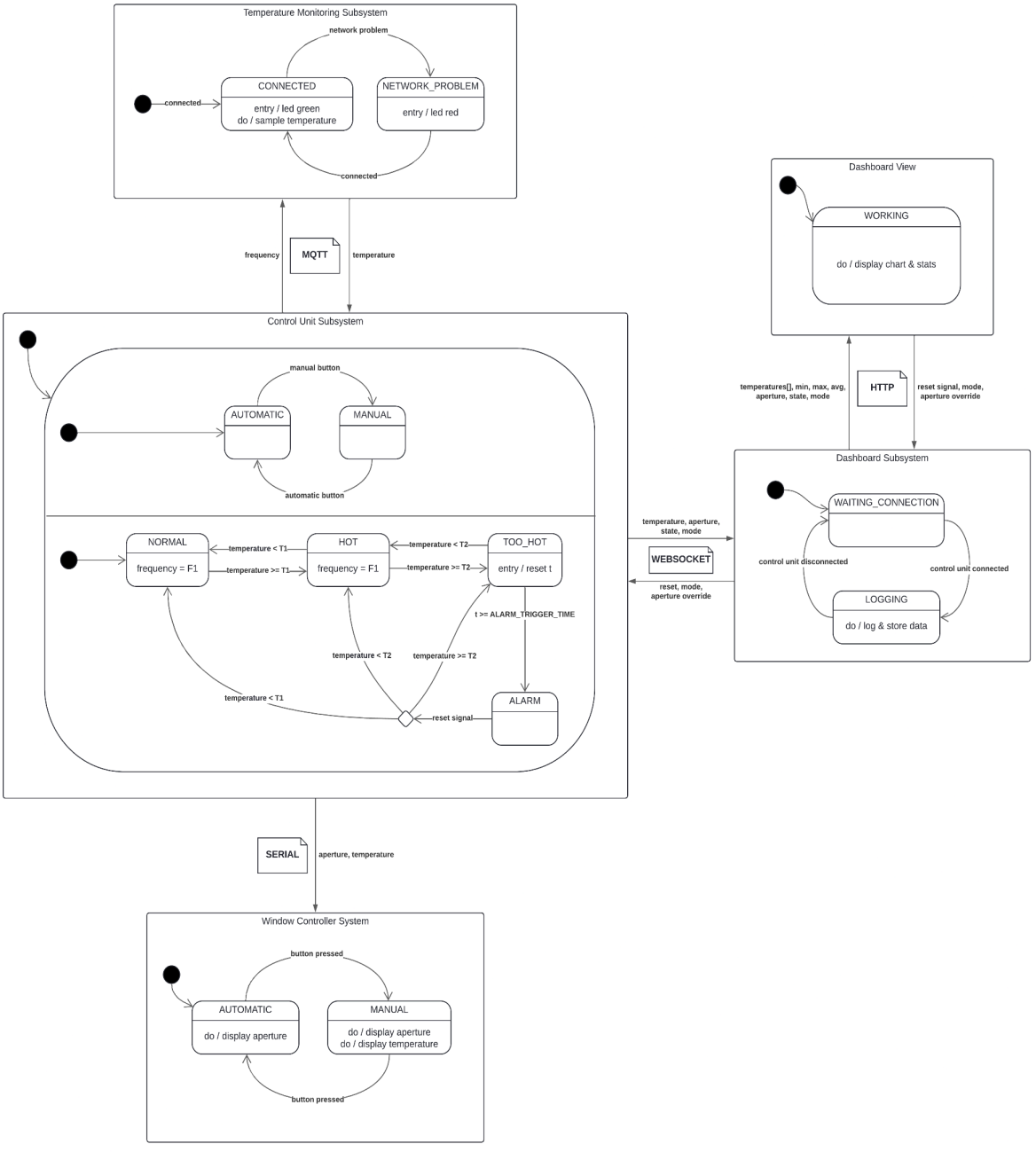
Assignment 03 - *Smart Temperature Monitoring*

