Assignment 03 - "Smart Temperature Monitoring"

"Embedded Systems e Internet of Things" final report

Lorenzo Cinelli

April 11, 2025

Contents

1 Analysis

1.1 Description

The required software is a Smart Temperature Monitoring system.
It is made of of a temperature sensor that periodically sends the temperature sampled to a control unit. The control unit is in charge of deciding the percentage of the window opening depending on the temperature - in case the window is in *AUTOMATIC MODE*.
If the window is in *MANUAL MODE* it can be controlled manually by an user through the user panel, formed by a button to switch the window mode, a potentiometer to manually set the opening of the window, and an LCD screen that displays informations.
The whole system can be controlled by a dashboard which shows the current values of the system and tracks the temperature of the environment. An operator, by pressing a button on the dashboard, is able to restore the system in case of high temperature for prolonged time, which sets the system in an alarm state. From the dashboard, it is possible to switch the window mode and also to change the opening of the window.

Here is the complete description of the assignment.

1.2 Domain Model

The system is composed by four subsystems.
The control unit is the back-end of the system in charge of managing and coordinating the whole system.
It establishes the frequency of temperature sampling of the temperature sensor, with whom communicates via the `mqtt` protocol.
The control unit, based on the temperature, decides the window opening percentage if the window is in *AUTOMATIC MODE*. It communicates with the window subsystem via serial line.

The temperature subsystem has to sample periodically the temperature and to send it to the control unit. In case of connection problems a red led should turn on, otherwise, in case of connection working, a green led should turn on.

The window subsystem controls the window and has a user panel, from where an user can switch from *AUTOMATIC* to *MANUAL* mode and vice versa, it can set the opening of the window when it's in *MANUAL MODE* and it can show some basic information through an lcd display.

The dashboard is the front-end of the system and it enables the operators to remotely visualize the data and to interact with the system via the `http` protocol.
It visualizes the current data of the system, including the temperature, the state, the window mode and the window opening.
It also shows a graph representing the temperature history, and it highlights the maximum, minimum and average temperature measured.
The other main function of the dashboard is to interact with the system. It can manage a temperature alarm and it can switch window mode, setting the opening of the window too.

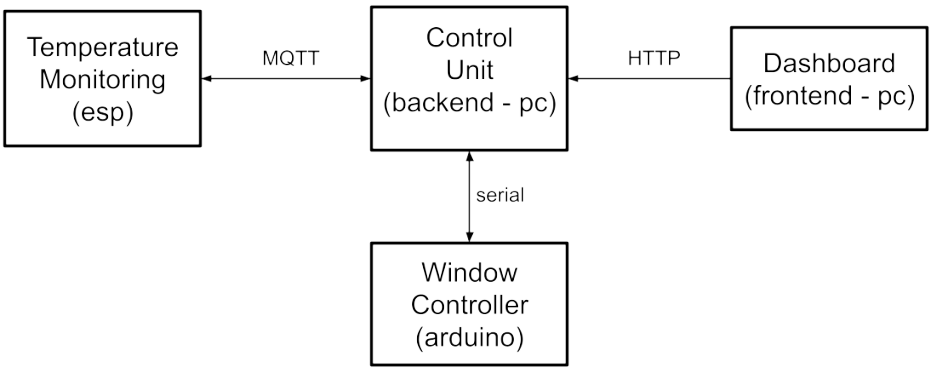

Figure 1.1: Smart Temperature Monitoring

1.3 Requirements

- The control unit and the temperature sensor have to communicate via the `mqtt` protocol.
- The control unit, the window and its user panel have to communicate via `serial line`.

- The control unit and the dashboard have to communicate via the `http` protocol.
- The control unit stores the current data of the entire system and tracks the temperature by storing the last N measurements, the maximum, the minimum and the average sampled.
- The temperature sensor must sample and send to the control unit the temperature at a frequency established by the control unit based on the current temperature state (*NORMAL, HOT, TOO_HOT, ALARM*).
- The temperature state is established by the temperature and duration of it.
- The window opening is decided by the control unit when the window is in *AUTOMATIC MODE*. It depends by the temperature state and the temperature itself.
- The window opening, when the window is in *MANUAL MODE*, is set manually by the user through the user panel or by an operator through the dashboard.
- The window mode can be switched by pressing a button in the user panel or in the dashboard.
- The dashboard displays the data stored by the control unit, in particular the temperature history, which is displayed in a chart.
- When the temperature is too high for too long the system goes in *ALARM* state. To return in *NORMAL* state an operator has to fix the issue from the dashboard.

