

Relazione Assignment 03

Embedded Systems and IoT

Smart Tank Monitoring System

Lucia Pola

Matricola: 0001118424

Leonardo Meloni

Matricola: 0001128110

Università di Bologna

Corso di Laurea in Ingegneria e Scienze Informatiche

A.A 2025/2026

14 febbraio 2026

Indice

1 Descrizione del Sistema	2
1.1 Componenti	2
1.2 Componenti Hardware	2
1.3 Descrizione del comportamento	3
1.3.1 Dettagli	3
1.4 L'Assignment	4
2 Progettazione dei sottosistemi	6
2.1 TMS - Tank Monitoring Subsystem	6
2.1.1 Hardware	6
2.1.2 Software	6
2.2 CUS - Control Unit Subsystem	8
2.2.1 Logica Funzionale	8
2.3 WCS - Water Channel Subsystem	9
2.3.1 Hardware	9
2.3.2 Software	10
2.4 DBS - Dashboard Subsystem	12
3 Protocolli di Comunicazione	14
3.1 Interfaccia Seriale (CUS - WCS)	14
3.1.1 Direzione CUS → WCS (Comandi)	14
3.1.2 Direzione WCS → CUS (Telemetria)	14
3.2 Interfaccia HTTP (CUS - DBS)	15
3.2.1 Endpoint API	15
3.3 Interfaccia MQTT (CUS - TMS)	15
4 Configurazioni	16
4.1 Mappa dei Collegamenti (Pinout)	16
4.1.1 TMS - Tank Monitoring Subsystem (ESP32)	16
4.1.2 WCS - Water Channel Subsystem (Arduino UNO)	16
4.2 Configurazione degli Scheduler	16
4.3 Parametri Funzionali	17

1 Descrizione del Sistema

1.1 Componenti

Il sistema è composto da quattro sottosistemi:

- **Tank Monitoring subsystem (TMS)** - basato su ESP
 - Sistema embedded per monitorare il livello dell'acqua piovana nel serbatoio.
 - Interagisce con il sottosistema Control Unit (CUS) tramite MQTT.
- **Control Unit subsystem (CUS)** - Back-end, in esecuzione su PC
 - Sottosistema principale, governa e coordina l'intero sistema.
 - Interagisce via MQTT con il Tank Monitoring Subsystem (TMS).
 - Interagisce attraverso la linea seriale con il Water Channels Subsystem (WCS).
 - Interagisce via HTTP con il Dashboard Subsystem (DBS).
- **Water Channel Subsystem (WCS)** - basato su Arduino
 - Sistema embedded che controlla il canale idrico che collega il serbatoio con la rete di canali.
 - Interagisce via linea seriale con il Control Unit Subsystem (CUS).
 - Fornisce un pannello per l'interazione in loco degli operatori umani.
- **Dashboard subsystem (DBS)** - Frontend/web app, su PC o device
 - Front-end per operatori remoti per visualizzare i dati e interagire con il sistema.
 - Interagisce via HTTP con il Control Unit Subsystem.

1.2 Componenti Hardware

• TMS

- Scheda SoC ESP32 (o ESP8266) che include:
 - 1 sonar
 - 1 led verde
 - 1 led rosso

• WCS

- Scheda Microcontrollore Arduino UNO che include:
 - 1 servomotore
 - 1 potenziometro
 - 1 pulsante tattile
 - 1 display LCD

1.3 Descrizione del comportamento

Il sistema è inteso per monitorare il livello dell'acqua piovana all'interno del serbatoio e - a seconda dei valori - controllare l'apertura di un canale idrico che collega il serbatoio a una rete di canali. L'intero sistema può trovarsi in due modalità diverse: **AUTOMATIC** o **MANUAL**.

In modalità **AUTOMATIC**, il sistema controlla automaticamente l'apertura del canale idrico, dipendendo dal livello attuale di acqua piovana nel serbatoio. In modalità **MANUAL**, l'apertura è controllata manualmente da un operatore. La modalità iniziale all'avvio è **AUTOMATIC**.

