



الجامعة الإسلامية العالمية ماليزيا  
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA

يُونُسُ بَرَسِيْتِي اِسْلَامُ اَنْتَا اِبْعَثْنَا مَلِيْسِيَا

*Garden of Knowledge and Virtue*

## LAB REPORT

**MCTA 3203**

**SECTION 1**

**GROUP D**

### EXPERIMENT 8

#### **Remote Temperature Monitoring And Control**

| MATIC NO. | NAME                                 |
|-----------|--------------------------------------|
| 2111633   | MUHAMMAD FARID BIN JAFRI             |
| 2110765   | MUHAMMAD ARIF SYAZWAN BIN MOHD RIZAL |
| 2112969   | MUHAMMAD HAZIQ HAIKAL BIN SUAIB      |
| 2115207   | MUHAMMAD AMIRUL SYAFIQ BIN RUSLANI   |
| 2116281   | MUHAMMAD FADHLUL WAFI BIN AHMAD NAIM |

**SUBMISSION DATE: 13/12/2023**

**INSTRUCTOR: DR. WAHJU SEDONIO**

## **Abstract**

The subsequent report offers comprehensive details on an experiment conducted in the field of mechatronics system integration. The experiment involves integrating data from Bluetooth and WiFi with microcontrollers and computer-based systems, encompassing data processing, sensors, and actuators. Utilizing a temperature sensor, Wi-Fi, and Arduino as the key components, the experiment primarily focuses on remote temperature monitoring and control.

The foundational elements of the experiment include a breadboard, jumper wires, and a Wi-Fi-equipped Arduino board. The essential components required for the experiment are a temperature sensor, a Bluetooth module, and a smartphone with Bluetooth support. Python is also integrated into the experiment, serving the main purpose of displaying and logging temperature data on both a computer and a smartphone.

## **Table of Contents**

|                                |           |
|--------------------------------|-----------|
| <b>Introduction</b>            | <b>4</b>  |
| <b>Materials and Equipment</b> | <b>5</b>  |
| <b>Experimental Setup</b>      | <b>6</b>  |
| <b>Methodology</b>             | <b>7</b>  |
| <b>Results (Observation)</b>   | <b>11</b> |
| <b>Discussion</b>              | <b>13</b> |
| <b>Conclusion</b>              | <b>14</b> |
| <b>Recommendations</b>         | <b>15</b> |
| <b>References</b>              | <b>16</b> |
| <b>Student's Declaration</b>   | <b>17</b> |

## **Introduction**

The experiment primarily focused on employing an Arduino board with Wi-Fi capabilities and a smartphone equipped with Bluetooth to create a wireless temperature monitoring system. Utilizing a DHT11 temperature sensor, the objective was to have the Arduino read temperature data from the thermistor, transmit it over Wi-Fi to a Python script, and have the script display and log the temperature. Real-time temperature monitoring via the Internet was achieved through a smartphone. The Arduino Mega, based on the ATmega2560 microcontroller, served as the central platform, offering numerous digital input/output pins, PWM outputs, analog inputs, hardware serial ports (UART), and other features. The ESP8266 Wi-Fi module was employed as a cost-effective wireless transceiver for internet connectivity in embedded applications.

The chosen temperature sensor, the DHT11, featured a dedicated NTC for temperature measurement and an 8-bit microcontroller for serial data output of temperature and humidity values. The experiment also incorporated Bluetooth technology using the HC-05 module, a standard for short-range wireless communication. The smartphone, with its built-in Bluetooth support, was deemed the ideal device. The HC-05 module operated in Data mode for data exchange and Command mode for adjusting settings using AT commands, communicating with microcontrollers in master or slave configuration through serial ports.

## **Materials and Equipment**

1. Arduino board with Wi-Fi capability (ESP32)
2. Temperature sensor (DHT11)
3. Smartphone with Bluetooth support
4. Wi-Fi network and internet access
5. Power supply for the Arduino
6. Breadboard and jumper wires

## Experimental Setup

### 1. Hardware Setup:

- The temperature sensor (thermistor) is connected to the ESP32.
- The Bluetooth module is connected to the ESP32.

