Relazione Elaborato #3

Sistemi Embedded and IOT

Corso di laurea in Ingegneria e Scienze Informatiche

Anno Accademico 2024/2025

Samuele Casadei
Matricola: 0001097772
Email: samuele.casadei3@studio.unibo.it

Michele Farneti
Matricola: 0001080677
Email: michele.farneti@studio.unibo.it

Prof. Alessandro Ricci
Università di Bologna

Maggio 2025

Indice

1 Introduzione

L'obiettivo di questo progetto è implementare un sistema IoT che controlli la temperatura in un ambiente chiuso e, basandosi su essa, regoli il livello di apertura di una finestra. Il sistema è composto da quattro sottosistemi:

- Sistema ESP per il controllo della temperatura
- Sistema di backend / Control Unit
- Sistema Arduino per la regolazione manuale della finestra
- Sistema di frontend / Dashboard

2 Specifiche

2.1 Temperature Monitoring Subsytem

Sistema responsabile per la continua rilevazione della temperatura, si occupa del campionamento e invio periodico della temperatura con una data frequenza F; F è determinata dallo stato attuale del sistema, che, a sua volta, è determinato dalla Control Unit. Un altro compito del sistema è mostrare che la comunicazione della Control Unit sta avvenendo correttamente tramite l'accensione di uno dei due LED:

- **Verde**: se la comunicazione sta avvenendo correttamente.
- **Rosso**: se avviene un errore di rete.

2.2 Window Controller Subsystem

Sistema responsabile per la regolazione della finestra tramite il controllo di un motore; lo stato della finestra è rappresentato in percentuale di apertura (0% chiuso, 100% aperto).
Il livello di apertura della finestra, di norma, è determinato dallo stato del sistema, determinato dalla Control Unit.
Tramite un pulsante è possibile cambiare la regolazione del livello di apertura a manuale, in questo modo si può aggiustare il livello tramite la regolazione di un potenziometro. Se il sistema si trova in modalità manuale, invece, una pressione del pulsante lo porterà in modalità automatica.
Un ulteriore compito del sistema è illustrare le informazioni dell'ambiente tramite un display LCD.

2.3 Control Unit Backend

Sistema responsabile della memorizzazione degli ultimi N campioni di temperatura raccolti, fornendo valori di temperatura media, minima e massima in un periodo. Un altro compito fondamentale del sistema è quello di determinare il livello di apertura della finestra (se in modalità automatica) basandosi sulla temperatura corrente; di conseguenza, la Control Unit dovrà fungere da punto cardine della comunicazione tra i sottosistemi descritti.
Nella Control Unit sono definiti gli stati del sistema:

- **NORMAL**: stato in cui la temperatura del sistema risulta nella media, la frequenza di campionamento è più bassa.
- **HOT**: stato in cui la temperatura del sistema risulta sopra la media, la frequenza di campionamento aumenta e il livello di apertura della finestra deve essere regolato in base alla temperatura (se in modalità automatica.
- **TOO HOT**: stato in cui la temperatura supera i valori attesi, la finestra viene aperta forzatamente (se in modalità automatica).
- **ALARM**: stato derivante da una permanenza prolungata nello stato TOO HOT, il sistema non risponde a comandi manuali e attende una risulozione dello stato dalla Dashboard.

2.4 Dashboard Frontend

Sistema responsabile della visualizzazione nel dettaglio dello stato del sistema, le informazioni mostrate sono:

- Un grafico rappresentante le ultime N temperature individuate.
- Lo stato attuale del sistema
- La temperatura media, minima e massima
- L'attuale livello di apertura della finestra

Oltre a questo è necessario che la dashboard permetta all'utilizzatore di cambiare modalità di utilizzo e impostare manualmente il livello di apertura della finestra, oltre a disattivare lo stato d'allarme del sistema.

