Relazione dell'Assignment 3: Smart-Tank-Monitoring

Lorenzo Antonioli
Eleonora Bianco
Giulia Fares
Luca Varale Rolla

3 febbraio 2026

Indice

1 Introduzione

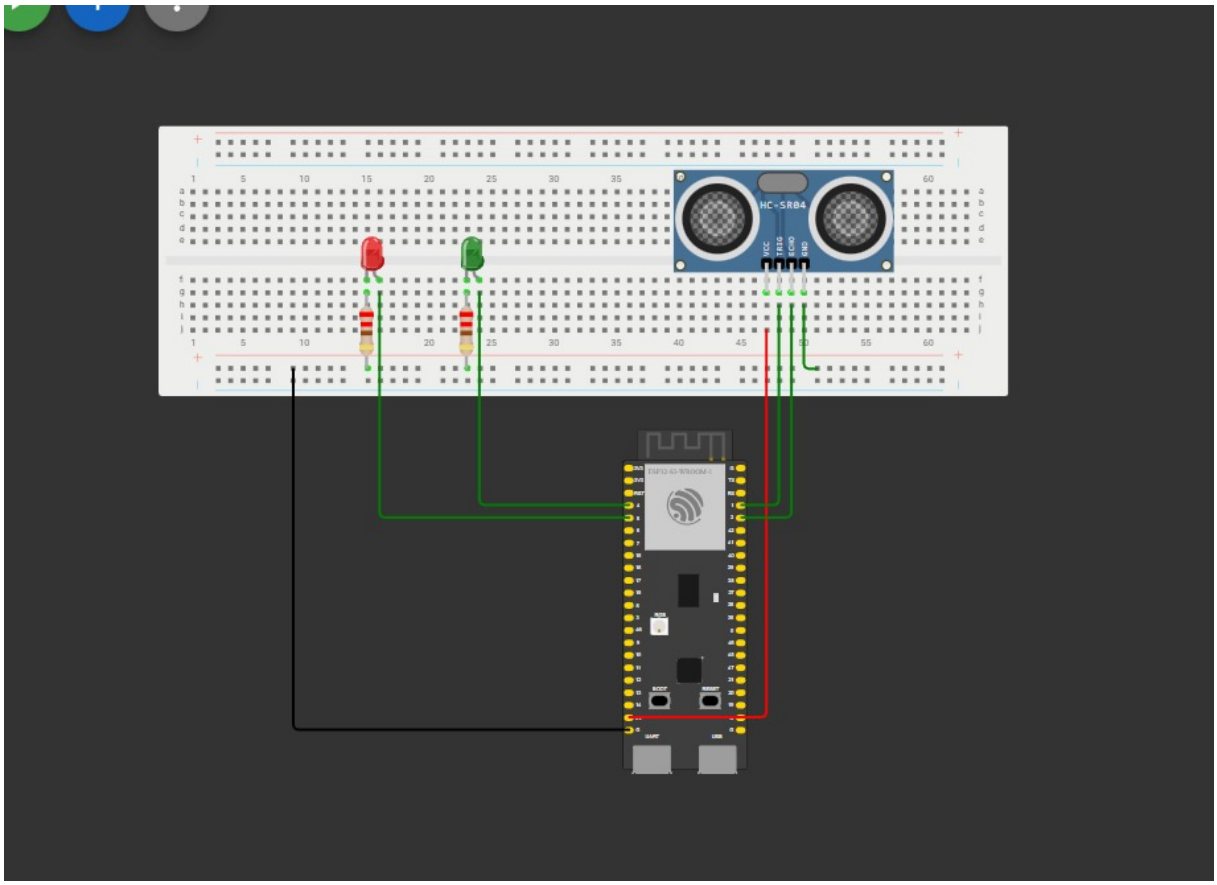
Il progetto consiste nell'implementazione di un sistema IoT per il monitoraggio e controllo di un tank d'acqua. Il sistema è suddiviso in quattro moduli principali:

1. **TMS**: Nodo sensore basato su SoC ESP32 che monitora il livello dell'acqua.
2. **WCS**: Nodo attuatore basato su Arduino Uno che manovra una valvola per controllare l'afflusso di acqua e permette a un operatore di interagire manualmente.
3. **CUS**: Backend di controllo sviluppato in Java, responsabile della logica e del bridging dei protocolli.
4. **DBS**: Frontend web per la visualizzazione dello stato del sistema ed eventuale controllo manuale del WCS.