Andrea Baldazzi 0001071149

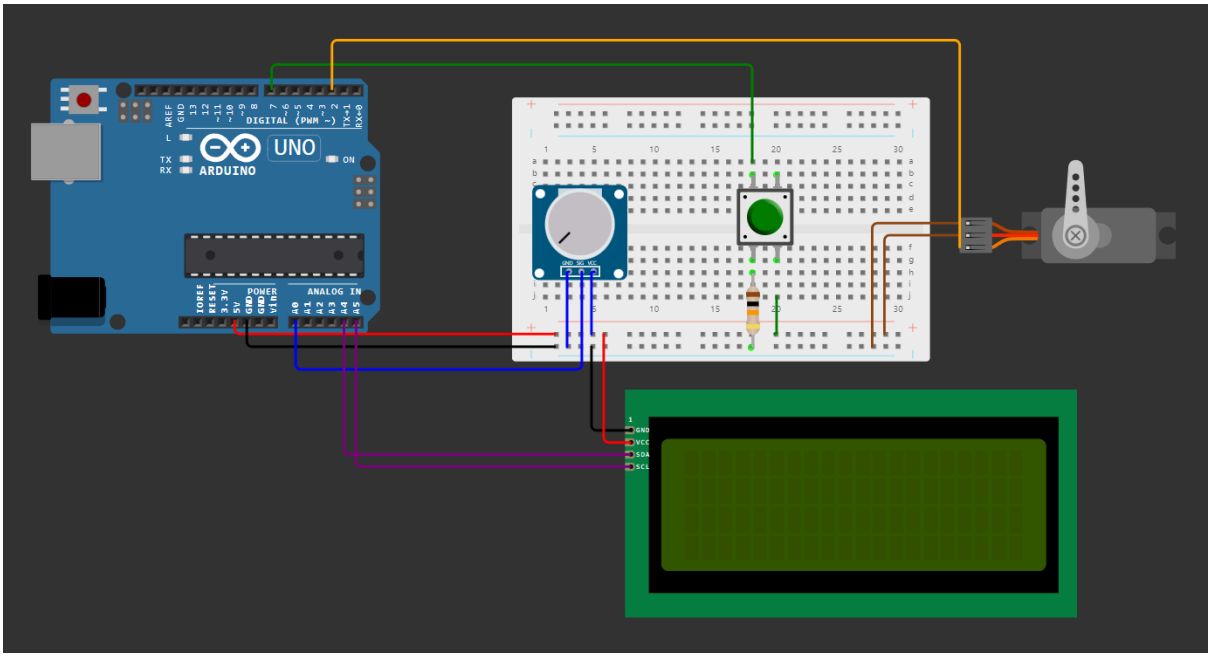
System description

The system is about the smart control of a window based on the temperature of the room. The temperature is sampled at regular intervals and then is processed, determining the opening level of the window. An operator can see real time stats in a dashboard and control the window opening manually. They system is composed of 4 submodules, whose Finite State Machines are represented by the following diagram (a standalone version is inside **src**):



Window Controller Subsystem

The Window Controller Subsystem is responsible for physically controlling the window opening. It is composed of an Arduino Uno system which has connected a potentiometer, a button, a servo motor (representing the window) and an LCD screen.
Here is the component scheme:



The subsystem is implemented with Tasks, using a synchronous FSM. Because the logic is very simple there is a single task, but the architecture allows for future expansion of the subsystem.

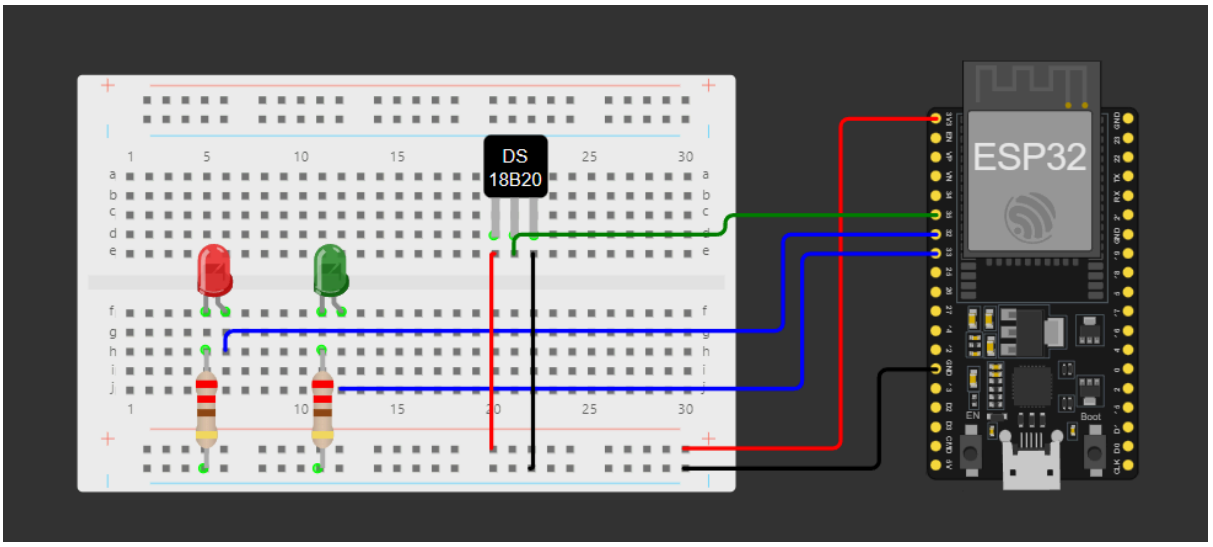
The Window Controller has the following states:

