EMBEDDED SYSTEM WITH IOT
EXPERIMENT NO. 3

IoT based Weather Monitoring System

Name: NAVARRO, ROD GERYK C.

Course/Section: CPE162P-4/E01

Date of Performance: 05/14/2025
Date of Submission: 05/18/2025

CYREL O. MANLISES, PH.D.
Instructor

DISCUSSION

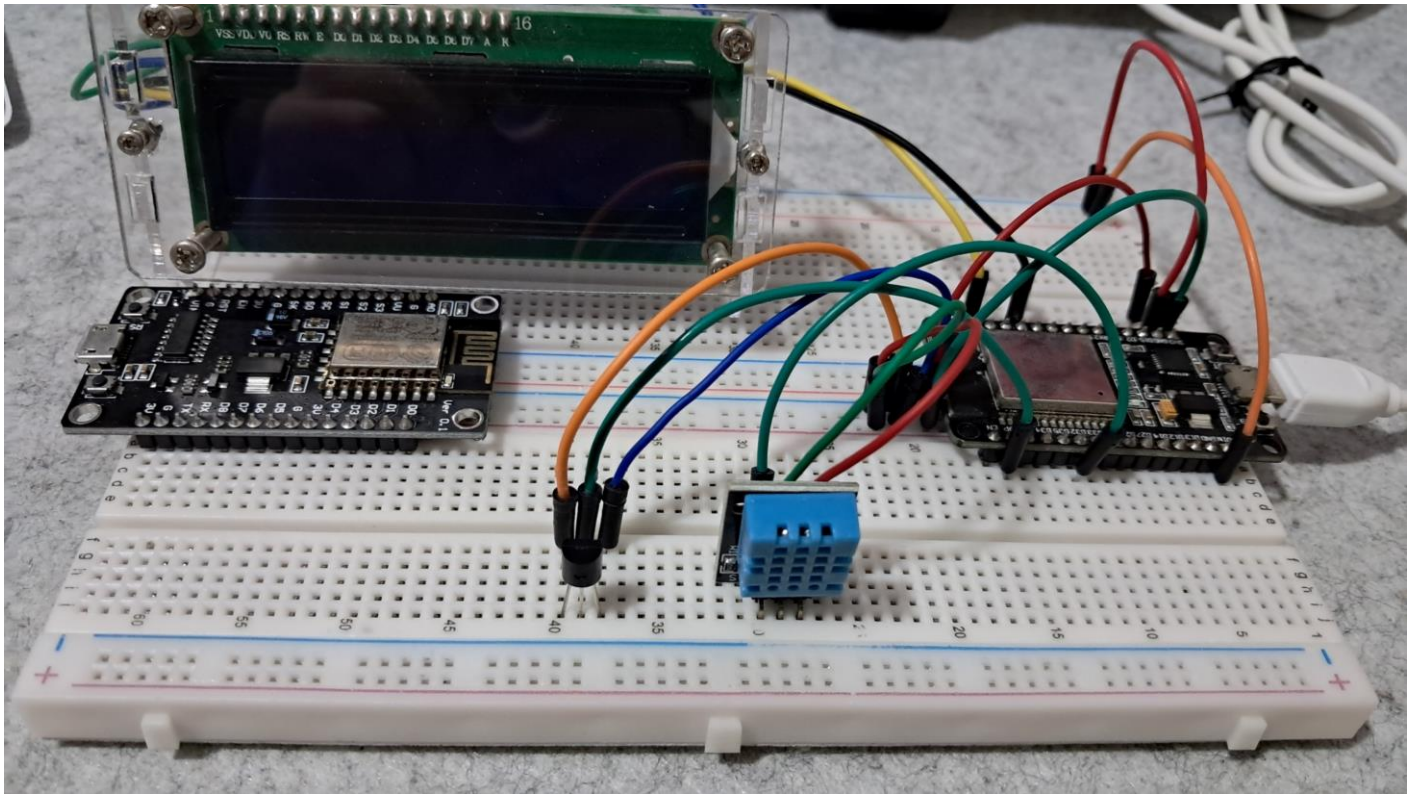


FIGURE 1: The connection of the LM35 sensor, DHT11 sensor, and LCD to the ESP32 dev module

Connections

This week, I worked on an IoT-based weather monitoring system. The system can sense the temperature and humidity of the surroundings. Just like in my previous experiments, I applied what I learned from subjects like logic circuits, microprocessors, switching theory, and embedded systems. For this experiment, I used an LM35 temperature sensor to get accurate temperature readings, and a DHT11 sensor to measure humidity. Even though the DHT11 can also detect temperature, I only used it for humidity because the LM35 is more accurate for temperature. I also used an LCD to display the current temperature and humidity in real-time, and the ESP32 Dev Module as the main microcontroller. The ESP32 has built-in Wi-Fi, which allowed me to send data to the internet.

The first thing I did was connect all the components to the ESP32. Once everything was wired correctly, I wrote the code using the Arduino IDE to control how each part works together. For monitoring and analyzing the data, I used ThingSpeak – an IoT platform that helps track and visualize changes in data. The ESP32 collects the data from the sensors and sends it to ThingSpeak, allowing me to monitor temperature and humidity from different locations.

