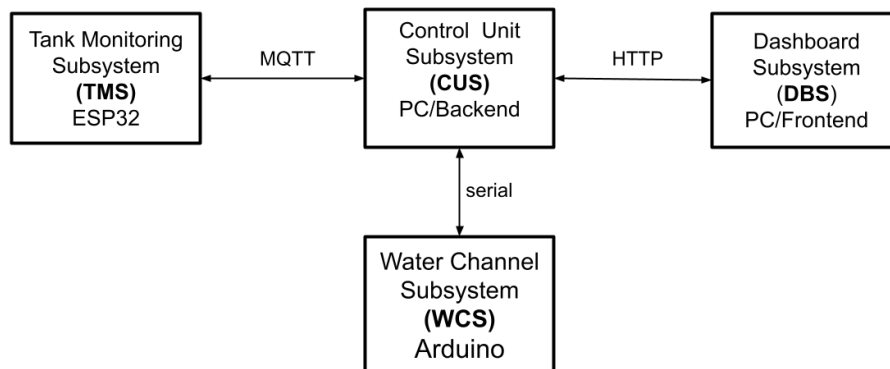


Assignment #03 - Smart Tank Monitoring System

Relazione per il corso di Sistemi Embedded e IoT



Indice

1	Traccia	3
1.1	Componenti	3
1.2	Comportamento	4
1.2.1	Dettagli Operativi	4
2	Progettazione dei Sottosistemi	5
2.1	TMS - Tank Monitoring Subsystem	5
2.1.1	Hardware	5
2.1.2	Software	6
2.1.2.1	Diagramma degli stati delle FSM	7
2.1.2.2	Diagramma delle classi	8
2.2	CUS - Control Unit Subsystem	9
2.2.0.1	Diagramma degli stati	10
2.2.0.2	Diagramma delle classi	11
2.3	WCS - Water Channel Subsystem	12
2.3.1	Hardware	12
2.3.2	Software	12
2.3.2.1	Diagramma degli stati delle FSM	14
2.3.2.2	Diagramma delle classi	15
2.4	DBS - Dashboard Subsystem	16
3	Comunicazione tra Sottosistemi	17
3.1	Protocollo Seriale	17
3.2	API HTTP	18
4	Test	19
A	Configurazioni Utilizzate	20
A.1	Cablaggio	20
A.1.1	TMS - ESP32	20
A.1.2	WCS - Arduino Uno	20
A.2	Periodo delle Task	21
A.2.1	TMS - ESP32	21
A.2.2	WCS - Arduino Uno	21
A.3	Parametri Operativi	21
A.3.1	Distanze	21

A.3.2 Tempi	21
-----------------------	----

Capitolo 1

Traccia

1.1 Componenti

Il sistema è composto da quattro sottosistemi:

- **Tank Monitoring Subsystem - TMS**
 - Basato su sistema embedded **ESP** per monitorare il livello dell'acqua piovana nel serbatoio.
 - Interagisce con il Control Unit Subsystem (CUS) via **MQTT**.
- **Control Unit Subsystem - CUS**
 - **Back-end** eseguito su PC.
 - È il sottosistema principale che governa e **coordina l'intero sistema**.
 - Interagisce via **MQTT** con il TMS.
 - Interagisce tramite 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 di scolo che collega il serbatoio alla rete idrica.
 - Interagisce via **seriale** con il CUS.
 - Fornisce un pannello per l'**interazione in loco** degli operatori umani.
- **Dashboard Subsystem - DBS**
 - **Frontend / Web app** eseguita su PC o qualsiasi dispositivo.
 - Interfaccia per **operatori remoti** per visualizzare i dati e interagire con il sistema via **HTTP** con il CUS.

1.2 Comportamento

Il sistema monitora il livello dell'acqua piovana nel serbatoio e controlla l'apertura di un canale di scolo.

Opera in due modalità: **AUTOMATICA** (default all'avvio) o **MANUALE**.

1.2.1 Dettagli Operativi

- **TMS**: Monitora costantemente il livello dell'acqua tramite il sonar
 - Campiona i dati a una frequenza F e li invia al CUS.
 - LED: Verde acceso / Rosso spento se il sistema è online; Verde spento / Rosso acceso in caso di problemi di rete.
- **WCS**: Controlla una valvola motorizzata
 - Apertura: Da 0% (0°) a 100% (90°).
 - Interazione: Un pulsante commuta tra modalità AUTOMATICA e MANUALE. In modalità manuale, l'apertura è regolata dal potenziometro.
 - LCD: Visualizza il livello di apertura attuale e la modalità corrente (**AUTOMATIC**, **MANUAL** o **UNCONNECTED**).
- **CUS**: Gestisce la logica di controllo
 - Se il livello supera $L1$ (ma $< L2$) per un tempo $> T1$, la valvola si apre al 50% finché il livello non scende sotto $L1$.
 - Se il livello supera $L2$, la valvola si apre immediatamente al 100% finché il valore non scende sotto $L2$.
 - Se non riceve dati dal TMS per un tempo $> T2$, entra nello stato **UNCONNECTED**.
- **DBS**: Dashboard di visualizzazione che include
 - Grafico del livello dell'acqua (ultime N misurazioni).
 - Percentuale di apertura della valvola e stato del sistema (**MANUAL**, **AUTOMATIC**, **UNCONNECTED** o **NOT AVAILABLE** se il CUS non è raggiungibile).
 - Controlli GUI per cambiare modalità e regolare l'apertura (in **MANUAL**).