3 Implementazione

3.1 Descrizione delle soluzioni proposte

3.1.1 Temperature Monitoring Subsystem

Una volta esaminato il dominio del problema, è stato possibile suddividere il flusso d'esecuzione in due task, il cui scheduling sarà gestito dal sistema operativo **FreeRTOS**:

- **Temperature Monitoring Task :** Qualora lo stato della connessione con la control unit sia ok, esegue la misurazione della temperatura del sistema utilizzando il **sensore di temperatura LM35**. Il sampling avviene periodicamente con una frequenza che muta volta per volta determinata dalla control unit.
- **Connection Check Task :** Controlla periodicamente l'invio di un messaggio da parte della control unit su un topic apposito, atto a segnalare il corretto funzionamento delle comunicazioni nel sistema.

Gestione delle connessioni mediante MQTT : La parte della comunicazione basata su protocollo **MQTT**, lato ESP32S3, è stata realizzata utilizzando alcune librerie fondamentali disponibili per l'ambiente di sviluppo *Arduino ESP32*. In particolare, sono state impiegate le seguenti librerie:

```
// Libreria per la connessione Wi-Fi
#include <WiFi.h>

// Libreria per la gestione di connessioni sicure (TLS/SSL)
#include <WiFiClientSecure.h>

// Libreria per la gestione del protocollo MQTT
#include <PubSubClient.h>
```

La libreria `WiFi.h` consente di connettere il dispositivo alla rete Wi-Fi, mentre `WiFiClientSecure.h` permette di instaurare connessioni sicure attraverso il protocollo TLS. Infine, `PubSubClient.h` è utilizzata per la pubblicazione e la sottoscrizione di messaggi secondo il protocollo MQTT, facilitando così la comunicazione con un broker remoto. La logica della comunicazione viene interamente incapsulata da oggetti che permetteranno alle task di compiere azioni inerenti alle comunicazioni attraverso una singola interfaccia.

Utilizzo di mutex : L'interfaccia relativa alla comunicazione verrà acceduta da entrambe le task in maniera concorrente, viene dunque utilizzata una mutex per gestione dell'esclusività temporale degli accessi. In particolare, viene utilizzato il `SemaphoreHandle_t`, tipo di dato utilizzato in FreeRTOS

Broker MQTT : Il broker utilizzato per implementare la comunicazione è stato HiveMQ, utilizzato in versione cloud. La connessione avviene per mezzo di credenziali. I topic sui cui sono pubblicati i messaggi sono 3, uno per le temperature, uno per i periodi di campionamento ed uno per i messaggi segnalazione dello stato della comunicazione.

Macchina a stati finiti : Per questo sottosistema non è stata implementata un'architettura a stati. Le due task vengono gestite parallelamente dal sistema operativo **FreeRTOS** e l'accesso al modulo di comunicazione viene protetto da una mutex.

3.1.2 Dashboard Frontend

Il frontend dell'applicativo, che permette all'utente di monitorare ed interagire col sistema, è stato realizzato mediante l'utilizzo di teconolgie web, quali HTML, JavaScript e CSS.

Comunicazione via HTTP : La comunicazione con la control unit viene effetuata, come da specifiche, sfruttando il protocollo **HTTP**. In particolare, La pagina viene aggiornata dinamicamente a intervalli di 100 millisecondi mediante richieste **AJAX** asincrone di tipo **GET** risovlte all'endpoint `../api/data`. La control unit risponderà a queste chiamate restituendo dei file in formato **JSON** che conterranno le specifiche monitorate del sistema, in particolare verranno visualizzate:

- Stato del sistema.
- Modalità di controllo.
- Temperatura media, minima e massima giornaliera.
- Livello di apertura della finestra.
- Ultime N misurazioni dell temperatura, inclusive di timestamp.

L'interfaccia web disporrà inoltre di componenti per gli input dell'utente. IN particolare sarà presente un pulsante per effettuare uno switch della modalità di controllo ed uno slider per modificare il livello di apertura della finestra. Sarà possibile interagire con quest'ultimo solo qualora il sistema si troverà in modalità di controllo automatica. Sarà infine presente un pulsante per indicare la risoluzione dei problemi, che apparirà solo qualora il pulsante entrerà in stato eccezionale d'allarme e permetterà al sistema di tornare ad uno stato ordinario. L'interazione con tali componenti causerà l'invio di richieste **POST** rivolte all'endpoint `../api/controls` fornito dalla Control Unit, che si occuperà di verificare la legalità dell'azione e modificherà il proprio stato di conseguenza.

