TECHNICAL UNIVERSITY OF MOLDOVA
FACULTY OF COMPUTERS, INFORMATICS
AND MICROELECTRONICS
DEPARTMENT OF SOFTWARE ENGINEERING AND
AUTOMATICS

Laboratory work nr. 7.3

Internet communication - MQTT english

Elaborated: **Cristian Brinza**
st. gr. FAF-212

Verified: **Moraru Dumitru**
lect. univ

Chișinău, 2024

THE TASK OF THE LABORATORY WORK:

To create an MCU-based application that will take a signal from a signal source, and display the physical parameter at a terminal (LCD and/or Serial).

Each student will select a sensor either analog or digital (not binary) from the attached PDF or: http://www.37sensors.com/.

- **To acquire the signal from the sensor;**
- **To display the data on the LCD and / or Serial display.**
- **To condition the signal involving digital filters and other methods**

PROGRESS OF THE WORK

1 Main functions/methods used to execute the task

Explication about this chapterIn this chapter I will explain the functionality of diferent parts of the executed task:

1. In the **sketch.ino** file:

- **setup() Function:** This function initializes the LCD and configures the sensor input pin. It is foundational in setting up the LCD display to ensure it's ready to show temperature readings and configuring the analog pin connected to the TMP36 temperature sensor, preparing it to read temperature data. The setup phase is crucial as it establishes the operational baseline for the sensor and display components, ensuring they are correctly initialized and ready for data acquisition and presentation.
- **loop() Function:** Manages the core operational cycle, including reading sensor data, evaluating control logic based on temperature readings, and updating the LCD display. This function embodies the continuous execution logic of the system, beginning with the acquisition of raw sensor data from the TMP36 sensor. It processes these readings to decide on controlling a relay for temperature regulation and displays the current and setpoint temperatures on the LCD, providing real-time feedback to the user. This loop ensures the system consistently monitors the environment and updates the display with the latest information.

2. **LCD Feedback Display:**

- Updates the LCD with the latest temperature reading. This interface between the system and the user is essential for communicating the real-time state of the monitored environment. It ensures that the temperature data, once acquired, processed, and calculated, is presented in an accessible and understandable format to the user, closing the loop on the system's functionality.

3. **Sensor Data Acquisition**:

- Directly reads the analog signal from the TMP36 temperature sensor. This critical operation translates the physical temperature into a digital signal that the Arduino can process. By reading the analog voltage across the sensor, the system can infer the temperature based on the known characteristics of the TMP36 sensor, such as its voltage-temperature relationship.

4. **Digital Signal Filtering:**

- **Relay Control:** Manages the power to a heating or cooling device through a relay to maintain the temperature around the setpoint with a specified hysteresis, preventing rapid toggling of the relay.

5. **Temperature Calculation:**

- Converts the analog signal from the TMP36 sensor into a meaningful temperature value. This process involves converting the sensor's voltage output into degrees Celsius using the specific linear relationship inherent to the TMP36 sensor. Leveraging the thermistor's electrical characteristics and the ambient conditions, this function calculates the precise temperature.

Block Diagram

- **Main Program (sketch.ino):** Serves as the heart of the system, initializing the Arduino pins for the LCD and sensor. It orchestrates the operational flow, including acquiring sensor data, processing this data through control logic, and displaying the results on the LCD.
- **Setup Function** Responsible for initializing the LCD display and configuring the sensor pin for input text message.
- **Loop Function**: Represents the continuous execution cycle of the Main Program. It handles reading sensor data, applying temperature control logic, and updating the LCD with the latest readings.
- **Sensor Data Acquisition:** Manages the reading of analog data from the TMP36 temperature sensor.
- **Temperature Control Logic:** Utilizes the relay to maintain temperature within a set range based on the current readings and setpoint.
- **Temperature Calculation:** Converts the processed sensor voltage to a physical temperature value.

