo NINA-B306) – Arm® Cortex®-M4F con FPU, 64 MHz.

**Memoria:** 1 MB Flash + 256 kB RAM.

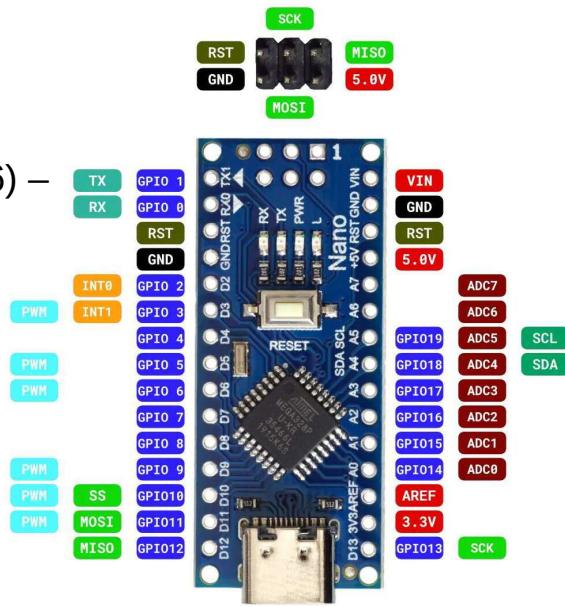
### Connettività wireless:

Bluetooth® 5 (2 Mbps, Long Range, Advertising Extensions), IEEE 802.15.4, supporto Thread e Zigbee.

### Alimentazione:

3.3V (non tollerante a 5V!)

Alimentabile tramite USB (Micro-B), pin VIN (fino a 21V con regolatore MP2322) o pin VUSB.



### GPIO:

30 pin totali (2x15 header), compatibili breadboard; inclusi PWM, SPI, I<sup>2</sup>C (A4/SDA, A5/SCL), UART, ADC/DAC.

### Sensori integrati:

- **IMU 9-axis:** BMI270 (accelerometro  $\pm 2/4/8/16\text{g}$  + giroscopio  $\pm 125\text{--}2000\text{ dps}$ ) + BMM150 (magnetometro  $\pm 1300/2500\text{ }\mu\text{T}$ ).
- **Barometro/temperatura:** LPS22HB (260–1260 hPa, 24-bit;  $\pm 0.1^\circ\text{C}$ ).
- **Umidità/temperatura:** HS3003 (0–100% RH,  $\pm 1.5\%$  RH;  $\pm 0.1^\circ\text{C}$ )
- **Gesto/prossimità/luce:** APDS-9960 (ALS, RGB, prossimità IR, gesture recognition).
- **Microfono digitale:** MP34DT06JTR (122.5 dB SPL, 64 dB SNR, omnidirezionale).

### Interfacce:

USB Full-Speed (12 Mbps), NFC-A, QSPI, TWI (I<sup>2</sup>C), I<sup>2</sup>S, PDM, QDEC.

**Sicurezza:** Arm® CryptoCell CC310 (AES-128/ECB/CCM).

**Temperatura di esercizio:** -40 °C - +85 °C.

### Note:

- I/O a 3.3V **non tolleranti a 5V**.
- Pin A4/A5 preconfigurati come I<sup>2</sup>C.
- Recupero bootloader: doppio tap su reset all'accensione.

# Air quality 5 Click (MIKROE-3056)

## Sensore principale:

MiCS-6814 – MOS tripla con tre elementi indipendenti:

- **RED** (riduttore: CO, H<sub>2</sub>, C<sub>2</sub>H<sub>5</sub>OH, CH<sub>4</sub>, C<sub>3</sub>H<sub>8</sub>, C<sub>4</sub>H<sub>10</sub>).
- **OX** (ossidante: NO<sub>2</sub>, O<sub>3</sub>).
- **NH<sub>3</sub>** (ammoniaca).

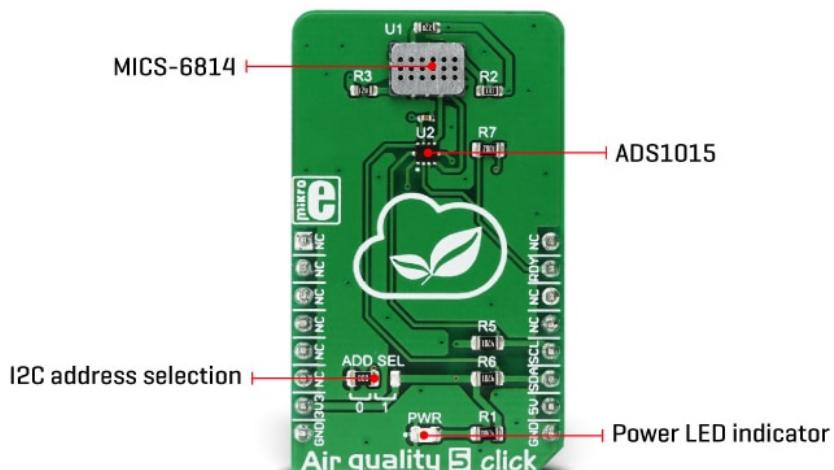
## Convertitore ADC:

ADS1015 – ADC a 12 bit, low-power, con riferimento interno e comparatore programmabile (Texas Instruments)

Interfaccia: I<sup>2</sup>C.