1.3.1 Dettagli

- Il **TMS** è responsabile del monitoraggio continuo del livello dell'acqua piovana, per mezzo di sensori adeguati (es. un sonar).
 - Il livello dell'acqua piovana viene campionato a frequenza F e inviato al sottosistema **CUS**.
 - Quando il sistema funziona correttamente (rete ok, invio dati ok) il led verde è acceso e il rosso è spento; altrimenti – nel caso di problemi di rete – il led rosso dovrebbe essere acceso e il verde spento.
- Il sottosistema **WCS** è responsabile del controllo di una valvola (con un motore) che apre/chiude un canale idrico drenando l'acqua dal serbatoio alla rete.
 - Il range di apertura è in percentuale: da 0% = canale chiuso (posizione motore 0 gradi), fino a 100% = canale completamente aperto (posizione motore 90 gradi).
 - Il livello di apertura del canale dipende dallo stato del sistema, stabilito dal sottosistema **CUS**.
 - Il **WCS** include un pulsante per abilitare la modalità **MANUAL**:
 - * Quando il pulsante viene premuto, il sistema entra in modalità **MANUAL**, così che il livello di apertura possa essere controllato manualmente dagli operatori usando un potenziometro.
 - * Per uscire dalla modalità **MANUAL**, il pulsante deve essere premuto nuovamente.
 - Il **WCS** è equipaggiato anche con un display LCD che riporta:
 - * L'attuale livello di apertura della valvola.
 - * La modalità corrente (**AUTOMATIC** o **MANUAL**), o **UNCONNECTED**.
- Il sottosistema **CUS** è incaricato della politica che governa il comportamento del sistema. In particolare:
 - **Monitoraggio del livello dell'acqua piovana**:
 - * Quando il livello dell'acqua supera il livello L_1 (ma sotto L_2 , con $L_1 < L_2$) per più di un tempo T_1 , il canale idrico viene aperto al 50% finché il livello è sotto L_1 .

- * Se il livello supera il livello L_2 , il canale viene immediatamente aperto al 100%, finché il valore è sotto L_2 .
- Se il **CUS** non riceve dati per più di T_2 unità di tempo dal **TMS** (a causa di problemi di rete), allora il sistema entra in uno stato **UNCONNECTED**, che viene ripristinato a uno stato normale solo se/quando i problemi di rete vengono risolti.
- Il sottosistema **DBS** è una dashboard per visualizzare lo stato del sistema, includendo:
 - Un grafico del livello dell'acqua piovana, considerando le ultime N misurazioni.
 - Il valore attuale della percentuale di apertura della valvola.
 - Lo stato del sistema: **MANUAL**, **AUTOMATIC**, **UNCONNECTED** o **NOT AVAILABLE**.
 - Lo stato è etichettato come **NOT AVAILABLE** quando il **CUS** non è raggiungibile.

Inoltre, include:

- Un pulsante GUI per cambiare la modalità (**MANUAL**, **AUTOMATIC**).
- Un widget GUI per controllare il livello di apertura della valvola nel **WCS** (quando in modalità **MANUAL**).

1.4 L'Assignment

Progettare e sviluppare un prototipo del sistema di Monitoraggio del Serbatoio, considerando come ulteriori requisiti:

- **Sottosistema TMS:**
 - Deve girare su un ESP32 (o una scheda SoC equivalente).
 - La logica di controllo deve essere progettata e implementata usando macchine a stati finiti (sincrone o asincrone) e (possibilmente, se utile) architetture basate su task, sfruttando l'RTOS.
 - Deve usare MQTT per comunicare con il sottosistema **CUS**.
- **Sottosistema WCS:**
 - Deve girare su un Arduino (o una scheda MCU equivalente).
 - La logica di controllo deve essere progettata e implementata usando macchine a stati finiti (sincrone o asincrone) e (possibilmente, se utile) architetture basate su task.
 - Deve comunicare con il sottosistema **CUS** via linea seriale.
- **Sottosistema CUS:**
 - Deve girare su un PC, come server/back-end.
 - Nessun vincolo specifico riguardo la tecnologia di programmazione/sw da usare.

- Deve usare:
 - * MQTT per comunicare con il sottosistema **TMS**.
 - * HTTP per comunicare con il sottosistema **DBS**.
 - * La linea seriale per interagire con il **WCS**.

- **Sottosistema DBS:**

- Deve girare su un PC.
- Nessun vincolo specifico riguardo la tecnologia di programmazione/sw da usare.
- Deve usare HTTP per interagire con il sottosistema **CUS**.

Per qualsiasi aspetto non specificato, si è liberi di scegliere l'approccio considerato più utile o valido.