Capitolo 2

Progettazione dei Sottosistemi

2.1 TMS - Tank Monitoring Subsystem

2.1.1 Hardware

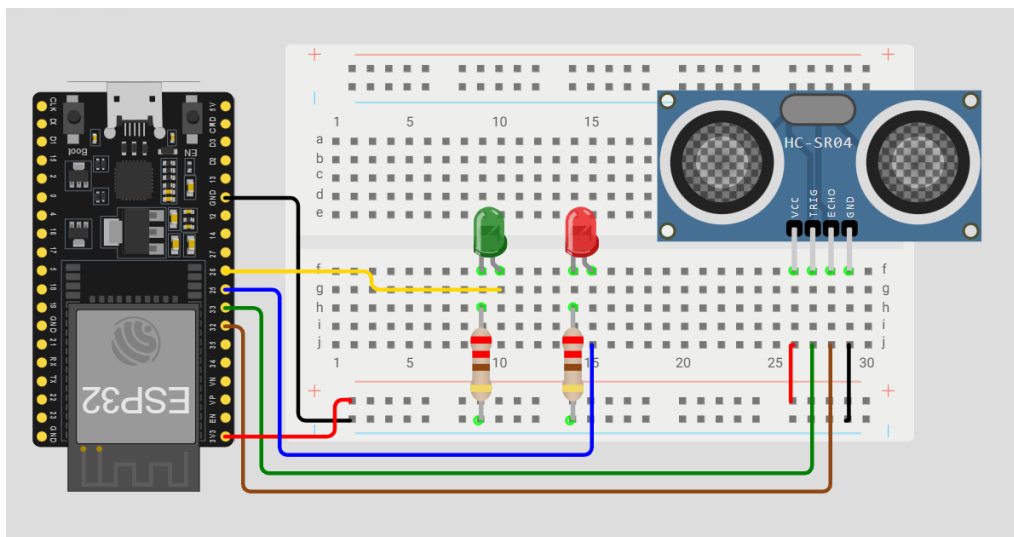


Figura 2.1: Rappresentazione schematica dell'hardware del TMS

Lato hardware sono stati utilizzati i seguenti componenti:

- SoC: ESP32
- Sensore livello acqua: Sonar HC-SR04
- LED verde e rosso

Nel circuito reale è stato utilizzato anche un Logic Level Converter per convertire il livello logico dell'ESP32 di 3.3V con quello del sensore di 5V, permettendo di alimentare il sensore con la tensione da lui richiesta.

Per l'elenco esaustivo dei pin utilizzati vedi Tabella A.1.

2.1.2 Software

Lato software l'architettura del TMS è basata su delle **task**, eseguite dal sistema operativo dell'ESP32.

Ogni task è basata su una **macchina a stati finiti** (FSM) (vedi schema in Fig. 2.2 a Pag. 7), da eseguire ogni certo periodo di tempo (vedi Tabella A.3). La comunicazione tra task avviene grazie a una struttura dati condivisa (la classe **SharedData**).

L'esecuzione periodica delle task avviene nella seguente maniera:

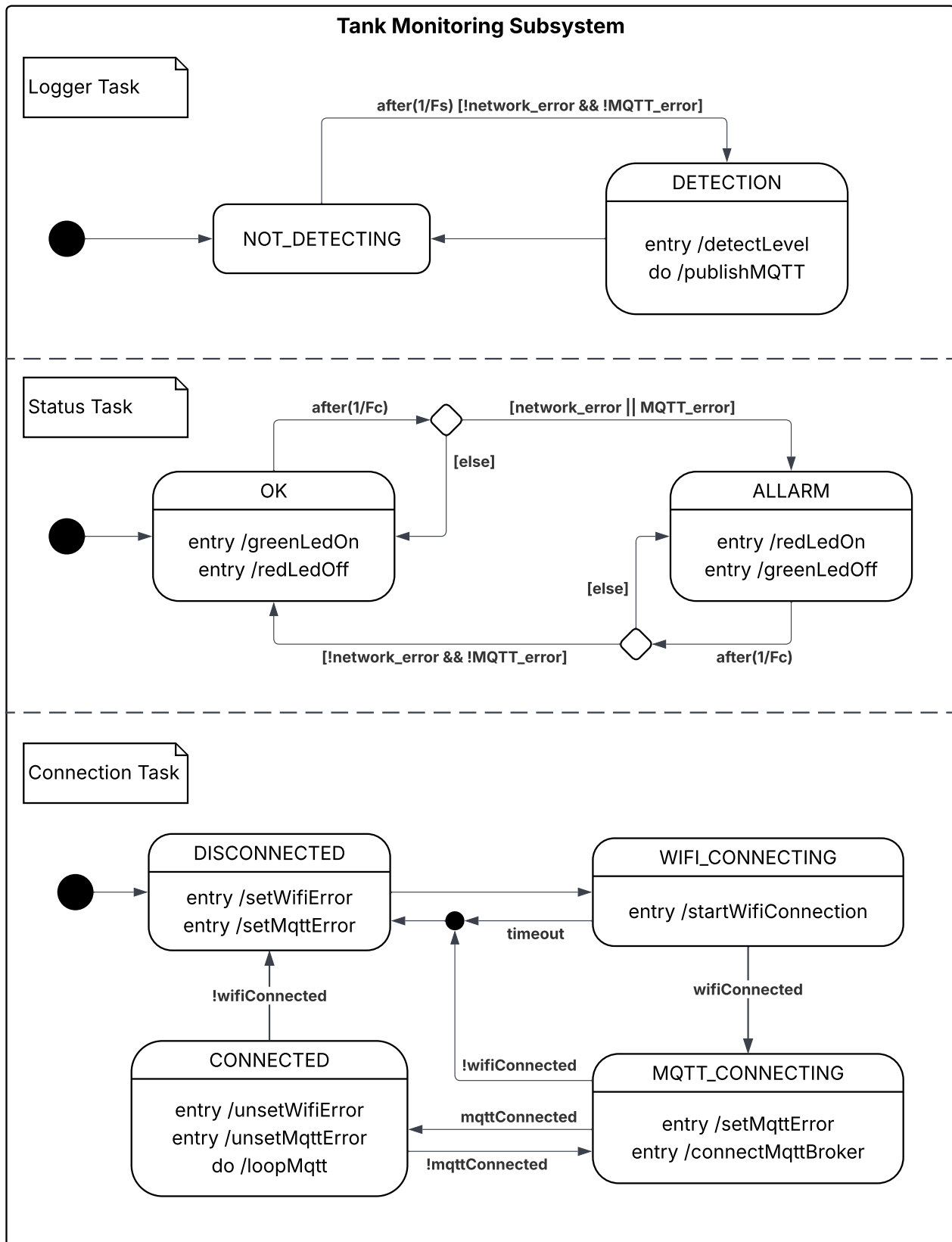
1. All'oggetto **Scheduler** viene passata la **task da eseguire** con il relativo **periodo**
2. Lo scheduler crea un oggetto **TaskExecutor**, contenente la task da eseguire, il periodo e la struttura dati condivisa
3. Tramite la **chiamata al sistema operativo** `xTaskCreatePinnedToCore` lo scheduler **mette in esecuzione** il metodo statico `execute` del **TaskExecutor**, passando come parametro l'oggetto **TaskExecutor** creato precedentemente
4. Il metodo `execute` esegue il metodo `init` della Task associata
5. Entra successivamente in un **ciclo infinito** in cui a ogni iterazione viene prima eseguito il metodo `tick` della Task, poi messa in pausa l'esecuzione sfruttando la primitiva `vTaskDelayUntil` di FreeRTOS per il periodo di tempo scelto