Gestione slider : Al fine di non intasare il sistema a causa di un elevato numero di chiamate asincrone dovute alla modifica dello stato dello slider, è stata realizzata una funzione di debouncing dedita all'inoltro della richiesta solo dopo una breve attesa dal termine del movimento del thumb.

Visualizzazione delle misurazioni : La tecnologia utilizzata per un'efficace visualizzazione delle misurazioni è stata la libreria **D3js**. Mediante quest'ultima è stato possibile realizzare un grafico dinamicamente aggiornato con i nuovi sample ricevuti dalla control unit per visualizzare l'andamento della temperatura. Sono inoltre rappresentate sul grafico le soglie di temperatura T1 e T2 su cui il sistema gestisce il proprio stato in modalità automatica.

Utilizzo di NGROK : Per facilitare la comunicazione tra la control unit e la dashboard web è stato utilizzato **Ngrok**. Siamo così riusciti a rendere raggiungibili le informazioni da un indirizzo pubblico superando le limitazioni date dal non utilizzo di un IP pubblico e rendendo dunque l'applicativo frontend semplicemente utilizzabile da browser.

3.1.3 Window Controller Subsytem

I compiti del Window Controller possono essere suddivisi in due *sotto-compiti*:

- Aggiustare il livello di apertura della finestra in base alle informazioni a disposizione (che siano input da dashboard o manuali).
- Rilevare se avviene un cambio di modalità manuale dalla pressione del pulsante.
- Richiedere informazioni alla Control Unit per mostrarle sul display LCD.

Essendo che aggiustare il livello di apertura della finestra e illustrare le informazioni sull'ambiente nel display LCD richiedono entrambi di ottenere informazioni dalla Control Unit, ho deciso di raggrupparli in un singolo task, in modo da interrogare la Control Unit una sola volta. Mentre la rilevazione della pressione del bottone è associata ad un task a sé stante.
I task definiti sono:

- **WindowRegTask**: si occupa di eseguire il caricamento dei dati inviati sul seriale, e di adattare lo stato del subsystem in base ai dati ricevuti (cambio di modalità, temperatura, livello di apertura);
- **ControlPanelTask**: si occupa di leggere una pressione del pulsante e, conseguentemente, inviare il segnale di cambio di modalità alla Control Unit.

Per evitare la perdita di messaggi in ricezioni veloci, il sistema di messaggistica lato Arduino è stato modificato, implementando una coda di messaggi di lunghezza statica (10), che incapsula al suo interno un array di oggetti **Msg**.

3.1.4 Control Unit Backend

La Control Unit, unità centrale del sistema, è stata realizzata in Java sfruttando il toolkit **Vert.x**. In particolare, è stata suddivisa in quattro moduli: tre dedicati alla comunicazione con i sottosistemi e uno principale per la gestione della logica applicativa. Ogni modulo è stato implementato come un **Vert.x Verticle**, e la comunicazione tra i moduli avviene tramite il **Vert.x Event Bus**, che consente uno scambio di messaggi asincrono ed efficiente all'interno del sistema.

MQTT Communication Module : È stato realizzato sfruttando l'**Mqtt Client** fornito da Vert.x. Una volta effettuata la configurazione mediante la lettura di un file CSV, contenente le credenziali per la connessione al broker MQTT (nel nostro caso HiveMQ), è incaricato di gestire i messaggi ricevuti sul topic in cui il subsystem di monitoring della temperatura caricherà le rilevazioni, inoltrandoli poi al modulo principale. Avrà poi il compito di pubblicare, quando richiesto dalla main unit, i nuovi periodi da utilizzare per intervallare il sampling della temperatura. Infine, comunicherà periodicamente un messaggio per segnalare il corretto funzionamento del collegamento via MQTT. Quest'ultima azione permetterà al sistema di verificare la mantenuta connettività anche in caso di utilizzo del sistema con frequenze di campionamento molto basse.

