Esiot-Assigment3 Report

River Monitoring System

Gardenghi Riccardo
0000970796 riccardo.gardenghi@studio.unibo.it

Penazzi Riccardo
0001021734 riccardo.penazzi@studio.unibo.it

Pracucci Filippo
0001020606 filippo.pracucci@studio.unibo.it

Anno accademico 2023-2024

Indice

Water Channel Controller

1.1 Schema Elettrico

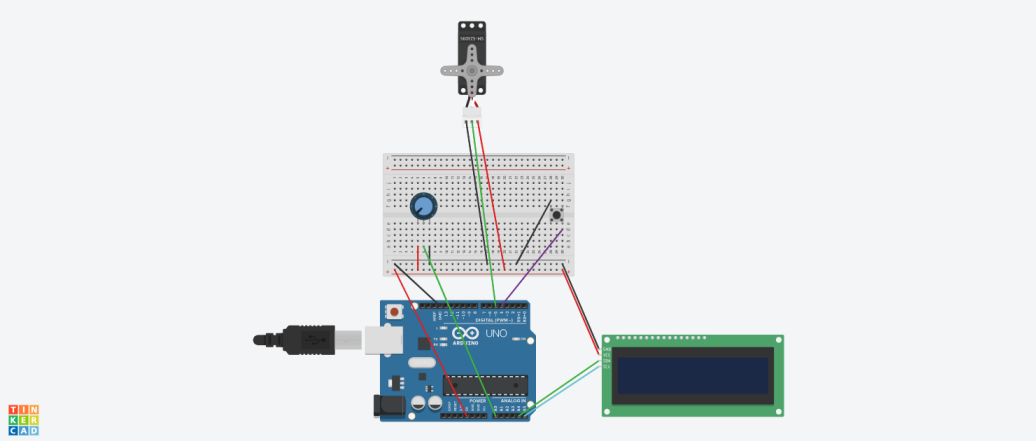

Figura 1.1: Schema Elettrico

1.2 Architettura

Abbiamo realizzato un'architettura basata sul concetto di macchine a stati finiti sincrone (synchronous FSM), sfruttando tre principali classi: *Scheduler*, *Task* e *Stato*. In particolare il *Water Channel Controller* presenta due soli stati:

1. automatico;
2. manuale.

In questa architettura si è cercato di scindere il concetto di stato da quello di task; nello specifico la classe *Stato* non eredita la classe *Task*, ma è invece una

classe a sé stante che fornisce un metodo *goNext()* per determinare quando bisogna passare allo stato successivo. A occuparsi delle transizioni di stato è il task *StateManager* che viene inserito nello scheduler all'avvio del sistema; esso controlla periodicamente la condizione del metodo *goNext()* per lo stato attuale ed eventualmente esegue la transizione al successivo. Ogni stato ha i comandi che vengono eseguiti una sola volta nel costruttore, mentre per i comandi periodici viene aggiunto un task allo scheduler, il quale verrà rimosso nel distruttore al cambiamento dello stato. Questa architettura ha il vantaggio di non fare una fuorviante associazione stato-task permettendoci di gestire in maniera più chiara le azioni di ingresso e uscita da uno stato; questo ci consente anche di riutilizzare gli stessi task su più stati senza così avere duplicazioni di codice.