## Tensione di alimentazione:

3.3V / 5V.



# Regolatore LM2596 HW411

## Tipologia:

Regolatore switching buck non isolato.

**Tensione di ingresso:** 4 V – 40 V.

**Tensione di uscita:** 1.25 V – 35 V (regolabile tramite trimmer)

## Corrente di uscita:

- 2 A continuo
- 3 A massimo (con dissipatore)
- Frequenza di commutazione: 150 kHz
- Efficienza: Fino al 92% (a tensioni di uscita elevate).

## Precisione regolazione:

- Regolazione di carico:  $\pm 0.5\%$
- Regolazione di linea:  $\pm 2.5\%$

**Dropout minimo:** 2 V.

**Protezioni integrate:** Limitazione corrente, Spegnimento termiche.

**Corrente di stand-by:** 80  $\mu$ A (tipica).

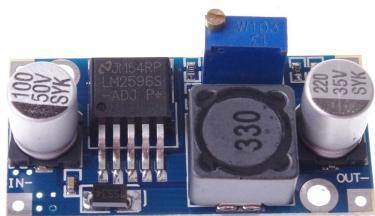
**Package del chip:** TO-220 / TO-263.

**Dimensioni modulo PCB:** 43.6 mm  $\times$  21 mm  $\times$  14 mm.

**Interfaccia utente:** Trimmer potenziometro per regolazione tensione d'uscita.

## Note:

- Uscita regolata tramite feedback a 1.25 V.
- Richiede solo 4 componenti esterni (induttore, diodo, condensatori, trimmer).
- Non isolato, ingresso e uscita condividono massa.



# LORA RYLR498 (Condivisi | A, B, C)

## **Modulo B (Galleria):**

### **Arduino Pro Mini(Condivisi | B, C)**

### **High Sensitivity Water Sensor (HW-038)**

#### **Funzionamento:**

Sensore resistivo a tracce esposte:  
rileva la presenza e il livello di acqua  
tramite la conduttività tra tracce interdigitate.



**Tensione di alimentazione:** 5 V.

**Corrente assorbita:** < 20 mA.

#### **Interfaccia:**

Analogica (segnale 0 – 4.2 V) –  
compatibile anche con ingresso digitale  
per rilevamento binario.

#### **Dimensioni sensore:**

- Area attiva: 40 mm × 16 mm.
- Dimensioni totali: 65 mm × 20 mm × 8 mm.  
Peso: 3 g.  
Temperatura di esercizio: 10 °C – 30 °C.  
Pinout: S: Segnale (analogico), +: (5 V), -: (GND).

#### **Note:**

- Uscita analogica proporzionale al livello del contatto con l'acqua
- Compatibile con Arduino UNO, Mega2560, ADK, ecc.

### **Pacco batterie 5xAA (Condivisi | B, C)**

### **DHT11 (Condivisi | B, C)**

### **Regolatore MCP1700-3302E (Condivisi | B, C)**

### **LORA RYLR498 (Condivisi | A, B, C)**

**Modulo C (Polveriera):**

**Arduino Pro Mini (Condivisi | B, C)**

**Pacco batterie 5xAA (Condivisi | B, C)**

**DHT11 (Condivisi | B, C)**

**Regolatore MCP1700-3302E (Condivisi | B, C)**

**LORA RYLR498 (Condivisi | A, B, C)**

# Condivisi

## Comuni ai Moduli A, B e C

### LORA RYLR498 Transceiver Module

**Frequenza operativa:**

~ 490 MHz



**Modulazione:**

LoRa® (Spread Spectrum) – configurabile con comandi AT.

**Sensibilità RF:** -129 dBm.

**Potenza RF regolabile:** 0 - 22 dBm (max 14 dBm per conformità CE).

**Interfaccia:** UART (300 - 115200 bps, 8N1).

**Antenna:** Integrata (versione base).

**Alimentazione:** 2.3 V – 3.6 V (tipico 3.3 V).

**Consumo:**

- Trasmissione (22 dBm): 144.7 mA.
- Ricezione: 17.5 mA.
- Modalità sleep: 10 µA.
- Smart RX (es. AT+MODE=2,1000,1000): 2.65 mA.

**Protocollo:** Comandi AT per configurazione (indirizzo, banda, crittografia, modalità).

**Memoria integrata:** Flash con 200k cicli di scrittura/cancellazione.

**Temperatura di esercizio:** -40 °C ÷ +85 °C.

**Connettore:** 5 pin (VDD, NRST, RXD, TXD, GND).

**Certificazioni:** FCC, CE RED, NCC.

**Note:**

- Il pin NRST è attivo basso (con pull-up interno da 100 kΩ).
- Per conformità CE, RFOP ≤ 14 dBm.
- Supporta crittografia dati nei comandi AT.

# Comuni ai moduli B e C

## Arduino Pro Mini

### Processore:

ATmega328P – 8-bit AVR RISC.

Frequenza: 16 MHz.