2 Progettazione dei sottosistemi

2.1 TMS - Tank Monitoring Subsystem

2.1.1 Hardware

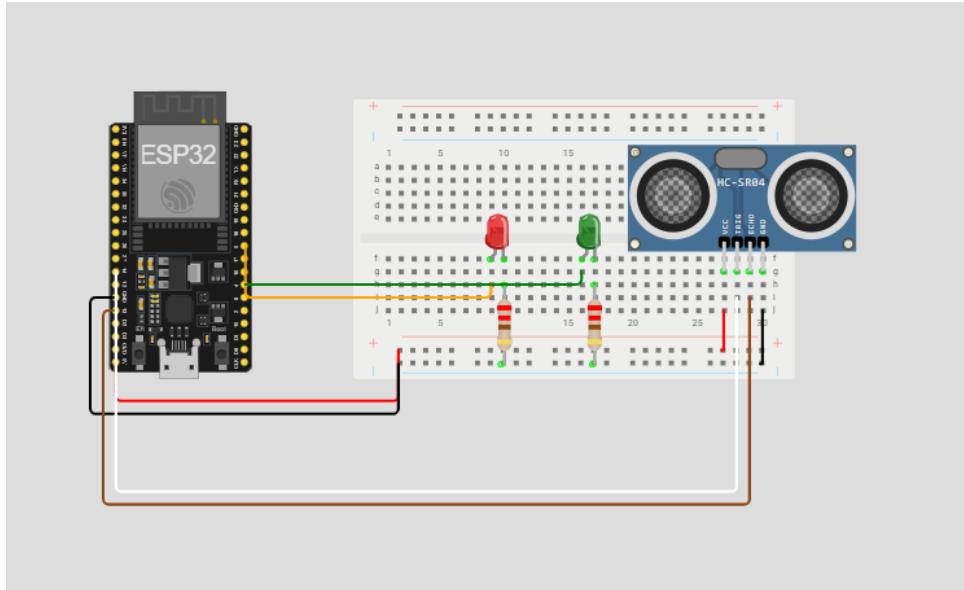


Figura 1: Schema del sistema Smart Tank Monitoring

Per l'implementazione del prototipo sono stati utilizzati i seguenti componenti hardware:

- **Soc:** ESP32.
- **Monitoraggio livello acqua:** Sonar HC-SR04
- **Stato sistema:** LED verde e LED rosso

2.1.2 Software

L'architettura software del **TMS** è basata su un modello di multitasking cooperativo gestito da uno *Scheduler*. La sincronizzazione tra i task avviene tramite un oggetto condiviso **Context**, che mantiene lo stato globale del sistema e funge da memoria condivisa tra i vari task. Ognuno dei task si basa su **una macchina a stati finiti (FSM)**.

I 2 Task Principali

- **ConnectionTask:** Gestisce l'intera logica di rete implementando una FSM a tre stati (**DISCONNECTED**, **CONNECTING_MQTT**, **CONNECTED**). Si occupa di stabilire la connessione WiFi, autenticarsi al broker MQTT e gestire automaticamente le riconnessioni in caso di timeout o errori.
- **StatusTask:** Esegue il campionamento periodico del sensore sonar e la trasmissione dei dati via MQTT. Inoltre, attiva il LED Verde durante il funzionamento del sistema e commuta sul LED Rosso se il sistema è offline o la trasmissione dei dati fallisce.

