MECHATRONICS SYSTEM INTEGRATION

EXPERIMENT 7: BLUETOOTH, XBEE AND WIFI DATA INTERFACING WITH MICROCONTROLLER AND COMPUTER BASED SYSTEM:

GROUP NUMBER: A

PROGRAMME: MECHATRONICS

GROUP MEMBERS	MATRIC NO
AISYAH SOFEA BINTI OTHMAN	2115386
SITI ALIAA BINTI IBRAHIM	2112618
MUHAMMAD ADLI FAHMI BIN TAJUL ARIS	2113095
ETHAR ABDALLA ABDELKARIM OSMAN	2111282
NURAIN AINAA AQILAH BINTI ROSLI	2114560

DATE OF EXPERIMENT:

Wednesday, 6th December 2023

DATE OF SUBMISSION:

Wednesday, 20th December 2023

Abstract

The ESP8266 serves as the central processing unit, facilitating wireless communication through its built-in Wi-Fi capabilities.The DH11 sensor accurately measures ambient temperature and humidity, providing critical environmental data for the system. A web-based interface allows users to access temperature information from anywhere with internet connectivity, enhancing convenience and accessibility. The Thingspeak part has been successfully executed with the wifi module connectivity and the task part with addition of components like LED representing the heater and LM35 that substitutes the DHT11 although manage to be executed but the value appears to be too high due to calibration error or else. Hence, it was recommended to go through the codes and various approaches.

Table of Contents

Objectives

To create a wireless temperature monitoring system using Wi-Fi, Arduino, Thingspeak, DHT11 and LM35. The Arduino reads temperature data from the DHT11, sends it to a Python script and Thingspeak over Wi-Fi which then displays and logs the temperature.

Introduction

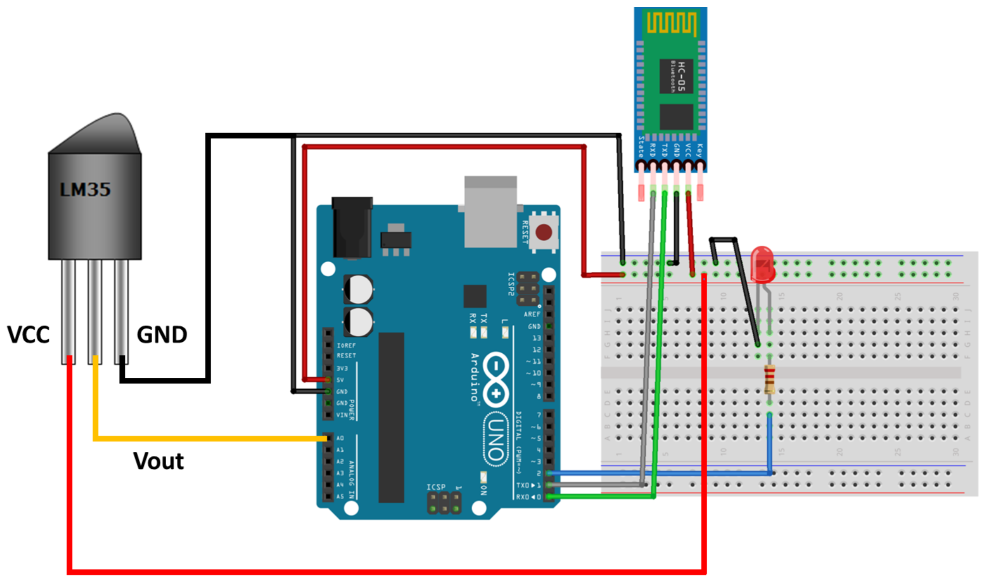
Wi-fi stands for Wireless Fidelity which is a wireless technology standard for wireless Internet access. It is used as a replacement for cable connections and other types of wires.The ESP8266 can be configured to connect to an existing Wi-Fi network (station mode) or act as an access point for other devices to connect to it (soft access point mode).

Bluetooth is a wireless communication standard used for short-range communication between devices. In this experiment ESP8266 serves as a wifi module and HC-06 acts as a bluetooth server to send data (temperature readings from the LM35 sensor or DHT11 sensor) to an android device application ('Serial Bluetooth Terminal') as a bluetooth client. Bluetooth typically has a shorter range (tens of metres) compared to Wi-Fi, which can cover longer distances.