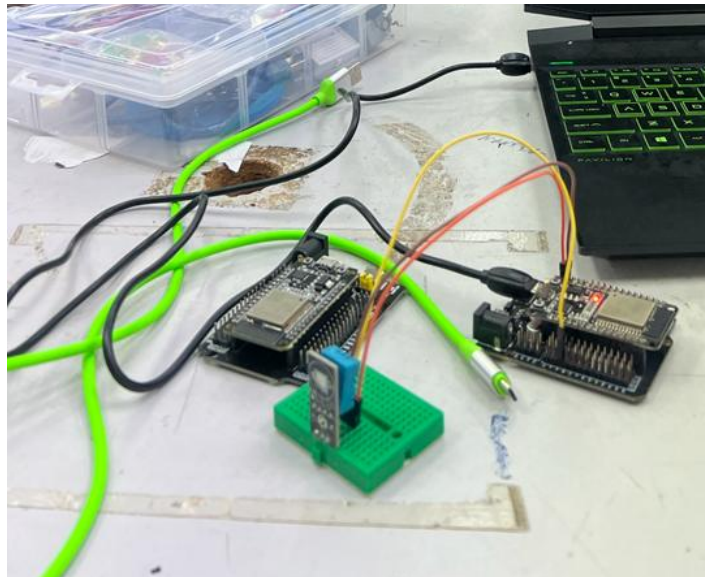
### 2. Arduino Programming:

- An Arduino sketch is written to read temperature data from the sensor.

### 3. Bluetooth Programming:

- An ESP32 sketch is written to enable Bluetooth communication.
- The task below is executed:

Develop a simple smartphone application (or use an existing one) that communicates with the Arduino via Bluetooth. This app should allow you to send commands to control a connected device, such as a fan or heater, based on the temperature data received from the Arduino. For this experiment, an LED is used as a placeholder for the fan; if the fan is switched on, the LED will be switched on as well.