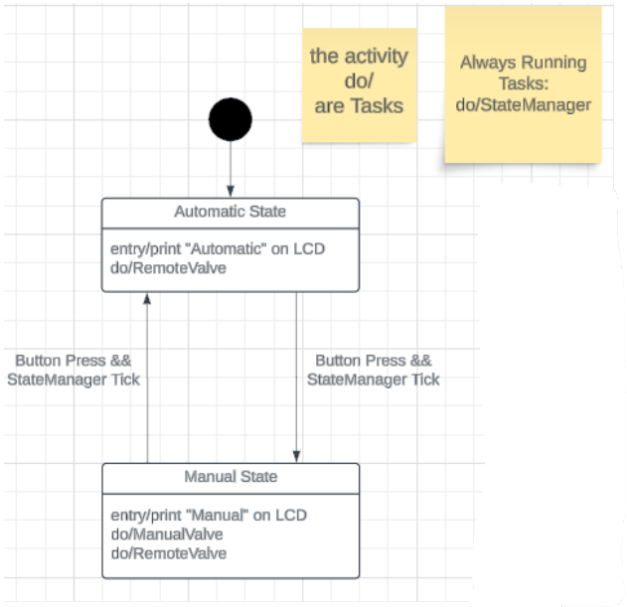

Figura 1.2: FSM water channel controller

1.3 Tasks

1.3.1 Remote valve Task

Questo task si occupa di controllare periodicamente se ha ricevuto un messaggio sulla seriale con una nuova posizione per la valvola; in tal caso la valvola viene impostata all'angolo specificato.

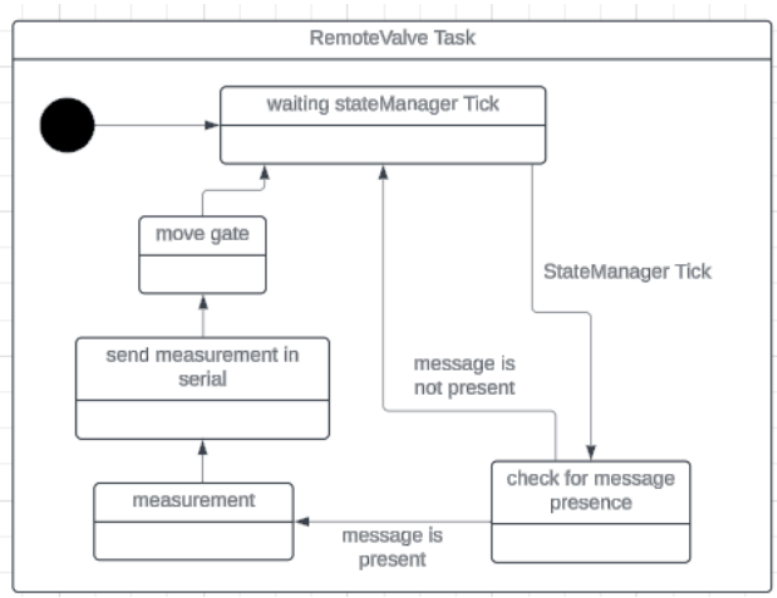

Figura 1.3: Remote Valve Task

1.3.2 Manual valve Task

Questo task si occupa di muovere la valvola in base alla posizione del potenziometro. Nel nostro caso però la dashboard ha una priorità maggiore rispetto al potenziometro, perciò anche muovendo quest'ultimo, se è stato impostato un valore dalla dashboard, la modifica verrà ignorata. Questa scelta è stata fatta per evitare che le modifiche effettuate dalla dashboard vengano subito sovrascritte dal valore del poteziometro.

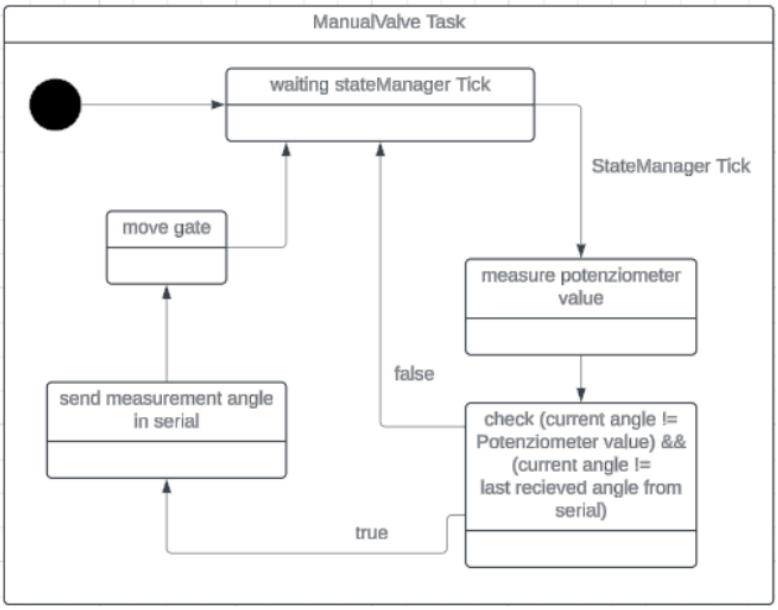

Figura 1.4: Manual Valve Task

River Monitoring Service

Questa parte è gestita da un server Java. Esso ha due file principali:

