HACETTEPE UNIVERSITY

BBM460- Wireless and Mobile Networks Laboratory

2023-2024 Spring

Final Project Report

IoT Project - Environmental Alert System with NodeMCU
Temperature and Humidity Monitoring

Cavit Bora Öztekeşin

2200356855

Table of Contents

1. Abstract

This project aims to measure temperature and humidity using an ESP8266 microcontroller and a DHT11 sensor. Data is transmitted via MQTT protocol through a local Mosquitto broker and monitored using a Flutter application. The project demonstrates the integration of IoT devices with a mobile application to provide real-time monitoring of environmental conditions.

2. Introduction

The Internet of Things (IoT) has enabled the connection of everyday objects to the internet, allowing them to send and receive data. This project focuses on monitoring temperature and humidity using an ESP8266 microcontroller and a DHT11 sensor. The data collected by the sensor is transmitted to a Mosquitto MQTT broker over a local Wi-Fi network. A Flutter mobile application is used to display the real-time data received from the broker. An important note is that the project was originally set to be implemented using Arduino, however, it was later replaced with NodeMCU in the final stage because it is decided to be easier to use in order to reach the internet , cheaper, and overall more suitable for this project.

3. Technologies

The following technologies were used in this project:

- **ESP8266 Microcontroller/NodeMCU:** A low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability.
- **DHT11 Sensor:** A basic, low-cost digital temperature and humidity sensor.

- **MQTT (Message Queuing Telemetry Transport):** A lightweight messaging protocol for small sensors and mobile devices, optimized for high-latency or unreliable networks.
- **Mosquitto Broker:** An open-source MQTT broker used to manage the messages sent by the sensors and received by the mobile application.
- **Flutter:** An open-source UI software development toolkit created by Google, used to develop the mobile application for displaying the sensor data.

4. System Design

The system is designed to collect environmental data (temperature and humidity) and transmit it to a mobile application for monitoring. The components used in this system include the ESP8266 microcontroller, DHT11 sensor, Mosquitto MQTT broker, and a Flutter mobile application.

The ESP8266 microcontroller reads data from the DHT11 sensor and sends it to the Mosquitto MQTT broker. The Flutter application subscribes to the MQTT topics to receive the data and display it in a user-friendly interface.

Hardware Configuration:

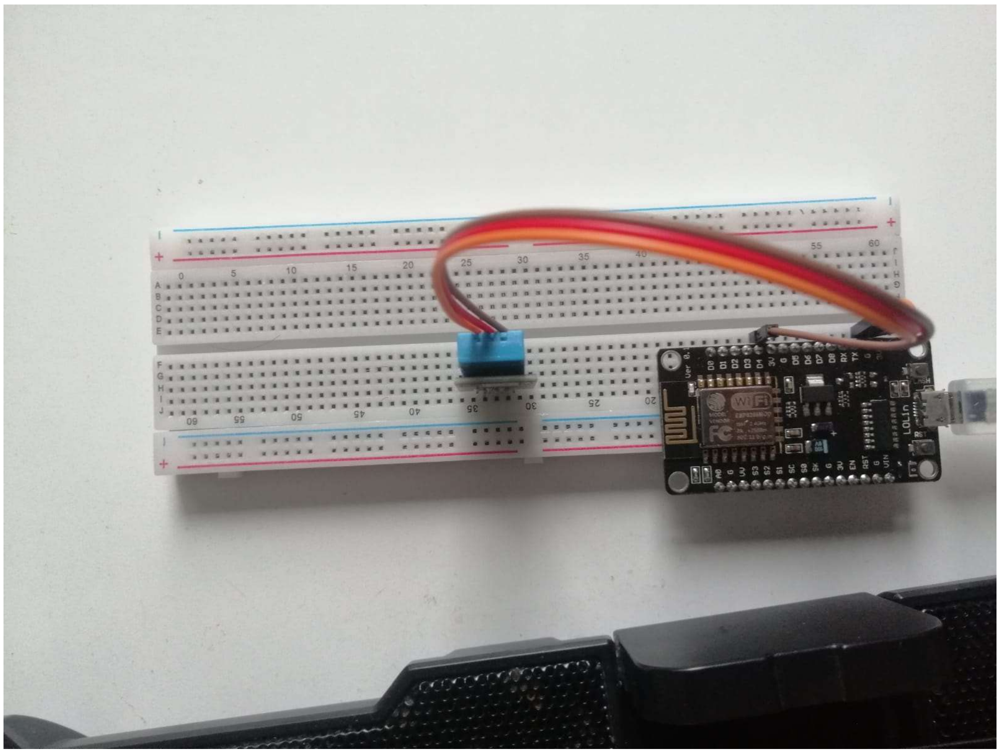
- **ESP8266:** A microcontroller that provides wireless communication capabilities.
- **DHT11:** A sensor that measures temperature and humidity.
- **Connections:**
    - DHT11 VCC -> ESP8266 3.3V
    - DHT11 GND -> ESP8266 GND
    - DHT11 Data -> ESP8266 D4 (GPIO 2)

5. Method

The following steps were taken to implement the project:

Step 1: Connect the DHT11 sensor to the ESP8266 microcontroller.