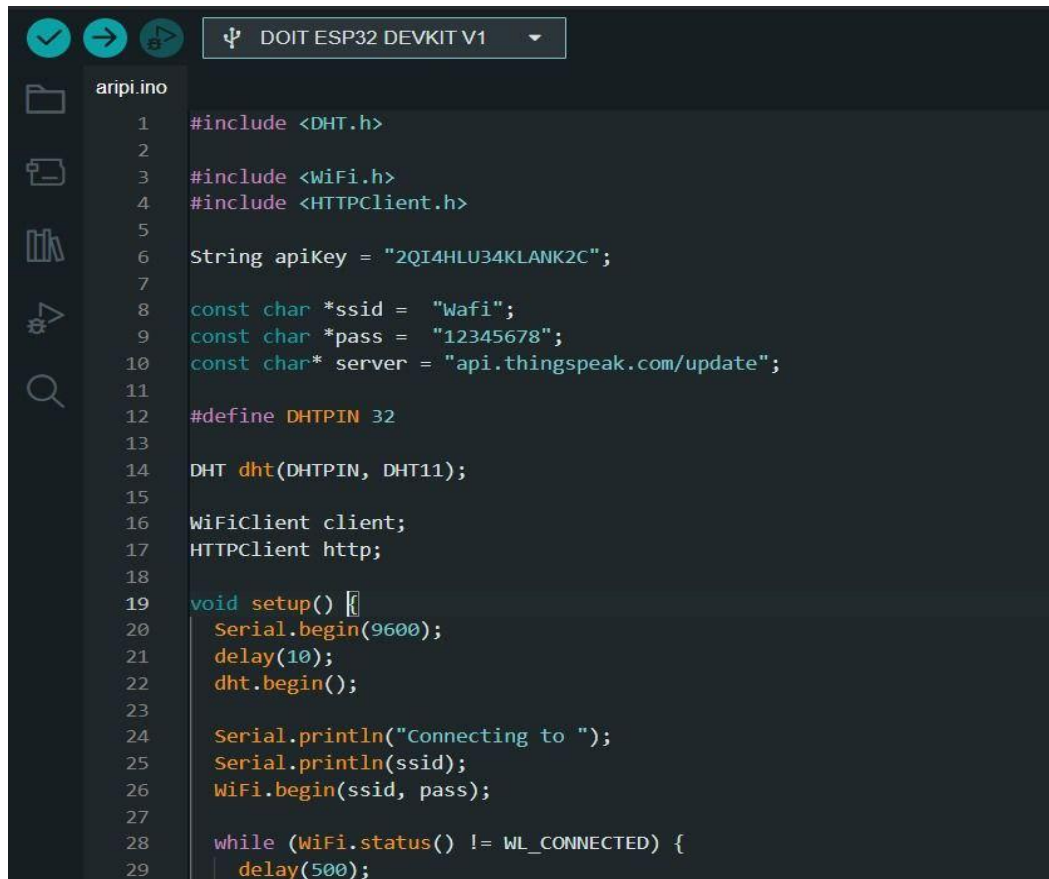


## Methodology

### Procedure:

1. The temperature sensor was placed in a room or area that was monitored and controlled for temperature.
2. The Arduino was connected to a power source, and it was ensured that it was connected to the Wi-Fi network.
3. The room's temperature was monitored in real-time using the ThingSpeak dashboard.

### Part A: Arduino Code (WIFI)



```
DOIT ESP32 DEVKIT V1

aripi.ino
1  #include <DHT.h>
2
3  #include <WiFi.h>
4  #include <HTTPClient.h>
5
6  String apiKey = "2QI4HLU34KLANK2C";
7
8  const char *ssid = "Wafi";
9  const char *pass = "12345678";
10 const char* server = "api.thingspeak.com/update";
11
12 #define DHTPIN 32
13
14 DHT dht(DHTPIN, DHT11);
15
16 WiFiClient client;
17 HTTPClient http;
18
19 void setup() {
20   Serial.begin(9600);
21   delay(10);
22   dht.begin();
23
24   Serial.println("Connecting to ");
25   Serial.println(ssid);
26   WiFi.begin(ssid, pass);
27
28   while (WiFi.status() != WL_CONNECTED) {
29     delay(500);
```

```
DOIT ESP32 DEVKIT V1

aripi.ino
29   delay(500);
30   Serial.print(".");
31   }
32   Serial.println("");
33   Serial.println("WiFi connected");
34   }
35
36 void loop() {
37   float h = dht.readHumidity();
38   float t = dht.readTemperature();
39
40   if (WiFi.status() == WL_CONNECTED) {
41     delay(2000);
42     http.begin(client, server);
43     http.addHeader("Content-Type", "application/x-www-form-urlencoded");
44     String httpRequestData = "api_key=" + apiKey + "&field1=" + String(t) + "&field2=" + String(h);
45     int httpResponseCode = http.POST(httpRequestData);
46
47     Serial.print("Temperature: ");
48     Serial.print(t);
49     Serial.print(" degrees Celcius, Humidity: ");
50     Serial.print(h);
51     Serial.println("%. Send to Thingspeak.");
52     http.end();
53     delay(2000);
54   }
55   else {
56     Serial.println("WiFi Disconnected");
57   }
58 }
```

## Part B: Arduino Code (Bluetooth)

```
1  #include <DHT.h>;
2  #define DHTPIN 8
3  #define DHTTYPE DHT11
4  DHT dht(DHTPIN, DHTTYPE);
5
6  #define LED_PIN 7
7  char receivedValue = 0;
8  float temp;
9
10 void setup()
11 {
12   Serial.begin(9600);
13   dht.begin();
14   pinMode(LED_PIN, OUTPUT);
15   digitalWrite(LED_PIN, LOW);
16 }
17
18 void loop()
19 {
20   temp=dht.readTemperature();
21   Serial.println(temp);
22
23   if (Serial.available() > 0)
24   {
25     receivedValue = Serial.read();
26
27     if (receivedValue == '1')
28       digitalWrite(LED_PIN,HIGH);
29     else if(receivedValue == '0')
30       digitalWrite(LED_PIN,LOW);
31   }
32   delay(1000);
33 }
```