Material and Equipment

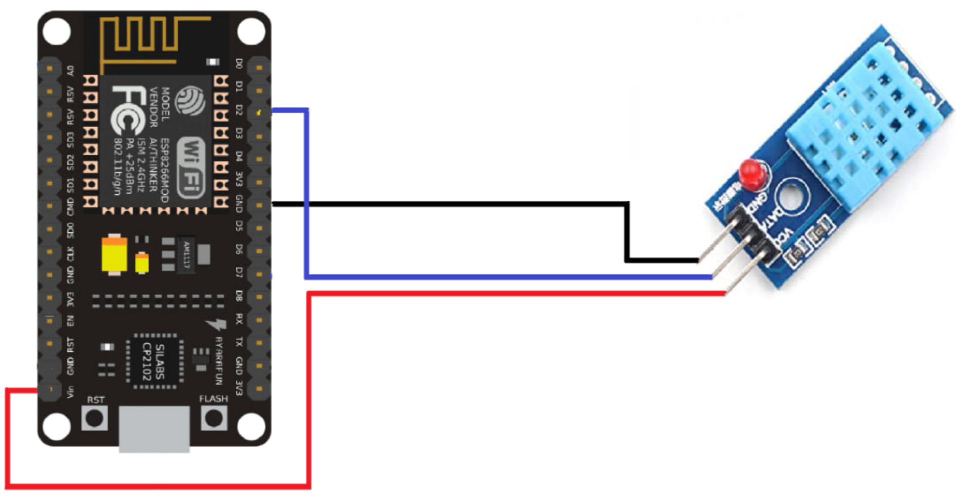
1. Arduino uno
2. Temperature sensor (DHT11 and LM35)
3. Bluetooth module (HC-06)
4. Smartphone with Bluetooth support
5. Wi-Fi network and internet access
6. Power supply for the Arduino
7. Breadboard
8. Jumper wires
9. Led (Red)
10. ESP8266
11. Resistor

Experimental Setup



LM35, Led & Bluetooth Connection For Task

The LM35 is an analog temperature sensor that provides an analog voltage output proportional to the temperature. In this experiment, we connect the sensor's VCC to the microcontroller's 5V, GND to GND, and the sensor's output pin to an analog input pin on the microcontroller which is A0. The Bluetooth module allows wireless communication between the microcontroller and another device, like a smartphone or computer. The module's TX pin goes to the Arduino pin which is digital pin 0 and RX pin goes to digital pin 1. Connect VCC to 5V and GND to GND. The LED will serve as an output indicator based on the temperature readings. We connect the longer leg (anode) of the LED to a digital output pin on the Arduino through a current-limiting 220 ohm resistor. Connect the shorter leg (cathode) to GND.



ESP 8266 & DHT 11 Connection for Thingspeak

ESP8266 is connected to the computer for power supply and to perform serial communication. DHT11 is a digital sensor that provides temperature and humidity data. We connect the DHT11 to the ESP8266 by connecting the sensor VCC to Vin, GND to GND, and the sensor data pin to a digital input/output pin which is D2.

Methodology

To set up the temperature and humidity sensor experiment using an ESP8266 and DHT11, first connect the DHT11 sensor to the ESP8266 by wiring the VCC to Vin, GND to GND, and DATA to a digital pin (e.g., D2). Power up the ESP8266, and upload the provided Arduino code with your Wi-Fi credentials and ThingSpeak API key. Monitor the Serial Monitor in the Arduino IDE to ensure a successful connection, and observe real-time temperature and humidity readings. Check ThingSpeak for data uploads and conduct experiments to evaluate sensor accuracy.