### Memoria:

- Flash: 32 kB
- SRAM: 2 kB
- EEPROM: 1 kB



### Alimentazione:

- RAW input: 5 V – 16 V (consigliato 6 V – 12 V).
- VCC: 5 V regolata.
- Corrente massima: 150 mA.

### GPIO:

- 14 pin digitali (di cui 6 con PWM).
- 8 ingressi analogici (A0–A7, 10-bit ADC).
- Pin dedicati: TX, RX, SCL (A5), SDA (A4).

### Interfacce:

- UART (seriale hardware).
- SPI (su pin 10–13).
- I<sup>2</sup>C/TWI (A4/SDA, A5/SCL).
- Interruzioni esterne: su D2 (INT0), D3 (INT1).  
Tensione I/O: 5 V (3.3 V solo tramite livellatori)
- Correnti massime:
  - Per pin: 40 mA (consigliato ≤ 20 mA)
  - Totale chip: 200 mA

### Programmazione:

Tramite header FTDI (6 pin: GND, VCC, RX, TX, GND, DTR) – richiede adattatore USB-seriale esterno.

### LED integrati:

- Power: rosso.
- Pin 13 (D13): verde.

### Note:

- Non ha connettore USB integrato.
- Reset manuale tramite pulsante.

# DHT11 – Sensore di Temperatura e Umidità

## Tipologia:

Sensore digitale combinato (umidità relativa + temperatura) con uscita seriale single-bus.

## Alimentazione:

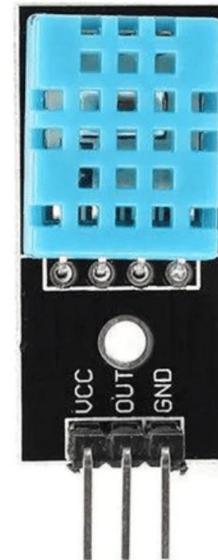
3.3 V – 5.5 V DC.

## Corrente assorbita:

- In misura: 0.3 mA
- In standby: 60 µA

## Interfaccia:

Single-bus digitale (1 filo, open-drain, richiede resistenza di pull-up da 5.1 kΩ).



## Dati trasmessi(40 bit):

- Umidità: 8 bit interi + 8 bit decimali
- Temperatura: 8 bit interi + 8 bit decimali
- Bit di parità: 8 bit

(Nota: la parte decimale è sempre zero nel DHT11)

## Precisione:

- Umidità: ±5% RH @ 25 °C
- Temperatura: ±2 °C @ 25 °C

### Ripetibilità:

- Umidità: ±1% RH
- Temperatura: ±1 °C

### Tempo di risposta:

- Umidità: 6 s (1/e, 1 m/s d'aria)
- Temperatura: 10 s (1/e)

## Frequenza di campionamento:

Minimo intervallo tra letture: **2 secondi** (consigliato ≥5 s per accuratezza)

## Pinout:

- VDD – Alimentazione (3.3–5.5 V)
- DATA – Segnale seriale (single-bus)
- NC – Non connesso
- GND – Massa

## Condizioni operative:

- Temperatura: 0 °C ÷ 50 °C.
- Umidità: 20–90% RH.

**Note:**

- Segnale digitale.
- Non usare in condensa, ambienti aggressivi (fumi acidi, SO<sub>2</sub>, HCl)
- o con esposizione prolungata a UV.
- Per cavi >20 m: ridurre il valore della resistenza di pull-up.
- Con alimentazione a 3.3 V: lunghezza cavo <100 cm.

## Regolatore MCP1700-3302E

**Tipologia:**

Regolatore lineare LDO (Low Dropout) CMOS.

**Tensione di ingresso:** 2.3 V – 6.0 V.

**Tensione di uscita fissa disponibile:**

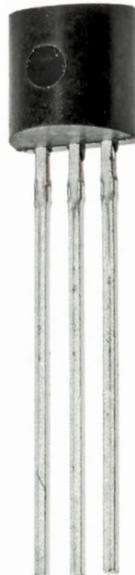
1.2 V, 1.8 V, 2.5 V, 2.8 V, 2.9 V, 3.0 V, 3.3 V, 5.0 V

**Corrente di uscita massima:**

- 250 mA (per V<sub>OUT</sub> ≥ 2.5 V)
- 200 mA (per V<sub>OUT</sub> < 2.5 V)

**Corrente di riposo (quiescent):**

1.6 µA (tipica)

**Dropout voltage:**

- 178 mV @ 250 mA (V<sub>OUT</sub> = 2.8 V)
- 150 mV @ 200 mA (V<sub>OUT</sub> < 2.5 V)

**Precisione tensione di uscita:**

±0.4% @ 25 °C (tipica), ±3% su tutta la gamma di temperatura

**Stabilità:**

Garantita con condensatore di uscita da **1 µF** (ceramico, tantalio o elettrolitico)

**Protezioni integrate:**

- Cortocircuito
- Sovratemperatura (intervento a 140 °C, ripristino a 130 °C)
- Rumore in uscita: 3 µV/V/Hz @ 1 kHz
- Reiezione ripple (PSRR): 44 dB @ 100 Hz
- Package disponibili: SOT-23, SOT-89, TO-92, DFN-6 (2×2 mm)
- Temperatura di esercizio: -40 °C - +125 °C