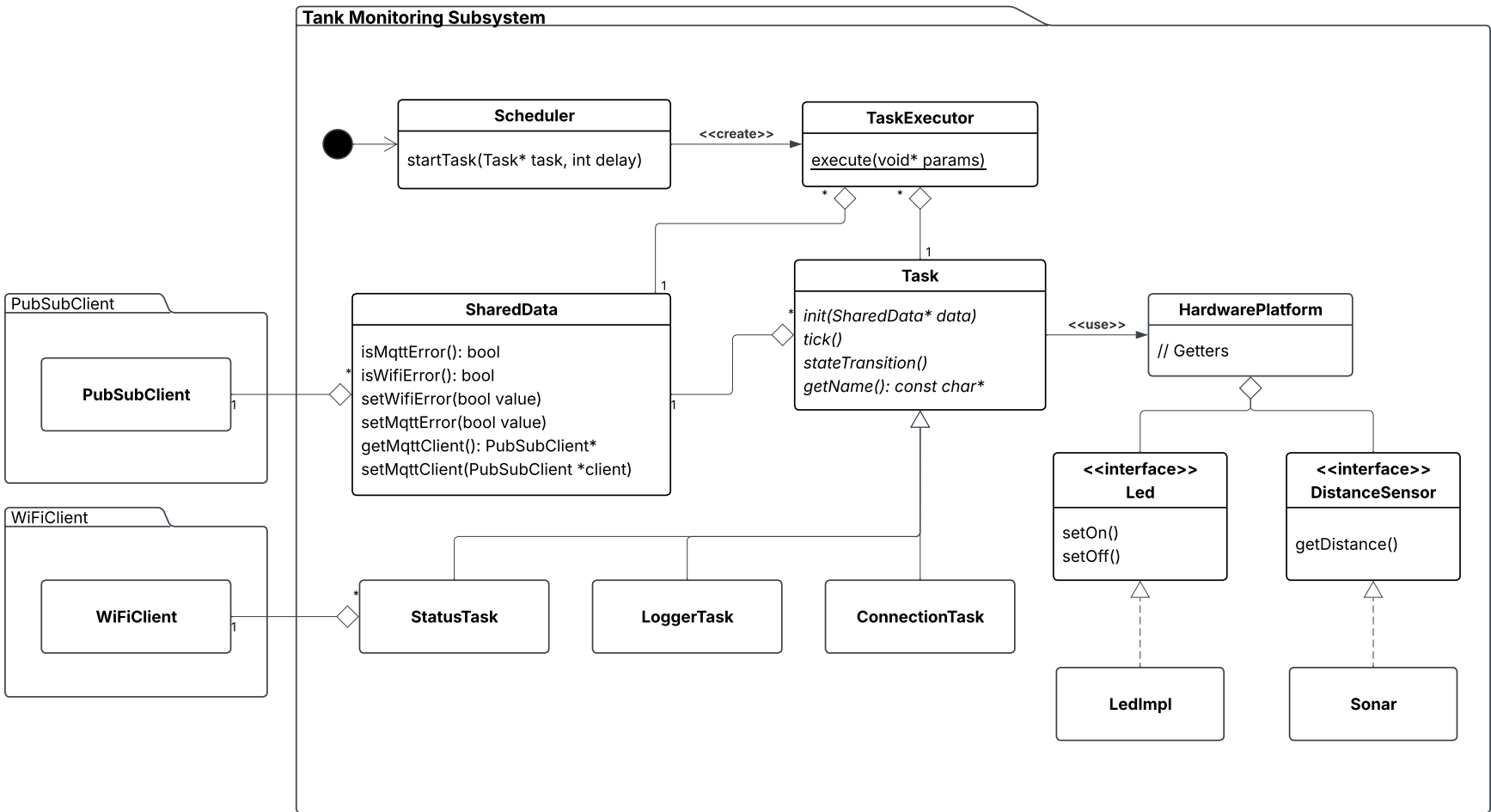
Le classi sono descritte nel diagramma delle classi Fig. 2.3, Pag. 8.