Arduino Code - ESP 8266 & DHT 11

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


String apiKey = "BJNET4LNB1Y1RQFM"; // write your "Write API key"
```

```
const char* ssid = "aisyahsofeeeeaaa"; // write your "wifi name"
const char* password = "Taylorsw@89"; // write your "wifi password"
const char* server = "api.thingspeak.com";
WiFiClient client;

DHT dht(D2, DHT11);  // (dht pin no, dht sensor type)
float Hum, Temp;

void setup()
{
  Serial.begin(9600);
  Serial.println("Serial Begin");
  dht.begin();

  WiFi.begin(ssid, password);
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);


  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
    Serial.println("");
    Serial.println("WiFi connected");
}

void loop()
{
  DHT_data();

  delay(1000);
```

```
 if (client.connect(server,80))
   {
     fwd_to_Thingspeak();
   }
 client.stop();
 Serial.println("Waiting");
 delay(1000);
}

void DHT_data()
{
 Hum = dht.readHumidity();
 Temp = dht.readTemperature();
}
void fwd_to_Thingspeak()
{
 String postStr = apiKey;
 postStr +="&field1=";
 postStr += String(Hum);  // Humidity data
 postStr +="&field2=";
 postStr += String(Temp); // Temperature Data
 postStr += "\r\n\r\n";

 client.print("POST /update HTTP/1.1\n");
 client.print("Host: api.thingspeak.com\n");
 client.print("Connection: close\n");
 client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
 client.print("Content-Type: application/x-www-form-urlencoded\n");
 client.print("Content-Length: ");
 client.print(postStr.length());
 client.print("\n\n");
 client.print(postStr);

 Serial.print("Send data to channel-1 ");
 Serial.print("Content-Length: ");
 Serial.print(postStr.length());
```

```
  Serial.print("Field-1: ");
  Serial.print(Hum);
  Serial.print("Field-2: ");
  Serial.print(Temp);
  Serial.println(" data send");
}
```

BLUETOOTH & LM 35 :

To set up the temperature sensor experiment, connect the LM35 sensor, HC-06 bluetooth module, and Red LED to Arduino uno using jumper wires and a breadboard. Upload the provided Arduino code to the Arduino Uno and power it up. Pair the Bluetooth module with the smartphone and monitor temperature readings on the Arduino Serial Monitor. Use a bluetooth terminal app to verify data transmission which can be confirmed when the prompt ‘connected’ shows up. Observe the LED change when the button ON/OFF is pressed in the application. LED is used as a heater that can be controlled by the application by monitoring the temperature.

Code for bluetooth and LM35 connection :

Arduino Code	Python Code

#include <SoftwareSerial.h>
 	
char switch_read = 0;
float calibration;
const int Led = 2; //Red : Turn on the Heater
float tempc; // Celcius
const int Tempsensor_LM35 = A1; //LM 35 used to represent temperature sensor
 	
void setup() {
  pinMode(Led, OUTPUT); // (pin_num on Arduino, OUTPUT)
	import serial
	import matplotlib.pyplot as plt
	from time import sleep
	
	serial_port = "COM11"
	baud_rate = 9600
	dt = 0.05
	
	ser = serial.Serial('COM11', 9600) # adjust as needed
	temperatures = []

```
  Serial.begin(9600);  // baud rate

}



void loop() {
  // Check if data is available to read from the
Bluetooth module
  if (Serial.available() > 0) {
    switch_read = Serial.read();
  }



//Switch for the LED , 1 = On (App) , 0 = Off
(App) to control it, Apps : "Serial Bluetooth
Terminal"
  if (switch_read == '1') {
    digitalWrite(Led, HIGH);

  } else if (switch_read == '0') {
    digitalWrite(Led, LOW);