## Part B: Python Code (Bluetooth)

```
import serial
import matplotlib.pyplot as plt
ser = serial.Serial('COM5', 9600)
temperatures = []
try:
    while True:
        data = ser.readline().decode('utf-8').strip()
        try:
            temperature = float(data)
            temperatures.append(temperature)
            # Display real-time temperature
            print(f"Temperature: {temperature} °C")
            if temperature >= 30:
                print("Please turn on yor fan")

        except ValueError as e:
            print(e)
        ...
except KeyboardInterrupt:
    # Plot the recorded temperatures when the user interrupts the script
    plt.plot(temperatures, marker='o')
    plt.title('Temperature Monitoring')
    plt.xlabel('Time (s)')
    plt.ylabel('Temperature (°C)')
    plt.show()
    ...
finally:
    ser.close()
```

## Data Collection

### Part A (WiFi)

| Time (s) | Temperature (°C) | Humidity (%) |
|----------|------------------|--------------|
| 0        | 25.30            | 70.00        |
| 10       | 25.30            | 70.00        |
| 14       | 25.30            | 69.00        |
| 17       | 25.30            | 69.00        |
| 20       | 25.30            | 69.00        |
| 23       | 25.30            | 69.00        |
| 26       | 26.00            | 76.00        |

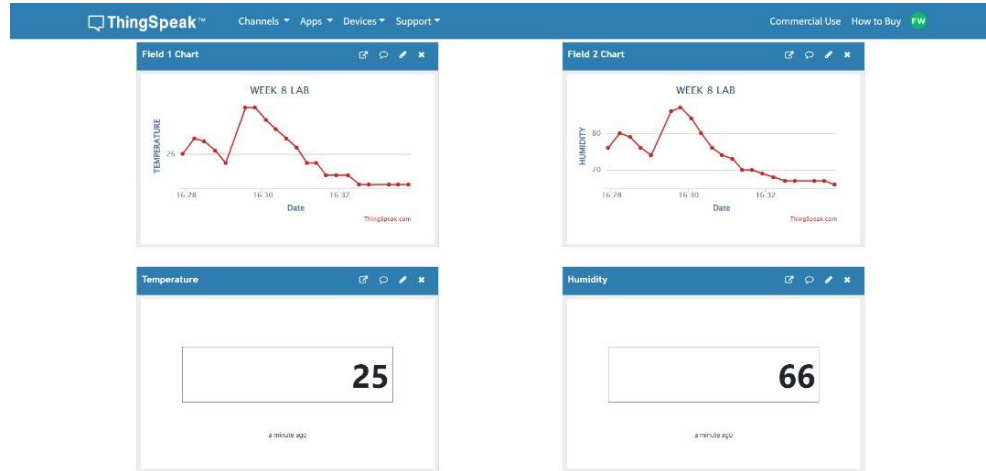
|    |       |       |
|----|-------|-------|
| 29 | 26.10 | 77.00 |
| 32 | 26.10 | 78.00 |
| 35 | 26.50 | 79.00 |
| 38 | 26.50 | 80.00 |
| 44 | 26.50 | 80.00 |
| 47 | 26.50 | 80.00 |
| 50 | 26.50 | 80.00 |