Code Development

Navarro_EXP3_CPE162P.ino

```
#include <DHT.h>
#include <LiquidCrystal_I2C.h>
#include <WiFi.h>
#include <ThingSpeak.h>

// WiFI credentials
const char* ssid = "PLDTHOMEFIBRt6ucX";
const char* password = "PLDTWIFItLK72";

// ThingSpeak setup
WiFiClient client;
long myChannelNumber = 2964917;
const char myWriteAPIKey[] = "18A6ZBU4I3OLZZ44";

#define DHTPIN 32   // DHT11 data pin connected to GPIO32
#define DHTTYPE DHT11
#define THERMISTOR_PIN 36 // Analog sensor pin GPIO36

DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);

unsigned long lastUpdate = 0;
const unsigned long updateInterval = 15000; // 15 seconds minimum for thingspeak

// average temperature
float tempSum = 0;
int tempCount = 0;
unsigned long avgStartTime = 0;
unsigned long avgInterval = 300000; // 5 minutes in milliseconds
```

FIGURE 2: The Initialization of Variables and Components

In this experiment, I started by including the necessary libraries for the sensors, LCD, Wi-Fi, and ThingSpeak. I defined the Wi-Fi network name and password so that the ESP32 can connect to the internet. Then, I set the

channel number and API key for ThingSpeak so the data could be sent properly. I also set the pins for the DHT11 sensor and LM35 sensor. The DHT11 is connected to GPIO32, and the LM35 analog sensor is connected to GPIO36. I created objects for the DHT sensor and the LCD display to make it easier to use them in the code. Lastly, I initialized some variables to help with updating data every 15 seconds and computing the average temperature every 5 minutes.

Navarro_EXP3_CPE162P.ino

```
30
31  void setup() {
32    Serial.begin(9600);
33    dht.begin();
34    lcd.init();
35    lcd.backlight();
36
37    lcd.setCursor(0, 0);
38    lcd.print("   My Weather");
39    lcd.setCursor(0,1);
40    lcd.print("System is ON");
41    delay(3000);
42    lcd.clear();
43
44    Serial.print("Connecting to ");
45    Serial.println(ssid);
46    WiFi.begin(ssid, password);
47    while(WiFi.status() != WL_CONNECTED){
48      delay(500);
49      Serial.print(".");
50    }
51    Serial.println("\nWiFi connected");
52    Serial.print("IP address: ");
53    Serial.println(WiFi.localIP());
54
55    ThingSpeak.begin(client);
56
57    avgStartTime = millis(); // start timer for average temperature
58  }
```

FIGURE 3: The Setup of the Components

Inside the setup() function, I started by opening the serial monitor using Serial.begin() to help me see the sensor readings later. Then, I initialized the DHT11 sensor and the LCD, turned on the LCD backlight, and displayed a welcome message for 3 seconds. After that, I connected the ESP32 to Wi-Fi using the credentials I provided earlier. The code keeps checking until the connection is successful. Once connected, it shows the IP

address of the ESP32 on the serial monitor. I also initialized ThingSpeak with the client object so it would be ready to receive and send data. Lastly, I started the timer that would be used to calculate the average temperature every 5 minutes.

Navarro_EXP3_CPE162P.ino

```
60  void loop() {
61    // reading analog temperature - LM35
62    int sensorValue = analogRead(THERMISTOR_PIN);
63    float voltage = sensorValue * (3.3 / 4095.0); // esp32 ADC is 12-bit
64    float temperature = voltage * 100.0;   // LM35: 10mv per Celsius
65
66    float humidity = dht.readHumidity();  // read DHT11 sensor
67
68    // Handle errors
69    if (isnan(humidity)) {
70      Serial.println("Failed to read from DHT sensor!");
71      humidity = 0;
72    }
73
74    // debug output
75    Serial.print("Voltage: ");
76    Serial.println(voltage, 2);
77    Serial.print("Temperature: ");
78    Serial.print(temperature, 1);
79    Serial.println(" °C");
80    Serial.print("Humidity: ");
81    Serial.print(humidity, 1);
82    Serial.println(" %");
83
84    // display on LCD
85    lcd.setCursor(0, 0);
86    lcd.print("   Monitoring  ");
87    lcd.setCursor(0, 1);
88    lcd.print("T:");
89    lcd.print(temperature, 1);
90    lcd.print((char)223);
91    lcd.print("C H:");
92    lcd.print(humidity, 1);
93    lcd.print("%");
94
95    // accumulate for average
96    tempSum += temperature;
97    tempCount++;
```

Navarro_EXP3_CPE162P.ino

```
92    lcd.print(humidity, 1);
93    lcd.print("%");
94
95    // accumulate for average
96    tempSum += temperature;
97    tempCount++;
98
99    //Send current data every 15 seconds
100   if (millis() - lastUpdate >= updateInterval){
101     lastUpdate = millis();
102
103     ThingSpeak.setField(1, temperature);   //Field 1 = temperature
104     ThingSpeak.setField(2, humidity);  //Field 2 = humidity
105
106     // only send average if 5 minutes have passed
107     if (millis() - avgStartTime >= avgInterval){
108       float avgTemp = tempSum / tempCount;
109       ThingSpeak.setField(3, avgTemp); // Field 3 = average temp
110
111       Serial.print("5-minute average temp");
112       Serial.println(avgTemp, 2);
113
114       // Reset accumulators
115       tempSum = 0;
116       tempCount = 0;
117       avgStartTime = millis();
118     }
119     int statusCode = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
120     if (statusCode == 200){
121       Serial.println("Channel update successfull.");
122   } else {
123     Serial.print("ThingSpeak update failed. Code: ");
124     Serial.print(statusCode);
125       }
126     }
127     delay(1000); // reduce flicker and allow sensor to settle
128   }
129
```

FIGURE 4: The Loop Function

The loop() function is where the main configuration happens. First, the ESP32 reads the temperature from the LM35 sensor using analog input and converts that reading into a voltage and then into degrees Celsius. It also reads the humidity from the DHT11 sensor. If the DHT11 fails to give a reading, it prints an error and sets the humidity to 0 to avoid crashing the program. After that, the LCD shows the current temperature and humidity in real-time. It also adds the temperature to a running total so that we can compute the average later. Every 15 seconds, the ESP32 sends the current temperature and humidity to ThingSpeak. If 5 minutes have passed, it calculates the average temperature and also sends that to ThingSpeak. After sending, it resets the total and count for the next average calculation. The delay(1000) at the end helps avoid too much flickering on the LCD and gives the sensors time to update.