HTTP Communication Module : Modulo del backend incaricato di comunicare periodicamente alla dashboard le informazioni sullo stato del sistema che andranno visualizzate, nonché di gestire la ricezione di comandi da parte della stessa. Viene utilizzato **l'HTTPServer di Vertex**, in combinazione con il suo Router, inizialmente configurato mediante un **CorsHandler** per gestire eventuali browser-warning mandati da ngrok e segnalare gli header consentiti per i messaggi. Ad ogni aggiornamento dello stato del sistema, il modulo mantiene al proprio interno una copia aggiornata dei valori interessati, che la dashboard potrà ottenere periodicamente, in base al proprio refresh rate, per mezzo di richieste GET. L'HTTPServer avrà inoltre un handler per la gestione dei messaggi POST che permetteranno, in base al campo messageType, di gestire le diverse azioni effettuabili da dashboard inoltrandole alla main unit.

Serial Line Communication Module : Modulo del backend incaricato di trasferire le informazioni ricevute dalla main unit sulla linea seriale, con un'opportuna codifica. Anch'esso utilizza **Vertex** per ricevere messaggi sui topic di interesse. Legge periodicamente la linea seriale per caricare i messaggi ricevuti dal Window Controller, parsarli e inviarli alla main unit, sempre tramite l'uso Vertex topic appositi.

Main unit : Modulo principale del backend con la funzione di coordinatore dei moduli di comunicazione e di gestore della logica di sistema. Alla ricezione di ogni nuovo sample, procede a memorizzarlo e, conseguentemente, a memorizzare le statistiche aggiornate, modificando lo stato del sistema. Le nuove informazioni vengono dunque comunicate ai sottosistemi, passando attraverso i moduli di comunicazione ai quali comunicherà per mezzo degli event bus di **Vertx**.

Macchina a stati finiti : Di seguito viene rappresentata la macchina a stati finiti del sottosistema :

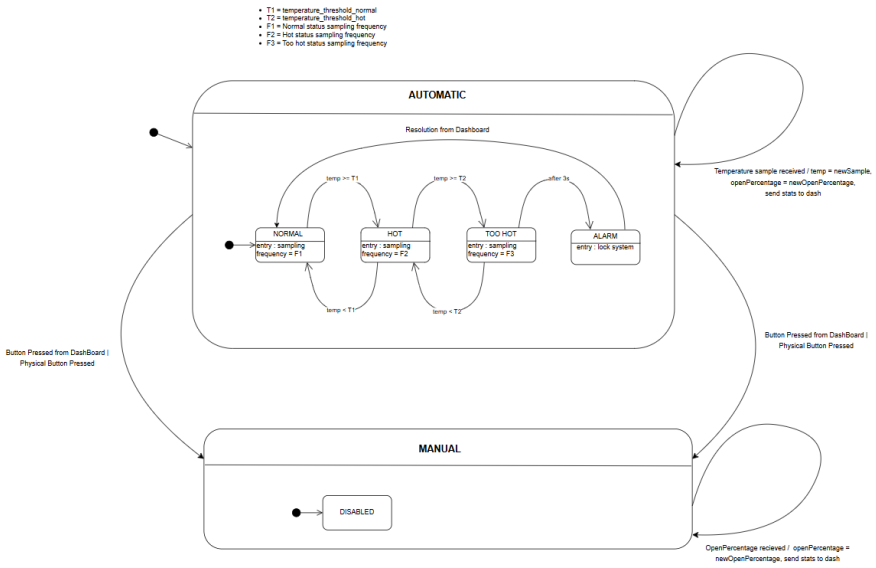

Figura 1: Diagramma a stati finiti della Control Unit.

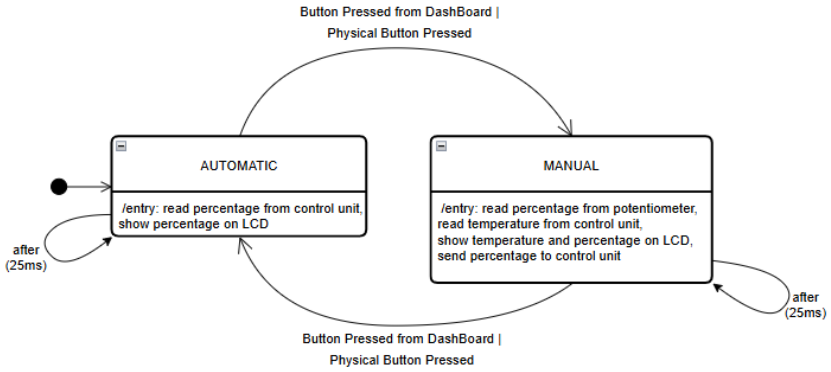

Figura 2: Diagramma a stati finiti del Window Controller.