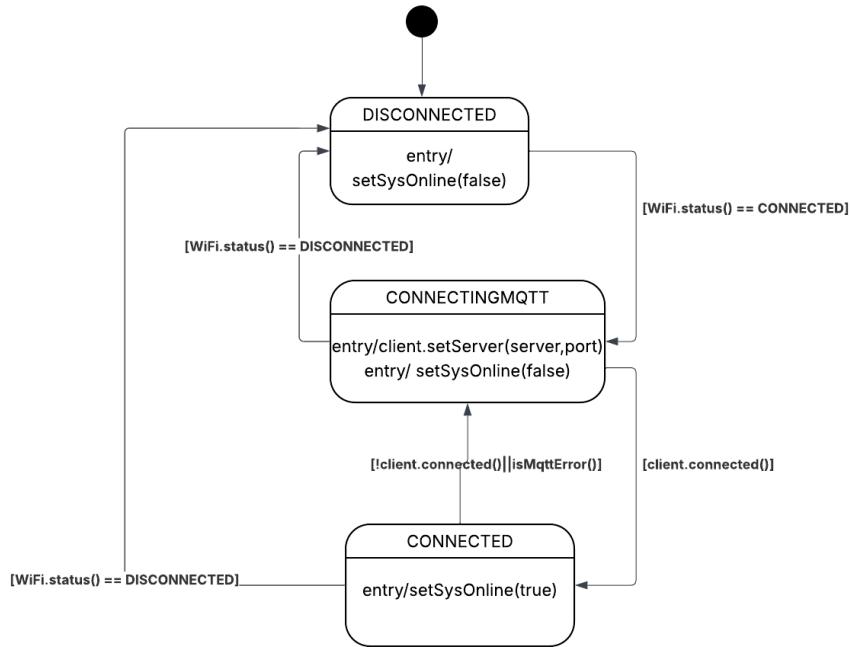


Figura 2: Schema degli stati connectionTask

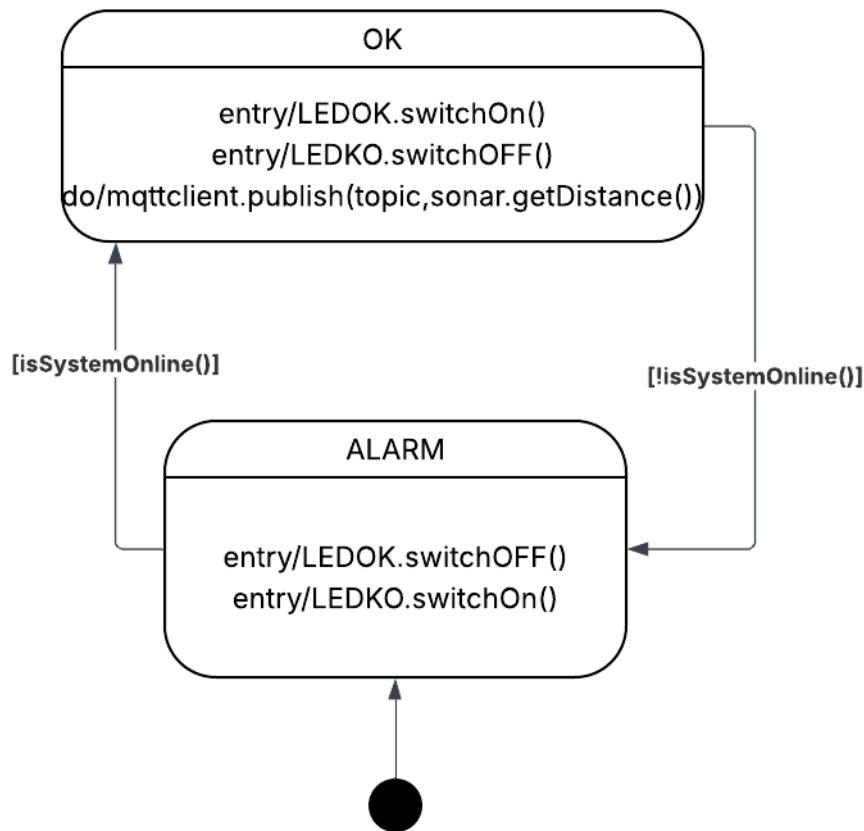


Figura 3: Schema degli stati statusTask

2.2 CUS - Control Unit Subsystem

Il **CUS** rappresenta il nodo centrale dell'architettura IoT, agendo come server di backend in esecuzione su PC. Il suo ruolo principale è quello di coordinare i sottosistemi distribuiti, gestire i dati e applicare la logica di controllo decisionale.

L'architettura software è di tipo *Event-Driven* centralizzata, progettata per gestire tre flussi di comunicazione asincroni concorrenti:

1. **Ingresso Dati (MQTT)**: Ricezione dei dati sul livello dell'acqua dal TMS.
2. **Controllo Attuatori (Seriale)**: Invio comandi di apertura valvola al WCS e ricezione input manuali.
3. **Interfaccia Utente (HTTP)**: Esposizione dello stato del sistema alla Dashboard (DBS).