  }

  int sensorValue =
analogRead(Tempsensor_LM35);
  //float calibration =
100*(5/1023)*sensorValue;  // 100:(ref. LM35
datasheet), use 5V or Arduino, ADC : 1023
  tempc = (analogRead(Tempsensor_LM35)
/1023)*5000 ;
  tempc = tempc/10;
  Serial.print("Temperature is  ");
  Serial.print(tempc, 1); // one decimal place
resolution is all you get
  Serial.print(" Celcius  ");
```

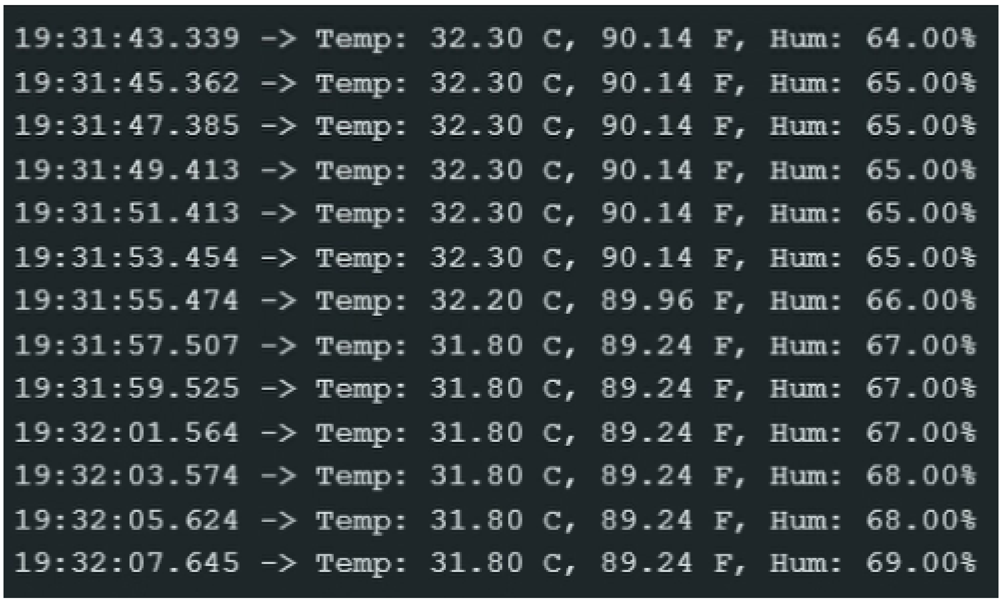
```
try:
    while True:
      data = ser.readline().decode('utf-8').strip()
      temperature = float(data)
      temperatures.append(temperature)

      # Display real-time temperature
      print(f"Serial port{ser name} is open")

except KeyboardInterrupt:
     # Plot the recorded temperatures when the
user interrupts the script

     plt.plot(data)
     plt.title('Temperature Monitoring')
     plt.xlabel('Time(s)')
     plt.ylabel('Temperature (°C)')
     plt.grid(True)
     plt.show()

finally:

   ser.close()
```

// Send temperature to the serial port
Serial.println(tempc);
delay(100); //faster on/off
}	

Data Collection

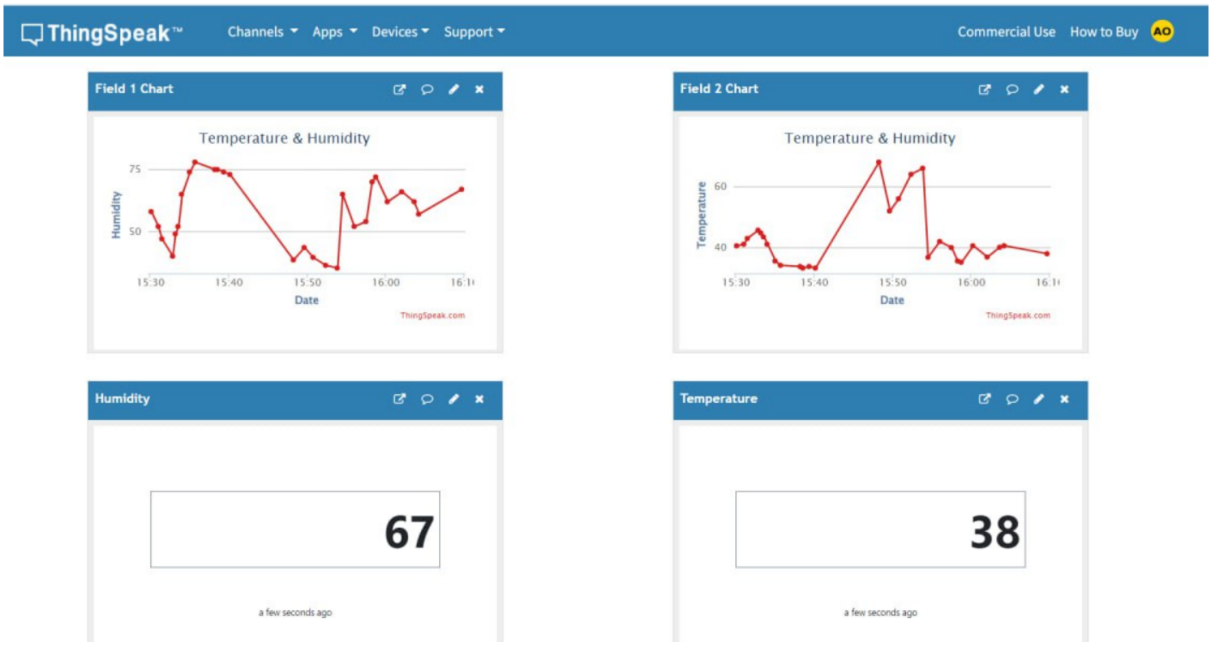
<u>ThingSpeak</u>

<u>Temperature and humidity change over time in room's temperature</u>



