***Phil Nevins******ECE 558 – Embedded Systems Programming******February 19, 2025***[*GitHub Repository*](https://github.com/Dawgburt/ECE558-Project2)

# **IoT Weather Station & LED Controller Project Report**

## **1. Introduction**

The ECE 558 IoT Weather Station is an integrated **Raspberry Pi 5 and Android** application that collects and displays **real-time environmental data**. This project was developed for **embedded systems programming and IoT communication using MQTT**.

Additionally, **LED control functionality** was added to allow users to **adjust an RGB LED connected to the Raspberry Pi 5** using the Android app. The app features **manual text input fields** for **R, G, B values (0-255)** as well as a **color spectrum slider** that smoothly transitions across the **entire RGB spectrum**.

## **2. Features**

### **2.1 Sensor Data Collection & Transmission**

* **BME688 Sensor**: Reads **temperature, humidity, pressure, and gas resistance** values.
* **Python MQTT Publisher**: The Raspberry Pi 5 reads sensor data and **publishes it to an MQTT broker every second**.

### **2.2 MQTT Communication**

* **Mosquitto MQTT Broker** hosted on the Raspberry Pi 5.
* **Android app subscribes** to "weather/station" topic and receives **live updates**.
* **LED Control**: The app publishes **RGB values** to the topic "led/control", and the Raspberry Pi **adjusts the LED accordingly**.

### **2.3 Android App Display & LED Control**

* **Real-time sensor readings** dynamically updated.
* **Graph Visualization** using **MPAndroidChart**.
* **Graph Selection**: Users can tap a sensor reading to **switch between graphs**.
* **RGB LED Control**:
  + **Text Inputs**: Users manually enter **R, G, and B values (0-255)**.
  + **Color Slider**: A **hue-based slider** dynamically adjusts the LED color **across the entire RGB spectrum**.
  + **"Set LED" Button**: Publishes the entered **RGB values to MQTT**.
  + **"Clear" Button**: Turns **off the LED and resets the controls**.

### **2.4 Adaptive Graph Scaling**

* **Dynamic Y-axis scaling** to accommodate different sensor ranges.
* **Auto-scrolling X-axis** for real-time data flow.

## **3. Technologies Used**

### **3.1 Programming Languages & Frameworks**

* **Python 3**: Raspberry Pi sensor data processing & MQTT publishing.
* **Kotlin (Jetpack Compose)**: Android app development.
* **Mosquitto MQTT**: Private MQTT broker for data transmission.

### **3.2 Libraries & Tools**

* **Adafruit CircuitPython BME680**: Sensor communication.
* **Paho-MQTT**: MQTT data publishing.
* **MPAndroidChart**: Real-time graphing.
* **GPIOZero**: GPIO control for LED lighting.

## **4. System Architecture**

### **4.1 Raspberry Pi 5: Sensor Data Collection & MQTT Publishing**

* Reads sensor values and publishes data to "weather/station" every second.
* Configured **Mosquitto MQTT broker** for **private communication**.
* Listens for LED control messages from the Android app on the **"led/control"** topic.

### **4.2 Android Application: MQTT Subscription & Data Display**

* Subscribes to "weather/station" and updates UI in **real-time**.
* Displays **temperature, humidity, pressure, and gas resistance**.
* Implements **graph selection UI** for interactive data visualization.
* Sends **RGB LED control messages** via MQTT.

## **5. Implementation Details**

### **5.1 Raspberry Pi 5: Python MQTT Publisher**

import paho.mqtt.publish as publish

MQTT\_SERVER = "iotweatherstation.ddns.net"

MQTT\_TOPIC = "weather/station"

data = {

"temperature": bme680.temperature,

"humidity": bme680.humidity,

"pressure": bme680.pressure,

"gas": bme680.gas

}

publish.single(MQTT\_TOPIC, payload=json.dumps(data), hostname=MQTT\_SERVER)

### **5.2 LED Control via MQTT on Raspberry Pi**

import paho.mqtt.client as mqtt

from gpiozero import PWMLED

import threading

# Define RGB LED pins

red\_led = PWMLED(16)

green\_led = PWMLED(20)

blue\_led = PWMLED(21)

def handle\_led\_message(payload):

try:

if payload.lower() == "clear":

red\_led.value = 0

green\_led.value = 0

blue\_led.value = 0

else:

r, g, b = map(int, payload.split(","))

red\_led.value = r / 255.0 # Convert 0-255 to 0-1

green\_led.value = g / 255.0

blue\_led.value = b / 255.0

except ValueError:

print("Invalid RGB values received.")