1. LogicImpl
2. DashboardServer

La comunicazione del *River Monitoring Service* con gli altri sistemi è gestita con tre tecnologie differenti:

1. **MQTT**: usata per la comunicazione con il *Water Level Monitoring*.
2. **HTTP**: sfruttata per la comunicazione con la *River Monitoring Dashboard*.
3. **Seriale**: utilizzata per la comunicazione con il *Water Channel Controller*.

2.1 LogicImpl

Questo contiene tutti i controlli e la logica per poter aggiornare correttamente la frequenza di campionamento, l'apertura della valvola e lo stato di allerta del sistema. Il suo compito principale è quello di aggiornare l'ambiente in base al livello dell'acqua, il quale viene monitorato tramite il *Water Level Monitoring*. L'ambiente a cui si fa riferimento consiste principalmente in tre valori:

- stato del sistema;
- frequenza;
- livello di apertura della valvola.

Questi valori vengono modificati su richiesta del server; i primi due vengono sempre modificati in base a soglie predeterminate del livello dell'acqua, mentre l'apertura della valvola viene riassegnata solamente se il sistema si trova in modalità manuale. Inoltre questa classe fornisce la possibilità di impostare il valore di apertura della valvola tramite il metodo *setValveLevel(int)* e un metodo *get* per ognuno dei tre valori sopracitati.

2.2 DashboardServer

Si occupa di instaurare le connessioni, inviare e ricevere i dati per poi passarli alla classe di logica. Il server ha il compito di rimanere sempre in attesa di messaggi da parte dell'ESP tramite MQTT e di Arduino tramite Seriale. Non appena riceve un messaggio dal *Water Level Monitoring* blocca l'ascolto della Seriale per aggiornare i valori presenti nella parte di logica. Il server si occupa di inviare un messaggio all'ESP solo nel caso in cui il nuovo valore della frequenza sia diverso rispetto al precedente, altrimenti non è richiesto alcun aggiornamento e quindi non si sprecano risorse. Per quanto riguarda l'aggiornamento del valore di apertura della valvola, questo viene effettuato leggendo il valore inviato dalla dashboard (modificato dall'utente manualmente) solo nel caso in cui il sistema si trovi nello stato manuale, altrimenti il valore viene costantemente modificato tramite i criteri presenti nella logica partendo dai valori letti dall'ESP. I dati vengono gestiti dal server come oggetti *JSON* grazie alla libreria *org.json.simple*.

River Monitoring Dashboard

Lo scopo di questo componente è quello di mostrare all'utente un'interfaccia grafica che consenta la gestione del sistema e l'analisi in tempo reale del suo stato. La dashboard ha due funzioni principalmente:

- visualizzare varie informazioni sul sistema:
  - lo stato del sistema (in base al livello dell'acqua)
  - il valore di apertura della valvola
  - un grafico raffigurante il livello dell'acqua in un intervallo di tempo
- modificare l'apertura della valvola se il sistema è in modalità manuale

La comunicazione con il server avviene tramite protocollo HTTP e soprattutto è la dashboard che attende aggiornamenti dal server, infatti quest'ultimo invia i dati solo quando è disponibile un messaggio sul topic a cui è iscritto. Una volta ricevuti gli aggiornamenti la dashboard scompone il messaggio, ricevuto in formato *json*, e lo utlizza per modificare la propria grafica, nello specifico per aggiornare il grafico e lo stato del sistema. Nel caso in cui invece il sistema sia in modalità manuale, la dashboard consente la modifica del livello di apertura della valvola; perciò in tal caso si occuperà di inviare, usando sempre il protocollo HTTP, il nuovo valore al server.