- **AUTOMATIC:** default, the window opening level is decided by the Control Unit and it's shown on the LCD screen. Pressing the button switches to manual mode.
- **MANUAL:** in this mode the window opening level is decided by an operator by rotating the potentiometer. The LCD screen shows the current opening level and also the temperature. Pressing the button switches back to automatic mode.

The subsystem communicates with the Control Unit through the serial line.

Temperature Monitoring Subsystem

The Temperature Monitoring Subsystem is responsible for sampling the temperature and communicating it to the Control Unit. It is composed of an ESP32 which has two LEDs and an analog TMP36 temperature sensor..
Here is the component scheme:



The subsystem is a synchronous FSM with the following states:

- **CONNECTED:** in this state the system samples temperature at regular intervals and sends the values to the Control Unit. The green LED is on, indicating that everything works as intended.
- **NETWORK_ERROR:** the system goes into this state whenever there is a network problem and tries to reconnect to the broker. The red LED is on, indicating that the system is not connected to the network.

The Temperature Monitor Subsystem communicates with the MQTT protocol with the Control Unit, receiving the sampling frequency and sending the sampled values.

Control Unit Subsystem

The Control Unit Subsystem is the core of the whole system. It hosts the logic behind the behaviour of the application.

The subsystem is a synchronous FSM with the following states:

- **NORMAL:** this state means that temperature is under control. The window opening level is 0%.
- **HOT:** when the temperature goes over some threshold T1 the system becomes hot. The window opening level is proportional to the distance to T2.
- **TOO_HOT:** when the temperature goes over some threshold T2 the system becomes too hot. The window is open at 100%.
- **ALARM:** if the system is TOO_HOT for longer than a predetermined time goes into this state, requiring the intervention of the operator through the dashboard. When the reset signal is received, it goes back to the appropriate state, depending on the temperature reported.

The are also two functioning modes:

- **AUTOMATIC:** the system decides the appropriate window opening level based on the temperature.
- **MANUAL:** the opening level is decided by an operator using the dashboard.

The Control Unit, implemented with a Java program, communicates the other subsystems in different ways:

- **Window Controller: serial line.** The Java library JSSC is used to communicate with Arduino.
- **Temperature Monitoring: MQTT.** The Java library Vert.x is used to communicate with an MQTT broker, with an independent verticle managing the connection.
- **Dashboard Subsystem: HTTP Websocket.** The Java library Vert.x is used to communicate with the dashboard server, with an independent verticle managing the websocket connection.

Dashboard Subsystem

The Dashboard Subsystem is responsible for displaying the interface for the operator to control the system, showing values over time and offering some control options. It also stores data received from the Control Unit, acting as a sort of a very simple DB.

The GUI, made with HTML and Javascript, is a SPA (Single Page Application) and can be accessed from every device allowed to connect to the server.

The subsystem is implemented in Java using the Vert.x library and hosts an HTTP server which allows for two types of connection:

- **HTTP Websocket:** this special full-duplex http connection is reserved for the communication with the Control Unit, receiving regular updates and sending commands.
- **Regular HTTP:** by connecting with an http client (a browser, for example) the server sends the necessary files required for running the Web App.