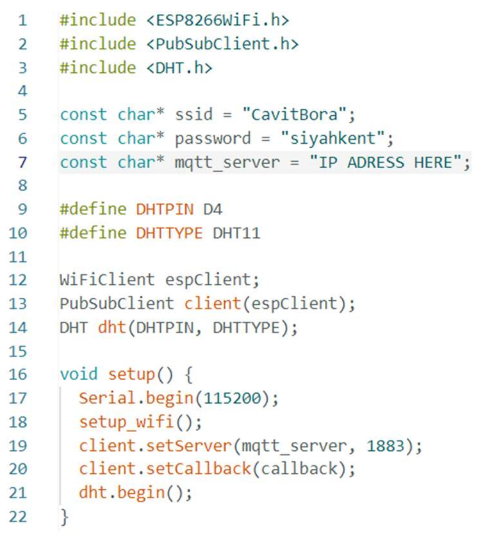
1. **Hardware Setup:**
    - Place the ESP8266 and DHT11 sensor on a breadboard.
    - Connect the VCC pin of the DHT11 sensor to the 3.3V pin of the ESP8266.
    - Connect the GND pin of the DHT11 sensor to the GND pin of the ESP8266.
    - Connect the Data pin of the DHT11 sensor to the D4 pin of the ESP8266.



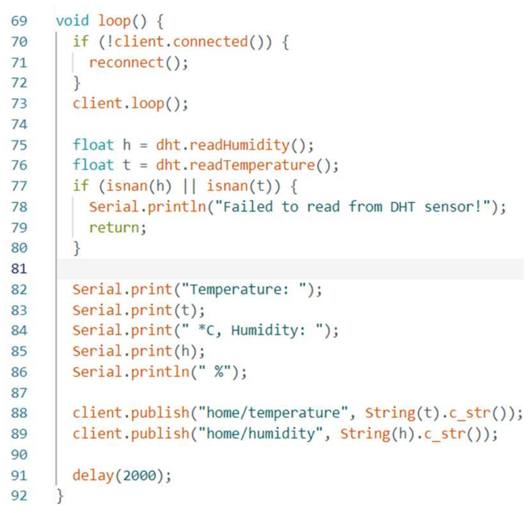
Step 2: Write code for the ESP8266 to read data from the DHT11 sensor and send it to the Mosquitto MQTT broker.

2. **Code for ESP8266:**

    - Include the necessary libraries: **ESP8266WiFi.h**, **PubSubClient.h**, and **DHT.h**.
    - Define the Wi-Fi SSID and password, as well as the MQTT broker address.
    - Initialize the DHT11 sensor and set up the Wi-Fi and MQTT connections.
    - In the **loop** function, read the temperature and humidity data from the DHT11 sensor and publish it to the MQTT topics.



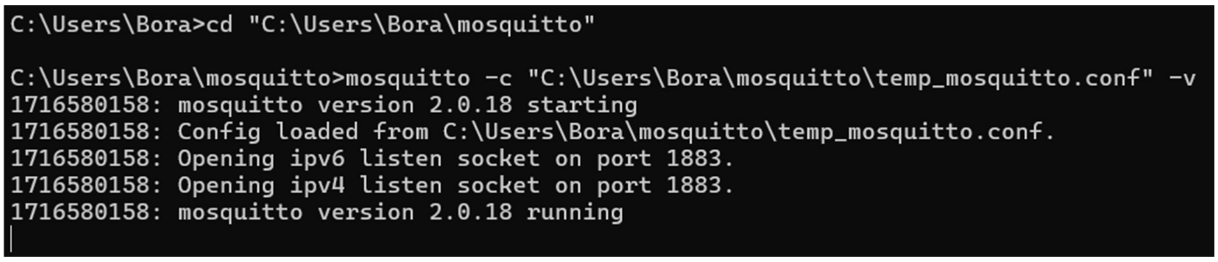
```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

const char* ssid = "CavitBora";
const char* password = "siyahkent";
const char* mqtt_server = "IP ADRESS HERE";

#define DHTPIN D4
#define DHTTYPE DHT11

WiFiClient espClient;
PubSubClient client(espClient);
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  dht.begin();
}
```



```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  float h = dht.readHumidity();
  float t = dht.readTemperature();
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C, Humidity: ");
  Serial.print(h);
  Serial.println(" %");

  client.publish("home/temperature", String(t).c_str());
  client.publish("home/humidity", String(h).c_str());

  delay(2000);
}
```

Step 3: Set up the Mosquitto MQTT broker on a local machine.