2.2.1 Logica Funzionale

Il funzionamento del CUS è regolato da un ciclo di controllo principale (Control Loop) che esegue periodicamente le seguenti operazioni logiche:

- **Gestione della Connettività (Heartbeat):** Il sistema monitora costantemente l'arrivo dei messaggi dal TMS. Implementa un meccanismo di *watchdog*: se non vengono ricevuti dati entro una soglia temporale critica ($T_{timeout}$), il sistema transita nello stato di errore UNCONNECTED, segnalando l'anomalia sia alla Dashboard che al WCS per attivare le procedure di sicurezza.
- **Gestione degli Stati Operativi:** Il sottosistema mantiene la macchina a stati finiti globale, gestendo la transizione tra le modalità AUTOMATIC e MANUAL.
 - In modalità **AUTOMATIC**, il CUS elabora i dati del livello dell'acqua applicando le soglie (L_1, L_2) e i vincoli temporali (T_1) definiti nei requisiti per calcolare la percentuale di apertura della valvola ottimale.
 - In modalità **MANUAL**, il CUS agisce come *gateway*, inoltrando i comandi ricevuti dalla Dashboard o dal pannello fisico del WCS direttamente al corrispondente.
- **Servizio API REST:** Un modulo server HTTP integrato risponde alle richieste della Dashboard, fornendo in tempo reale lo stato istantaneo del sistema (livello, stato valvola, modalità) e lo storico delle misurazioni per la visualizzazione grafica.

2.3 WCS - Water Channel Subsystem

2.3.1 Hardware

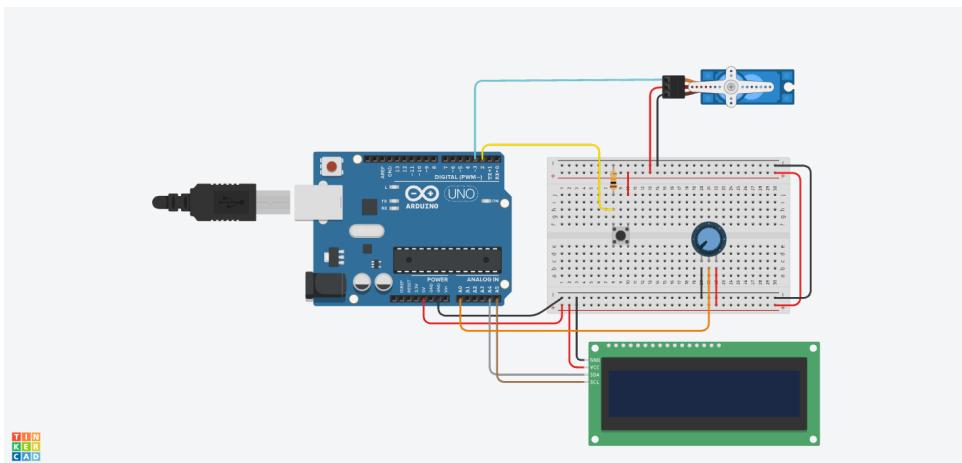


Figura 4: Schema del sottosistema WCS

Per l'implementazione del prototipo sono stati utilizzati i seguenti componenti hardware:

- **Microcontrollore:** Arduino UNO R3.
- **Attuatore Valvola:** Microservo HS-53.
- **Switch Automatico/Mauale:** Bottone tattile.
- **Valvola:** Potenziometro.
- **Interfaccia Utente:** Display LCD 16 × 2 (I2C).

2.3.2 Software

L'architettura software è basata su un modello di multitasking cooperativo gestito da uno *Scheduler*. La sincronizzazione tra i task avviene tramite un oggetto condiviso **Context**, che mantiene lo stato globale del sistema e funge da memoria condivisa tra i vari task. Ognuno dei task si basa su **una macchina a stati finiti (FSM)**.

I 2 task principali

- **SerialMonitorTask:** Gestisce la comunicazione seriale bidirezionale con il CUS.
 - **Ricezione:** Decodifica i messaggi in arrivo per aggiornare lo stato del sistema (cambio modalità AUTOMATIC/MANUAL e setpoint valvola).
 - **Trasmissione:** Invia periodicamente al CUS lo stato corrente (modalità e percentuale apertura valvola) per mantenere sincronizzata la Dashboard remota.
- **StateTask:** È il task di controllo principale che implementa la logica operativa.
 - **Input Utente:** Monitora la pressione del pulsante per il cambio modalità e legge il valore del potenziometro per il controllo manuale locale.
 - **Attuazione:** Comanda il servomotore mappando il valore percentuale (0 – 100%) in gradi (0 – 90°). In modalità UNCONNECTED, forza l'apertura totale per sicurezza.
 - **Interfaccia:** Gestisce l'aggiornamento del display LCD, visualizzando la modalità corrente e lo stato della valvola.