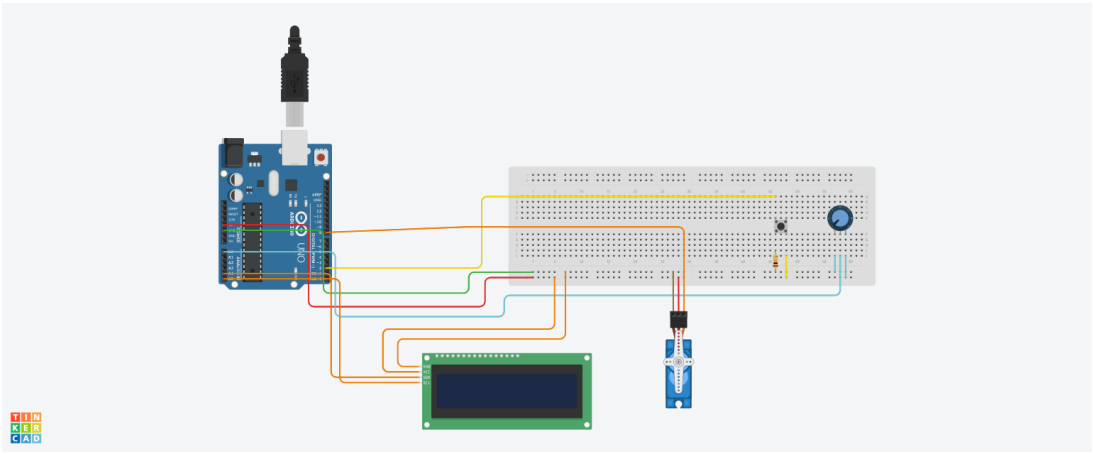
2 Architettura e Schema del Sistema

Il sistema implementa l'architettura indicata nel testo dell'assignment con al centro il CUS che funge da smistamento e gestione dei messaggi.

- **MQTT su WiFi**: Protocollo utilizzato per la comunicazione dei dati al CUS di cui il TMS è responsabile.
- **Seriale**: Implementata per il WCS utilizzando un protocollo a messaggi di testo (es. "CMD:MANUAL").
- **HTTP/JSON**: Permette al frontend, tramite il CUS, di interagire e inviare comandi al WCS per il controllo del servo motor.



(a) TMS (ESP32)



(b) WCS (Arduino Uno)

Figura 1: Schemi dei sottosistemi WCS e TMS.

3 Dettagli Implementativi dei Sottosistemi

3.1 TMS - Tank Monitoring Subsystem

Il TMS è stato sviluppato sfruttando le capacità dual-core dell'ESP32 tramite il sistema operativo **FreeRTOS**. L'architettura è task-based e divide i compiti computazionali da quelli di comunicazione:

- **Gestione Multicore**: Il task di comunicazione (`CommTask`) è stato assegnato al Core 0 per gestire lo stack WiFi e MQTT senza interferire con le letture dei sensori. I task applicativi (Sonar e LED) risiedono sul Core 1.
- **Sincronizzazione**: La comunicazione tra il task di lettura del sonar e il task di invio dati avviene tramite una *Queue* FreeRTOS thread-safe, garantendo thread-safety tra i due core.
- **FSMs**: È stata implementata una macchina a stati finiti sincrona per gestire gli stati WORKING e NOT-WORKING.

3.2 WCS - Water Channel Subsystem

Anche in questo caso l'architettura è task-based ed è stato utilizzato uno scheduler per la gestione dei task:

- Lo scheduler gestisce l'esecuzione ciclica dei task basandosi sul tempo di sistema (`millis()`). Ogni task (Input, Logic, Valve, Display, Comm) possiede un proprio periodo di esecuzione.
- **Context Pattern**: Per evitare l'uso indiscriminato di variabili globali, è stata implementata una classe `Context` che funge da memoria condivisa. Tutti i task leggono e scrivono lo stato del sistema (livello valvola, modalità operativa) su questo oggetto.
- **FSMs**: Realizzata una finite state machine sincrona per gli stati del WCS: AUTOMATIC, MANUAL e UNCONNECTED.

3.3 CUS - Control Unit Subsystem

Il CUS è un'applicazione Java back-end che gestisce l'intero sistema.

- **Comunicazione Seriale**: Il sottosistema utilizza la libreria `jSerialComm` per stabilire la connessione con il WCS. Attraverso questo canale vengono scambiati messaggi testuali per la sincronizzazione degli stati e l'invio dei comandi di attuazione.
- **Protocollo MQTT**: Tramite la libreria Paho Eclipse, il CUS agisce come subscriber sul topic `tms/waterlevel`. Questo permette la ricezione dei dati inviati dal TMS.
- **Server HTTP**: È stato implementato un server HTTP per il collegamento con il DBS. Il server gestisce richieste GET per il polling dello stato e POST per i comandi manuali, utilizzando JSON come formato standard di interscambio dati.

