SCIENCES ET TECHNOLOGIES

Projet Multidisciplinaire 3

Temperature control

Aoufia Samir

Informatique industrielle

2023 - 2024

HELHa
Haute École Louvain en Hainaut
Développe tes talents

Table des matières

1.Introduction

This year, as part of my multidisciplinary project course, I build a project on temperature control.

The principle of this project is the integration of hardware and software components to create an efficient and autonomous system.

I will start by explaining my design brief, then I will talk about my management and finally I will explain the autonomous system, which is divided into 2 parts hardware and software.

2. Design brief[1]

Objective: The aim of this project is to enable temperature control, either automatically or manually with a mobile application.

How: There will be a box, inside the box the temperature is always read by a sensor. If the temperature differs from the preset setpoint, the microcontroller with a relay activated the heat lamp or the fan until the temperature of the setpoint is reach. User can with the mobile application control the process. He can start or stop the automatic mode with a button, switch the lamp or fan on or off, and choice a new setpoint for the temperature. In the mobile application the will a history of all temperatures entered by the user.

[1] Full design brief in appendix

3. Project management

3.1 Gantt

ID	Name	Start Date	End Date
1	Multidisciplinary Project 3	Oct 02, 2023	Dec 29, 2023
2	Design brief	Oct 02, 2023	Oct 06, 2023
3	List of components	Oct 08, 2023	Oct 08, 2023
12	Ordering components	Oct 09, 2023	Oct 09, 2023
5	Waiting components	Oct 10, 2023	Oct 15, 2023
6	Research	Oct 16, 2023	Oct 23, 2023
7	Connecting components	Oct 24, 2023	Oct 31, 2023
8	Arduino development	Nov 01, 2023	Nov 08, 2023
9	Creating a Mqtt server	Nov 09, 2023	Nov 11, 2023
10	Mobile application development	Nov 12, 2023	Dec 01, 2023
11	Report	Nov 26, 2023	Dec 29, 2023