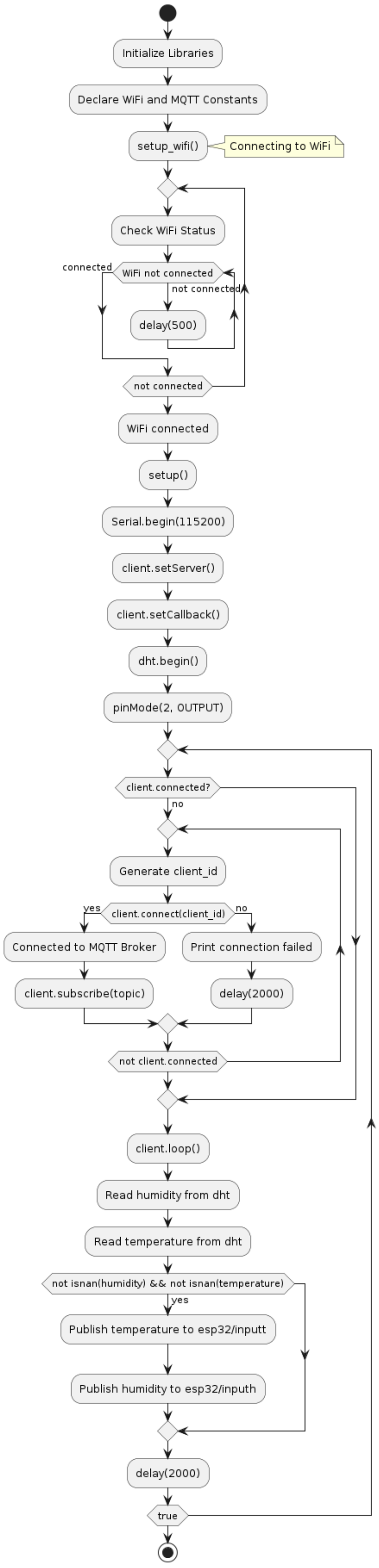

Figure 1 Program schema.

Simulated or real assembled electrical schematic diagram :

The Figure 2 depicted the phisical board schema.

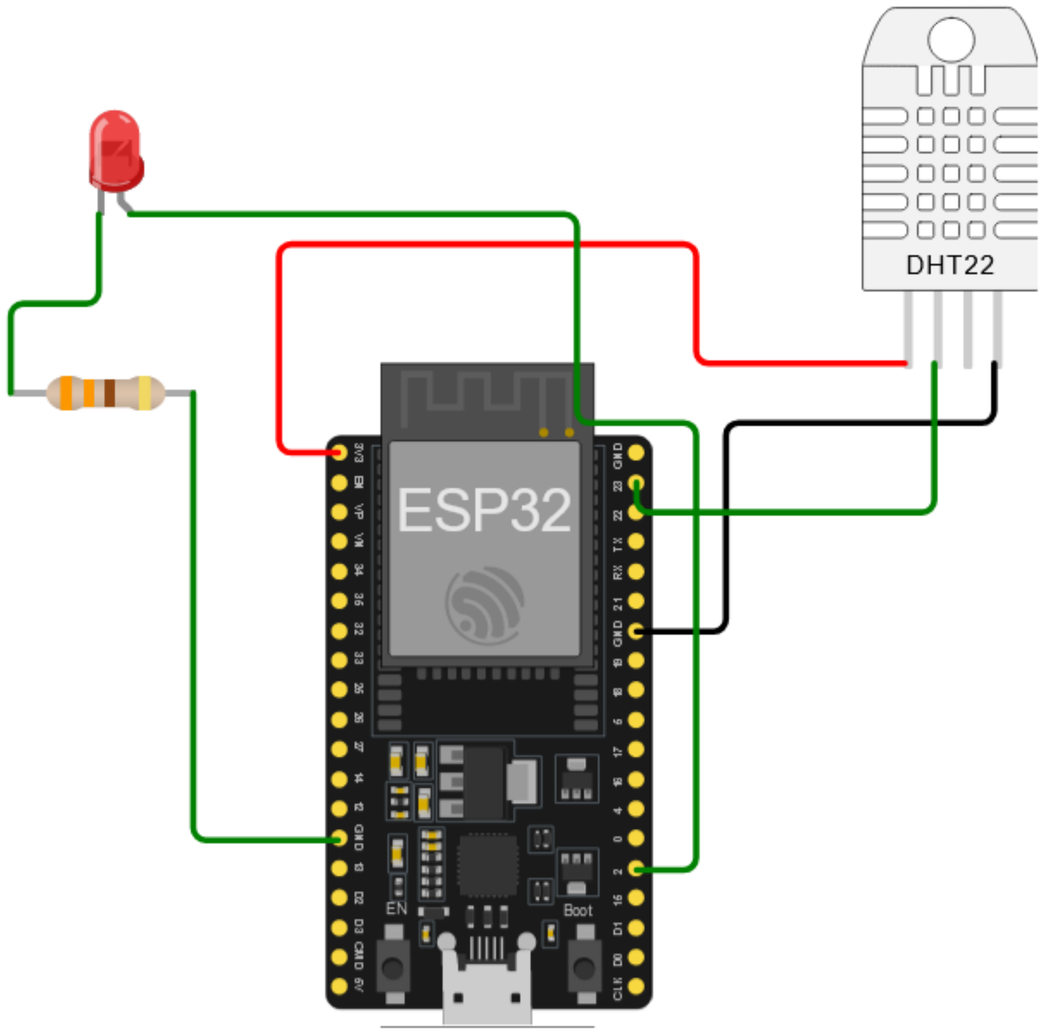

Figure 2 Phisical board schema.

CONCLUSION

In completing this lab, we have navigated through the detailed intricacies of working with Arduino, particularly focusing on temperature regulation and display using a TMP36 temperature sensor and relay control logic. This project has exemplified the harmonious integration of hardware and software components, illustrating our capability to collect raw sensor data, implement control mechanisms using hysteresis, and visually communicate system statuses through an LCD.