FIGURE 5: Serial Monitor

The serial monitor is very helpful in this experiment. It shows the voltage from the LM35, the current temperature in Celsius, and the humidity percentage from the DHT11 sensor. This lets me check if the sensors are working correctly. It also prints a message if the DHT11 fails to read. When the data is sent to ThingSpeak, the serial monitor shows if the update was successful or not, including any error codes. After every 5 minutes, it prints the average temperature so I can see the trend over time. These outputs help me understand what's happening in real time and are useful for debugging the system.

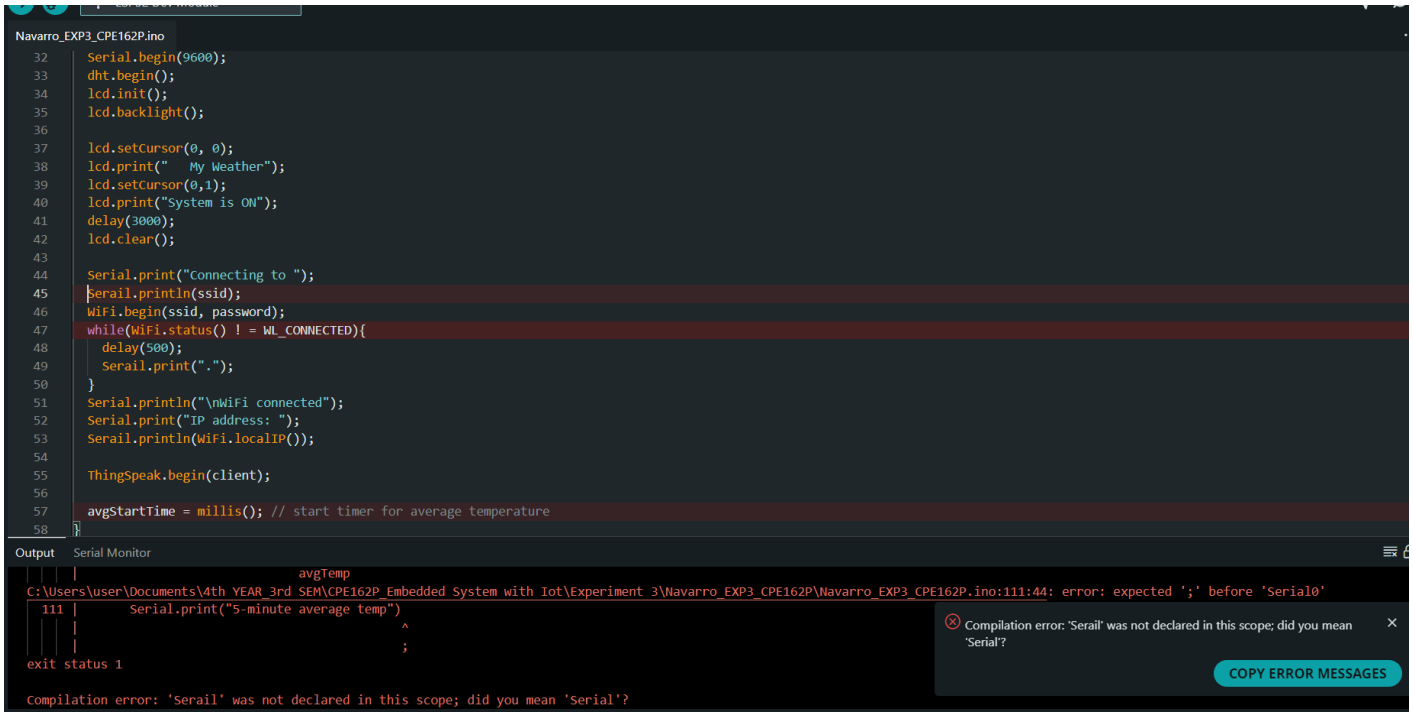


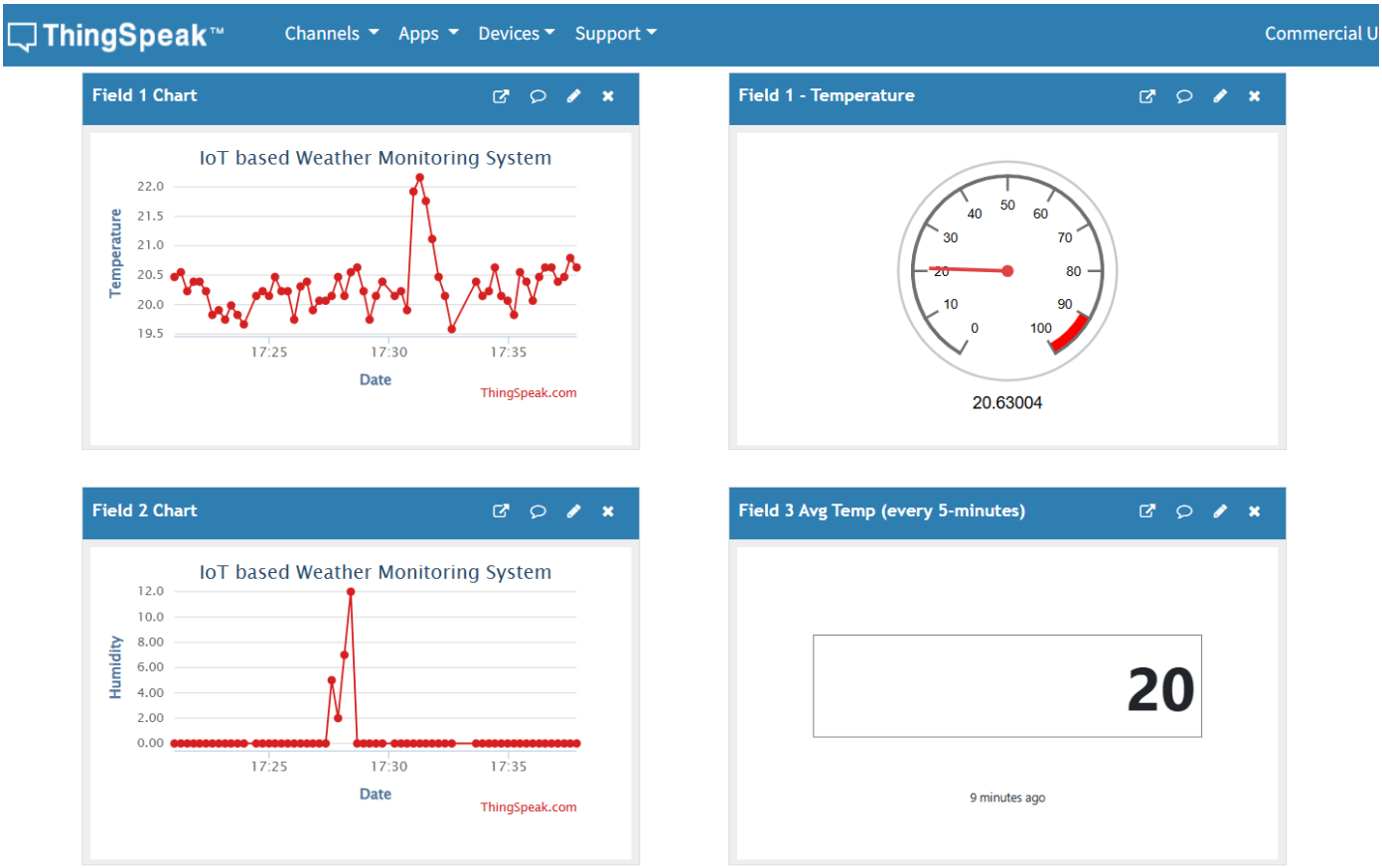
FIGURE 6: Fixing the Errors

The image shown in Figure 6 is one of the errors I encountered during the creation of the code. This error is caused by many misspelled words in the code, as well as many lines that need semi colons are missing. So, I reviewed the code and find the origin of each errors. Another thing I faced is the DHT11 sensor not detecting any humidity even if I try to blow directly it or exposed it to high humidity placed. However, I solved the problem by reconnecting the pins, the problem was the connections of the pins. There are various pin connections of the DHT11 sensor in the internet and I copied a different variant of DHT11 sensor connections which I thought it was the same as mine. After this, my DHT11 sensor began to work.

Difficult Aspect of the Experiment's Development

