Smart River Monitoring

Nicolò Monaldini

Febbraio 2024

1 Introduzione

Smart River Monitoring è un sistema composto da 4 componenti: River Monitoring Service, Water Level Monitoring, Water Channel Controller e River Monitoring Dashboard. River Monitoring Service è collegato tramite MQTT a Water Level Monitoring, tramite linea seriale a Water Channel Controller e tramite HTTP (in particolare, con l'utilizzo di un WebSocket) a River Monitoring Dashboard.

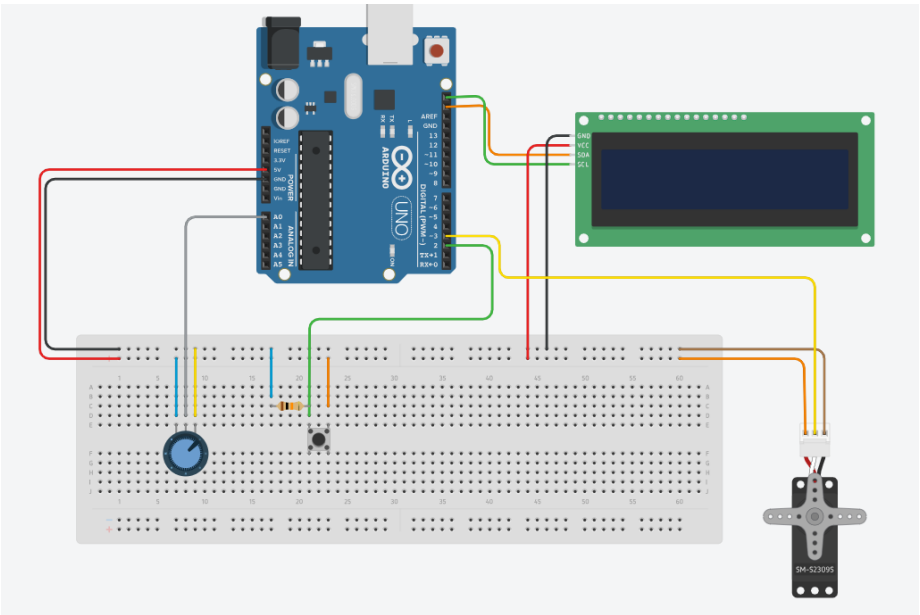

Figura 1: Schema del circuito Arduino

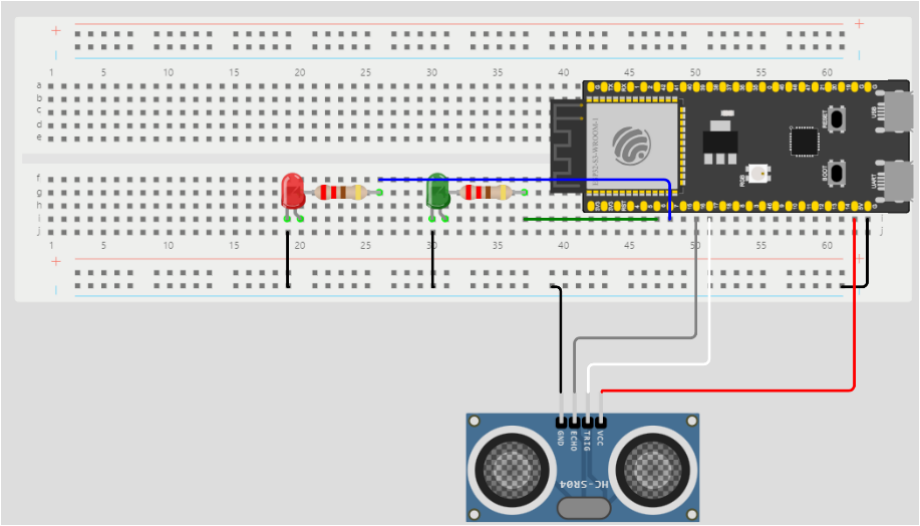

Figura 2: Schema del circuito ESP32

2 River Monitoring Service

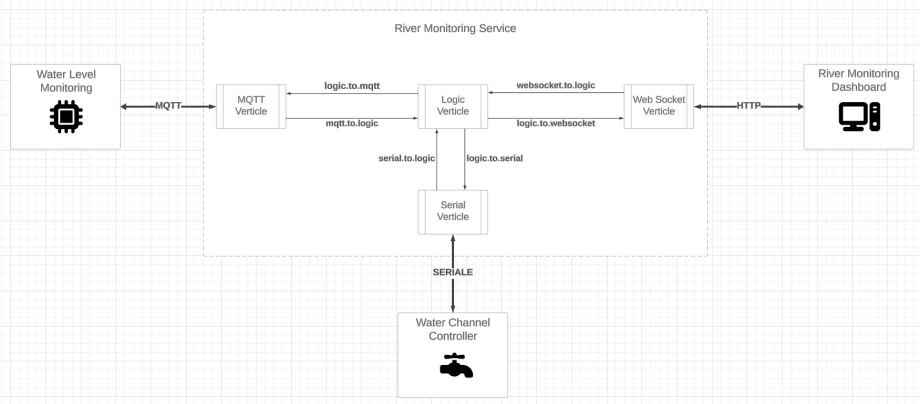

Figura 3: Schema dell'applicazione backend

L'applicazione backend è stata realizzata in Java, utilizzando il framework Vert.x. In particolare, sono stati utilizzati quattro Verticle. Tre di questi, WebSocketVerticle, MQTTVerticle e SerialVerticle, si occupano della comunicazione con i vari componenti del sistema; il loro compito è l'invio