## Pacco batterie 5xAA

alcaline 1800 - 2800 mAh

5 in serie 8 V 9000 mAh - 14000 mAh

# Software

Questa sezione ha lo scopo di illustrare a livello generale il funzionamento del codice e la gestione/formattazione dei dati, per informazioni in dettaglio sull'argomento si rimanda alla repo GIT <https://github.com/Simv135/Rivellino-Smart-Sensors>

## Software LoRa

### Spiegazione codice

Il software dedicato al modulo LoRa gestisce la comunicazione wireless tra i tre nodi del sistema: Modulo A (tirante), Modulo B (galleria) e Modulo C (polveriera). Tutti e tre i moduli integrano il transceiver RYLR498, configurato tramite comandi AT e interfacciato al microcontrollore locale via UART.

Il funzionamento si basa su una topologia a stella: i moduli B e C operano esclusivamente in trasmissione, inviando periodicamente i dati raccolti dai propri sensori; il modulo A, invece, agisce da nodo centrale, ricevendo i messaggi LoRa dai nodi remoti e inoltrandoli al modulo di elaborazione (Raspberry Pi) attraverso una connessione seriale diretta.

### Configurazione iniziale

All'avvio, ciascun nodo esegue una fase di inizializzazione del modulo RYLR498, inviando una sequenza fissa di comandi AT per impostare:

- l'indirizzo univoco del nodo (1 per il modulo A, 2 per il modulo B, 3 per il modulo C);
- l'identificativo di rete (NETWORKID), identico per tutti i nodi per garantire l'interoperabilità;
- la potenza di trasmissione RF, limitata a 14 dBm per rispettare la normativa CE;
- la modalità operativa (MODE=0 per funzionamento normale con ricezione e trasmissione contemporanee).

Tali impostazioni vengono memorizzate nella flash interna del RYLR498 e non richiedono quindi di essere ripetute ad ogni riavvio, a meno che il modulo non venga resettato o riconfigurato manualmente.

## Formato dei dati trasmessi

I dati vengono inviati in formato testuale ASCII, strutturato come una stringa delimitata da virgole, nel seguente ordine:

- temperatura in gradi Celsius;
- umidità relativa in percentuale;
- valore relativo di CO<sub>2</sub>
- valore normalizzato del sensore gas (MiCS-6814);
- stato del sensore di presenza acqua (0 = assente, 1 = presente).

Un esempio di pacchetto valido è il seguente:

- moduloA: **c28f58j0.06k15.511.9i0.07**
- moduloB: **a61d27g56m1**
- moduloC: **b62e28h42**

## Gestione della ricezione (modulo A)

Il modulo A è costantemente in ascolto sulla porta seriale del RYLR498. Alla ricezione di un nuovo pacchetto:

- verifica che l'identificativo del mittente corrisponda a uno dei nodi remoti (2 o 3);
- controlla l'integrità del messaggio (se presente, viene validato un checksum semplice basato su XOR);
- aggiunge eventuali informazioni mancanti, come un timestamp locale in caso di dati non sincronizzati;
- inoltra il messaggio così composto al Raspberry Pi mediante la seriale hardware (pin TX dell'Arduino Nano 33 BLE).

## Gestione degli errori

Il software include diverse misure per garantire l'affidabilità della comunicazione:

- timeout nella risposta ai comandi AT durante la fase di inizializzazione;
- tentativi ripetuti di invio in caso di mancata conferma (solo nei moduli B e C);
- reset hardware del modulo RYLR498 tramite il pin NRST, pilotato da un GPIO del microcontrollore, in caso di malfunzionamenti persistenti;
- log minimo su seriale di debug, disattivabile in produzione per ridurre il consumo energetico.

Nei moduli alimentati a batteria (B e C), l'alimentazione del modulo LoRa è gestita dinamicamente: il trasceiver viene alimentato solo durante la

fase di trasmissione, grazie a un regolatore di tensione abilitato dal microcontrollore.

## Software Modulo A

### Spiegazione codice

Il software caricato sul **Modulo A** (Arduino Nano 33 BLE, installato in prossimità del tirante) ha un ruolo centrale all'interno dell'intero sistema. Esso è infatti responsabile di tre funzioni principali:

- la lettura periodica dei sensori ambientali integrati nell'unità;
- la ricezione e il filtraggio dei dati provenienti in modalità wireless dai **Moduli B e C** tramite il transceiver LoRa;
- l'inoltro di tutti i dati, sia locali che remoti, al **modulo di elaborazione** (Raspberry Pi 4) attraverso una connessione seriale diretta.

Il file principale è **Modulo\_A.ino**, che include la libreria personalizzata **LoRa\_RYLR498.h** (già descritta nella sezione precedente) e sfrutta le librerie ufficiali Arduino per la gestione dei sensori integrati (BMI270, BMM150, LPS22HB, HS3003, APDS-9960 e MiCS-6814).

## Ciclo operativo

All'avvio, il microcontrollore esegue le seguenti fasi in sequenza:

- inizializzazione della seriale di debug (opzionale, disattivabile);
- configurazione e inizializzazione di tutti i sensori onboard;
- inizializzazione e configurazione del modulo LoRa (RYLR498) in modalità ricezione;
- avvio del timer principale, impostato su un intervallo di campionamento di **60 secondi**.