2 Design

2.1 Architecture

The system is composed by four modules: the control unit, the temperature monitoring, the window controller and the dashboard. The control unit is the centre of the system, able to exchange information between itself and any other module.

The temperature monitoring is a task-based architecture with only one task, which is a synchronous finite state machine. Despite having only one task produces overhead, it offers great flexibility and modularity.

The window controller is a task-based architecture. Tasks are concurrent and they are communication task via serial line and window controlling task. Each task interacts with the model that offers encapsulation to physical devices and subsystem data.

The dashboard is based on an MVC architectural pattern. The view is formed by two modules: the active GUI and the communication module.
The first one is in charge of displaying updated data in the GUI, while the second submodule periodically sends an HTTP request to the control unit to obtain updated data and eventually send commands.

The control unit is based on an MVC architectural pattern. The view is formed by the communication modules that communicates with the other three subsystem.
The control unit can be seen as the central server of the system, where each one of the communication tasks remains in a listening state, ready to process and to respond to an upcoming message.

2.2 Detailed Design

Temperature Subsystem

The task in the Temperature Monitoring subsystem sends periodically the current temperature of the environment via the `mqtt` protocol. The period is sent via `mqtt` from the control unit.
The task is described by a `synchronous finite state machine`.

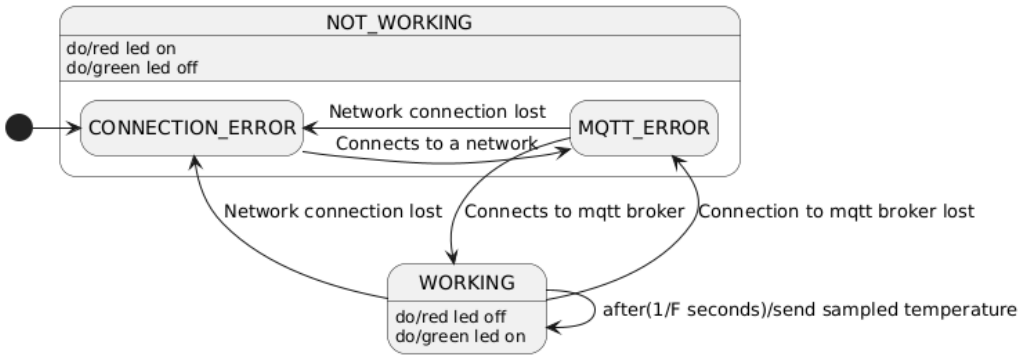

Figure 2.1: Temperature Monitoring Connection State

- NOT_WORKING: in this state the sensor cannot communicate with the via the `mqtt` broker and, consequently, with the Control Unit. To highlight the connection problem a red led turns on.
- CONNECTION_ERROR: it's the initial state. It means that the sensor couldn't connect to any network. Each time the task is executed it will try to connect to the designated network.
- MQTT_ERROR: in this state the sensor is connected to a network but is not connected to the via the `mqtt` broker. Each time the task is executed it will try to connect to the broker.
- WORKING: the sensor can communicate with the Control Unit. To signal that the connection is working a green led turns on. Each time the task is executed the sensor samples the current temperature and send it to the Control Unit. Based on the temperature received, the Control Unit replies with the frequency of temperature sampling.

Window Subsystem

The Window Controller has two tasks: the communication task, that sends periodically the subsystem data to the Control Unit and reads any reply from it, and the window task, which deals with the window mode and sets the opening of the window, based on the Control Unit value if the window is in *AUTOMATIC_MODE* or on the percentage set by the user or by operator if the window is in *MANUAL_MODE*.
The window task is described by a `synchronous finite state machine`.