3. **MQTT Broker Setup:**
    - Download and install Mosquitto on the local machine.
    - Create a configuration file (**mosquitto.conf**) to allow anonymous connections and specify the port (1883).
    - Start the Mosquitto broker using the command: **mosquitto -c "path_to_mosquitto.conf" -v**.

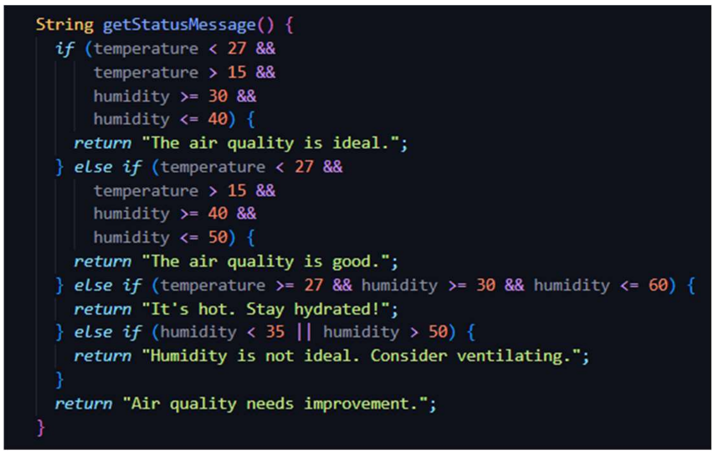


```
C:\Users\Bora>cd "C:\Users\Bora\mosquitto"

C:\Users\Bora\mosquitto>mosquitto -c "C:\Users\Bora\mosquitto\temp_mosquitto.conf" -v
1716580158: mosquitto version 2.0.18 starting
1716580158: Config loaded from C:\Users\Bora\mosquitto\temp_mosquitto.conf.
1716580158: Opening ipv6 listen socket on port 1883.
1716580158: Opening ipv4 listen socket on port 1883.
1716580158: mosquitto version 2.0.18 running
```

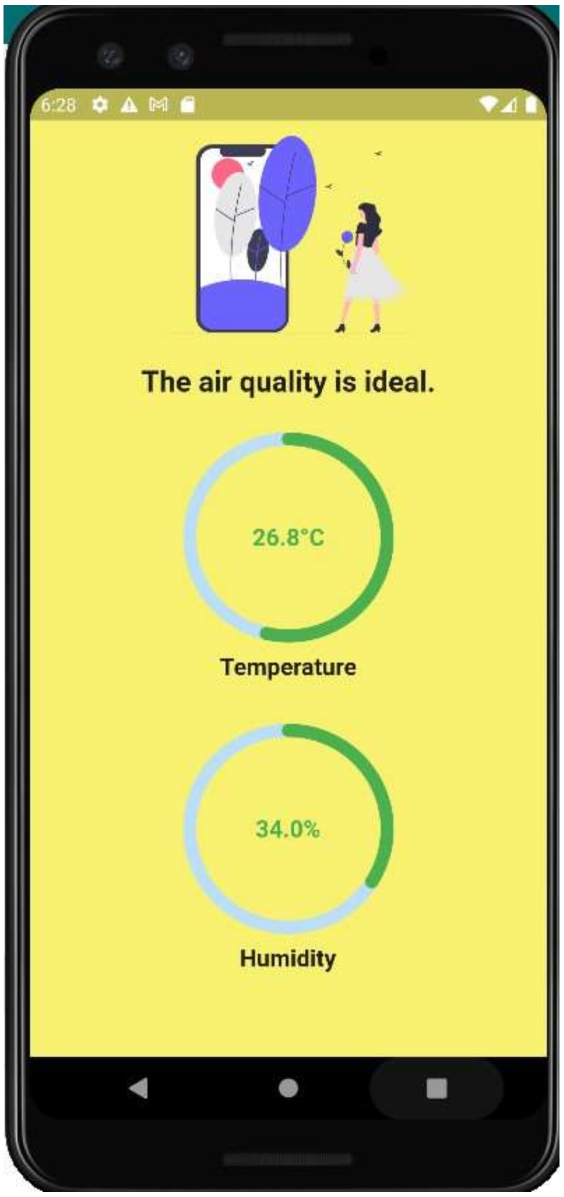
Step 4: Develop a Flutter application to subscribe to the MQTT topics and display the data.