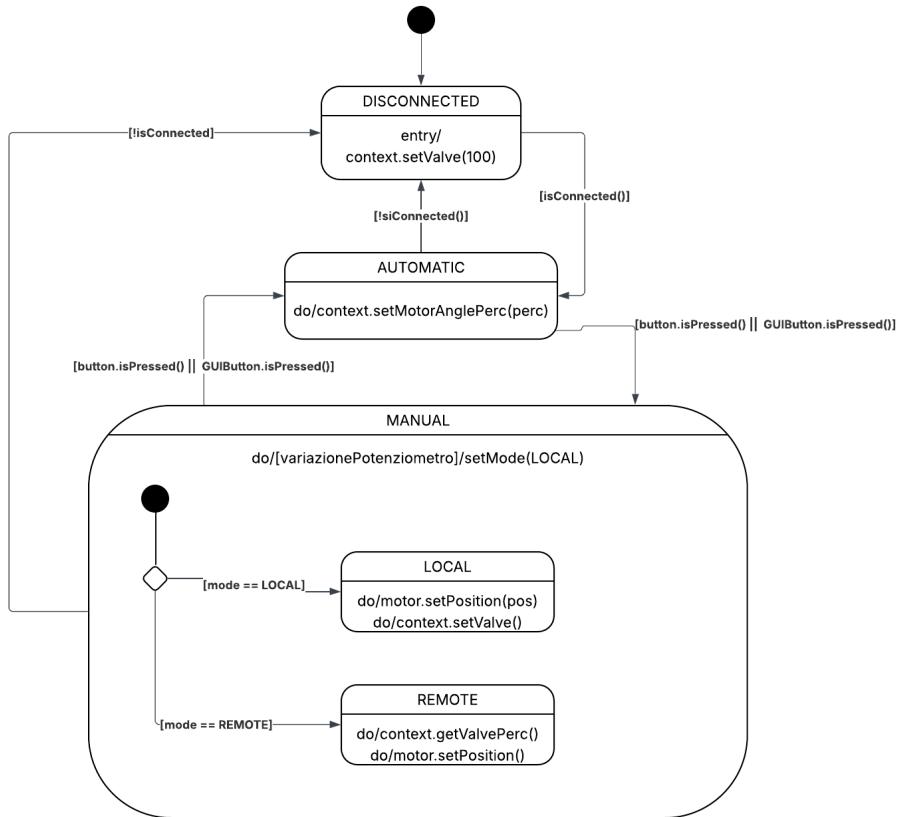


Figura 5: Schema degli stati stateTask

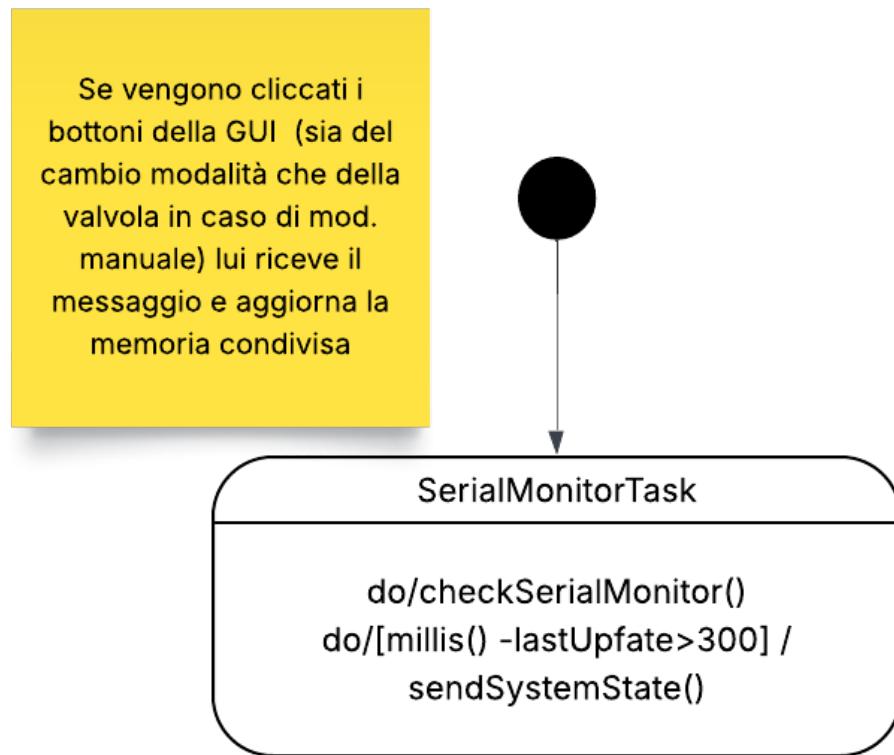


Figura 6: Schema degli stati serialMonitorTask

2.4 DBS - Dashboard Subsystem

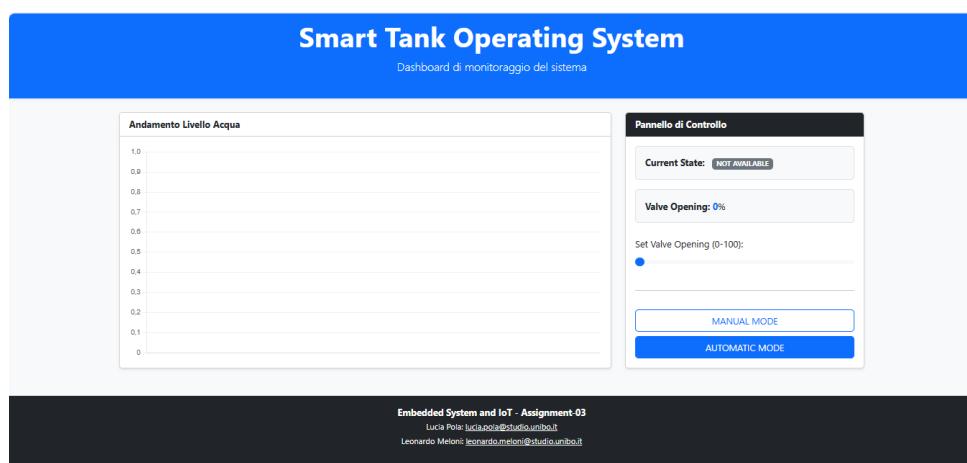


Figura 7: Schema del sottosistema dbs

Il **DBS** è un’interfaccia web realizzata con HTML5, Bootstrap 5 e JavaScript, che funge da pannello di controllo remoto per gli operatori. L’applicazione comunica con il CUS tramite richieste HTTP asincrone (REST API) sfruttando un meccanismo di polling periodico (1Hz) per aggiornare i dati in tempo reale.

Le funzionalità principali sono:

- **Monitoraggio:** Visualizza lo storico del livello dell’acqua su un grafico interattivo (libreria *Chart.js*) e riporta lo stato istantaneo del sistema e la percentuale d’apertura della valvola.
- **Controllo:** Fornisce pulsanti per commutare la modalità (AUTOMATIC/MANUAL). In modalità manuale, abilita uno slider (range 0-100%) che permette di inviare comandi di apertura valvola al CUS.

3 Protocolli di Comunicazione

Il sistema distribuito si avvale dei seguenti protocolli di comunicazione.

3.1 Interfaccia Seriale (CUS - WCS)