di oggetti JSON dall'applicazione backend ai vari componenti e viceversa. Il quarto Verticle, LogicVerticle, è quello in cui si processano i dati ricevuti e si istanziano gli oggetti JSON che saranno inviati. I vari Verticle comunicano tramite Event Bus.

3 Water Channel Controller

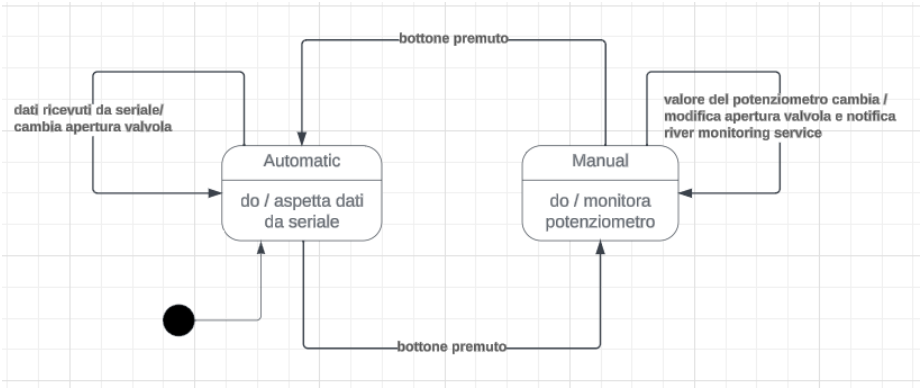

Figura 4: Diagramma di stato del task ManageValve

Water Channel Controller è composto da un task, ManageValveTask, con due stati: Manual e Automatic. Nello stato Manual il task rileva il valore da un potenziometro, cambia l'apertura della valvola di conseguenza e notifica l'applicazione backend. Nello stato Automatic il task legge i dati dalla seriale in attesa di ottenere il valore di apertura della valvola impostato dall'utente attraverso River Monitoring Dashboard e cambia l'apertura della valvola di conseguenza. Il cambio tra i due stati avviene tramite la pressione di un bottone.

4 Water Level Monitoring

Water Level Monitoring è composto da due task gestiti dallo scheduler di FreeRTOS.