**Part B (Bluetooth)**

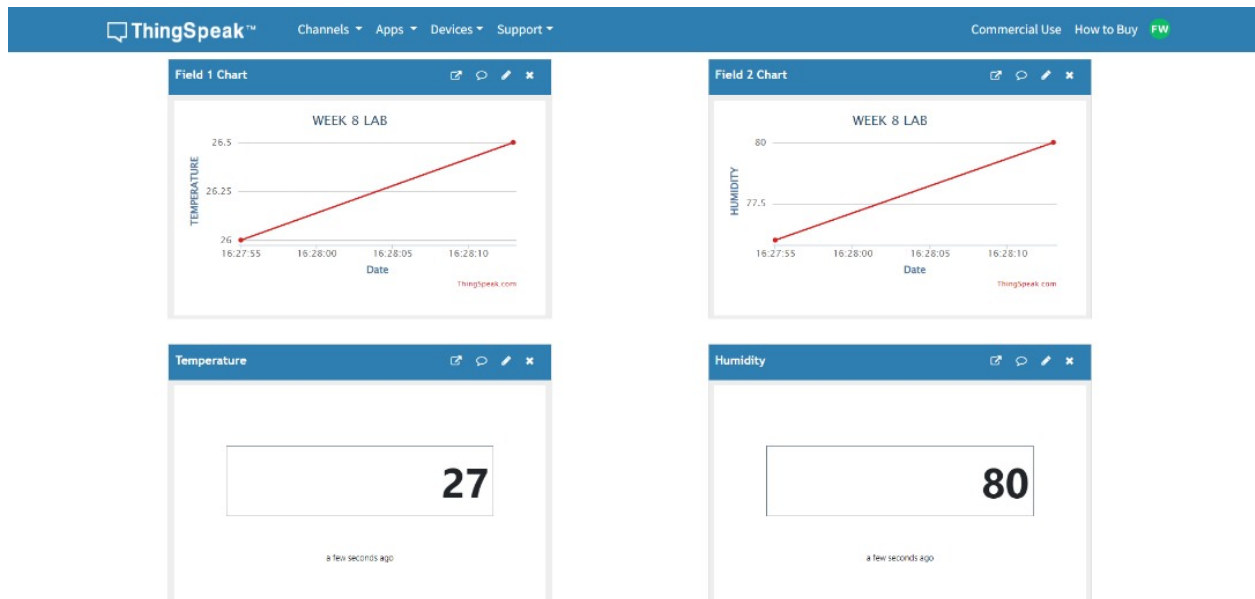
| Time (s) | Temperature (°C) |
|----------|------------------|
| 0        | 29.4             |
| 10       | 30.9             |
| 20       | 33.3             |
| 30       | 32.2             |
| 40       | 30.9             |
| 50       | 30.4             |
| 60       | 30.2             |
| 70       | 29.7             |
| 80       | 29.5             |