```
19:31:43.339 -> Temp: 32.30 C, 90.14 F, Hum: 64.00%
19:31:45.362 -> Temp: 32.30 C, 90.14 F, Hum: 65.00%
19:31:47.385 -> Temp: 32.30 C, 90.14 F, Hum: 65.00%
19:31:49.413 -> Temp: 32.30 C, 90.14 F, Hum: 65.00%
19:31:51.413 -> Temp: 32.30 C, 90.14 F, Hum: 65.00%
19:31:53.454 -> Temp: 32.30 C, 90.14 F, Hum: 65.00%
19:31:55.474 -> Temp: 32.20 C, 89.96 F, Hum: 66.00%
19:31:57.507 -> Temp: 31.80 C, 89.24 F, Hum: 67.00%
19:31:59.525 -> Temp: 31.80 C, 89.24 F, Hum: 67.00%
19:32:01.564 -> Temp: 31.80 C, 89.24 F, Hum: 67.00%
19:32:03.574 -> Temp: 31.80 C, 89.24 F, Hum: 68.00%
19:32:05.624 -> Temp: 31.80 C, 89.24 F, Hum: 68.00%
19:32:07.645 -> Temp: 31.80 C, 89.24 F, Hum: 69.00%
```

<u>Plotted graph on 'ThingSpeak' of Temperature and Humidity with Wi-fi connection</u>



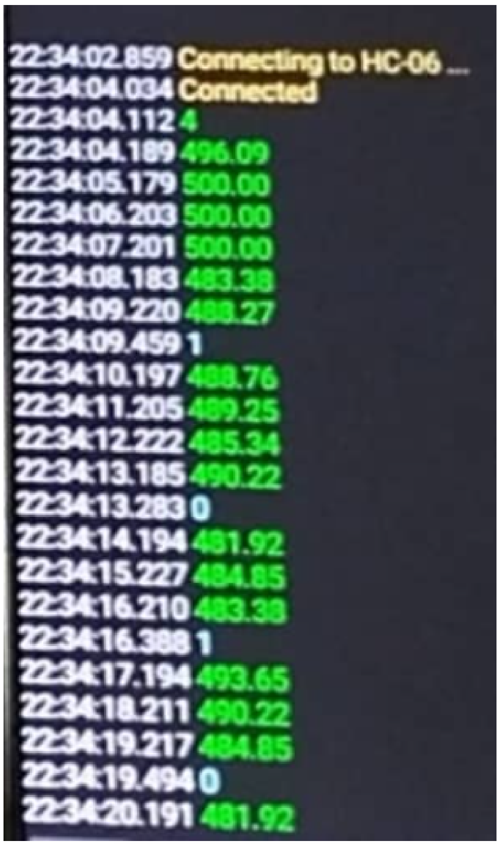
Task

‘Serial Bluetooth Terminal ‘		Heater
Button	**Value**	**Red LED (Circuit)**
ON	1	Light up
OFF	0	Light off

Table 8.1 : Representation of heater device to control the temperature by monitoring the temperature receive in the application from Arduino

**The Red LED is the representation of the switch for the heater.*

**Serial Bluetooth Terminal is connected to the HC-06*



