Assignment nr.3
Towards IoT: Smart Temperature Monitoring

Luca Casadei - 0001069237
Francesco Pazzaglia - 0001077423

Last modified: March 1, 2025

Contents

1 Introduction

The third assignment represents a smart temperature monitoring system by using the appropriate communication protocols between different devices with each other. In particular:

- **Temperature Monitoring Subsystem** (ESP32): It continuously sends temperature data and communicates via MQTT and using LEDs that describe the status of the connection;
- **Control Unit Backend** (Java/Vert.x): It acts as the brain of the system by managing the state, communicating the temperature to other subsystems by managing their coordination;
- **Window Controller** (Arduino): Manages the physical window (the servomotor) optionally through the use of the potentiometer(if in MANUAL mode) and the operator interface with the LCD display;
- **Dashboard** (Python/Tkinter): Provides GUI for remote monitoring and control;

The system has two operating modes: *AUTOMATIC* with window control based on temperature and *MANUAL* for direct operator input. Communication uses MQTT for thermometer data, HTTP for dashboard updates, and serial protocol for window control.

2 Temperature Monitoring Subsystem

The Temperature Monitoring subsystem is based on ESP32 and is responsible for continuously sending the temperature of the environment to the Control Unit Backend. It uses a temperature sensor to acquire data and sends it to the Control Unit via the MQTT protocol.

The subsystem continuously collects temperature readings from the LM35 temperature sensor. Some fault tolerance is ensured by the choice of including a secondary broker, meaning that if the primary MQTT broker becomes unreachable, after a certain number of attempts(by default 3), the system can switch to a secondary broker, maintaining uninterrupted operation, an essential feature in critical IoT applications.

In addition, the use of indicator LEDs adds an additional layer of feedback as they immediately inform the operator about the network status, thus whether it is connected or if there have been connection problems by the SoC.

3 Control Unit Backend

The Control Unit is created in Java, everything is done through the use of Vert.x, a framework that simplifies the management of threads and their interaction, more specifically, each thread ("verticle") represents an interface with other devices, for example, there is the serial verticle that sends and receives data from the window controller

(Arduino) through the serial line, the HTTP one to send and receive data from the user dashboard, the MQTT verticle that exchanges information with the temperature monitoring system (ESP32) and lastly the most important one that manages all the logic of the whole system, the control unit verticle, which has the the task of checking the temperatures received from the ESP32 and use them to update the state of the machine, then it needs to send all the updated data to the various verticles, that's why we needed a solution that let the verticles communicate between one-another without creating race conditions, so we decided to use the "event bus" that Vert.x provides to send data in an event-based way between verticles, preventing every major trouble that comes with thread programming. The control unit has also the task of keeping memory of the last N received temperatures (we chose 50), and with that collection it has to compute the average temperature, the minimum and the maximum value that has been received. To do so, we decided to use java stream operations.

4 Window Controller

The Window Controller subsystem is implemented on an Arduino UNO board and is responsible for managing the physical opening of the window. The actuator is controlled by a servo motor, which can adjust the window opening from 0% (window closed) to 100% (window fully open, corresponding to 90°).

This subsystem also includes a push-button to switch between automatic and manual modes and a potentiometer for manual fine controlling(active only in MANUAL mode otherwise window opening is dictated automatically by the Control Unit). Also through the use of an LCD display we print the mode type, its status, and the updated temperature on the screen.

5 Dashboard Frontend

The Frontend Dashboard is a GUI for remote monitoring and control of the system. Developed in Python with Tkinter, it uses the Model-View-Controller (MVC) architecture to separate data management, control logic, and presentation. It periodically makes HTTP calls in order to gather and send data to the Control Unit. Through the dashboard, the operator has several functions available including: displaying a graph with temperatures, the average of the last N temperatures, changing the mode from AUTOMATIC to MANUAL, and possibly through a slider changing the opening of the Window.