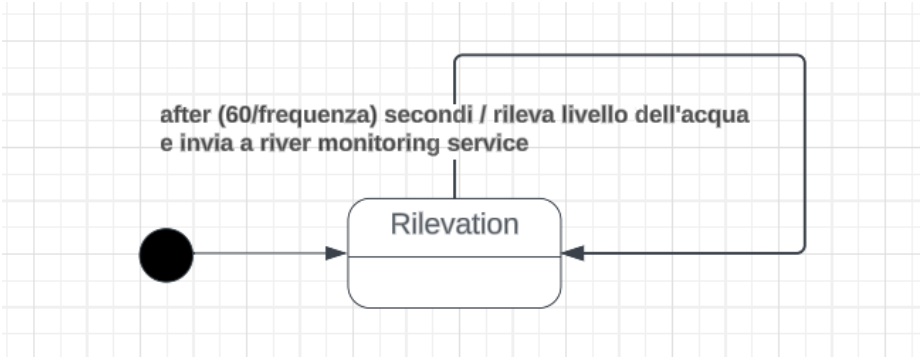

Figura 5: Diagramma di stato del task Rilevation

Il task Rilevation, composto da un solo stato, si occupa della rilevazione del livello dell'acqua. La rilevazione viene effettuata con una frequenza che dipende dal valore inviato dall'applicazione backend.

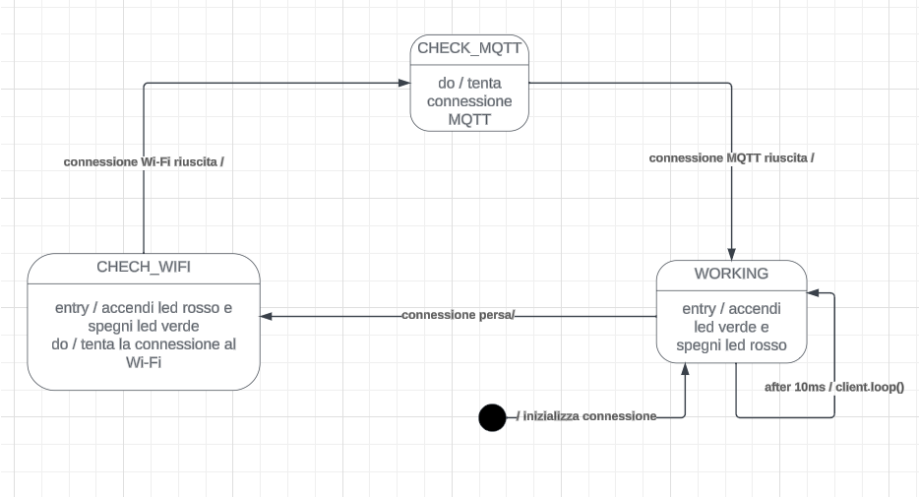

Figura 6: Diagramma di stato del task Connection

Il task Connection si occupa di gestire la connessione. Normalmente in stato Working, chiama la funzione client.loop() per assicurarsi il corretto funzionamento della comunicazione MQTT. In caso di problemi, vengono verificate le connessioni al Wi-Fi e al broker MQTT attraverso gli stati Check_Wifi e Check_MQTT. La frequenza di rilevazione viene modificata dalla funzione di callback di MQTT a seguito di un messaggio inviato dall'applicazione backend. Una variabile mutex è stata utilizzata per evitare race condition sulla variabile che contiene il valore attuale della frequenza.

5 River Monitoring Dashboard

Figura 7: Schermata di funzionamento della dashboard

River Monitoring Dashboard è una semplice web app implementata in JavaScript, che comunica con l'applicazione backend attraverso un WebSocket. La web app si occupa di ricevere i dati di rilevazione e mostrarli su un grafico implementato con la libreria ChartJS. Si occupa, inoltre, di manipolare la pagina web a seconda dei messaggi ricevuti dall'applicazione backend, ad esempio mostrando lo stato del fiume o non permettendo all'utente di impostare l'apertura della valvola in caso vi sia la modalità manuale attiva.