3.2 Schema implementato (WOKWI)

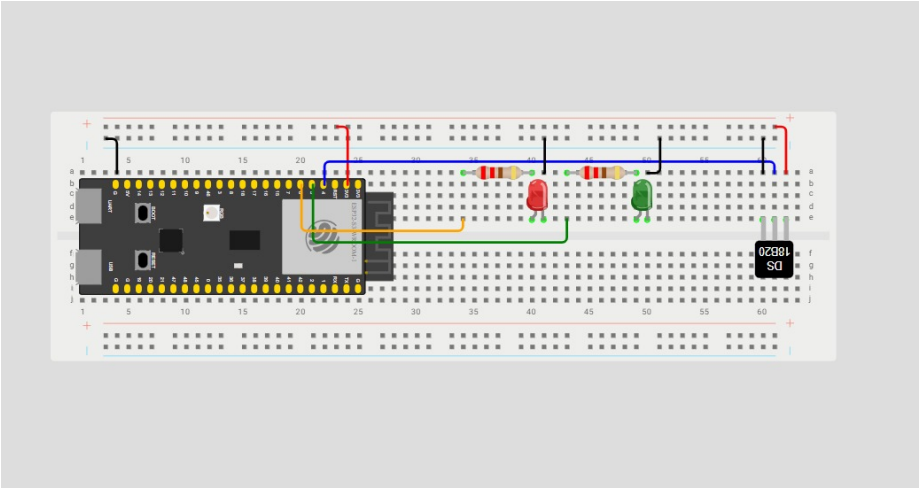

Figura 3: Circuito del temperature monitoring subsystem.

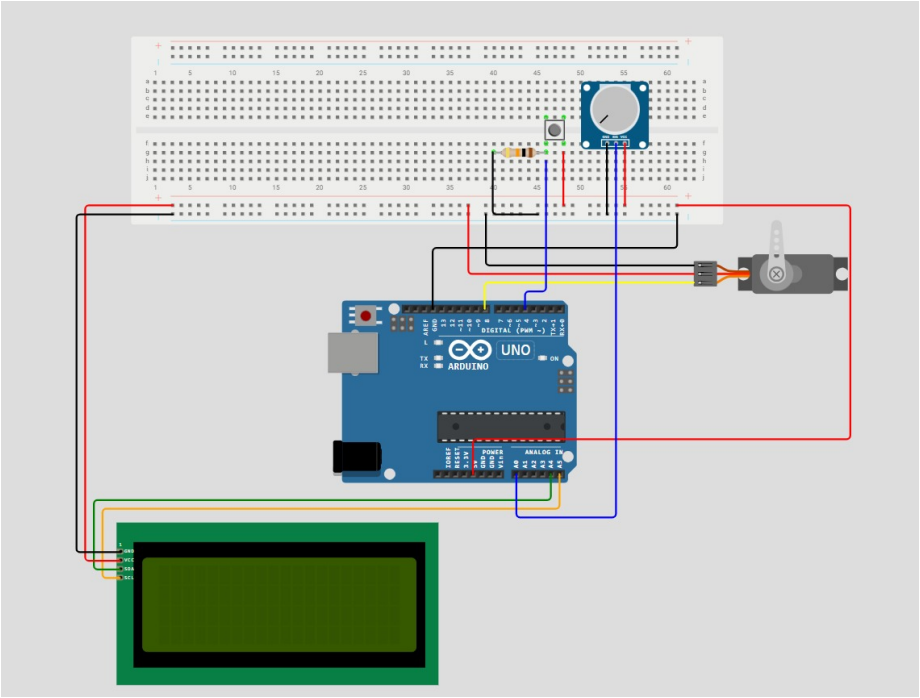

Figura 4: Circuito del window controller.