```
22:34:02.859 Connecting to HC-06 ...
22:34:04.034 Connected
22:34:04.112 4
22:34:04.189 496.09
22:34:05.179 500.00
22:34:06.203 500.00
22:34:07.201 500.00
22:34:08.183 483.38
22:34:09.220 488.27
22:34:09.459 1
22:34:10.197 488.76
22:34:11.205 489.25
22:34:12.222 485.34
22:34:13.185 490.22
22:34:13.283 0
22:34:14.194 481.92
22:34:15.227 484.85
22:34:16.210 483.38
22:34:16.388 1
22:34:17.194 493.65
22:34:18.211 490.22
22:34:19.217 484.85
22:34:19.494 0
22:34:20.191 481.92
```

Timestamp for 'Serial Bluetooth Terminal' when Heater in On/Off

Data Analysis

<u>ThingSpeak</u>

DHT11 was used as a temperature sensor. The reading is close since it's in the same source of heat which is the room's temperature. From the reading it can be seen that, at first it was at 32.30 degree Celsius and it slightly decreased and remained constant after some time. The graph was automatically plotted using the integration between the DHT11 and ESP8266.

<u>Task</u>

For this part, we use LM35 to substitute DHT11 as the temperature sensor. HC-06 was integrated with the Arduino to receive reading from LM35 and shown in the monitor of the 'Serial Bluetooth Terminal'. From there, it can be seen the value was over 400 which is not accurate and did not meet the expected readings even through calibration. The '1' was to turn on the heater switch while '0' was to turn it off. The red led is used as the switch representation. It was a controlled button to send commands by monitoring the reading of the temperature from the application.

Result

The ESP8266 module successfully connected to the internet, as well as succeeded in transmitting temperature and humidity data from the DHT11 sensor to Thingspeak. Thingspeak web-based interface provided real-time monitoring and plotting of the environmental conditions readings over time. In the task part involving Bluetooth HC-06, the wireless transmission of temperature data to a smartphone for remote monitoring was achieved. However, the Bluetooth connection faced instability issues, leading to higher-than-expected temperature readings. The experiment encountered challenges with Arduino port connectivity when transferring temperature data to the smartphone through Bluetooth. It was observed that the process reliably worked when connected to the COM4 port. The LM35 temperature sensor effectively captured temperature values, despite the obtained readings being too high. The collected data was successfully transmitted to a smartphone using HC-06 bluetooth module. Python (used to receive and process data transmitted from Arduino through Bluetooth) faced difficulties in generating and plotting the temperature graph using the matplotlib library. This aspect requires improvement in future experiments. Controlling the LED as a heater indicator through the serial Bluetooth terminal application on the smartphone was successful, allowing users to turn the heater on (LED on) or off (LED off) remotely.

Discussion

- ESP8266 (WiFi)

  The ESP8266 module includes Wi-Fi capabilities that allow it to connect to the internet. It can connect to the Wi-Fi network, allowing communication with internet platforms such as Thingspeak. Through Arduino code, the ESP 8266 was programmed to channel the values received from the temperature sensor DHT11. Then, the collected data will be transferred to the Thingspeak using the API key assigned to the Thingspeak channel. The graph of humidity and temperature over time was plotted in the Thingspeak.

- Temperature sensor DHT11

  The DHT11 sensor is a sensor that was used in this experiment to capture or measure the surrounding temperature and humidity values. This sensor was connected to the ESP8266 to transmit the data to the Thingspeak.

