Assignment 3 – ESIOT 2023/2024

Alessandro Coli (alessandro.coli2@studio.unibo.it)

Chiara Giangiulli (chiara.giangiulli@studio.unibo.it)

Giovanni Pisoni (giovanni.pisoni@studio.unibo.it)

Mirco Terenzi (mirco.terenzi@studio.unibo.it)

Architettura

Il progetto è suddiviso in quattro parti principali che comunicano tra di loro al fine di raggiungere l'obiettivo:

- water-level-monitoring-subsystem su ESP, il quale rileva il livello dell'acqua tramite un sonar ad una data frequenza e comunica con il back-end tramite MQTT, grazie all'appoggio della libreria ESP-MQTT
- water-channel-controller su Arduino, il quale gestisce l'apertura della diga, tramite controllo manuale con un potenziomentro oppure automaticamente, ricevendo la percentuale di apertura dal backend tramite seriale.
- river-monitoring-dashboard: una web app il cui compito è quello di mostrare le informazioni attuali, ricevute dal back-end tramite http, e fornisce la possibilità di modificare l'apertura della diga da remoto.
- river-monitoring-service ossia il back-end, interamente implementato in java, che collega le tre parti ed ha il ruolo di gestire automaticamente l'apertura della diga al variale del livello dell'acqua e di fare comunicare le altre 3 componenti in maniera corretta e coerente.

1. Arduino

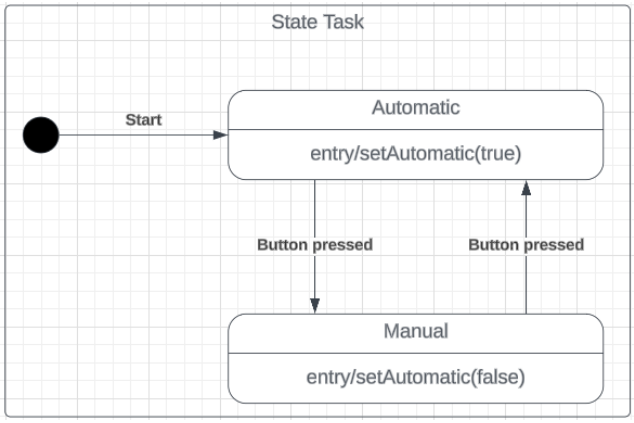
Nel sottosistema Water Channel Controller si trova la parte di Arduino, la quale agisce come parte operativa del sistema Smart River Monitoring. L'implementazione realizzata sfrutta una FSM (Finite State Machine) che ha il compito di controllare la valvola tramite un servomotore in riposta a ciò che gli viene comunicato (tramite seriale) dal River Monitoring Service.

Inoltre, nell'implementazione, si è deciso di attuare una suddivisione delle operazioni in diversi task, ognuno dei quali si occupa di realizzare determinate funzioni del progetto. Per gestire i task è stato implementato uno scheduler che si occupa di eseguirli in maniera concorrente.

Di seguito vengono elencati i task nello specifico:

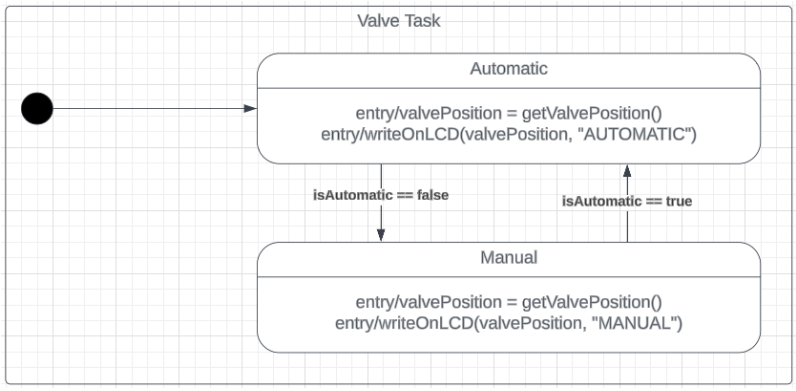
1.1. State Task

Questo task è utilizzato per cambiare stato da automatico a manuale o viceversa alla pressione del tasto presente nel circuito di Arduino.



1.2. Valve Task

Valve Task è responsabile di controllare l'angolo di apertura del servomotore che deriva da una percentuale (dallo 0% che equivale a 0° al 100% che equivale a 180°) passata al Water Channel Controller tramite seriale dallo Smart River Monitoring nel caso in cui il sistema è in modalità automatica; mentre, se il sistema è in modalità manuale, la percentuale (mappata nello stesso modo del caso automatico) viene decisa da un potenziometro. Il task non fa altro che controllare se la macchina è in stato automatico o manuale, dopo di che, a seconda dello stato, tramite uno switch setta l'apertura della valvola e scrive sul LCD I2C lo stato della macchina.