La comunicazione tra il PC (**CUS**) e la scheda Arduino (**WCS**) avviene tramite connessione seriale (USB/UART). Il protocollo è di tipo asincrono, basato su messaggi testuali ASCII terminati dal carattere `newline` ('`\n`').

3.1.1 Direzione CUS → WCS (Comandi)

Il CUS invia comandi al WCS per impostare lo stato del sistema o controllare l'apertura della valvola. I messaggi seguono il formato `<header>:<payload>`.

Header	Valore	Descrizione Funzionale
m	1	Set UNCONNECTED: Forza il WCS in stato di errore (timeout connessione rilevato dal CUS).
m	2	Set AUTOMATIC: Imposta il WCS in modalità automatica.
m	3	Set MANUAL: Imposta il WCS in modalità manuale.
v	0-100	Set Valve (Auto): Imposta la percentuale di apertura della valvola (comando generato dalla logica automatica).
r	0-100	Set Valve (Remote): Imposta la percentuale di apertura della valvola e attiva il controllo manuale remoto (comando generato dalla Dashboard).

Tabella 1: Tabella dei comandi seriali CUS verso WCS.

3.1.2 Direzione WCS → CUS (Telemetria)

Il WCS invia periodicamente (ogni 300ms) o su evento lo stato corrente del sistema per mantenere sincronizzata la logica centrale e la dashboard.