The most difficult part of the experiment was fixing the accuracy of the sensors. First, I used the DHT11 sensor to detect humidity. I already knew that this sensor wasn't very accurate, but I still tried different ways to improve its performance. However, I kept getting the same results. I noticed that the DHT11 struggles to detect very low humidity levels and needs to be placed very close to a humid area to work properly. Another challenge was improving the accuracy of the LM35 temperature sensor. Sometimes it gave correct readings, but most of the time, the temperature didn't match the actual temperature in my location. I tried several solutions like recalibrating the sensor, adjusting the formula to convert voltage to degrees Celsius, and using shorter wires for better connection. Even with these efforts, the readings were still inconsistent. Although both sensors worked, their readings were slightly off compared to the real temperature and humidity in my area.

Interpretations of ThingSpeak Visualization



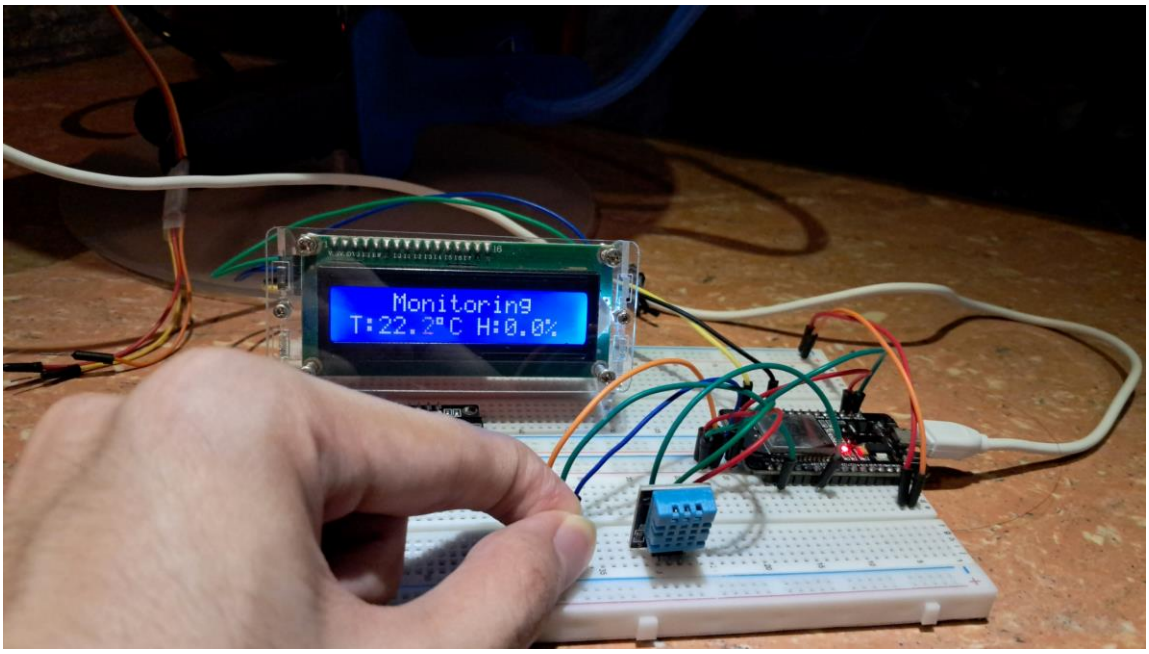


FIGURE 7: ThingSpeak Visualization: Covering the LM35 Temperature Sensor with my Hand

In the image above, the temperature graph shows a big increase when I covered the LM35 sensor with my hand. Normally, the temperature stays around 20 degrees Celsius, but when I removed my hand, the temperature went back to normal. Meanwhile, the humidity stayed at 0%. However, before covering the LM35, I blew air directly onto the DHT11 sensor, which caused a sudden increase in the humidity shown in the graph.