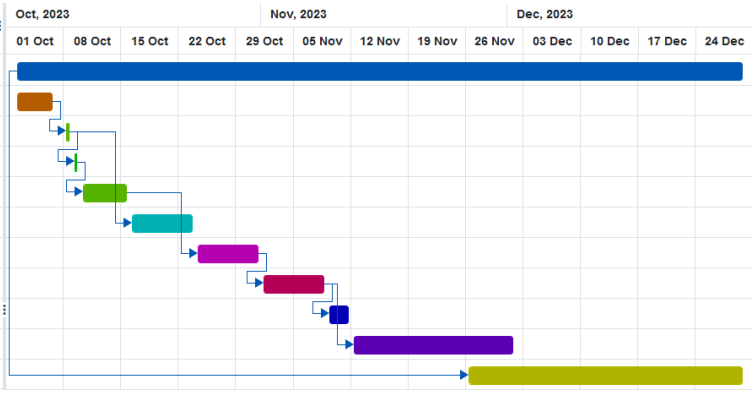


Figure 1 This is a Gantt of my project

3.2 Mind map

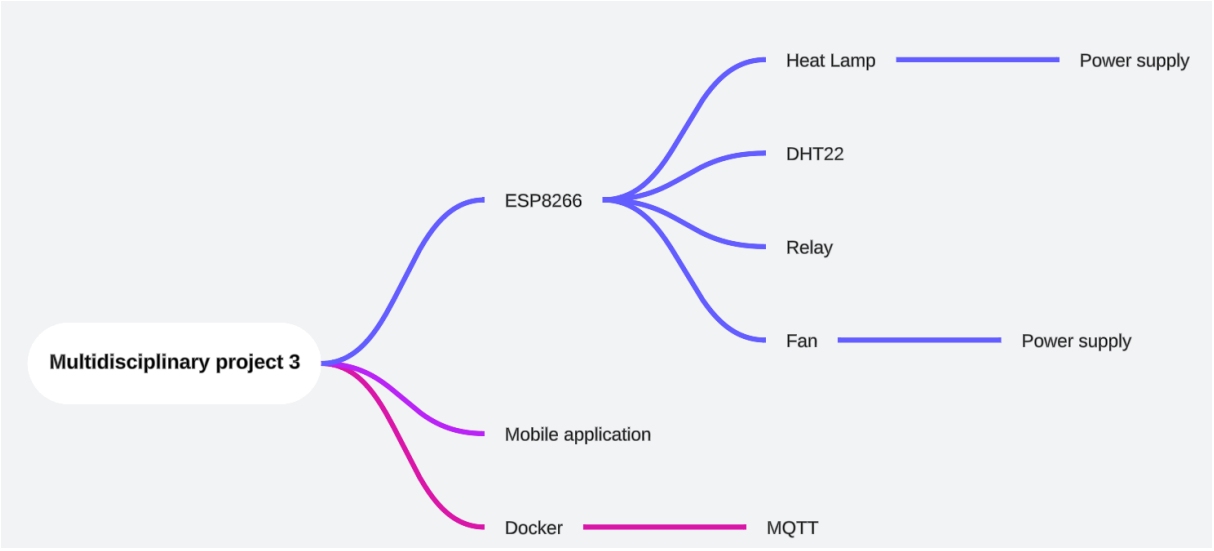


Figure 2 This is a Mind Map of my project

4. Development

In this part, I going to explain how I build the different parts of the project, starting with the embedded system, the docker and the mobile application.

4.1 Embedded system

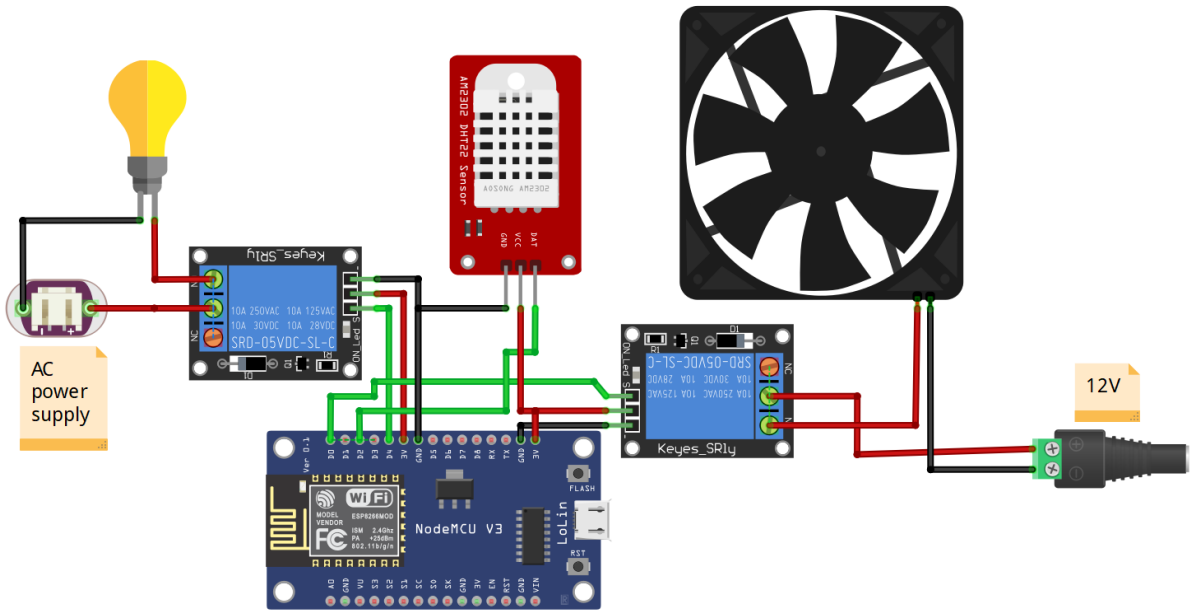


Figure 3 Schema of my system

4.1.1 Esp8266

The esp8266 is a wifi module based on a Tensilica Xtensa LX106 microcontroller clocked at 80 MHz with a 16 MO flash memory and a 32K + 80K ram memory, it has 16 inputs/outputs and 10 analog inputs.

I use an esp8266 because it was a requirement to comply with the design brief, it uses to connect by wifi, send and receive data from MQTT.

The esp8266 allows me to execute code in a loop and recover temperature information and turn the lamp or fan on and off.

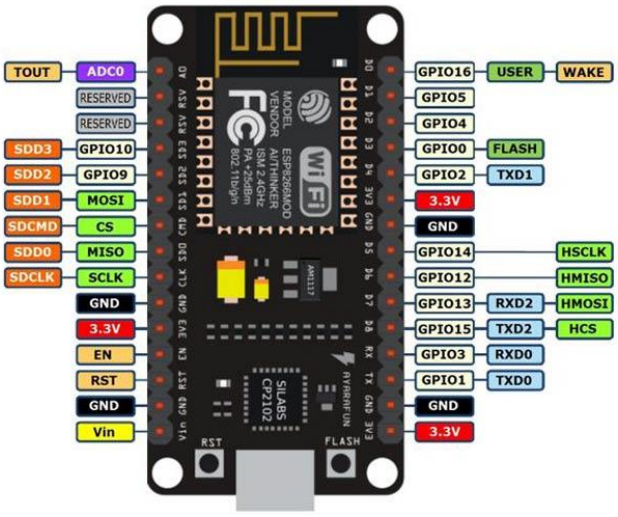


Figure 4 Picture of an esp8266

4.1.2 DHT22

The DHT22 is a temperature and humidity sensor. This sensor originally has 4 pins, but when is mounted on a PCB, it has only 3 pins to connect V+, V- and data. The data it sends is digital.