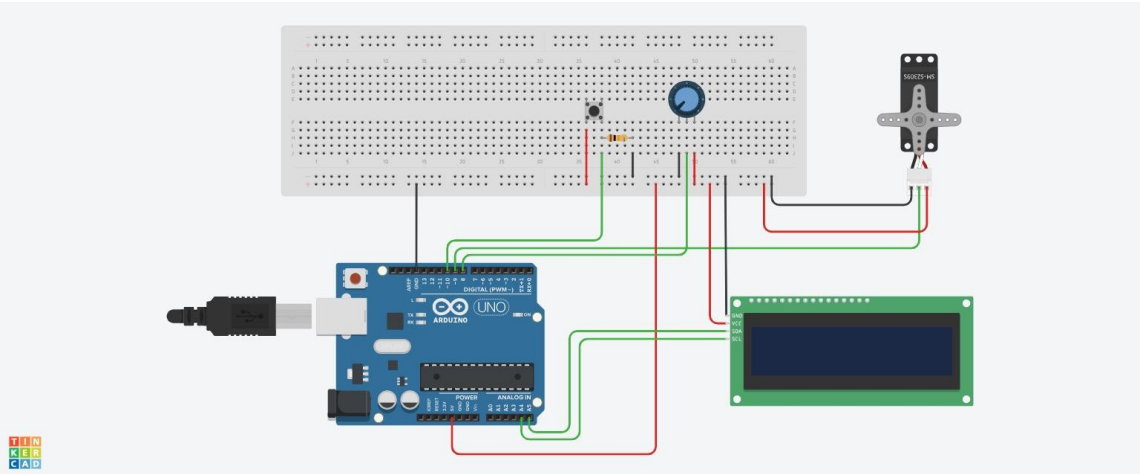


1.3. Comunication Task

Comunication Task ha il compito di gestire le interazioni con il back-end sulla serial line. Se presenti, legge i messaggi inviategli dallo Smart River Monitoring e fornisce al controller la percentuale di apertura del servomotore. Inoltre, in caso di stato della macchina manuale, prepara i messaggi da inviare all'unità centrale spendendoli poi sulla serial line.

1.4. Rappresentazione dello schema

Di seguito una rappresentazione del progetto:



2. ESP32

Il water-level-monitoring-subsystem è realizzato utilizzando un ESP32 che comunica con il river-monitoring-service attraverso il protocollo MQTT. Il suo compito è controllare e inviare periodicamente al backend il livello dell'acqua. Per fare questo, viene implementato un monitoringTask.
Questo si occupa di controllare continuamente il livello dell'acqua attraverso un sonar e inviarlo ad una frequenza F, stabilita e comunicata attraverso MQTT dal back end, al river-monitoring-service.

Altro compito del water-level-monitoring-subsystem è controllare lo stato della comunicazione e visualizzarlo attraverso l'accensione e lo spegnimento di due led, uno rosso e uno verde.
Questo compito viene svolto dal ledControlTask che, in base allo stato del WiFi e della connessione del client MQTT, accende il green led (il sistema sta lavorando correttamente) o il red led (il sistema sta riscontrando problemi di connessione).

3. Back-end

Il ruolo di river-monitoring-service è unire le altre tre componenti tra loro fungendo da tramite per le loro comunicazioni e per decidere in maniera automatica il livello del gate e la frequenza di update.

Ad ogni tick il back-end chiede ad esp tramite mqtt il livello dell'acqua e in base a quello determina uno stato e di conseguenza decide il livello della diga e la frequenza di aggiornamento. Come broker per mqtt si è scelto di usare broker.mqtt-dashboard.com e come libreria lato java Eclipse Paho.

Manda poi tramite seriale ad Arduino il livello del gate e alla dashboard tutte le informazioni necessarie alla visualizzazione tramite HTTP.

4. Dashboard

Tramite la dashboard, implementata come web app, possiamo visualizzare tutte le informazioni circa il sistema. Le informazioni visualizzate comprendono lo stato del sistema, il livello di apertura della valvola e lo storico del livello dell'acqua, presentato tramite un grafico.

Sempre utilizzando la dashboard siamo in grado di cambiare il tipo di controllo della valvola: da automatico a manuale e viceversa. Tramite il controllo manuale siamo in grado di decidere la percentuale di apertura della valvola, interrompendo la gestione automatica del sistema.

Il sistema implementa la comunicazione tra applicativo web e back-end tramite connessione HTTP, gestendo i vari tipi di messaggi e dati scambiati: oltre ai dati necessari per la visualizzazione dello stato del sistema viene utilizzata una variabile controlType che permette di interpretare la funzionalità del messaggio corrente.