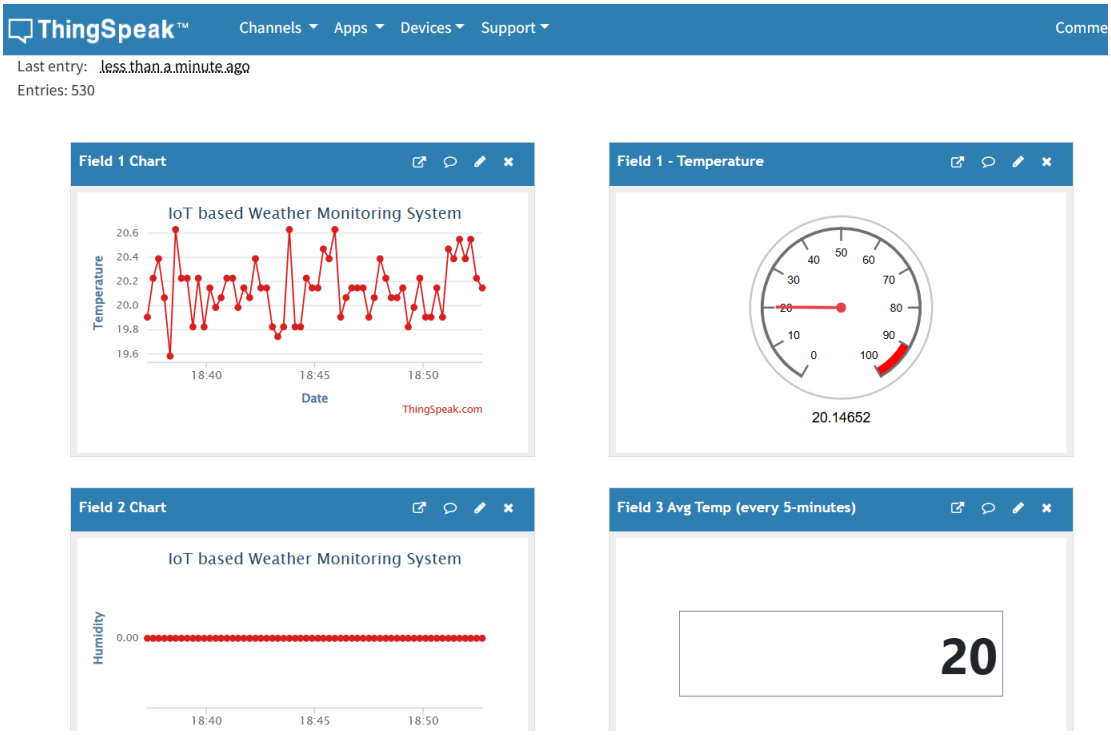


FIGURE 8: ThingSpeak Visualization: Placing the Circuit in our Kitchen

When we look at the temperature graph for the kitchen, we can see that the temperature readings are not stable. The highest temperature goes a little above 20.6 C, while the lowest drops slightly below 19.6 C. Even with these changes, the average temperature stays around 20 C. Meanwhile, the humidity in this area remains steady at 0%.

FIGURE 9: ThingSpeak Visualization: Placing the Circuit in a Bedroom without Aircon

In this location, there was a small increase in temperature. Most of the recorded temperatures were between 20.4 and 21 degrees Celsius. However, there was one time when the temperature dropped to 19 degrees Celsius. Since most of the temperatures were on the higher side, the average temperature ended up being 21 degrees Celsius, which is one degree higher than the previous locations. The humidity in this area stayed at 0%.

FIGURE 10: ThingSpeak Visualization: Placing the Circuit in a Bag with Things Inside

Most of the temperature readings in this location stayed steady at 20 degrees Celsius. There were only a few times when the temperature dropped below 20. Based on these readings, the average temperature was 20 degrees Celsius. The humidity stayed constant at 0%.