Il formato del messaggio è strutturato come segue:

`st:vo:<STATO>:<VALVOLA>`

Dove:

- `st:vo:` è il prefisso standard per lo stato della valvola e del sistema.
- `<STATO>` indica la modalità operativa corrente rilevata dal microcontrollore. Valori possibili:
 - **AUTOMATIC:** Controllo autonomo attivo.
 - **MANUAL:** Controllo manuale (locale o remoto) attivo.

- UNCONNECTED: Stato di emergenza/disconnessione.
- <VALVOLA> è un valore intero (0 – 100) che rappresenta l’effettiva apertura del servomotore.

Esempio di messaggio: st:vo:MANUAL:50 indica che il WCS è in modalità Manuale con la valvola aperta al 50%.

3.2 Interfaccia HTTP (CUS - DBS)

Il CUS espone un’API REST su porta 8080 (framework **Eclipse Vert.x**) interrogata dalla Dashboard tramite polling.

3.2.1 Endpoint API

La comunicazione utilizza payload **JSON** per le seguenti operazioni:

Endpoint	Metodo	Funzione e Payload
/api/status	GET	Restituisce lo stato corrente: {level, valve, mode}.
/api/history	GET	Restituisce lo storico dei livelli: [{time, value}, ...].
/api mode/switch	POST	Commuta la modalità operativa (AUTOMATIC/MANUAL).
/api valve/set	POST	Imposta l’apertura valvola (solo Manual). Payload richiesto: {"value": <int>}.

Tabella 2: Sintesi dell’API REST.

3.3 Interfaccia MQTT (CUS - TMS)

Il protocollo **MQTT** gestisce la comunicazione *Publish-Subscribe* tra il TMS (Publisher, ESP32) e il CUS (Subscriber, Java).

- **Payload:** I dati sono inviati come stringhe ASCII nel formato Distance:<VALORE> (es. Distance:45.50).
- **Interruzione comunicazione (T_2):** Il CUS implementa un *controllo* applicativo: se il tempo trascorso dall’ultimo messaggio ricevuto supera la soglia T_2 , il sistema transita forzatamente nello stato UNCONNECTED.

4 Configurazioni

4.1 Mappa dei Collegamenti (Pinout)

Di seguito sono riportate le connessioni fisiche realizzate per i due sottosistemi embedded.

4.1.1 TMS - Tank Monitoring Subsystem (ESP32)

Componente	Pin (GPIO)	Tipologia
Sonar Trigger	14	Digital Output
Sonar Echo	13	Digital Input
LED Verde	6	Digital Output
LED Rosso	5	Digital Output

Tabella 3: Pinout del TMS su scheda ESP32.

4.1.2 WCS - Water Channel Subsystem (Arduino UNO)

Componente	Pin	Tipologia
Servomotore (Valvola)	3	PWM Output
Potenziometro (Manuale)	A0	Analog Input
Pulsante (Mode Switch)	2	Digital Input
LCD SDA	A4	I2C Data
LCD SCL	A5	I2C Clock

Tabella 4: Pinout del WCS su scheda Arduino UNO.

4.2 Configurazione degli Scheduler

Poiché il sistema embedded è distribuito, TMS e WCS eseguono due istanze indipendenti dello Scheduler cooperativo. La tabella seguente riporta la frequenza di esecuzione dei task specifici:

Sottosistema	Task	Periodo	Descrizione
TMS	ConnectionTask	100 ms	Gestione WiFi e MQTT
	StatusTask	250 ms	Lettura Sonar e invio dati
WCS	SerialMonitorTask	200 ms	Comunicazione UART con CUS
	StateTask	150 ms	Logica FSM e controllo Servo

Tabella 5: Pianificazione dei Task nei sistemi embedded.

4.3 Parametri Funzionali

Il comportamento logico del sistema è governato dai seguenti parametri di soglia e temporizzazione, gestiti principalmente dal Control Unit Subsystem (CUS).

Parametro	Descrizione	Valore
L_1	Livello acqua moderato (Apertura valvola 50%)	10 cm
L_2	Livello acqua critico (Apertura valvola 100%)	20 cm
T_1	Tolleranza temporale per superamento soglia L_1	6000 ms
T_2	Timeout connessione TMS (Heartbeat)	5000 ms
F	Frequenza campionamento livello acqua	1/T ms
N	Numero campioni nello storico Dashboard	20

Tabella 6: Parametri di configurazione del sistema.