Ad ogni ciclo, il software:

- legge i valori dai sensori ambientali (temperatura, umidità, pressione, gas, luce, movimento, campo magnetico);
- formatta i dati locali in una stringa conforme allo schema definito (vedi sezione Software LoRa);
- verifica se è stato ricevuto un nuovo pacchetto LoRa dai nodi remoti;
- in caso affermativo, applica un controllo di validità e lo accoda alla coda di trasmissione;
- invia al Raspberry Pi, in sequenza, prima il proprio pacchetto locale, quindi eventuali pacchetti remoti ricevuti nel ciclo corrente.

La trasmissione seriale avviene a **9600 baud**, velocità sufficiente a garantire un trasferimento rapido e affidabile senza sovraccaricare il bus.

## Gestione dei sensori

Il Modulo A integra un set completo di sensori ambientali, ciascuno gestito tramite una specifica libreria:

- **BMI270 + BMM150**: IMU a 9 assi per rilevamento di movimento, vibrazioni o spostamenti anomali della struttura;
- **LPS22HB**: sensore barometrico ad alta precisione, utilizzato anche per la misura della temperatura ambiente;
- **HS3003**: sensore di umidità con accuratezza  $\pm 1.5\%$  RH;
- **APDS-9960**: sensore di prossimità, luce ambientale e gesti – impiegato principalmente per monitorare variazioni di illuminazione o presenza di oggetti vicini;
- **MiCS-6814 (Air quality 5 Click)**: sensore MOS per il rilevamento di gas riducenti (CO, NO<sub>2</sub>).

Tutti i valori grezzi vengono convertiti in unità fisiche significative prima della trasmissione. In particolare, il sensore MiCS-6814 richiede una fase di calibrazione iniziale (effettuata in laboratorio), dopo la quale i valori vengono restituiti come indici normalizzati, privi di unità ma proporzionali alla concentrazione del gas rilevato.

## Comunicazione con il Raspberry

La connessione seriale tra Arduino Nano 33 BLE e Raspberry Pi è di tipo seriale **UART**, con i pin TX (D0) e RX (D1) del microcontrollore direttamente collegati ai pin corrispondenti del Raspberryi (GPIO 15 e 14).

Il protocollo di comunicazione è testuale: ogni riga terminata da \n rappresenta un pacchetto completo.

Non sono previsti frame complessi né meccanismi di riscontro, in quanto il Raspberry Pi è in grado di ricevere e processare i dati in modo asincrono e tollera eventuali perdite occasionali.

In caso di riavvio del Raspberry Pi o di interruzione temporanea della seriale, il modulo A non effettua buffering dei dati: ciò è una scelta progettuale volta a ridurre al minimo l'uso della memoria e il consumo energetico. Eventuali lacune nei dati vengono gestite a livello di software di elaborazione.

## Ottimizzazione energetica

Sebbene il Modulo A sia alimentato dalla rete elettrica, il codice include comunque alcune ottimizzazioni per ridurre il carico termico e prolungare la vita utile dei componenti:

- i sensori non utilizzati in un determinato ciclo vengono messi in modalità sleep;
- il microcontrollore non entra in deep sleep, ma disattiva temporaneamente i sottosistemi non essenziali tra un ciclo e l'altro.

## Note

Tutti i parametri (frequenza di campionamento, indirizzo LoRa, chiavi di crittografia, ecc.) sono definiti in fase di compilazione tramite `#define`.

Eventuali aggiornamenti firmware possono essere caricati tramite la porta USB Micro-B dell'Arduino Nano 33 BLE.

## Software Modulo B

### Spiegazione codice

Il software del **Modulo B** (installato nella galleria) è caricato su un **Arduino Pro Mini** ed è progettato per operare in autonomia, alimentato esclusivamente da un pacco batteria da 5xAA. La sua unica funzione è raccogliere periodicamente i dati dai sensori locali e trasmetterli in modalità wireless al **Modulo A** attraverso il transceiver **LoRa RYLR498**.

Il file principale è **Modulo\_B.ino**, che include la libreria **LoRa\_RYLR498.h** per la gestione del modulo radio e utilizza una semplice routine per la lettura del sensore **DHT11** e del **sensore resistivo di presenza acqua (HW-038)**.

## Ciclo operativo

All'accensione, il microcontrollore esegue una breve sequenza di inizializzazione:

- avvio della seriale di debug (solo per fase di sviluppo, disattivata nella versione definitiva);
- configurazione del pin analogico per la lettura del sensore di acqua;
- inizializzazione del sensore DHT11;
- configurazione del modulo LoRa con indirizzo fisso (ID = 2) e parametri di rete predefiniti.

Terminata l'inizializzazione, il modulo entra in un ciclo infinito strutturato come segue:

- lettura della temperatura e dell'umidità dal DHT11;
- lettura del valore analogico dal sensore HW-038, convertito in uno stato binario (0 = asciutto, 1 = presenza di acqua);
- assemblaggio del pacchetto dati secondo lo schema standard (vedi sezione Software LoRa);
- alimentazione del modulo LoRa (fino a quel momento spento per risparmiare energia);
- invio del pacchetto tramite UART al RYLR498;
- attesa della conferma di trasmissione;
- spegnimento del modulo LoRa;
- messa in **deep sleep** del microcontrollore per **600 secondi** (10 minuti).