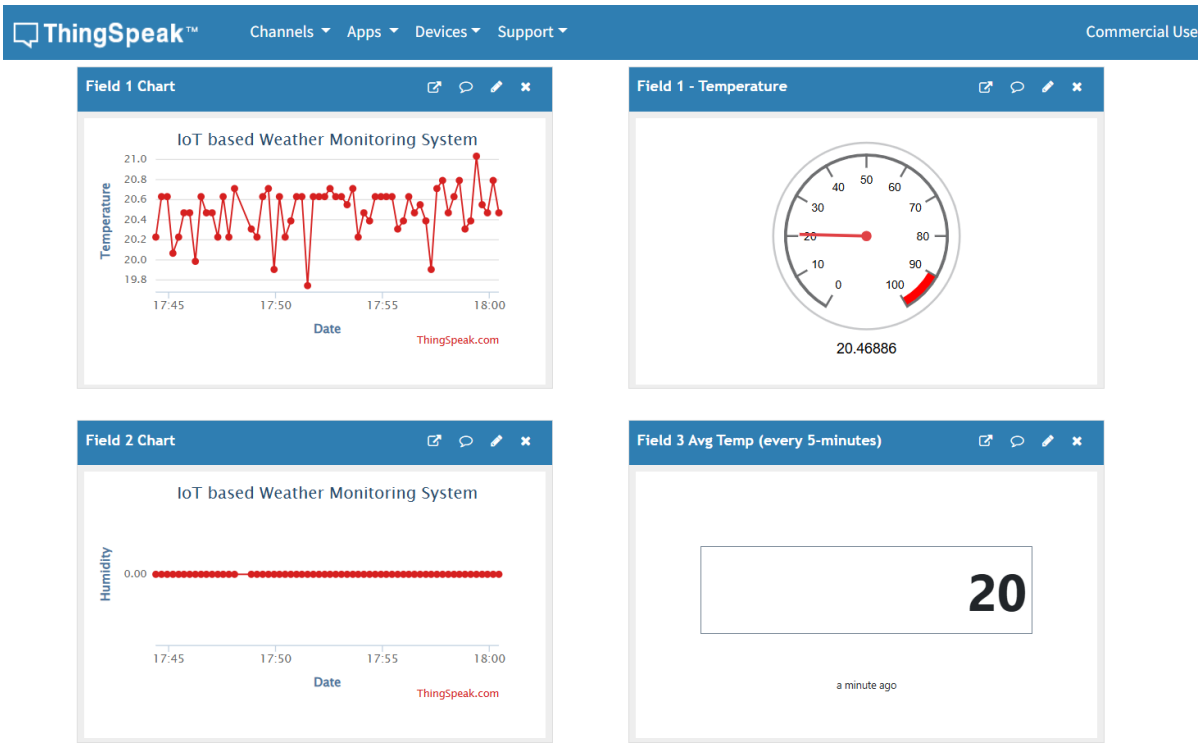
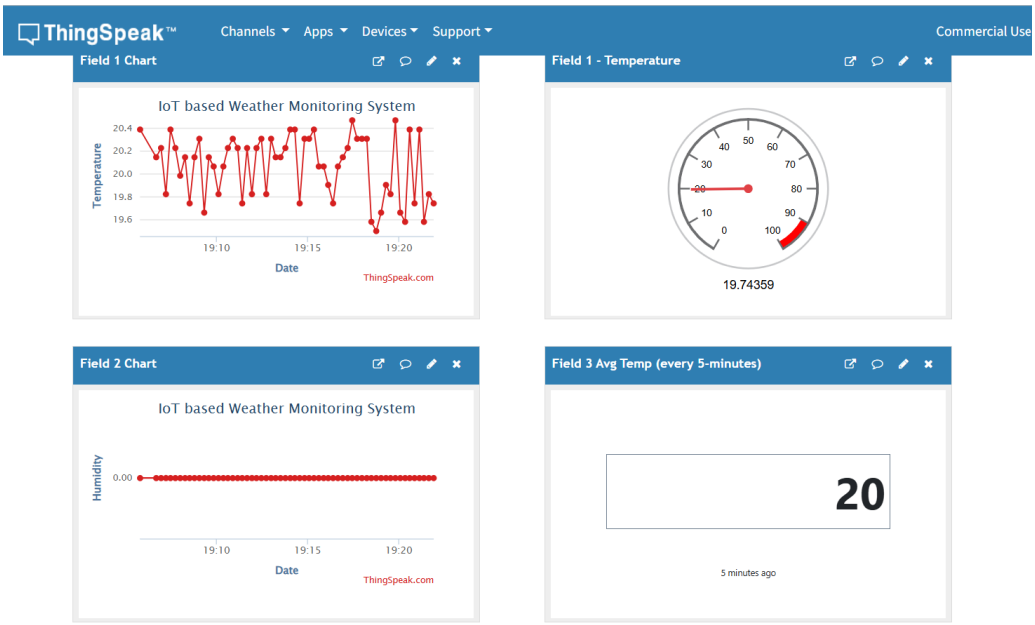


FIGURE 11: ThingSpeak Visualization: Placing the Circuit in a Bag with no Things Inside

In the graph, we can see the difference between a bag with items inside and a bag without them. When the bag is empty, most of the temperature readings stay between 20 to 20.6 degrees Celsius, which is slightly lower than when there are items inside. Also, there are more points where the temperature drops below 20 degrees. Even with these changes, the average temperature stays around 20 degrees Celsius. The humidity remains at 0% in both cases.