Flutter Application:

- Create a new Flutter project.
- Add dependencies for **mqtt_client**, **percent_indicator**, and **flutter_svg**.
- In the main Dart file, connect to the MQTT broker and subscribe to the temperature and humidity topics.
- Display the data using circular progress indicators, with different colors indicating the status of the environment.
- Update the temperature and humidity information every couple of seconds to keep the information updated.
- Depending on the current levels of humidity and temperature, inform the user about whether the air quality is ideal , or good enough or bad. In case of bad air quality, warn the user about it and suggest actions.



```
String getStatusMessage() {
  if (temperature < 27 &&
      temperature > 15 &&
      humidity >= 30 &&
      humidity <= 40) {
    return "The air quality is ideal.";
  } else if (temperature < 27 &&
      temperature > 15 &&
      humidity >= 40 &&
      humidity <= 50) {
    return "The air quality is good.";
  } else if (temperature >= 27 && humidity >= 30 && humidity <= 60) {
    return "It's hot. Stay hydrated!";
  } else if (humidity < 35 || humidity > 50) {
    return "Humidity is not ideal. Consider ventilating.";
  }
  return "Air quality needs improvement.";
}
```



6:28

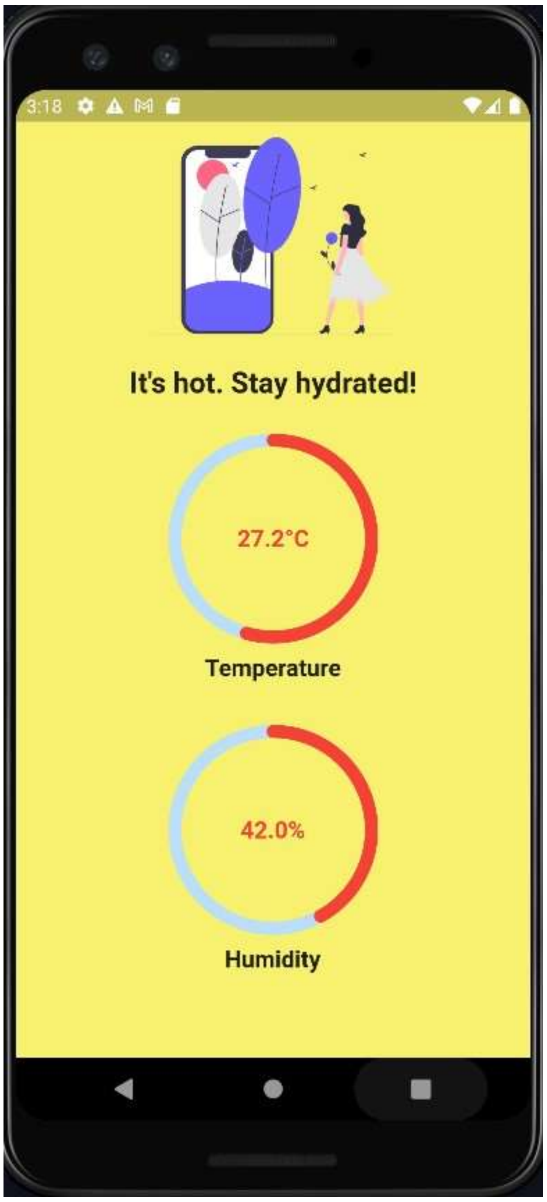
The air quality is ideal.

26.8°C

Temperature

34.0%

Humidity



3:18

It's hot. Stay hydrated!

27.2°C

Temperature

42.0%

Humidity

6. Conclusion

This project successfully demonstrated the integration of IoT devices with a mobile application to monitor environmental conditions in real-time. The ESP8266 microcontroller and DHT11 sensor effectively collected temperature and humidity data, which was transmitted via MQTT to a Mosquitto broker and displayed on a Flutter application. Future improvements could include adding more sensors and enhancing the mobile application's features.

7. References

1. MQTT Protocol: https://mqtt.org/
2. Mosquitto Broker: https://mosquitto.org/
3. ESP8266 Documentation: https://www.espressif.com/en/products/socs/esp8266
4. DHT11 Sensor: https://www.adafruit.com/product/386
5. Flutter: https://flutter.dev/