Task Part

- Bluetooth HC-06

  For the task part, the Bluetooth HC-06 was used as a part of the setup. This Bluetooth HC-06 acts as a bridge to transmit the temperature data collected by the sensor wirelessly to the chosen device. In this experiment, we were using the smartphone as a remote to control and monitor the temperature values. For this experiment, the Bluetooth connection was unstable as we successfully transferred the data to the Arduino serial monitor and the device, however, the values of the temperature obtained were too high.

- Arduino port

  In this task part experiment, we encountered a problem with connecting the correct Arduino port to smoothly transfer the collected temperature data from the sensor to the smartphone through Bluetooth. In this case, the process only seems to work when we connect it to the com4 port.

- Temperature sensor LM35

  The LM35 sensor is used to measure the temperature value. The connections for the LM35 are Vcc, GND, and Vout. The analog voltage The sensor was connected to the Arduino microcontroller to read the analog voltage output. As the temperature changed, the analog voltage was generated at its output pin. This voltage was used to obtain the surrounding temperature. The sensor has successfully captured the temperature values even though the values are too high. The collected data will be transmitted to a smartphone using wireless Bluetooth.

- Python

Python was used to receive and then process the data that was transmitted from the Arduino. As shown in the Arduino code, the Python code interprets the data it has received from the sensor through Bluetooth. To generate and plot the graph we should use the matplotlib library. When the temperature changes, the data will be plotted in the Python script. However, we could not obtain the graph through Python.

- Led conduct

  After the transmission of the temperature data from the sensor to the smartphone we could control the LED. This LED acts as the heater indicator. Using the serial Bluetooth terminal application that we have pre-installed in the smartphone, we could connect with the Bluetooth HC-06. In the application, we could control the LED by simply clicking 1 to turn on the LED and clicking 0 to turn off the LED. When the LED is on, it indicates that the heater is on while if the LED is off, it indicates that the heater is off.

Conclusion

In conclusion, the experiment aimed to create a wireless temperature monitoring system using Wi-Fi, Arduino, and temperature sensors such as DHT11 and LM35 was successful. The ESP8266 module connected to the internet, transmitting temperature and humidity data from the DHT11 sensor to Thingspeak, providing real-time monitoring and plotting capabilities. However, challenges were encountered in the Bluetooth HC-06 setup, resulting in unstable connections and higher-than-expected temperature readings. The experiment revealed issues with Arduino port connectivity, requiring specific adjustments for reliable data transfer. The LM35 temperature sensor effectively captured temperature values but produced readings that were too high. Despite these challenges, the experiment successfully modelled the remote control of a heater using a Bluetooth terminal application on a smartphone.

Recommendation

The need for improvement is always needed to make sure the accuracy, reliability, and adaptability of the experiment results. Researchers can enhance their techniques and cut down on possible error sources through continuous improvement. The findings become more reliable as a result, yielding more exact and accurate outcomes. Below are a few recommendations that we can improve in the future.