Water Level Monitoring

4.1 Schema elettrico

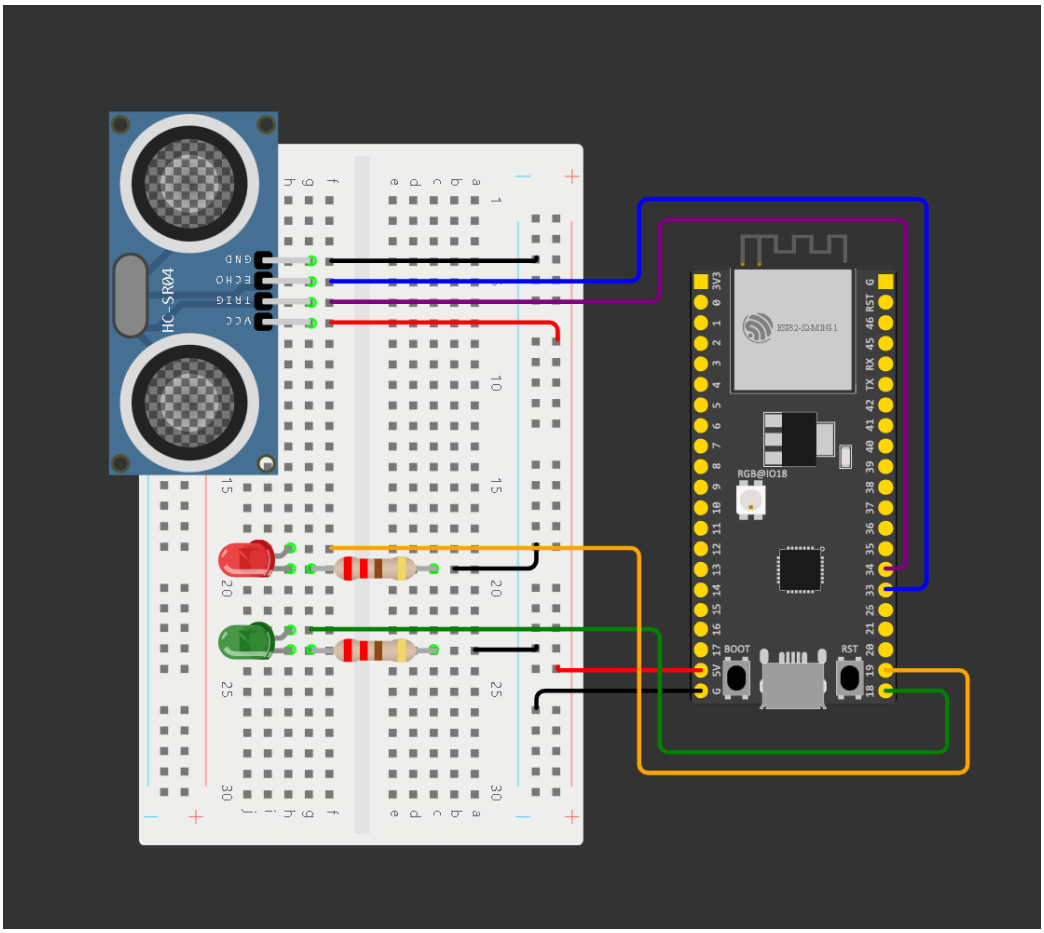

Figura 4.1: Schema Elettrico

4.2 Architettura

Questa parte del sistema si occupa di monitorare il livello dell'acqua e comunicarlo al server, in modo da tenerlo aggiornato. Per comunicare utilizza il protocollo MQTT. Nel setup vengono impostati i parametri necessari alla connessione wifi e viene stabilita la connessione con il broker MQTT, oltre ad inizializzare l'harware necessario. Dopodichè il sistema esegue un loop per effettuare le seguenti operazioni:

- controllo della stabilità della connessione con il broker, in caso negativo si provvede a ristabilirla;
- controllo della connessione wifi e visualizzazione della disponibilità tramite un led rosso e uno verde;
- misurazione del livello dell'acqua e conseguente invio al broker.

Il sistema agisce sia da publisher, in quanto comunica il livello dell'acqua sul topic *sensor/sonar/water_level*, sia da subscriber, infatti il server, dopo aver processato i dati ricevuti, comunica sul topic *sensor/sonar/frequency* la frequenza di campionamento che il sistema deve utilizzare per monitorare il livello.

Link video

Un video dimostrativo del sistema in funzione è visionabile qui.