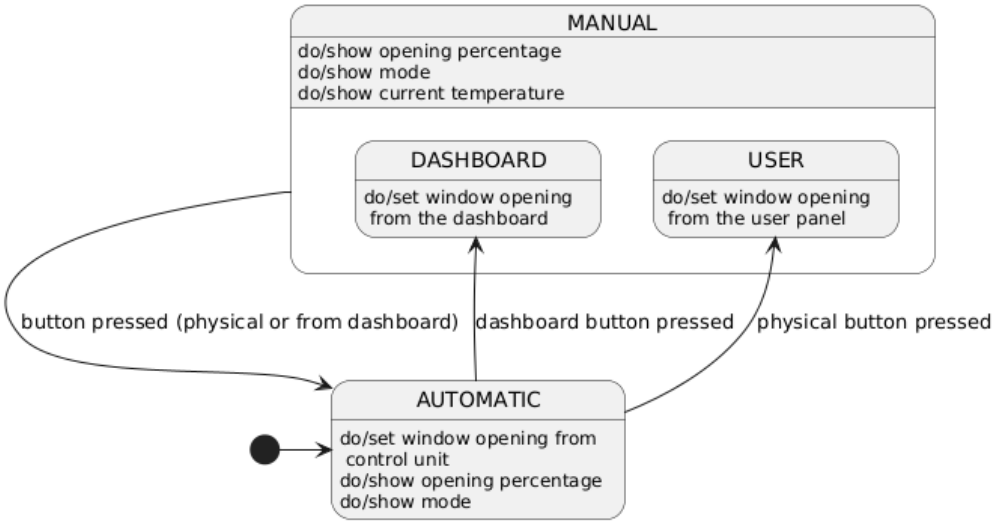

Figure 2.2: Window Mode

- AUTOMATIC: it's the initial mode of the window. In this mode the opening of the window is established by the Control Unit depending on the current temperature and its state.
- MANUAL: in this mode the window opening is established by an user or an operator, depending on where the command came from.
- USER: in this mode an user sets the opening of the window by the provided user panel.
- DASHBOARD: in this mode an operator sets the opening of the window by the dashboard.

Control Unit Subsystem

The Control Unit is designed to store the data of the system, to compute the frequency of temperature sampling and the opening of the window, as well as managing communication between itself and the rest of the system.
The frequency of sampling and the opening of the window are calculated by the temperature state, determined by the current temperature.
The temperature task is described by a `synchronous finite state machine`.

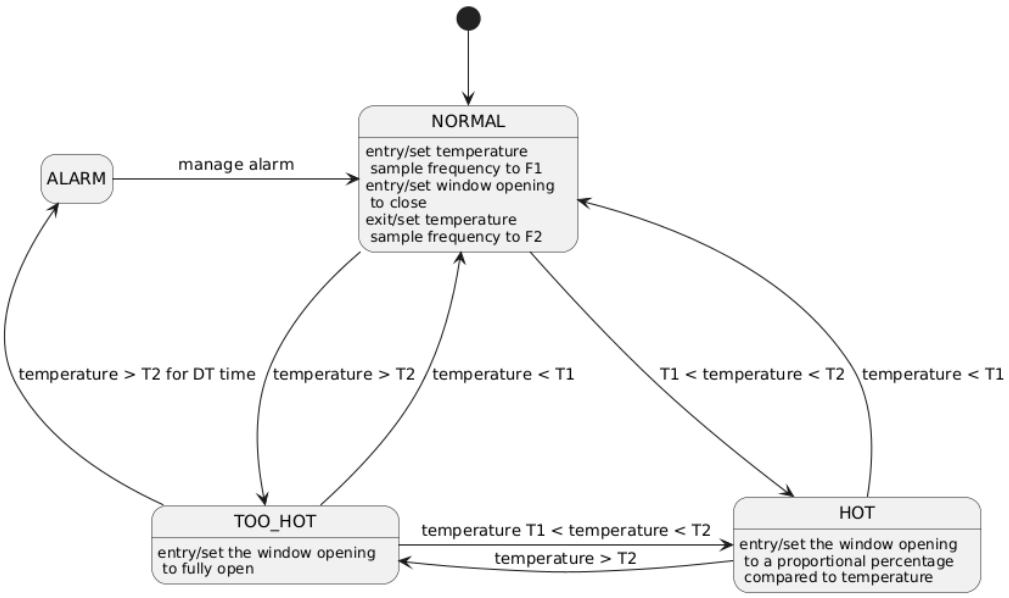

Figure 2.3: Temperature State

- NORMAL: it's the initial state. In this state the temperature is normal, so the window is closed.
- HOT: in this state the temperature begins to be hot, so depending on the current temperature the window will be opened for a certain percentage.
- TOO_HOT: in this state the environmental temperature is too hot. The window must be fully open.
- ALARM: if the temperature state remains in *TOO_HOT* for *DT* time the system goes in *ALARM* state. In this state the window stays fully open until an operator manually manages the alarm from the dashboard.