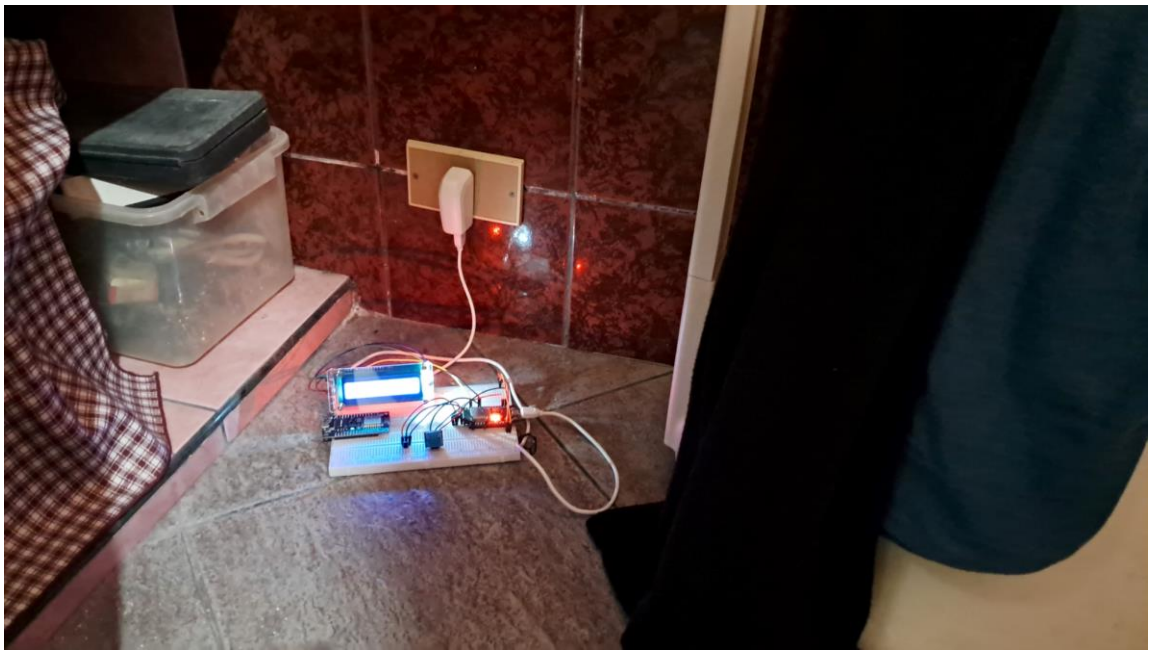
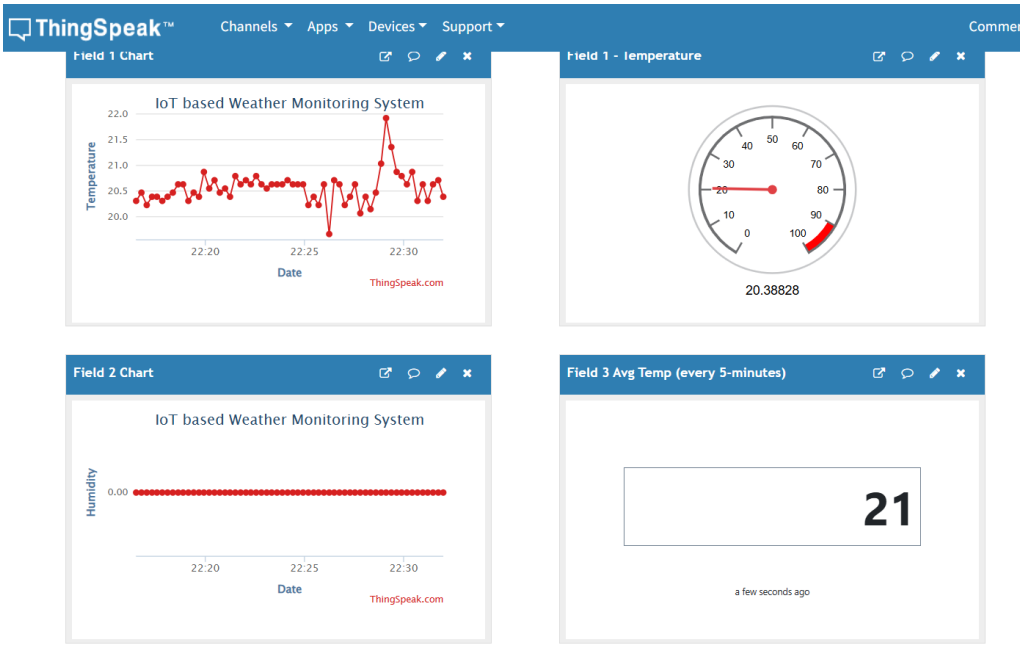


FIGURE 12: ThingSpeak Visualization: Placing the Circuit Inside the Bathroom

The temperature graph is hard to understand because the readings go up and down a lot, with some values much higher or lower than normal. This big change in temperature could be caused by different things like wind from the window, moisture from water, and other factors. Even though this part of the house has higher humidity compared to other areas, the DHT11 sensor still shows 0% humidity. The average temperature recorded is 20 degrees Celsius.



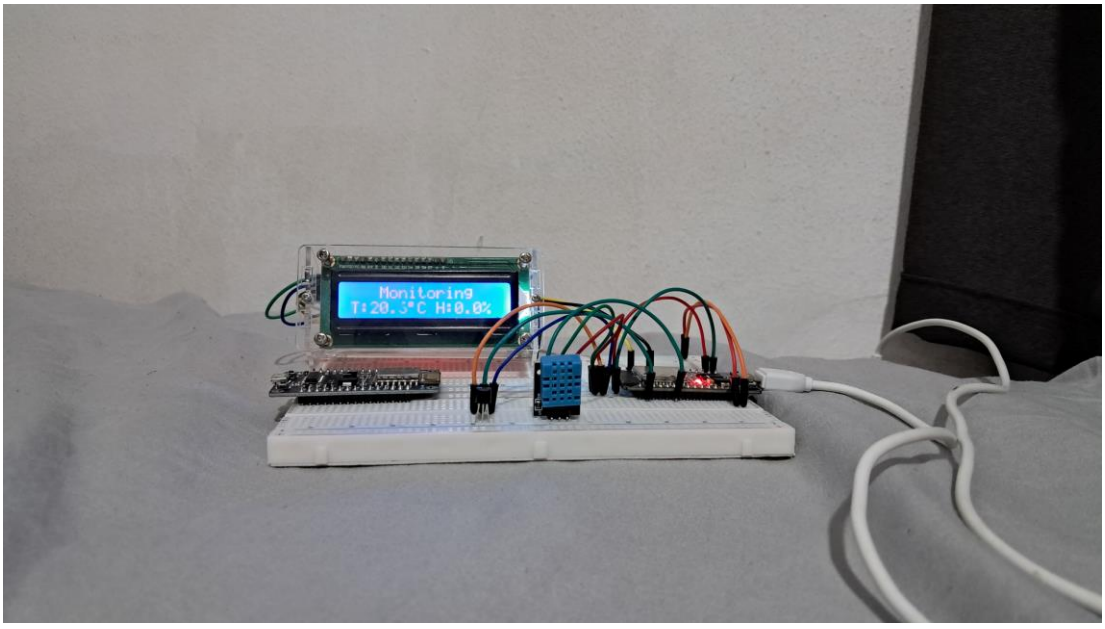


FIGURE 13: ThingSpeak Visualization: Placing the Circuit Inside a Closed Room with No Windows

Lastly, the final location is inside a room with poor air circulation. Based on the graph, we can see that the temperature only went below 20 degrees Celsius once. Most of the readings were above 20, with the highest at 22 degrees Celsius. The average temperature in the room was 21 degrees Celsius, and the humidity level was expected to be 0%.