La comunicazione dei dati rilevati al CUS avviene tramite protocollo **MQTT**, pubblicando la lettura nel topic `water_level`.

Breve descrizione delle task:

- **LoggerTask:** In assenza di errori di rete, esegue la lettura per poi pubblicarla tramite MQTT.
- **StatusTask:** Accende e spegne i LED di stato, verde in caso di connessione corretta e rosso in caso di problemi.
- **ConnectionTask:** Controlla che le connessioni WiFi e MQTT siano funzionanti, tentando di riconnettersi in caso di problemi.





2.2 CUS - Control Unit Subsystem

Il CUS è il cuore del sistema: per questo necessita di avere una propria rappresentazione interna dello stato complessivo, descritto nel diagramma degli stati in Fig. 2.4 a Pag. 10. I valori dei parametri utilizzati nel diagramma degli stati sono specificati nell'Appendice A.3.

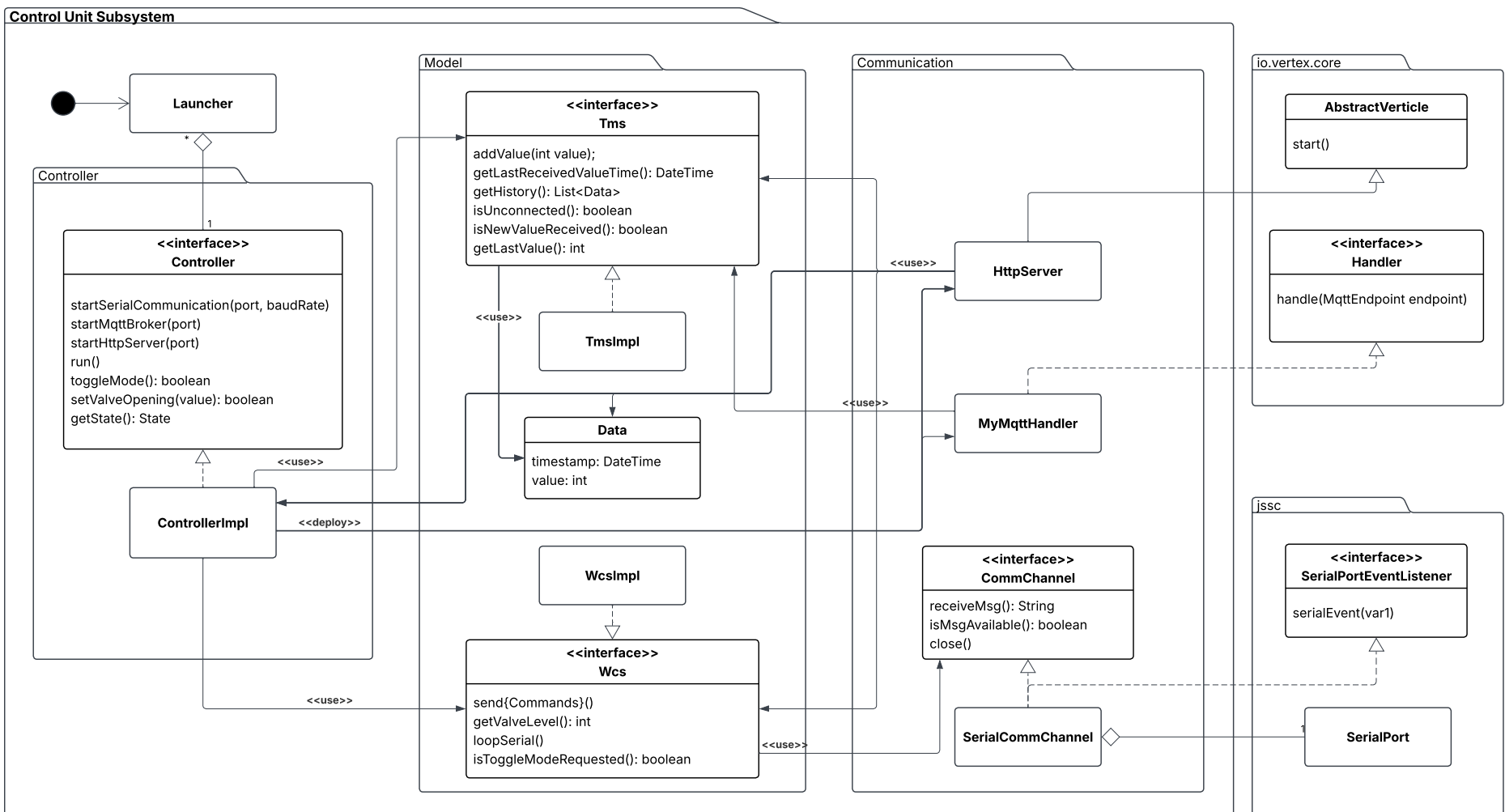
È un **processo server** che controlla e mette in comunicazione i vari sottosistemi, realizzato utilizzando il linguaggio **Java** e sfruttando **Gradle** per la compilazione. Per i server HTTP e MQTT è stata sfruttata la libreria **VertX**, specificando le azioni da eseguire per ogni evento da gestire, mentre per la comunicazione seriale viene utilizzata la libreria **jssc**. Si riporta il diagramma delle classi in Fig. 2.5 a Pag. 11.