Questa strategia di “wake-up -> misura -> trasmetti -> sleep” permette di ridurre il consumo medio a pochi microampere, consentendo un'autonomia stimata di oltre **6 mesi** con batterie alcaline da 5xAA.

## Gestione dei sensori

Il Modulo B integra due soli sensori:

- **DHT11**: fornisce temperatura ( $\pm 2^{\circ}\text{C}$ ) e umidità ( $\pm 5\% \text{ RH}$ ) con una frequenza massima di 1 lettura ogni 2 secondi. Nel codice, l'intervallo è impostato a 1 lettura per ciclo (ogni 10 minuti), ben al di sotto del limite, per garantire stabilità e accuratezza.
- **HW-038**: sensore resistivo a tracce esposte. Il valore analogico letto è confrontato con una soglia fissa (circa 300 su scala 0–1023), al di sopra della quale si considera presente acqua. L'uscita è quindi binarizzata per semplificare il formato del messaggio.

Non è richiesta alcuna calibrazione in campo. I dati vengono trasmessi così come letti, con la sola conversione necessaria per il formato comune.

## Ottimizzazione energetica

L'intero sistema è ottimizzato per il risparmio energetico:

- il **modulo LoRa** è alimentato tramite un regolatore di tensione controllato da un pin digitale (D7). Viene acceso solo per il tempo strettamente necessario all'invio del pacchetto (circa 500 ms).
- l'**Arduino Pro Mini** entra in modalità **power-down** tramite la libreria **LowPower.h**, che sfrutta il timer watchdog per il risveglio periodico.
- il **sensore DHT11** e il **HW-038** non richiedono alimentazione continua; vengono letti solo durante la fase attiva e non dissipano corrente in sleep.

Questa architettura permette al modulo di consumare, in media, meno di **100 µA**, con picchi inferiori a 30 mA durante la trasmissione.

## Comportamento in caso di errore

Il software include una gestione minima ma efficace degli errori:

- se la lettura del DHT11 fallisce (errore di checksum o timeout), il sistema riprova una volta prima di inviare un pacchetto con valori di default (**temp=-99, hum=-99**);
- se il modulo LoRa non risponde ai comandi AT, il microcontrollore esegue un reset hardware tramite il pin NRST e tenta nuovamente l'invio;
- dopo tre tentativi falliti consecutivi, il sistema comunque va in sleep, per non compromettere l'autonomia.

Non è previsto alcun sistema di logging persistente, né di notifica all'utente: l'assenza di dati sul sistema centrale è l'unico indicatore di malfunzionamento, da verificare in fase di manutenzione.

## Note

Gli aggiornamenti firmware richiedono la rimozione del modulo e la programmazione tramite adattatore FTDI, collegato all'header seriale (6 pin: GND, VCC, RX, TX, GND, DTR). Non è prevista alcuna forma di aggiornamento over-the-air

# Software Modulo C

## Spiegazione codice

Il software del **Modulo C** (installato nella polveriera) è strutturalmente identico a quello del **Modulo B**, in quanto entrambi sono basati sullo stesso hardware: **Arduino Pro Mini**, **sensore DHT11**, **regolatore MCP1700-3302E**, **pacco batteria da 5xAA** e **modulo LoRa RYLR498**. Anche la logica operativa è la medesima: il modulo si risveglia periodicamente, acquisisce i dati ambientali, trasmette un pacchetto via LoRa al **Modulo A** e torna in stato di riposo per conservare l'energia.

Il file sorgente principale è **Modulo\_C.ino**. Pur essendo quasi identico a **Modulo\_B.ino**, differisce in due aspetti fondamentali:

- l'**indirizzo LoRa** è impostato a **3** (invece che **2**), per identificare univocamente il nodo;
- il **sensore di presenza acqua (HW-038)** non è presente, pertanto il relativo campo nel pacchetto dati viene sostituito con il valore fisso **0**.

## Ciclo operativo

Il ciclo del Modulo C segue la stessa logica del Modulo B:

- al risveglio (ogni **600 secondi**, cioè 10 minuti), il microcontrollore esegue l'inizializzazione dei sensori e del modulo LoRa;
- legge temperatura e umidità dal **DHT11**;
- compone il pacchetto dati nel formato standard:  
**3 ,<timestamp>, <temp>, <hum>, 0, 0, 0**  
(i campi per CO2, gas e acqua sono impostati a 0, in quanto i corrispondenti sensori non sono installati);
- alimenta il modulo LoRa, invia il messaggio e attende la conferma;
- spegne il transceiver;
- torna in **deep sleep** tramite la libreria **LowPower.h**.

L'intero processo attivo dura meno di un secondo, dopodiché il consumo scende a livelli trascurabili.

## Gestione dei sensori

L'unico sensore attivo nel Modulo C è il **DHT11**.