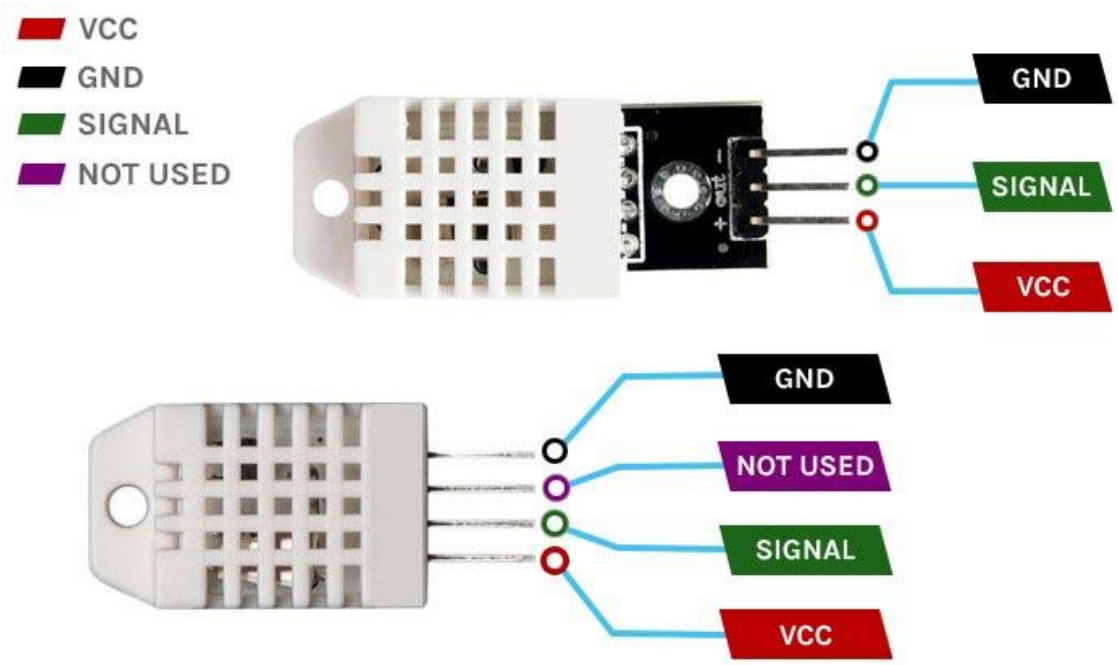


Figure 5 Picture of a DHT22

4.1.3 Relay

I use a relay as a switch controlled by the esp8266 to switch on the lamp for heating or the fan for cooling. To power it, I use the 3V3 of the esp.

Figure 6 Picture of a relay

4.1.4 Code Arduino

Here is the code for the connection between the wifi and the esp8266

We start by including the library. Then we enter the wifi name and password and use a function called setup_wifi to connect to our wifi.

```
#include <ESP8266WiFi.h>

const char* ssid = "SAMIR";
const char* password = "Samir1601";

WiFiClient espClient;

Serial.begin(9600);
setup_wifi();


void setup_wifi(){
WiFiMulti.addAP(ssid, password);
while ( WiFiMulti.run() != WL_CONNECTED ) {
delay ( 500 );
Serial.print ( "." );
}
Serial.println("");
Serial.println("WiFi connecté");
Serial.print("MAC : ");
Serial.println(WiFi.macAddress());
Serial.print("Adresse IP : ");
Serial.println(WiFi.localIP());
}
```