Nello specifico:

- **Memorizza** i dati ricevuti dal TMS via **MQTT**, controllando che non passi troppo tempo dall'ultima comunicazione (entrando altrimenti nello stato **UNCONNECTED**).
- **Risponde** alle richieste **HTTP** create dal DBS (descritte nella Sezione 3.2):
 - Restituisce le ultime N letture del TMS
 - Restituisce lo stato del sistema e il livello di apertura della valvola
 - Cambia la modalità tra **AUTOMATIC** e **MANUAL**
 - Se in manuale, imposta l'apertura della valvola sulla base del parametro comunicato
- Quando in modalità automatica, **comunica** al WCS tramite seriale il **livello di apertura della valvola** sulla base delle letture inviate dal TMS.
- Quando in modalità manuale, riceve tramite seriale il livello di apertura corrente della valvola (sia che sia stato impostato tramite potenziometro, sia che sia stato impostato tramite seriale).

All'avvio dell'app verranno richiesti all'utente di indicare tutti i parametri di connessione, ovvero porta seriale e velocità in baud del WCS, porta del server HTTP e porta del broker MQTT.





2.3 WCS - Water Channel Subsystem

2.3.1 Hardware

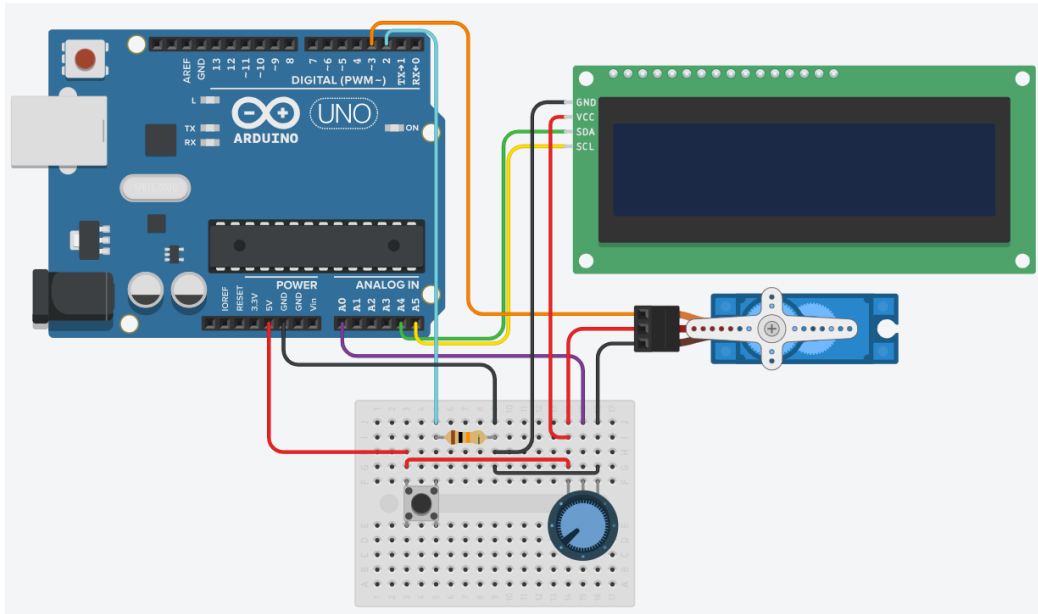


Figura 2.6: Rappresentazione schematica dell'hardware del WCS

Lato hardware sono stati utilizzati i seguenti componenti:

- Microcontrollore: Arduino Uno
- Display: LCD con interfaccia I2C
- Servomotore: Microservo SG90
- Selettore apertura valvola: Potenzziometro analogico
- Switch tra modalità: Pulsante tattile

Per l'elenco esaustivo dei pin utilizzati vedi Tabella A.2.

2.3.2 Software

Come per il TMS, l'architettura del WCS è basata su delle **task**: l'Arduino non è però dotato di un sistema operativo, rendendo quindi necessaria l'implementazione di uno **scheduler**, basato in questo caso sul Timer2 del microcontrollore.

Data la semplicità di questo sottosistema, è stata sviluppata una sola task basata su una **macchina a stati finiti** (FSM), di cui si riporta lo schema in Fig. 2.7 a Pag. 14.

Lo scheduler ogni 50ms controlla se è passato abbastanza tempo per eseguire una task, e in caso affermativo la esegue. Durante questi 50ms l'arduino viene posto in **sleep mode** di tipo IDLE, e viene risvegliato tramite il Timer2. La scelta della sleep mode è ricaduta sulla IDLE così che non vengano persi eventuali comandi ricevuti sulla seriale.