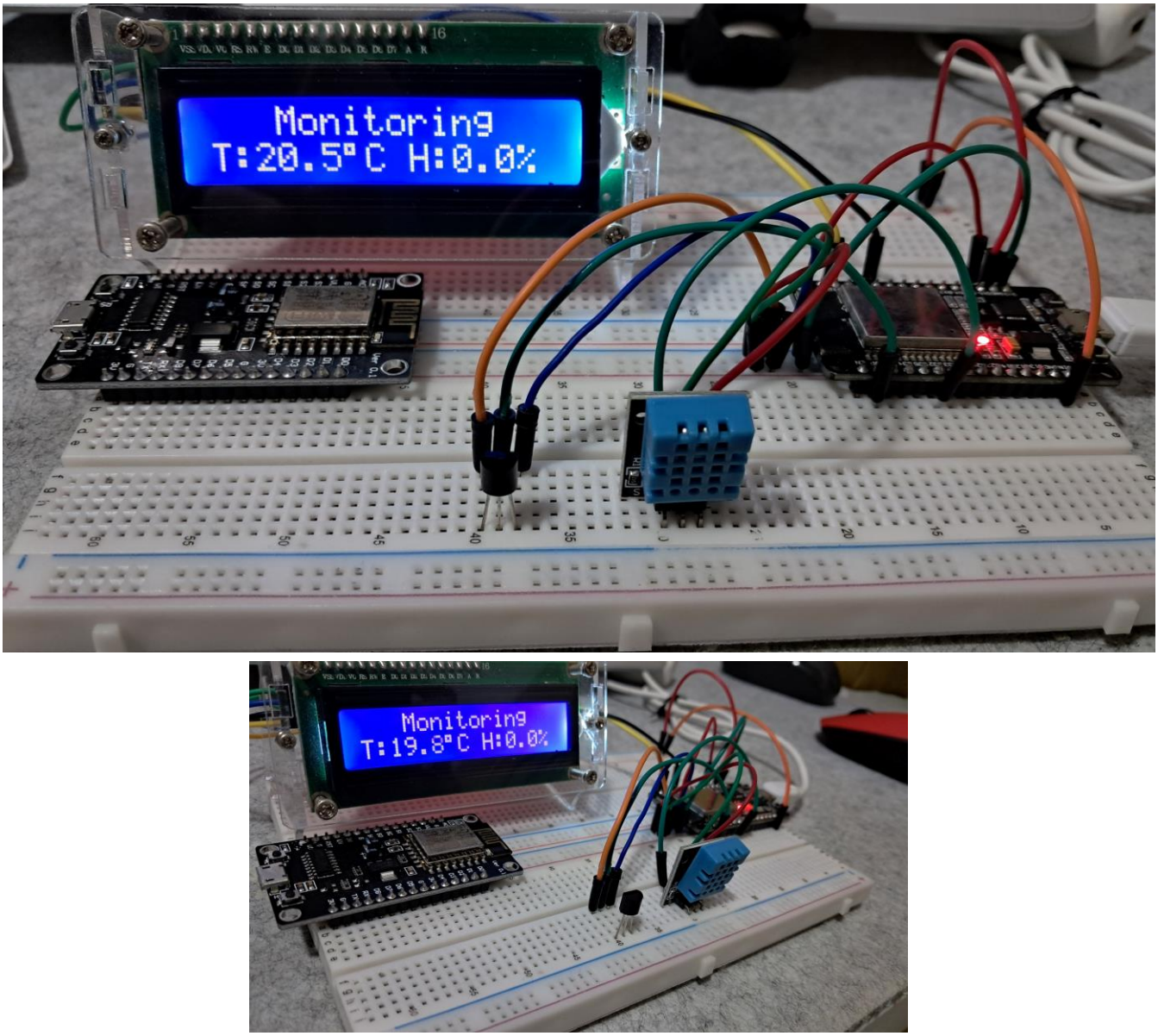


FIGURE 14: Working Prototype

This is the final working prototype of the IoT based weather monitoring system experiment, which successfully meets all the requirements.

CONCLUSION

As a student, this project helped me understand how to build a real-time weather monitoring system using basic sensors and IoT tools. I learned how to properly connect the LM35 temperature sensor, DHT11 humidity sensor, LCD display, and the ESP32 microcontroller. Through trial and error, I figured out the right wiring and setup, especially after encountering problems with the DHT11 sensor, which at first gave wrong readings. Fixing those errors taught me the importance of checking datasheets and verifying pin configurations. Using my knowledge from previous subjects like logic circuits and microprocessors made this process smoother, and seeing the actual data being sent to ThingSpeak gave me a real sense of how IoT systems work in real life. Throughout the code development, I practiced organizing libraries, setting up Wi-Fi connections, and using APIs like ThingSpeak to visualize data. I added timers to track the temperature averages every five minutes and also made sure the system could recover from errors like sensor failures. While coding, I made many mistakes like missing semicolons and typos, but reviewing and fixing those helped sharpen my debugging skills. The serial monitor became a very helpful tool for checking whether my sensors were working properly and whether the data was sent successfully online. It gave me confidence that my circuit was working as planned and that my code was doing the right job.

Lastly, the observations from different places like the kitchen, bathroom, and inside bags showed me how environmental factors can affect sensor readings. Although the temperature data was mostly accurate, I found it challenging to get consistent humidity readings because of the DHT11's limitations. Still, I was able to draw useful insights from the graphs on ThingSpeak, like how temperature changes when covering the sensor with a hand or putting it in a room without ventilation. Despite the small issues with accuracy, I'm proud that my prototype

worked and gave real-time weather data both on the LCD and online. This experiment helped me grow not just in coding and electronics, but also in solving problems and thinking like an engineer.