Here is the code for the connection between the MQTT and the esp8266

We start by including the library. Then I give the ip and port of the MQTT, I use a function called setup_MQTT to connect our MQTT and the callback function which is called each time a new message is received from MQTT.

```
#include <PubSubClient.h>


const char* mqtt_server = "192.168.0.153";
const int mqttPort = 1883;


PubSubClient client(espClient);


setup_mqtt();



void setup_mqtt(){
client.setServer(mqtt_server, mqttPort);
client.setCallback(callback);
reconnect();
}


void reconnect(){
while (!client.connected()) {
Serial.println("Connection au serveur MQTT ...");
if (client.connect("ESP32Client")) {
Serial.println("MQTT connecté");
client.subscribe("Temperature");
client.subscribe("OnOff");
}
else {
Serial.print("echec, code erreur= ");
Serial.println(client.state());
Serial.println("nouvel essai dans 2s");
delay(2000);
}
}
}
```

```
void callback(char* topic, byte *payload, unsigned int length) {
  Serial.println("-------Nouveau message du broker mqtt-----");
  Serial.print("Canal:");
  Serial.println(topic);
  Serial.print("donnee:");
  Serial.write(payload, length);
  Serial.println();

  char message[length + 1];
  memcpy(message, payload, length);
  message[length] = '\0';

  if (strcmp(topic, "Temperature") == 0) {
    consigne_temp = atof(message);
    Serial.print("Nouvelle valeur de consigne de température : ")
    Serial.println(consigne_temp);
  } else if (strcmp(topic, "OnOff") == 0) {
    mode_automatique = false;
    commande_manuelle_active = true;

    switch ((char)payload[0]) {
      case '1':
        Serial.println("Allumer le ventilateur (manuel)");
        digitalWrite(RELAY_venti, HIGH);
        digitalWrite(RELAY_lamp, LOW);
        break;
      case '2':
        Serial.println("Allumer la lampe (manuel)");
        digitalWrite(RELAY_lamp, HIGH);
        digitalWrite(RELAY_venti, LOW);
        break;
      case '3':
        Serial.println("Éteindre les deux (manuel)");
        digitalWrite(RELAY_venti, LOW);
        digitalWrite(RELAY_lamp, LOW);
        break;
      case '4':
        Serial.println("Passage en mode automatique");
        mode_automatique = true;
        commande_manuelle_active = false;
        break;
      default:
        Serial.println("Commande non reconnue");
        break;
    }
  }
}
```

4.2 Docker

Docker is a conainerization software which lets you deploy applications in isolated containers.

I use docker to have an MQTT server, it runs on my pc locally.

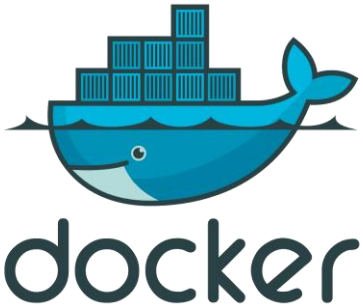


Figure 7 Picture of the logo of docker

4.2.1 Mosquitto

Mosquito is an open source MQTT (Message Queuing Telemetry Transport) developed by Eclipse Foundation.

Mosquitto manage communications between different devices like connected objects and servers.

I use mosquito to communicate between my esp8266 and my mobile application.

Figure 8 Picture of the logo of moquitto

4.3 Mobile application

My mobile application is made with Expo Go using react-native and Typescript.

4.3.1 Home

When you launch the application, you come to a login page, to access another part of the application you must login. If you make a mistake, an alert appears.

Name: Admin

Password: Admin

To make the connection, I did an If that checks If the name is “Admin” and the password is “Admin”.

```
function test() {
if (text == "Admin" && text2 == "Admin") {
navigation.navigate('Log')
} else {
alert("Wrong ID")
```