## Results (Observation)

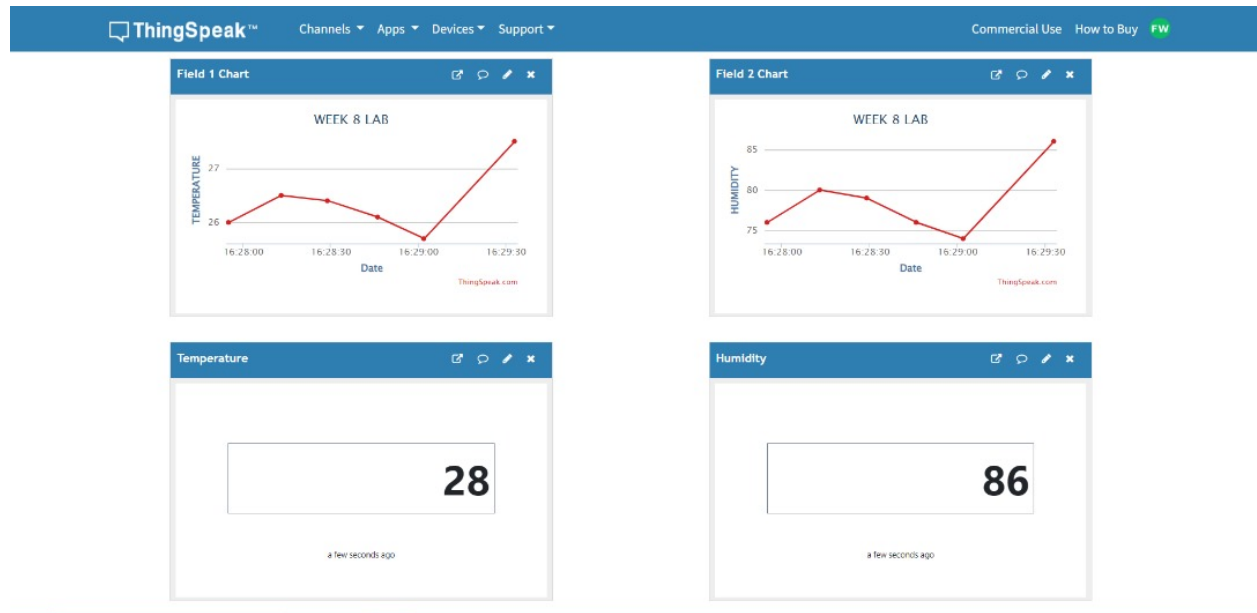
### Part A: WiFi



*Fig. 1.1: Graph of Temperature and Humidity vs Time*



*Fig. 1.2: Value temperature and humidity before blow*



*Fig. 1.2: Value temperature and humidity after blow*

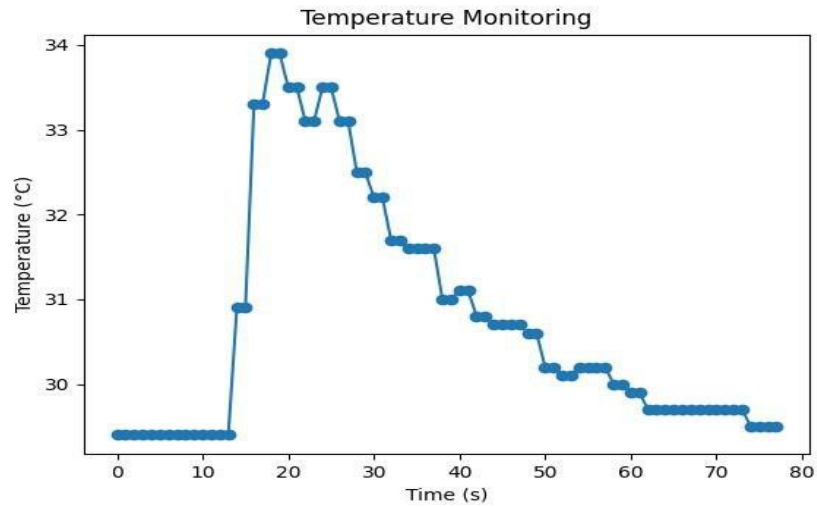
## Part B: Bluetooth

```

IDLE Shell 3.12.0
File Edit Shell Debug Options Window Help
Temperature: 29.4 °C
Temperature: 29.4 °C
Temperature: 29.4 °C
Temperature: 29.4 °C
Temperature: 29.4 °C
Temperature: 29.4 °C
Temperature: 29.4 °C
Temperature: 29.4 °C
Temperature: 29.4 °C
Temperature: 29.4 °C
Temperature: 29.4 °C
Temperature: 29.4 °C
Temperature: 29.4 °C
Temperature: 29.4 °C
Temperature: 30.9 °C
Please turn on yor fan
Temperature: 30.9 °C
Please turn on yor fan
Temperature: 33.3 °C
Please turn on yor fan
Temperature: 33.3 °C
Please turn on yor fan
Temperature: 33.9 °C
Please turn on yor fan
Temperature: 33.9 °C
Please turn on yor fan
Temperature: 33.5 °C
Please turn on yor fan
Temperature: 33.5 °C
Please turn on yor fan
Temperature: 33.1 °C
Please turn on yor fan
Temperature: 33.1 °C
Please turn on yor fan
Temperature: 33.5 °C
Please turn on yor fan
Temperature: 33.5 °C
Please turn on yor fan
Temperature: 33.1 °C
Please turn on yor fan
Temperature: 33.1 °C
Please turn on yor fan
Temperature: 32.5 °C
Please turn on yor fan
Temperature: 32.5 °C
Please turn on yor fan
Temperature: 32.2 °C
Please turn on yor fan

```

***Fig. 2.1: Result of Temperature***



***Fig. 2.2: Graph of Temperature(°C) vs Time(s)***

Part A video link:

<https://drive.google.com/file/d/1QD07q63Pw36f1L9TXEcReejG81myariU/view?usp=drivesdk>

Part B video link:

[https://drive.google.com/file/d/1J1bv6\\_yhiU9dhdLbcJ64WeN6yx0QgF1H/view?usp=drivesdk](https://drive.google.com/file/d/1J1bv6_yhiU9dhdLbcJ64WeN6yx0QgF1H/view?usp=drivesdk)

## **Discussion**

### Part A: WIFI

- Temperature and humidity are sent serially from the temperature sensor (thermistor) to ESP32, which then sends the information to the ThinkSpeak via wifi and API keys
- The graph shows the values of temperature and humidity changing over time in a separated graph.
- The values read by the temperature sensor (thermistor), which keeps updating in the Arduino IDE compiler are directly transferred to the ThinkSpeak and the graph of the result changes in time.