1. **Testing and calibration:** In the future, we should ensure that the sensor LM35 is calibrated correctly in the experiment so that the temperature values collected from the data are accurate. Calibration can help to achieve more accurate and reliable data for our experiment results.

2. **Bluetooth troubleshooting:** In the task part, we received temperature data from the sensor however the value of temperature was not as we expected. Using a more reliable Bluetooth module or troubleshooting in the future should be considered.

3. **Python code improvement:** The Python should be enhanced to handle errors correctly and provide meaningful errors statements**.** The python code should contain necessary libraries including matplotlib so that the generated and plotted graph will be displayed in the python script.

4. **Ensure connection and components:** Ensure the components are correctly connected. The connection between bluetooth and colour sensor are perfectly connected so that the data is precise and reliable.

By implementing these recommendations, we can improve the system's reliability and efficiency. Hence, it will reduce the errors that might occur in the future.

References

Link:

1.LM35 Temperature Converting

https://forum.arduino.cc/t/lm35-temp-sensor-equation/596512/2

2.LM35 with Arduino

https://www.youtube.com/watch?app=desktop&v=3Xc2sPhwWEc&ab_channel=BINARYUPDATES

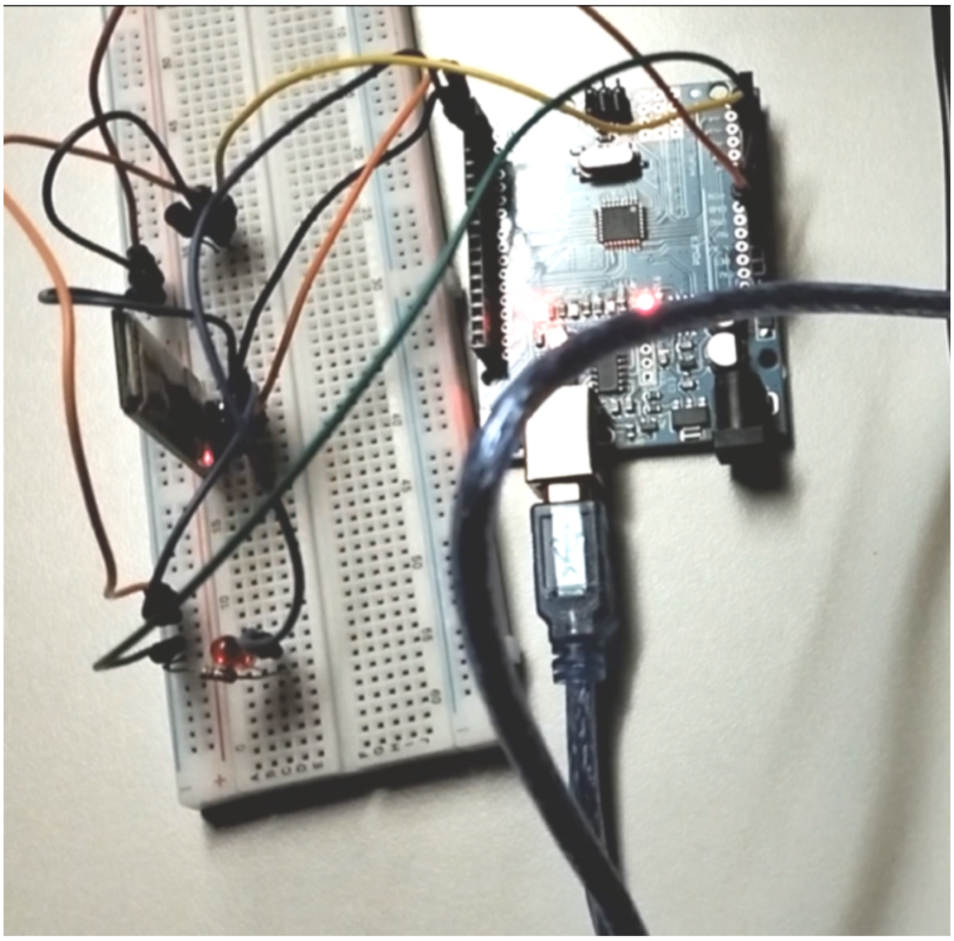
3.HC-05 Bluetooth Module with Arduino

https://www.youtube.com/watch?v=uT8-HPMS1cU&ab_channel=TechatHome

PDF:

Zulkifli. (2014). *Mechatronics Interfacing Lab Manual*, (Rev. ed.). Unpublished Class Materials

Appendices



Circuit for Bluetooth and LM35 using Arduino

Acknowledgments

We would like to express our sincere gratitude to the individuals who provided invaluable assistance, guidance, and support during this experiment. First and foremost, we extend our appreciation to Dr. Wahju Sediono for their comprehensive instruction and mentorship throughout the experiment. Their insights, feedback, and enthusiasm played an important role in our understanding of Arduino programming.

Our fellow group members also deserve special acknowledgment for their collaboration and support. Our discussions, knowledge-sharing, and problem-solving sessions greatly enriched our understanding of this experiment's concepts and enhanced the overall learning experience. The collective contributions of our group members have not only enriched our learning experience but have also significantly contributed to the successful completion of this project.

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 **read** and **understand** the content of the report and no further improvement on the report is needed from any of the individual's contributions 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	**sofea**	
Name	AISYAH SOFEA OTHMAN	**Read** ✅
Matric Number	2115386	**Understand** ✅

Contribution	- Experimental setup - Methodology - Material & Equipment	**Agree** ✅

Signature	**aliaa**	
Name	SITI ALIAA BINTI IBRAHIM	**Read** ✅
Matric Number	2112618	**Understand** ✅
Contribution	- Recommendation - Discussion	**Agree** ✅

Signature	**adli**	
Name	ADLI FAHMI	**Read** ✅
Matric Number	2113095	**Understand** ✅
Contribution	- Abstract - Introduction	**Agree** ✅

Signature	**ethar**	
Name	ETHAR OSMAN	**Read** ✅
Matric Number	2111282	**Understand** ✅
Contribution	- Conclusion - Result	**Agree** ✅

Signature	**ainaa**	
Name	NURAIN AINAA AQILAH BINTI ROSLI	**Read** ✅

Matric Number	2114560	**Understand** ✅
Contribution	-Data Collection -Data Analysis	**Agree** ✅