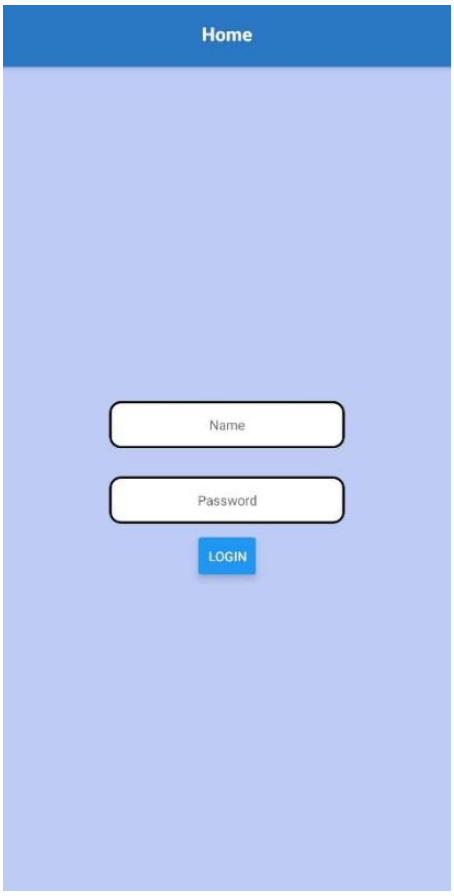


Figure 10 Picture of my login page

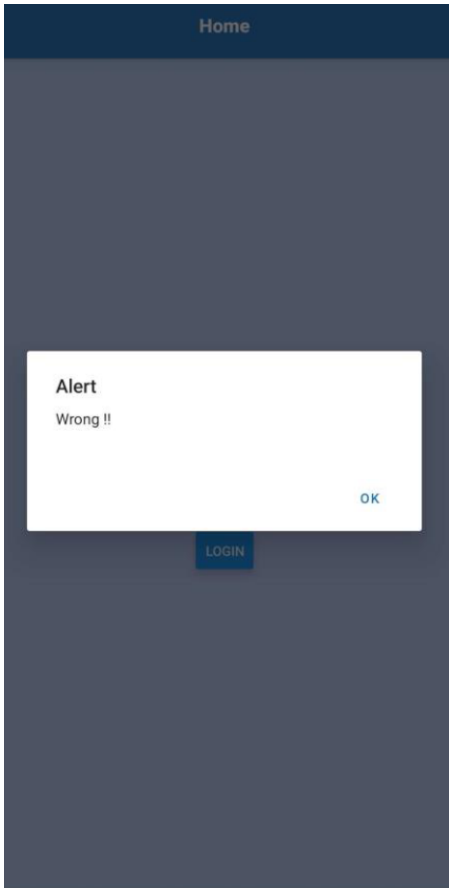


Figure 9 Picture of my login page

4.3.2 Login

After logging in, you appear on a page with 3 buttons, each buttons sends you to a different page.

Figure 11 Picture of my home page

4.3.2 Project

On this page, you will find a short text explaining my project.

Figure 12 Picture of my project page

4.3.4 Value

Figure 13 Picture of my value page

The value page lets you enter the desired temperature, shows the current temperature inside the box, and there are 4 buttons.

The first button is the Off lamp / On fan button, turns the fan on and the lamp off.

The second button is the On lamp / off fan button, turns the fan off and the lamp on.

The third button is all off button, turn everything off.

The last button is the automatic button, puts the system in automatic mode, it looks at the current temperature and compares it with the desired temperature. If the temperature is too high the fan is activated and if it's too low the lamp is activated.

Here is the code for the desired temperature

First, I check if I'm connected to MQTT

After that I use a useState to get value in a variable, I send the message with "Client.send(mqttMessage)" and the topic with "mqttMessage.destinationName".

```
const [temp, onChangeText] = React.useState('');
const val = temp;
const topic = 'Temperature';
if (client.isConnected()) {
const mqttMessage = new Paho.Message(val.toString());
mqttMessage.destinationName = topic;
client.send(mqttMessage);
console.log(`Message sent to topic ${topic}: ${val}`);
```

Here is the code for the current temperature

First, I subscribe to the value topic, I use a useState to get the value in a variable, I use an onMessage function to get the value and set it in value, then I display it on the page.

```
const [value, setValue] = useState(0);
function subscribeToValue() {
client.subscribe('value');
client.onMessageArrived = onMessage;
}
function onMessage(message: Paho.Message) {
if (message.destinationName === 'value') {
setValue(parseInt(message.payloadString || '0', 10));
}
}
<Text style={{ fontSize: 20, marginTop: 20 }}>Temperature: {value}</Text>
```

Here is the code for the button

For the button I made a unique code for all button, which sends a value between 1 and 4because in my Arduino code I made a switch that says if the value is 4 it goes into automatic mode, at 1 I turn on the fan, at 2 I turn on the lamp and at 3 I turn all off.

```
function publishToMQTT(value: any) {
  const topic = 'OnOff';
  if (client.isConnected()) {
    const mqttMessage = new Paho.Message(value.toString());
    mqttMessage.destinationName = topic;
    client.send(mqttMessage);
    console.log(`Message sent to topic ${topic}: ${value}`);
  } else {
    console.log('Not connected to MQTT. Cannot send message.');
  }
}
```

4.3.5 History

For my history page, there is my history of the desired temperature with the time and a button to clear the history.