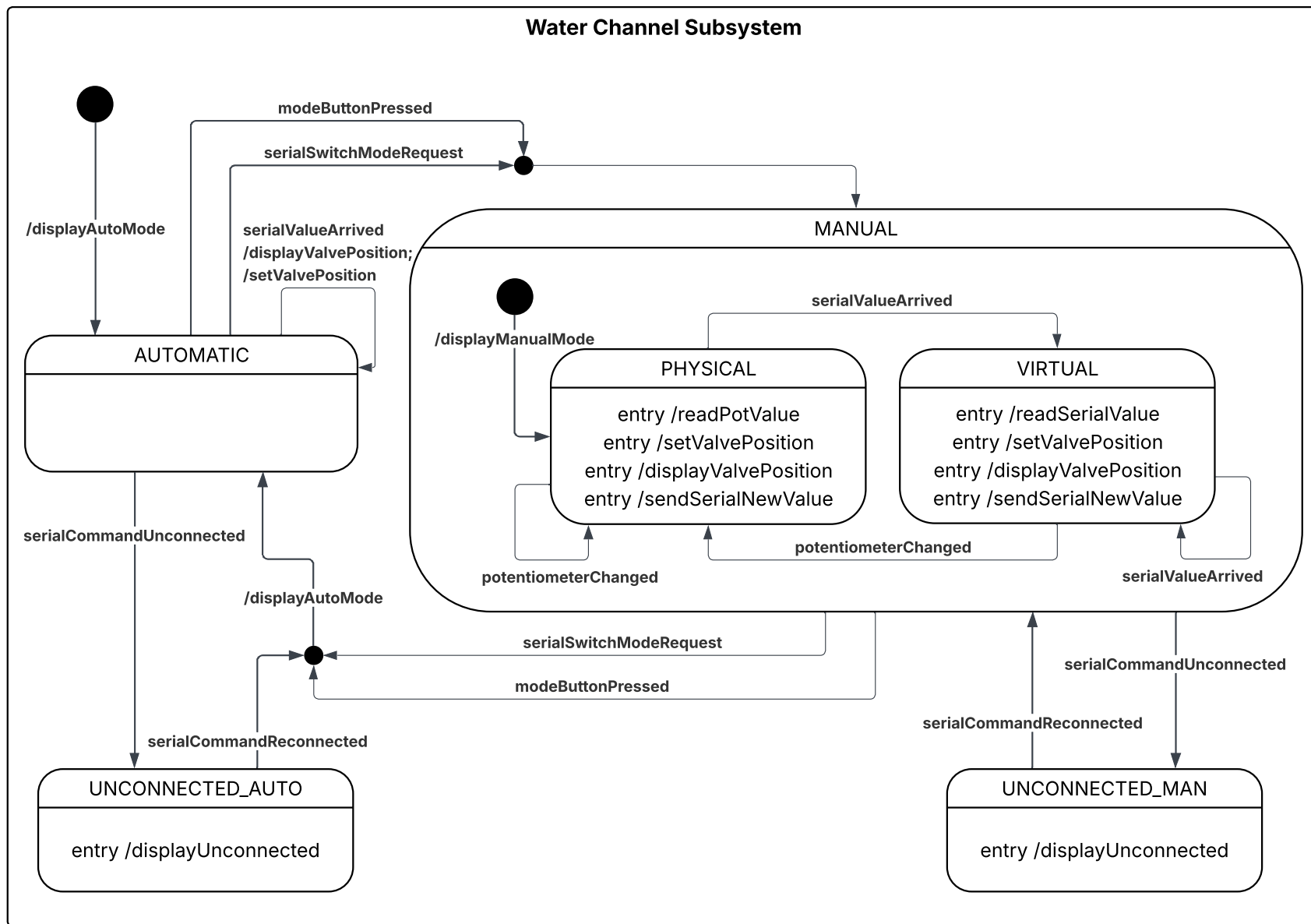
Si riporta il diagramma delle classi dell'**architettura software** in Fig. 2.8 a Pag. 15.

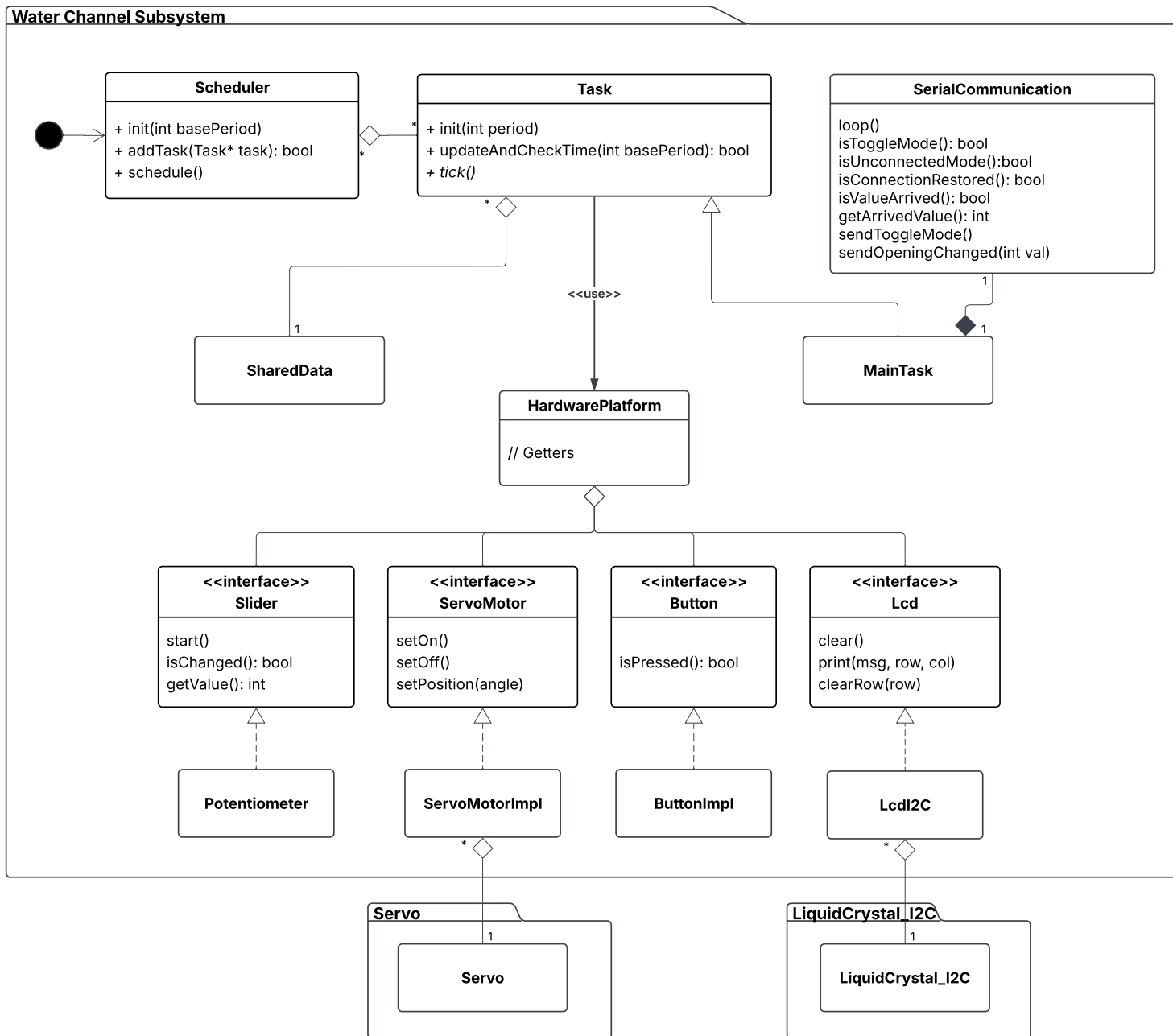
La comunicazione con il CUS avviene tramite **porta seriale**, seguendo il protocollo indicato nella Sezione 3.1.