3 Develop

The system is developed in two different languages:
The Temperature Monitoring and Window Controller subsystems are developed in `C++` by using `PlatformIO` IDE with `arduino` framework.
The Control Unit and the Dashboard subsystems are developed in `Java` by using `Gradle` as a build tool with the starting configuration given in the course `Object-Oriented Programming`.

The Temperature Monitoring subsystem uses the "PubSubClient" external library to use MQTT protocol.

The Window Controller subsystem uses three external libraries: "EnableInterrupt" to enable interrupt handling, "LiquidCrystal_I2C" to manage the display and "Servo" to manage the servo motor for the window.

The Control Unit uses "jssc" to communicate through the serial line and "mqttv3" to use mqtt protocol.

The Dashboard uses "jfreechart" to create and draw charts.

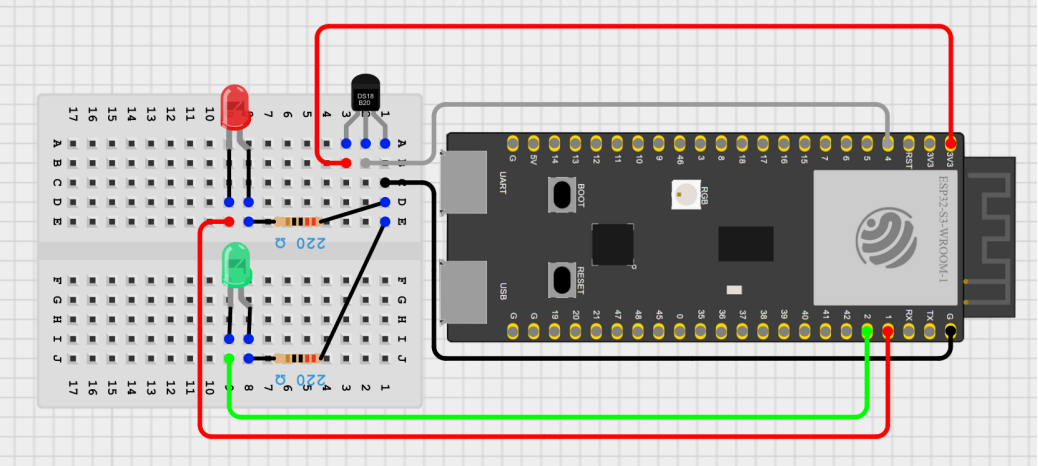

Figure 3.1: Temperature Monitoring schema

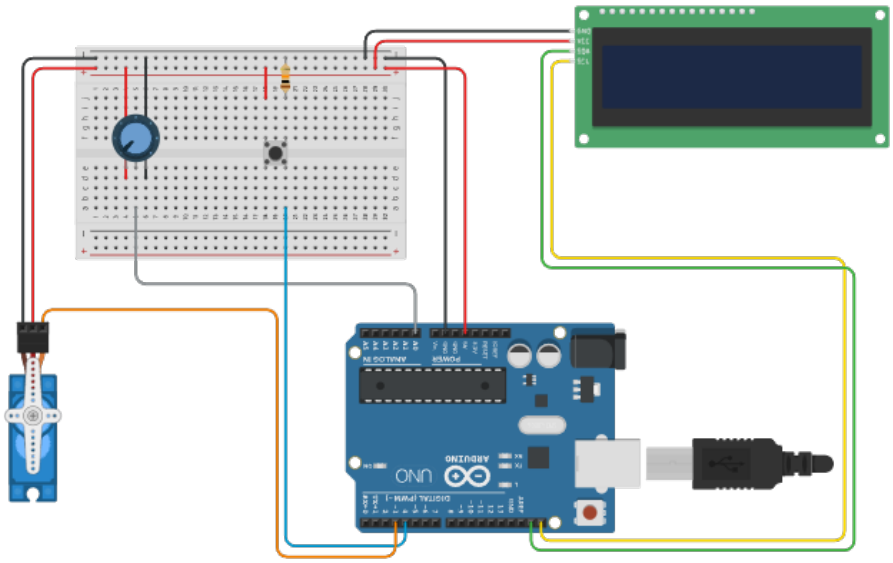

Figure 3.2: Window Controller schema

Here the link to the project's GitHub repo.
Here is the link to the demonstration video.