Smart Temperature Monitoring

Documentazione

Autori

Fattori Fabio, matricola 0001071463
Senni Mattia, matricola 0001079869
Tonelli Francesco, matricola 0001071531

Funzionamento del sistema

control-unit-backend

Scritto in Node.js con un server Express.js, il sistema utilizza le socket (libreria socket.io) per comunicare in tempo reale con la dashboard, inviando dati via WebSocket e ricevendo comandi tramite REST API. Per la comunicazione con l'ESP, impiega MQTT per trasmettere i valori delle frequenze di trasmissione e ricevere i dati sulle temperature registrate. Infine, sfrutta la comunicazione seriale con Arduino per inviare e ricevere dati e comandi relativi all'apertura della porta, trasmettere le temperature da visualizzare in modalità manuale e ricevere lo stato del sottosistema Arduino.

dashboard-frontend

L'applicazione è sviluppata in React (libreria JavaScript) con l'ausilio di Vite.js per il building e TypeScript per la tipizzazione del codice. Si occupa della visualizzazione dei dati ricevuti esclusivamente dal server backend, acquisiti tramite WebSocket e registrati nello stato centralizzato dell'applicazione. Al variare di quest'ultimo, l'interfaccia grafica si aggiorna automaticamente in base ai nuovi dati.
La dashboard gestisce anche la modalità MANUAL per il controllo della finestra, eseguendo prima una chiamata a una REST API per evitare conflitti tra lo stato di Arduino (che ha la priorità) e quello della dashboard. Inoltre, il passaggio da qualsiasi stato ad ALARM rende disponibile un'opzione per la regolazione del sistema, che esegue l'operazione correttiva tramite una chiamata a una REST API.
La pagina principale offre una panoramica sintetica dei dati ricevuti e consente di eseguire le azioni sopra descritte. La pagina "Temperatures" permette di visualizzare gli stessi dati in forma dettagliata, corredati da un grafico. Infine, la pagina "Window" mostra un grafico di tipo Gauge, rappresentando l'apertura della finestra in gradi.

window-controller

Il sottosistema Arduino implementa diversi pattern, tra cui un Controller per gestire la logica dell'applicazione, contenente tutti i modelli necessari. Inoltre, utilizza un Task Scheduler, dove un task si occupa della lettura dei dati ricevuti via seriale (temperatura e percentuale di apertura della finestra), mentre un altro gestisce la macchina a stati finiti time-based, responsabile della gestione dei due stati: AUTOMATIC e MANUAL.

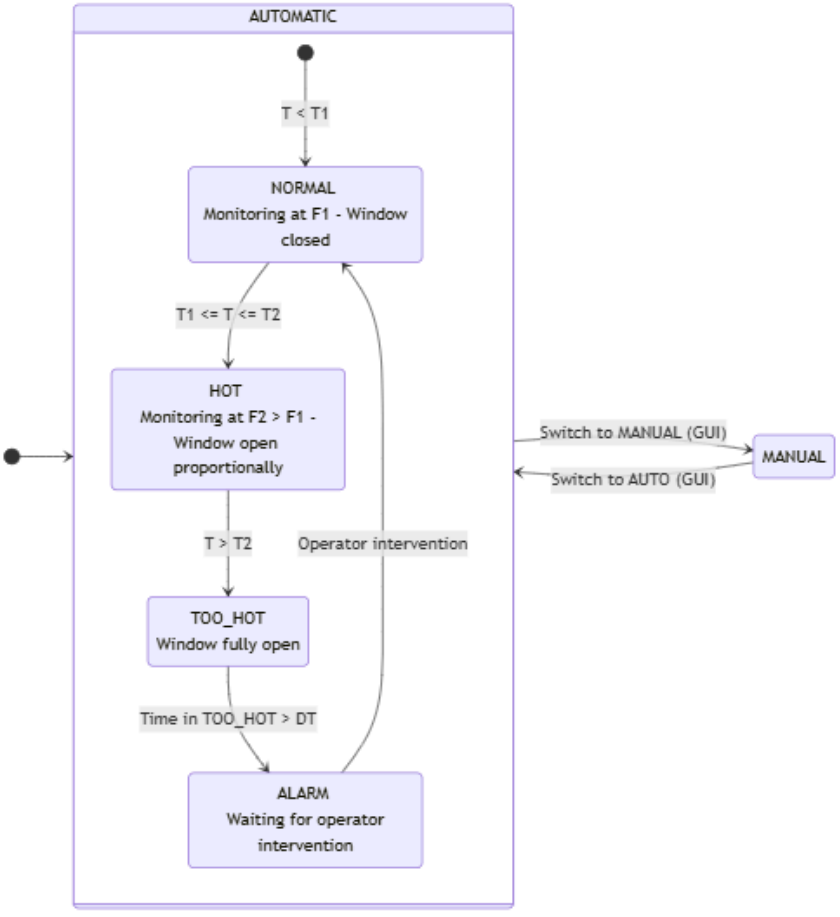
È presente anche un gestore di eventi, utile per eseguire codice specifico durante la transizione tra uno stato e l'altro. Infine, tutti i componenti di Arduino sono modellati tramite classi dedicate, garantendo una struttura modulare e organizzata.

temperature-monitoring-subsystem

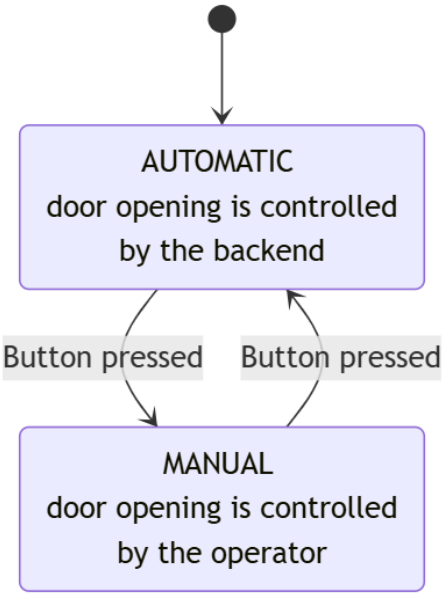
Il sistema implementa una macchina a stati finiti di piccole dimensioni, che gestisce i due stati di connessione: OK ed ERROR. Inoltre, utilizza un sistema di comunicazione basato su MQTT, operando su un server HiveMQ.
La macchina attende di ricevere il primo valore della frequenza di invio e, una volta ricevuto, inizia a trasmettere i dati della temperatura rilevati dal termometro sul medesimo topic. Alla ricezione di un nuovo valore di frequenza, il sistema aggiorna il proprio comportamento di conseguenza.
In caso di disconnessione dalla rete o dal server MQTT, il sistema tenta prima la riconnessione alla rete Wi-Fi (se necessario) e successivamente al server MQTT. Una volta ristabilita la connessione, la trasmissione riprende automaticamente seguendo l'ultima frequenza ricevuta.

Stati dei componenti

control-unit-backend & dashboard-frontend



window-controller



temperature-monitoring-subsystem