We initiated the project by setting up crucial components: preparing the LCD for user feedback and configuring the Arduino to interface with the TMP36 sensor and relay. The development of the sketch.ino file formed the backbone of our endeavor, incorporating both setup() and loop() functions that are essential for the system's operation. The setup() function was pivotal in ensuring that all components were correctly initialized and operational, whereas the loop() function played a continuous role in reading temperature data, applying logical control based on the setpoint and hysteresis, and updating the LCD with the latest temperature information and system status.

This lab not only underscored the importance of precise and responsive data handling in embedded systems but also highlighted the critical role of user interfaces in real-time monitoring. By implementing a straightforward yet effective ON/OFF control logic with hysteresis, we addressed the challenges associated with maintaining a stable temperature environment. Additionally, the practice of interpreting sensor output to enact control decisions reinforced the necessity of integrating accurate electronics principles and understanding the behavior of sensors and actuators.

The lab served as more than just a practical exercise; it was a robust demonstration of the foundational principles of embedded system design, encompassing sensor interfacing, control logic implementation, and user interaction. The skills and insights gained from this project provide a solid base for future explorations, whether they involve further applications in sensor data management, Internet of Things (IoT) implementations, or broader ventures within the expansive fields of electronics and embedded systems. Through this project, we have strengthened our understanding and are better prepared for the technological challenges and opportunities that lie ahead.

BIBLIOGRAPHY

1 EDUCBA: *Introduction to Embedded Systems.* Cursuri electronice online ©2020 [quote 20.03.2024] Available: https://www.educba.com/what-is-embedded-systems/

2 ARDUINO: *Arduino UNO.* The official website of Arduino modules, ©2020 [quote 20.03.2024]. Available: https://www.arduino.cc/.

3 TUTORIALSPOINT: *Embedded Systems Tutorial*. Comprehensive guide for beginners and professionals to learn Embedded System basics to advanced concepts, including microcontrollers, processors, and real-time operating systems. ©2023 [quote 20.03.2024] Available: https://www.tutorialspoint.com/embedded_systems/index.htm

4 GURU99: *Embedded Systems Tutorial: What is, History & Characteristics*. A detailed introduction to embedded systems, covering microcontrollers, microprocessors, and the architecture of embedded systems, as well as their applications and advantages. ©2023 [quote 20.03.2024] Available: https://www.guru99.com/embedded-systems-tutorial.html

5 JAVATPOINT: *Embedded Systems Tutorial.* Offers basic and advanced concepts of Embedded System, designed for beginners and professionals. ©2023 [quote 20.03.2024] Available: https://www.javatpoint.com/embedded-systems-tutorial

6 DEEPBLUE*: Embedded Systems Tutorials Introduction* | Embedded Systems Online Course. ©2023 [quote 20.03.2024] Available: https://deepbluembedded.com/embedded-systems-tutorials/

APPENDIX 1

Code of main.ino:

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
// WiFi credentials
const char* ssid = "Wokwi-GUEST";
const char* password = "";
// MQTT Broker
const char* mqtt_broker = "broker.hivemq.com";
const char* topic = "esp32/output";
const char* mqtt_username = "";
const char* mqtt_password = "";
const int mqtt_port = 1883;
WiFiClient espClient;
PubSubClient client(espClient);
DHT dht(23, DHT22);  // GPIO23 connected to the DHT22 data pin
void setup_wifi() {
delay(10);
// Connect to Wi-Fi
Serial.println("Connecting to WiFi..");
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");}
Serial.println("WiFi connected!");}
void callback(char* topic, byte* payload, unsigned int length) {
Serial.print("Message arrived in topic: ");
Serial.println(topic);
Serial.print("Message:");
for (int i = 0; i < length; i++) {
Serial.print((char)payload[i]); }
if ((char)payload[0] == '1') {
digitalWrite(2, HIGH);  // Turn the LED on
} else {
digitalWrite(2, LOW);  // Turn the LED off}
Serial.println();
Serial.println("-----------------------");}
void setup() {
Serial.begin(115200);
setup_wifi();
client.setServer(mqtt_broker, mqtt_port);
client.setCallback(callback);
dht.begin();
pinMode(2, OUTPUT);  // Initialize GPIO2 as an output for LED}
void loop() {
if (!client.connected()) {
while (!client.connected()) {
String client_id = "esp32-client-";
client_id += String(random(0xffff), HEX);
if (client.connect(client_id.c_str())) {
Serial.println("Connected to MQTT Broker!");
client.subscribe(topic);
} else {
Serial.print("failed with state ");
Serial.print(client.state());
delay(2000);}}}
client.loop();
float humidity = dht.readHumidity();
float temperature = dht.readTemperature();
Check if sensor readings are not NaN (not a number)
if (!isnan(humidity) && !isnan(temperature)) {
String tempToSend = String(temperature);
client.publish("esp32/inputt", tempToSend.c_str());
String humToSend = String(humidity);
client.publish("esp32/inputh", humToSend.c_str());}
delay(2000);}
```