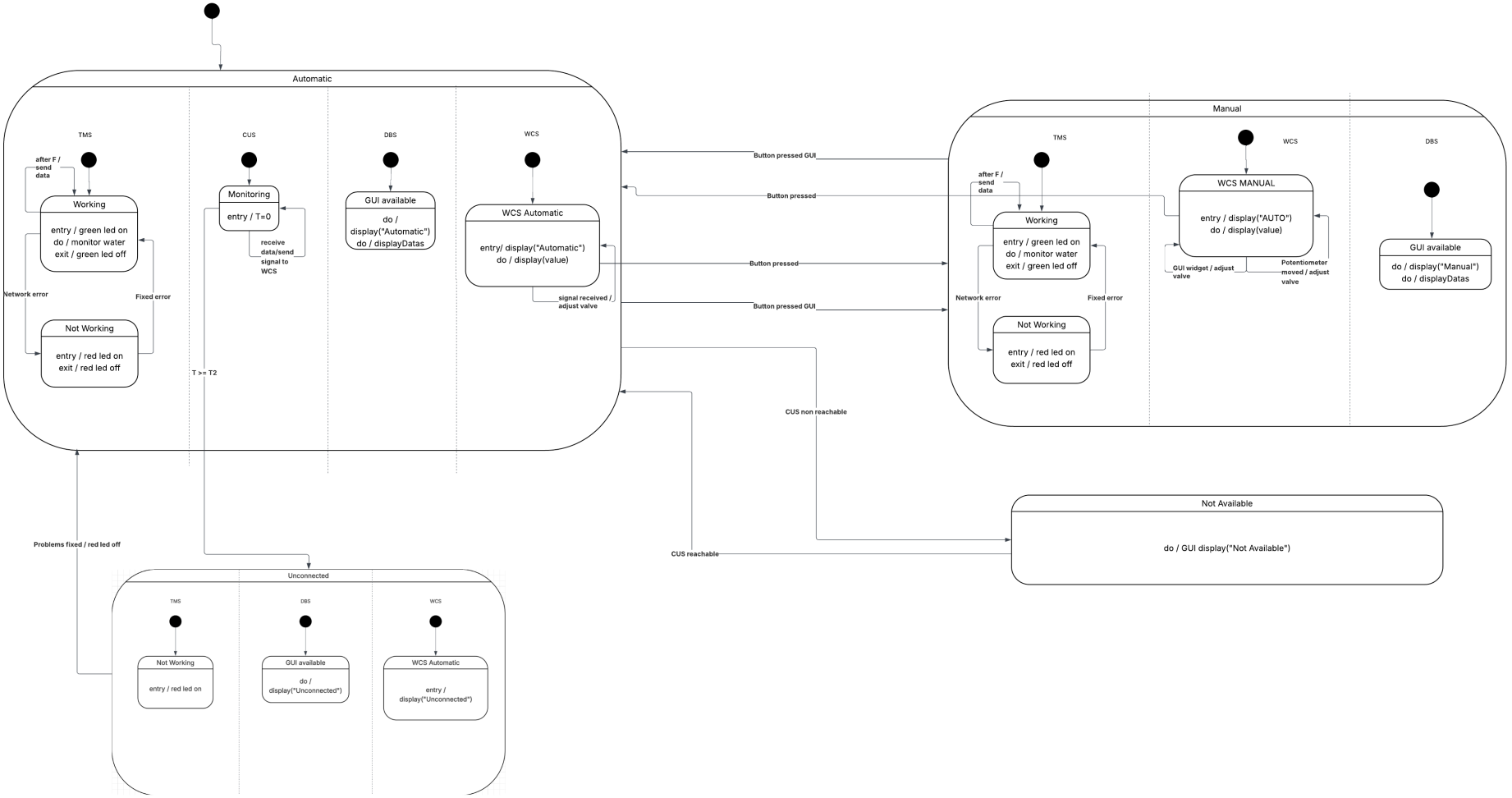
3.4 DBS - Dashboard Subsystem

La Dashboard è un'interfaccia sviluppata in HTML, CSS (usando anche la libreria Bootstrap) e JavaScript.

- **Polling Asincrono**: Non mantenendo una connessione socket persistente, il client utilizza chiamate `fetch` periodiche per sincronizzare lo stato dell'interfaccia con il CUS.
- **Rendering Dinamico**: Il DOM viene manipolato in tempo reale via JavaScript per aggiornare badge di stato e slider di controllo senza ricaricare la pagina. La visualizzazione del grafico dei livelli d'acqua monitorati avviene tramite la libreria Chart.js.

4 Macchine a Stati Finiti Sincrone (FSMs)

La logica di controllo distribuita è stata modellata tramite Macchine a Stati Finiti sincrone. Il diagramma seguente illustra le transizioni di stato globali e le interazioni tra gli stati interni dei singoli sottosistemi.


Automatic
TMS
CUS
DBS
WCS
after F / send data
Working
entry / green led on
do / monitor water
exit / green led off
Monitoring
entry / T=0
receive data/send signal to WCS
GUI available
do / display("Automatic")
do / displayDatas
WCS Automatic
entry/ display("Automatic")
do / display(value)
signal received / adjust valve
Network error
Fixed error
Not Working
entry / red led on
exit / red led off
T >= T2
Button pressed GUI
Button pressed
Button pressed
Button pressed GUI
CUS non reachable
CUS reachable
Problems fixed / red led off
Manual
TMS
WCS
DBS
after F / send data
Working
entry / green led on
do / monitor water
exit / green led off
WCS MANUAL
entry / display("AUTO")
do / display(value)
GUI widget / adjust valve
Potentiometer moved / adjust valve
GUI available
do / display("Manual")
do / displayDatas
Network error
Fixed error
Not Working
entry / red led on
exit / red led off
Not Available
do / GUI display("Not Available")
Unconnected
TMS
DBS
WCS
Not Working
entry / red led on
GUI available
do / display("Unconnected")
WCS Automatic
entry / display("Unconnected")

5 Link Al Video

Cliccare qui per il video