def on\_message(client, userdata, message):

payload = message.payload.decode().strip()

print(f"Received MQTT message: {payload}")

# Run LED handling in a separate thread

threading.Thread(target=handle\_led\_message, args=(payload,)).start()

## **7. Challenges & Solutions**

### **7.4 Pigpio Library Failure**

* **Problem**: The pigpio library **failed to start** properly on the Raspberry Pi 5, preventing GPIO control.
* **Solution**: **Uninstalled pigpio and switched to GPIOZero**, which worked without issues.

### **7.5 Implementing Real-Time LED Control**

* **Problem**: Sending **RGB values via MQTT** had **delays and incorrect formatting**.
* **Solution**:
  + Implemented **manual text input** for **precise RGB entry**.
  + Added **hue slider** for **smooth, continuous LED transitions**.
  + Used **GPIOZero PWMLED** to allow **smooth brightness control**.

### **7.6 Sensor Data Stopping After LED Control**

* **Problem**: Adjusting the LED **stopped sensor updates** from appearing.
* **Cause**: The **MQTT** on\_message() **function** for LED control was **blocking the main event loop**.
* **Solution**:
  + **Moved LED handling to a separate thread** using threading.Thread(), so it doesn't interfere with the sensor updates.
  + **Ensured the Raspberry Pi subscribes to both topics (**weather/station **and** led/control**)**.
  + **Throttled MQTT messages** from the Android app to prevent overload.

## **8. Future Improvements**

**Implement data logging** for historical analysis.  
**Add alerts** for extreme weather conditions.  
**Improve UI/UX** with smoother graph transitions.  
**Enhance LED control** with **preset color modes** (e.g., fading, pulsing).

## **9. Conclusion**

The **ECE 558 IoT Weather Station** successfully integrates a **Raspberry Pi 5, BME688 sensor, and Android app** to provide **real-time environmental monitoring**. The addition of **RGB LED control via MQTT** enhances user interaction. The **sensor update issue after LED control was resolved** by **moving LED handling to a background thread**. Future work will focus on **data logging and expanded LED effects**.

## Acknowledgements

ChatGPT

Eclipse Paho MQTT Client Libraries. (n.d.). *Eclipse Foundation*. Retrieved from <https://github.com/eclipse-paho>

HiveMQ. (n.d.). *MQTT Client Library Encyclopedia: Paho Android Service*. Retrieved from <https://www.hivemq.com/blog/mqtt-client-library-enyclopedia-paho-android-service/>

HiveMQ. (n.d.). *MQTT Client Library Encyclopedia: Paho Python*. Retrieved from <https://www.hivemq.com/blog/mqtt-client-library-paho-python/>

Mosquitto MQTT Broker. (n.d.). *Eclipse Foundation*. Retrieved from <https://mosquitto.org/>

Raspberry Pi Foundation. (n.d.). *Raspberry Pi 5 Technical Specifications*. Retrieved from <https://www.raspberrypi.org/products/raspberry-pi-5/>

Adafruit Industries. (n.d.). *BME688 Sensor Guide*. Retrieved from <https://learn.adafruit.com/adafruit-bme688>

GPIOZero Library for Raspberry Pi. (n.d.). *GPIOZero Documentation*. Retrieved from <https://gpiozero.readthedocs.io/en/stable/>

MPAndroidChart Library. (n.d.). *GitHub Repository for MPAndroidChart*. Retrieved from <https://github.com/PhilJay/MPAndroidChart>

Android Jetpack Compose. (n.d.). *Jetpack Compose Documentation*. Retrieved from <https://developer.android.com/jetpack/compose>

Adafruit AHT20 Sensor Documentation. (n.d.). *Adafruit Learning System*. Retrieved from <https://learn.adafruit.com/adafruit-aht20>

MQTT Essentials: A Lightweight IoT Protocol. (2017). *Gaston Hiller, Packt Publishing*. Retrieved from <https://subscription.packtpub.com/book/application-development/9781787287815>