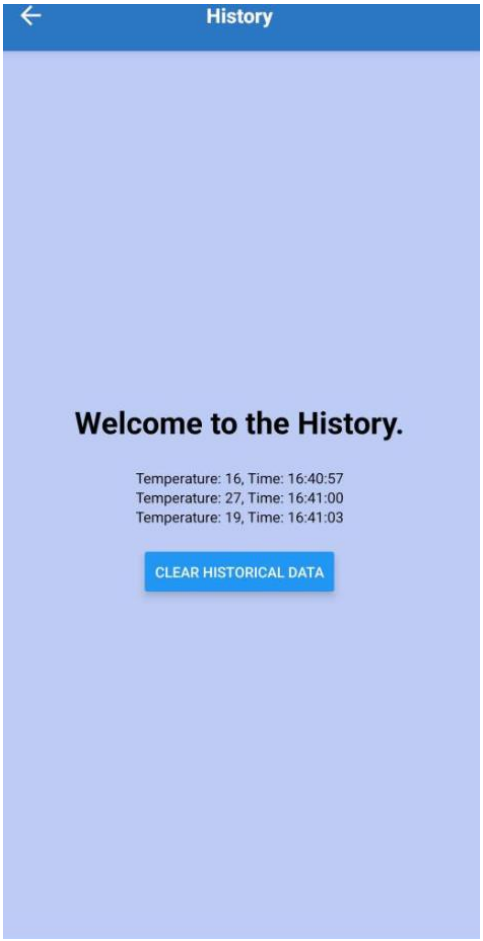


Figure 14 Picture of my history page

4.3.6 Database

I use AsyncStorage as a database to store and get data locally. The “clearHistoricalData” function deletes the history, while “storeHistoricalData” updates and saves the data. The history use a “useEffect” to load data at lauch and displays this data with the option of deleting it. The “formatTimePlusOneHour” function formats time by adding one hour to the original time.

```
const clearHistoricalData = async () => {
try {
await AsyncStorage.removeItem('historicalData');
console.log('Historical data cleared.');
} catch (error) {
console.error('Error clearing historical data:', error);
}
};
```

```
const currentTime = new Date();
setHistoricalData((prevData) => [...prevData, { value: val, time:
currentTime }]);
storeHistoricalData((prevData) => [...prevData, { value: val, time:
currentTime }]);
const storeHistoricalData = async (dataUpdateFunction) => {
try {
const currentData = await AsyncStorage.getItem('historicalData');
const currentDataArray = currentData ? JSON.parse(currentData) :
[];
const updatedDataArray = dataUpdateFunction ?
dataUpdateFunction(currentDataArray) : currentDataArray;
await AsyncStorage.setItem('historicalData',
JSON.stringify(updatedDataArray));
} catch (error) {
console.error('Error storing historical data:', error);
}
}
```

```
function Historique({ navigation }) {
const [historicalData, setHistoricalData] = useState([]);
useEffect(() => {
const retrieveData = async () => {
try {
const data = await AsyncStorage.getItem('historicalData');
if (data) {
setHistoricalData(JSON.parse(data));
}
} catch (error) {
console.error('Error fetching historical data:', error);
}
};
retrieveData();
}, []);
const handleClearHistoricalData = () => {
clearHistoricalData();
setHistoricalData([]);
};
return (
<View style={styles.homescreen}>
<Text style={{ fontSize: 30, fontWeight: 'bold' }}>Welcome to the
History.</Text>
<View style={{ marginBottom: 20, marginTop: 20 }}>
{historicalData.map((entry, index) => (
<Text key={index}>{`Temperature: ${entry.value}, Time:
${entry.time ? formatTimePlusOneHour(entry.time) : 'N/A'}`}</Text>
))}
</View>
<Button title="Clear Historical Data"
onPress={handleClearHistoricalData} />
</View>
);
}
```

```
const formatTimePlusOneHour = (time) => {
const newTime = new Date(time);
newTime.setHours(newTime.getHours() + 1);
const hours = newTime.getUTCHours().toString().padStart(2, '0');
const minutes = newTime.getUTCMinutes().toString().padStart(2, '0');
const seconds = newTime.getUTCSeconds().toString().padStart(2, '0');
return `${hours}:${minutes}:${seconds}`;
};
```

5. Conclusion

This project fascinated me because it made me use several aspects of computing.

For the embedded system part, I gained experience on how the esp8266 works and I also had to learn how to make the wifi and mqtt connection on an esp8266.

For the docker part, I had to create an MQTT (mosquitto), something I had never done before and that caused me problems because I had to configure all the settings.

I had a lot of problems with the history part and sending messages via MQTT, but thanks to the documentation it all work.

Overall, the project enabled me to learn and deepen my knowledge and gave me a taste of my future traineeship and that gave me some confidence.