La lettura avviene con la stessa routine del Modulo B: viene richiesto un campione, ne viene verificata l'integrità e, in caso di fallimento, viene effettuato un singolo tentativo di ripetizione.

Se anche questo fallisce, il pacchetto viene comunque inviato con valori marcati come non validi (-99), per consentire al sistema centrale di rilevare un'anomalia.

Non essendo presente alcun sensore aggiuntivo, non sono richiesti ADC esterni, librerie aggiuntive o calibrazioni in campo.

## Ottimizzazione energetica

- il **modulo LoRa** è alimentato a richiesta tramite un MOSFET comandato dal pin D7;
- il **microcontrollore** trascorre oltre il 99,9% del tempo in modalità **power-down**;
- il **DHT11**, pur non avendo un pin di alimentazione dedicato, non dissipava corrente significativa quando non interrogato;
- il **regolatore MCP1700** ha una corrente di riposo di soli **1,6 µA**, compatibile con un utilizzo a batteria prolungato.

Con queste caratteristiche, l'autonomia stimata del Modulo C supera i **7 mesi** con batterie alcaline nuove, grazie anche all'assenza del sensore di acqua, che riduce leggermente i consumi di picco.

## Gestione degli errori

La gestione degli errori riprende quella del Modulo B:

- fallimento nella lettura del DHT11 -> tentativo di ripetizione, poi invio con valori invalidi;
- mancata risposta del RYLR498 -> reset hardware via pin NRST e nuovo tentativo;
- dopo tre fallimenti consecutivi -> comunque sleep, per non compromettere l'autonomia.

Non è prevista alcuna memoria non volatile per il logging locale, né feedback visivo o acustico: la diagnosi avviene esclusivamente a livello centrale, tramite l'analisi dei dati ricevuti (o non ricevuti) dal Raspberry.

## Note

Come per il Modulo B, gli aggiornamenti firmware richiedono l'intervento fisico sull'unità e la programmazione tramite adattatore FTDI. Il codice, per sua natura, non prevede configurazioni runtime né interazione con l'utente.

## Linee generali di funzionamento del sistema

L'intero sistema è progettato per operare in modo autonomo, continuo e senza necessità di intervento umano in campo, eccetto per la manutenzione periodica delle batterie nei moduli remoti.

Il flusso dati si articola in quattro fasi distinte: **raccolta, trasmissione, elaborazione ed esportazione**.

## Raccolta

La raccolta avviene in modo decentralizzato, tramite i tre moduli dislocati nei punti strategici del sito:

- il **Modulo A (tirante)** acquisisce dati da un set completo di sensori ambientali (temperatura, umidità, pressione, gas, luce, movimento, campo magnetico);
- il **Modulo B (galleria)** rileva temperatura, umidità e presenza di acqua;
- il **Modulo C (polveriera)** monitora esclusivamente temperatura e umidità.

Tutti i moduli operano in modo indipendente. I moduli B e C, alimentati a batteria, si risvegliano ogni 10 minuti per eseguire una misura e trasmetterla. Il modulo A, alimentato dalla rete elettrica, opera in continuo e, oltre a raccogliere i propri dati, funge da gateway per i nodi remoti.

## Trasmissione

La trasmissione dei dati avviene su due livelli:

1. **Wireless (LoRa)**: i moduli B e C inviano i propri pacchetti al modulo A tramite il transceiver RYLR498, operante nella banda 490 MHz con modulazione LoRa. La comunicazione è unidirezionale, crittografata (AES-128) e configurata per massimizzare l'affidabilità entro i limiti normativi CE (potenza RF  $\leq$  14 dBm).
2. **Seriale cablata**: il modulo A inoltre, in tempo reale, tutti i dati — sia quelli raccolti localmente che quelli ricevuti via LoRa — al **modulo di elaborazione** (Raspberry Pi 4) attraverso una connessione UART hardware a 115200 baud. Questa connessione è fisicamente stabile, immune a interferenze e non soggetta a latenze imprevedibili.

Non è previsto alcun meccanismo di riscontro a livello applicativo: il sistema si basa sull'assunzione che la trasmissione seriale sia affidabile e che eventuali perdite LoRa possano essere tollerate, data la periodicità delle misure.

## Elaborazione

L'elaborazione è centralizzata sul **Raspberry Pi 4**, che esegue uno script Python dedicato (`data_handler.py`), avviato automaticamente all'avvio del sistema tramite `systemd`.

Lo script svolge le seguenti funzioni:

- apre la porta seriale `/dev/ttyS0` e legge i dati in arrivo riga per riga;
- verifica il formato del messaggio e la validità dell'identificativo del modulo;
- associa ogni pacchetto a un file CSV dedicato, basato sull'ID del nodo (es. `modulo_1.csv`, `modulo_2.csv`, `modulo_3.csv`);
- aggiunge un timestamp locale nel caso il pacchetto non ne contenga uno valido;
- salva i dati in modo atomico, per evitare corruzione in caso di interruzione di alimentazione;
- mantiene in memoria un buffer di ultimi N valori per eventuali estensioni future (es. allarmi in tempo reale).

I file CSV sono organizzati in una directory dedicata (`/home/pi/rivellino_data/`) e seguono una struttura uniforme, con intestazione fissa e valori separati da virgole. Il sistema supporta riavvii imprevisti, lo script riparte senza perdere lo stato e continua a scrivere nei file esistenti.