### Part B: Bluetooth

- The temperature sensor (thermistor) senses the temperature of the environment and sends the data to the Arduino IDE via Arduino UNO R3.
- The data from the temperature sensor (thermistor) are sent to the Python from Arduino IDE.
- The results from the temperature sensor (thermistor) reader can be viewed in Arduino IDE and also in Python.
- The circuit that has been constructed is linked with HC-05 Bluetooth devices and connected to the LED (symbolic as a fan).
- HC-05 is also connected to an application on a mobile phone called Arduino BlueControl.
- When the temperature sensor (thermistor) senses temperature above 31 degrees Celsius, the LED will light up as an indicator that temperature is high.
- The user can either turn on or turn off the LED(symbolic as a fan) by using a mobile phone that is connected via Bluetooth.

## **Conclusion**

In summary, the experiment effectively illustrated how to integrate data from an Arduino microcontroller with Bluetooth and WiFi. The results provide important new information about how to choose the best wireless communication protocol for a particular project based on its particular requirements. It is anticipated that forthcoming advancements in wireless technologies will augment the functionalities of Arduino-based systems, hence broadening the potential for inventive and effective uses within the Internet of Things (IoT) domain and other domains.

In this experiment, we used WiFi to send temperature and humidity data using ThingSpeak online website then plot graphs for the data. For the task, we used Bluetooth connectivity through HC-05 to turn on the fan (in our experiment we used LED just for simplicity) whenever the temperature displayed in the python exceeded 30°C. It's important to take the planned application's specific requirements into account when evaluating WiFi and Bluetooth integration. When it comes to situations when speed and range are important, WiFi shines, while Bluetooth provides a more power-efficient option for short-range connection that requires less setup work.

## **Recommendations**

One of the recommendations that we think really important to this lab report is to recognize the power-hungry features of Bluetooth and WiFi. Because Bluetooth uses less energy than other technologies, it can be a better choice for applications that depend on batteries or have restricted power sources. Incorporate power-saving techniques, such sleep modes, to improve energy efficiency in general.

Other than that, it is recommended to use the library while interfacing DHT11 instead of using manual calculation written in the Arduino IDE. This is necessary to increase data accuracy read from the DHT11 which are temperature and humidity. This is because the library has already included calibration and threshold values into the main formula, so the output value will be more accurate.



## **References**

1. (2020). *WiFi ESP32 LED Control Project (Arduino)*. YouTube. Retrieved December 14, 2023, from <https://m.youtube.com/watch?v=Hgq2KX5w-o&t=263s>
2. A. (2022, August 22). DHT11 Humidity Temperature Monitor on ThingSpeak With NodeMCU. How To Electronics. Retrieved December 14, 2023, from <https://how2electronics.com/dht11-humidity-temperature-nodemcu-thingspeak/>
3. Gamra. (2022, November 16). Enjoy-Mechatronics - Overview. GitHub. Retrieved December 14, 2023, from <https://github.com/Enjoy-Mechatronics>

## Student's Declaration

### Certificate of Originality and Authenticity

This is to certify that we are responsible for the work submitted in this report, that the original work is our own except as specified in the references and acknowledgment, and that the original work contained herein has not been untaken or done by unspecified sources or persons.

We hereby certify that this report has not been done by only one individual and all of us have contributed to the report. The length of contribution to the reports by Each individual is noted within this certificate.

We also hereby certify that we have read and understand the content of the total report and no further improvement on the reports is needed from any of the individual contributor to the report.

We, therefore, agreed unanimously that this report shall be submitted for marking and this final printed report has been verified by us

Signature: *arifsyazwan*

Name: Muhammad Arif Syazwan Bin Mohd Rizal

Matric Number: 2110765

Read ✓

Understand ✓

Agree ✓

Signature: *haziqhaikal*

Name: Muhammad Haziq Haikal bin Suaib

Matric Number: 2112969

Read ✓

Understand ✓

Agree ✓

Signature: *amirul*

Name: Muhammad Amirul Syafiq Bin Ruslani

Matric Number: 2115207

Read ✓

Understand ✓

Agree ✓

Signature: *Wafi*

Name: Muhammad Fadhlul Wafi Bin Ahmad Naim

Matric Number: 2116281

Read ✓

Understand ✓

Agree ✓

Signature: *faridjafri*

Name: Muhammad Farid Bin Jafri

Matric Number: 2111633

Read ✓

Understand ✓

Agree ✓