6. Mediagraphy

ESP8266 [Wiki]. (s. d.). https://reso-nance.org/wiki/materiel/esp8266/accueil#:~:text=Caract%C3%A9ristiques,-Documentation%20technique%20%3A%20ESP8266&text=L%27ESP8266%20est%20compos%C3%A9%20d,m%C3%A9moire%20RAM%20de%2032K%20%2B%2080K

Bulcke, A. (s. d.). *Mqtt avec Arduino – Arduino : l'essentiel*. https://arduino.blaisepascal.fr/mqtt-avec-arduino/

Relais - Arduino. (2023, août 4). https://arduinofactory.fr/relais/

Introduction · React native. (2024, 15 mai). https://reactnative.dev/docs/getting-started

P, P. (s. d.). *A Beginner's Guide to the ESP8266*. https://tttapa.github.io/ESP8266/Chap07%20-%20Wi-Fi%20Connections.html

Stéphane. (2024, 24 mars). *Installation du broker Mosquitto sur Docker*. Technologie Geek. https://technologie-geek.fr/installation-broker-mosquitto-docker/

7. Glossary

Esp8266 : Microcontroller

React native: Framework for mobile application

AsyncStorage: Local database

Expo Go: Tool for mobile development

Typescript: Programming language

MQTT: Message Queuing Telemetry Transport

8. Appendixes

8.1 Mobile application

https://github.com/SamirAoufia/Project3

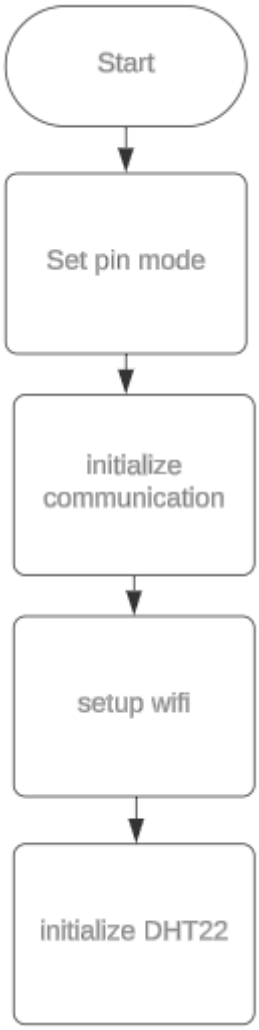
8.2 Arduino code

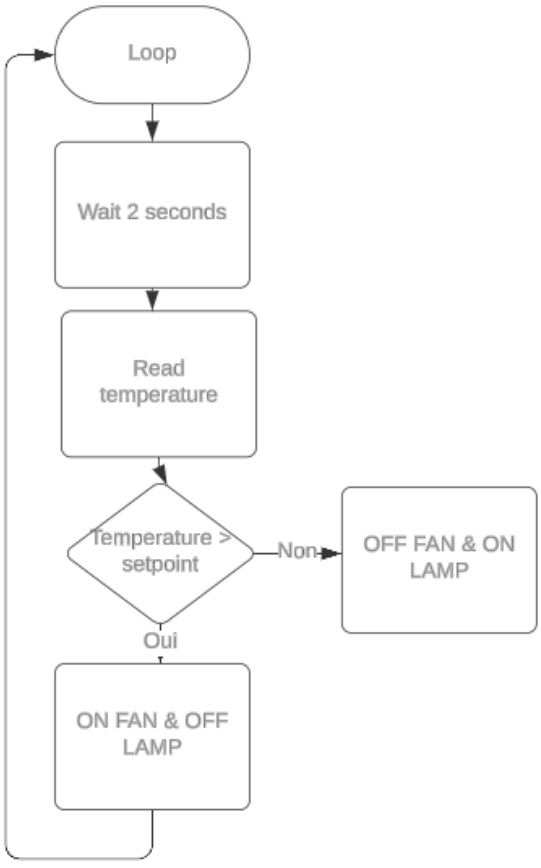
https://github.com/SamirAoufia/ArduinocodeProject3