## Esportazione

L'esportazione dei dati avviene in due modalità:

1. **Locale**: i file CSV sono accessibili direttamente sul Raspberry Pi 4 tramite rete LAN o Wi-Fi (SSH, Samba, o interfaccia web locale). Questo permette agli operatori di recuperare i dati in modo semplice durante le visite tecniche.
2. **Remota (opzionale)**: il sistema include uno script secondario ([upload\\_to\\_cloud.py](#)), non attivo di default, che può essere configurato per caricare periodicamente i dati su un server remoto (FTP, S3, o API REST). L'attivazione di questa funzione richiede la configurazione manuale delle credenziali e della destinazione, per ragioni di sicurezza.

Non è prevista alcuna visualizzazione grafica integrata nel sistema base.

## Sincronizzazione temporale

Il sistema non utilizza un orologio in tempo reale (RTC) né la sincronizzazione NTP nei moduli remoti, per motivi di costo e consumo. Di conseguenza:

- i moduli B e C inviano i dati **senza timestamp**;
- il modulo A inserisce un timestamp basato sull'ora del Raspberry Pi 4 (sincronizzata via NTP all'avvio);
- il Raspberry Pi 4, al ricevimento di un pacchetto senza timestamp, utilizza l'ora locale come riferimento.

Questa scelta introduce una imprecisione temporale (dell'ordine del secondo), ma è ritenuta accettabile per il tipo di monitoraggio.

## Tolleranza a imprevisti

Il sistema è progettato per essere tollerante a guasti parziali:

- la perdita di un nodo remoto non compromette il funzionamento degli altri;
- un'interruzione della rete elettrica ferma temporaneamente il modulo A e il Raspberry Pi, ma i moduli B e C continuano a operare (anche se i dati non vengono ricevuti finché il gateway non torna attivo);
- in caso di corruzione di un file CSV, lo script ne crea automaticamente uno nuovo con suffisso incrementale (es. [modulo\\_2\\_20251204.csv](#)).

La manutenzione periodica (sostituzione batterie, verifica connessioni, backup dati) è l'unico requisito per garantire il funzionamento a lungo termine.

# Glossario

**ADC** – Convertitore analogico-digitale.

**BLE** – Bluetooth Low Energy. Tecnologia wireless a corto raggio e basso consumo.

**CSV** – Formato testuale semplice per organizzare dati in righe e colonne, separati da virgole. È il formato scelto per l'archiviazione dei dati sul modulo di elaborazione

**Deep sleep** – Stato di minimo consumo in cui il microcontrollore disattiva quasi tutte le sue funzioni, mantenendo solo il timer necessario per il risveglio. Utilizzato nei moduli B e C per prolungare la durata delle batterie.

**DHT11** – Sensore digitale per la misura di temperatura e umidità relativa. Presente nei moduli B e C.

**GPIO** – Pin general-purpose di un microcontrollore o SoC.

**HW-038** – Sensore resistivo per la rilevazione di presenza d'acqua, basato sulla conducibilità tra tracce esposte. Installato solo nel modulo B (galleria).

**IMU** – Inertial Measurement Unit. Unità che combina accelerometro, giroscopio e, talvolta, magnetometro per rilevare movimento, orientamento e vibrazioni. L'Arduino Nano 33 BLE integra un'IMU a 9 assi (BMI270 + BMM150).

**LoRa** – Tecnologia di trasmissione radio a lungo raggio e basso consumo, impiegata tramite il modulo RYLR498 per la comunicazione tra i nodi.

**LDO** – Low Dropout Regulator. Tipo di regolatore di tensione lineare che mantiene un'uscita stabile anche con poca differenza tra ingresso e uscita.; di cui il MCP1700 ne è un esempio, usato nei moduli B e C.

**MOS** – Metal-Oxide-Semiconductor. Tipo di sensore chimico la cui resistenza varia in presenza di gas specifici. Il MiCS-6814, presente nel modulo A, ne integra tre elementi indipendenti.

**NRST** – Pin di reset attivo basso presente su molti chip, incluso il RYLR498. Nel sistema è collegato a un GPIO per consentire il reset hardware del modulo LoRa in caso di blocco.

**Power-down** – Modalità di risparmio energetico dell'ATmega328P (Arduino Pro Mini), equivalente al deep sleep, attivata tramite libreria LowPower.h.

**RYLR498** – Modulo transceiver LoRa operante a 490 MHz, utilizzato in tutti e tre i nodi per la comunicazione wireless. Configurabile tramite comandi AT.

**UART** – Interfaccia seriale asincrona, usata in questo sistema per la comunicazione tra Arduino e RYLR498, e tra Arduino Nano 33 BLE e Raspberry Pi.

**Wake-up** – Fase di riattivazione del microcontrollore dopo un periodo di sleep. Nei moduli B e C avviene ogni 600 secondi, grazie al timer watchdog.

**Watchdog** – Timer hardware che, se non resettato in tempo, provoca un reset del microcontrollore. Nel sistema è usato anche come base per il risveglio periodico dai modi di sleep.