4 Guida all'uso

4.1 Istruzioni per l'utilizzo del Temperature Monitroing Subsystem

1. **Avvio del monitor seriale**
   Avviare il monitor seriale con **baud rate 115200** per visualizzare i messaggi di log.

2. **Configurazione dei PIN**
   Configurare i PIN utilizzati dall'ESP32S3 modificando il file:

   `temperature-monitoring-subsytem/src/settings/HWInterfaces.h`

3. **Avvio del broker MQTT**
   Avviare un broker MQTT. In questo caso, è possibile utilizzare un server gratuito fornito dal servizio cloud **HiveMQ**.

4. **Connessione Wi-Fi**
   Avviare la connessione Wi-Fi a cui collegare l'ESP32S3.

5. **Configurazione delle credenziali**
   Aprire e modificare il file:

   `temperature-monitoring-subsytem/src/main.cpp`

   Impostare i seguenti valori:

```
const char *wifi_ssid = "<YOUR_WIFI_SSID>";
const char *wifi_password = "<YOUR_WIFI_PASSWORD>";
const char *mqtt_server = "<YOUR_MQTT_SERVER_ADDRESS>";
const char *mqtt_username = "<YOUR_MQTT_USERNAME>";
const char *mqtt_password = "<YOUR_MQTT_PASSWORD>";
const int mqtt_port = <YOUR_MQTT_PORT>;
const char *temperatures_topic = "<TEMPERATURES_TOPIC>";
const char *periods_topic = "<PERIODS_TOPIC>";
const char *connection_topic = "<CONNECTION_TOPIC>";
```

6. **Caricamento del progetto**
   Utilizzando **PlatformIO**, caricare sull'ESP32S3 il progetto presente nella cartella:

   `temperature-monitoring-subsytem`

4.2 Istruzioni per l'utilizzo del Dashboard Frontend

1. **Avvio del frontend**
   Aprire nel browser il sottoprogetto `dashboard-frontend`.
   Per impostazione predefinita, lo scambio di informazioni con il backend avviene tramite `localhost`.

2. **Accesso pubblico via ngrok (opzionale)**
   È possibile esporre il servizio della Control Unit pubblicamente in modo sicuro tramite **ngrok**.

   (a) Installare ngrok, se non già presente (`https://ngrok.com`).

   (b) Avviare ngrok in ascolto sulla porta 8080, usata dal modulo backend della Control Unit:

```
ngrok http 8080
```

   (c) Copiare l'indirizzo HTTPS generato da ngrok.

   (d) Assegnare l'indirizzo alla costante `CONTROL_UNIT_ADDRESS` nel file:

   `dashboard-frontend/js/main.js`

   (e) Esempio di configurazione nel file JavaScript:

```
const CONTROL_UNIT_ADDRESS = "https://1234-xx-yy.
    ngrok.io";
```

3. **Avvio del progetto**
   Dopo la configurazione, aprire nel browser la dashboard con:

   `dashboard-frontend/index.html`

4.3 Istruzioni per l'utilizzo del Control Unit Backend

1. **Configurazione del file CSV**
   Accedere al file:

   `control-unit-backend/src/main/resources/MQTTconfig.csv`

   Configurare i parametri per la connessione al broker MQTT (come fatto per il *temperature-monitoring-subsystem*).

   Inserire i dati nella **seconda riga** del file CSV nel seguente formato:

```
broker_address,port,username,password,topic_temperature,
    topic_period,topic_connection
```

2. **Avvio dell'applicativo Java**
   Avviare il `main` dell'applicativo Java.

4.4 Istruzioni per l'utilizzo del Window Controller Subsystem

1. **Configurazione dei PIN**
   Configurare i PIN utilizzati da Arduino tramite la modifica del file:

   `window-controller/src/settings/HwInterfaces.h`

2. **Caricamento del progetto**
   Utilizzando **PlatformIO**, caricare su Arduino il progetto presente nella cartella:

   `window-controller`

Riferimenti bibliografici

[1] HiveMQ Cloud MQTT Broker, `https://www.hivemq.com/mqtt-cloud-broker/`

[2] Ngrok - Secure Tunnels to Localhost, `https://ngrok.com/`

[3] Eclipse Vert.x Toolkit, `https://vertx.io/`