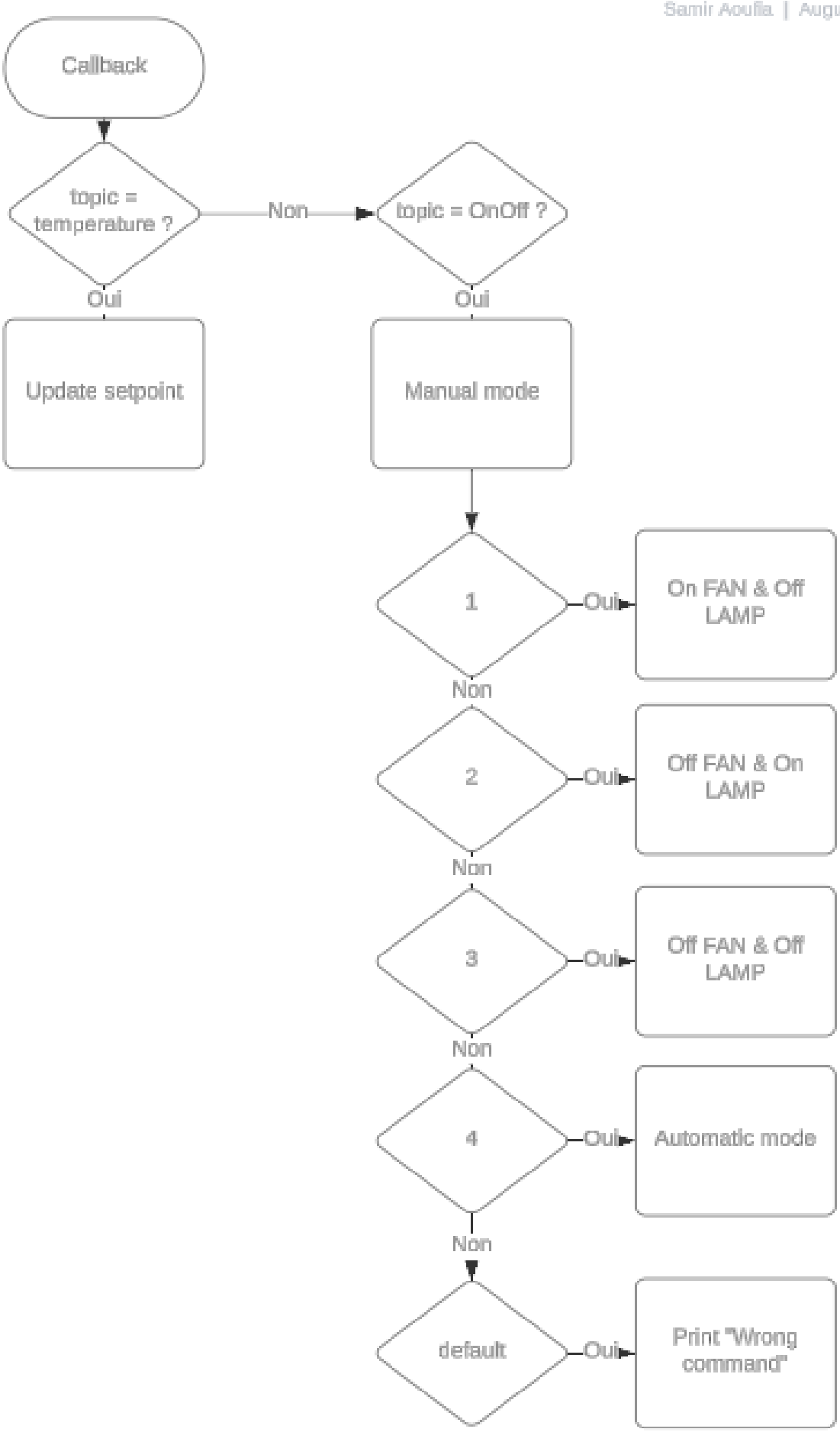
8.3 Docker

https://github.com/SamirAoufia/Docker_project3

8.4 Flowchart diagram






Callback
topic = temperature ?
Non
topic = OnOff ?
Oui
Oui
Update setpoint
Manual mode
1
Oui
On FAN & Off LAMP
Non
2
Oui
Off FAN & On LAMP
Non
3
Oui
Off FAN & Off LAMP
Non
4
Oui
Automatic mode
Non
default
Oui
Print "Wrong command"

8.5 Design brief

HELHa Haute École Louvain en Hainaut — SCIENCES ET TECHNOLOGIES CHARLEROI

Academic year 2023-2024

Informatique Industrielle - Bloc 3 - UE Projet Multidisciplinaire 3 - Epreuve Intégrée
Gestion de projet 2 - Internet des Objets (IOT) - Développement mobile - Anglais professionnel

First Name: Samir Last Name: Aoufia

Project 3 - Design Brief

Context

The aim of this project is to create an IoT-controlled temperature controller using an ESP8266, a temperature sensor, a ventilator and a heat lamp. The main objective is to enable total regulation via a mobile application or directly on the embedded system.

Objective

At first I'll have to set up the control system without iot, then I'll connect the system to the mobile app so that it displays the current temperature, the desired temperature, whether the lamp is on, whether the fan is running.

There'll be a section for setting the setpoint, another for switching lamp ot the ventilator on or off, and lastly for receiving notifications if there's too much temperature variation.

Technical limits

The only technical limit that could happen is to melt the box with heat, otherwise, there's no technical limit, because once the system is in place, it can work alone.

Functional description

The temperature inside the box is always read. If the temperature differs from the user-defined setpoint, the actuator is activated to cool or heat the space until the desired temperature is reached. Users can monitor and control the entire process via the mobile app (switching the lamp or ventilator on or off).