Nello specifico:

- All'accensione il sistema entra nello stato **AUTOMATIC**, aspettando un **valore di apertura** sulla linea **seriale**.
- Controlla se viene **premuto il pulsante** per il **cambio modalità** o se viene ricevuto il corrispondente **comando seriale**, permettendo di passare tra **AUTOMATIC** e **MANUAL**.
- Quando in **MANUAL imposta la valvola** sul valore scelto tramite **potenziometro** o comunicato tramite **seriale**: in entrambi i casi viene comunicato al CUS il livello di apertura corrente.
- Entra nello stato **UNCONNECTED** se richiesto dal CUS, ritornando allo stato precedente al ripristino della connessione

Dato che il sistema è dotato di una sua logica, anche in **assenza di connessione seriale** al server CUS, un operatore fisico può impostare la modalità manuale e regolare l'apertura della valvola tramite potenziometro.





2.4 DBS - Dashboard Subsystem

La dashboard è stata realizzata tramite una **pagina HTML**, utilizzando il framework **bootstrap** per lo stile, la libreria **Chart.js** per la visualizzazione dello storico delle letture e il linguaggio **JavaScript (API Fetch)** per la comunicazione in HTTP con il CUS (protocollo descritto nella Sezione 3.2).

All'apertura è richiesto l'indirizzo del server su cui sono messe a disposizione le API HTTP; è comunque possibile cambiarlo in qualsiasi momento tramite l'apposito pulsante.

Tutti i dati visualizzati vengono aggiornati a intervalli regolari tramite nuove richieste HTTP al CUS.

La dashboard è composta da due sezioni:

- **Distanza dall'acqua:** Riporta il grafico con le ultime letture del TMS
- **Pannello di controllo:**
 - Stato attuale del sistema (modalità e livello di apertura della valvola)
 - Comando per passare tra modalità **AUTOMATIC** e **MANUAL**
 - Se in **MANUAL**, slider per impostare manualmente l'apertura della valvola

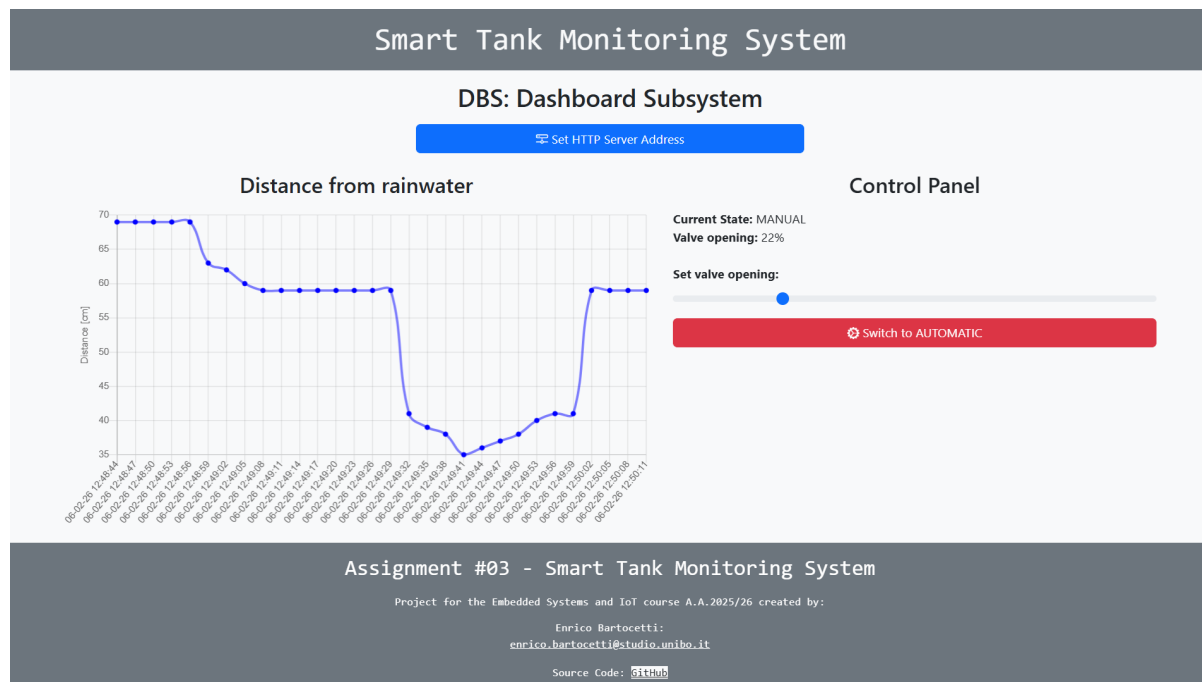


Figura 2.9: Screenshot della DashBoard mentre il sistema era in stato **MANUAL** con la valvola impostata al 22%

Capitolo 3

Comunicazione tra Sottosistemi

3.1 Protocollo Seriale

Per agevolare la comunicazione è stato stabilito un protocollo basato su stringhe separate da un carattere di `Line Feed`.

Comando	Direzione	Descrizione
0	CUS \leftrightarrow WCS	Passaggio di stato AUTOMATIC \longleftrightarrow MANUAL (e viceversa)
1	CUS \rightarrow WCS	Passaggio a stato UNCONNECTED
2	CUS \rightarrow WCS	Connessione ripristinata, ritorno allo stato precedente
3 + valore	CUS \leftrightarrow WCS	Nuovo valore di apertura della valvola

Tabella 3.1: Protocollo di comunicazione seriale tra CUS e WCS

Alcuni comandi sono bidirezionali, visto che:

- Il passaggio di stato **AUTOMATIC** \longleftrightarrow **MANUAL** può essere richiesto sia tramite DBS (quindi CUS \rightarrow WCS), ma anche tramite pulsante fisico nel WCS rendendo necessario l'aggiornamento del CUS (WCS \rightarrow CUS).
- Per impostare un valore manuale da remoto, è utilizzato il comando 3 (CUS \rightarrow WCS). In ogni caso, sia per valore impostato tramite potenziometro sia tramite DBS, quando in modalità manuale il WCS aggiorna il CUS sul valore di apertura corrente (WCS \rightarrow CUS).

3.2 API HTTP

Route	Metodo	Descrizione
api/data	GET	Restituisce un JSON con le ultime letture del TMS
api/status	GET	Restituisce un JSON con lo stato del sistema (modalità e apertura valvola)
api/toggle	POST	Richiede il passaggio di modalità AUTOMATIC \longleftrightarrow MANUAL
api/opening	POST	Imposta il nuovo valore di apertura contenuto nel campo value del JSON passato nel body della richiesta

Tabella 3.2: API HTTP messe a disposizione dal CUS

Nella Tabella 3.2 si riporta una breve descrizione delle API HTTP messe a disposizione dal CUS grazie alla classe `HttpServer`.

Capitolo 4

Test

Appendice A

Configurazioni Utilizzate

Tutti i parametri di configurazione possono essere impostati nei file `include/config.h` di ogni sottosistema. Si riportano in seguito tutti quelli utilizzati ai fini del test.

A.1 Cablaggio

A.1.1 TMS - ESP32

Componente	Pin	Tipo
Led Verde	26	Digital Output
Led Rosso	25	Digital Output
Sonar Trigger	33	Digital Output
Sonar Echo	32	Digital Input

Tabella A.1: Elenco completo dei pin utilizzati nell'ESP32

A.1.2 WCS - Arduino Uno

Componente	Pin	Tipo
Pulsante cambio modalità	2	Digital Input
Potenzimetro controllo manuale	A0	Analog Input
Servomotore valvola	3	Digital Output (PWM)
LCD I2C SDA/SCL	A4/A5	I2C

Tabella A.2: Elenco completo dei pin di Arduino utilizzati

A.2 Periodo delle Task

A.2.1 TMS - ESP32

Task	Period (ms)
Logger Task	3000
Connection Task	500
Status Task	1000

Tabella A.3: Periodo di esecuzione di ogni Task

A.2.2 WCS - Arduino Uno

Task	Period (ms)
Main Task	250

Tabella A.4: Periodo di esecuzione di ogni Task

A.3 Parametri Operativi

A.3.1 Distanze

Quando il sistema è in modalità **AUTOMATIC**, il livello di apertura della valvola viene stabilito sulla base dei seguenti valori, impostati nella classe **ControllerImpl** del CUS:

Descrizione	Nome parametri	Distanza dalla cima	Apertura valvola
Cisterna Vuota	$> L1$	> 50 cm	0%
Cisterna Semipiena	$> L2 \ \& \ \leq L1$	> 30 cm $\ \& \ \leq 50$ cm	50%
Cisterna Piena	$\leq L2$	≤ 30 cm	100%

Tabella A.5: Parametri per l'identificazione dello stato della cisterna

A.3.2 Tempi

Descrizione	Classe	Nome parametro	Tempo
Isteresti per transizione Cisterna Vuota \rightarrow Cisterna Semipiena	TmsImpl	T1 - HYSTERESIS	5s
Passaggio ad UNCONNECTED	ControllerImpl	T2 - TIMEOUT	7s

Tabella A.6: Tempi presenti nel CUS per effettuare i cambi di stato