
**MECHATRONICS SYSTEM INTEGRATION**

**MCTA 3203**

# LAB 8:

# BLUETOOTH AND WIFI DATA INTERFACING WITH MICROCONTROLLER AND COMPUTER BASED SYSTEM

**SECTION 1**

**SEMESTER 2, 2024/2025**

**INSTRUCTOR:**

**ASSOC. PROF. EUR. ING. IR. TS. GS. INV. DR. ZULKIFLI BIN ZAINAL ABIDIN**

**DR. WAHJU SEDIONO**

**GROUP 9**


| NAME | MATRIC NO |
|---|---|
| AMER WAFDI BIN ZAINIZAM | 2228703 |
| AMALIN BALQIS BINTI RUYUSNI | 2314684 |
| FARHA QAISARA BINTI MOHD ZAHIDI | 2315994 |


**DATE OF SUBMISSION: 5 MAY 2025**

**ABSTRACT**

This experiment aimed to design and test a Bluetooth-enabled temperature monitoring and fan control system using an Arduino Mega 2560, LM35 temperature sensor, and HC-05 Bluetooth module. The objective was to measure ambient temperature in real-time, transmit the data wirelessly, and control a fan based on temperature thresholds. The experimental setup involved connecting the LM35 sensor and Bluetooth module to the Arduino and developing code to automate data collection and fan control. Temperature readings were transmitted to both the Serial Monitor and a smartphone via Bluetooth, while a Python script logged and graphed the data over a 30-second interval. Results showed that the temperature ranged between 27.86°C and 29.33°C, remaining below the 30 °C threshold required to activate the fan automatically. Manual fan control via Bluetooth commands was successfully demonstrated, validating system responsiveness. The experiment confirmed the effectiveness of the LM35 sensor in detecting small temperature variations and the reliability of Bluetooth-based communication for remote monitoring and control. Overall, the system functioned as intended, offering a foundation for future development in automated environmental control applications.

**TABLE OF CONTENT** ..... **3**

## 1.0 INTRODUCTION

This experiment aims to develop and validate a Bluetooth-enabled temperature monitoring and control system using an Arduino Mega 2560, an LM35 temperature sensor, and an HC-05 Bluetooth module. The primary objective is to enable bidirectional communication between the smartphone and Arduino, measuring real-time ambient temperature, transmitting the data wirelessly to a smartphone, and controlling a fan based on predefined temperature thresholds. The system is designed to operate both automatically, activating the fan when the temperature exceeds a set limit and manually via Bluetooth commands sent from a smartphone. By plotting temperature trends using Python's data visualization tools, this project also explores how collected data can be visualized and analyzed over time.

In recent years, the integration of microcontrollers with wireless communication has opened doors to smart environmental monitoring systems. The LM35 temperature sensor, known for its linear output and ease of interfacing with microcontrollers, converts ambient temperature into a proportional voltage signal. The Arduino Mega 2560, equipped with a 10-bit analog-to-digital converter (ADC), digitizes this signal and processes it using embedded control logic. The HC-05 Bluetooth module facilitates seamless wireless communication, allowing remote access and control through Bluetooth-supported devices such as smartphones. This forms the foundation for many real-world Internet of Things (IoT) applications in home automation and environmental monitoring.

The hypothesis of this experiment is that the system will successfully monitor temperature variations in real time, accurately reflect changes via both serial and Bluetooth outputs, and respond to control commands appropriately. It is also expected that the automatic fan control logic will activate the fan when the temperature reaches or exceeds 30 °C, and that manual override commands will function effectively. This experiment not only tests the core functionality of temperature sensing and control but also verifies the reliability of wireless data transmission and command execution within a short observation window.

## 2.0 MATERIALS AND EQUIPMENT

The materials and equipment that we used in this experiment are:

a) Arduino Mega 2560 board

b) Temperature sensor - LM35

c) Bluetooth module - HC-05

d) Smartphone with Bluetooth support

e) Breadboard

f) Jumper wires

## 3.0 EXPERIMENTAL SETUP

Here are the steps of equipment and components that were set up for the experiment:

1) The LM35 temperature sensor was connected to the Arduino by connecting its VCC pin to the 5V pin, GND pin to the GND pin, and the output pin to analog pin A0 of the Arduino.
2) The Bluetooth module HC-05 was connected to the Arduino by connecting its VCC to 3.3V pin, GND to GND pin, RXD to TX3 pin and TXD to RX2 pin of the Arduino.
3) Jumper wires were used to establish all connections between the Arduino board, temperature sensor, Bluetooth module and the breadboard.
4) The Arduino board was connected to the PC via USB cable to power the board and enable serial communication.
5) The Arduino was connected to the Wi-Fi network using the built-in Wi-Fi capabilities.

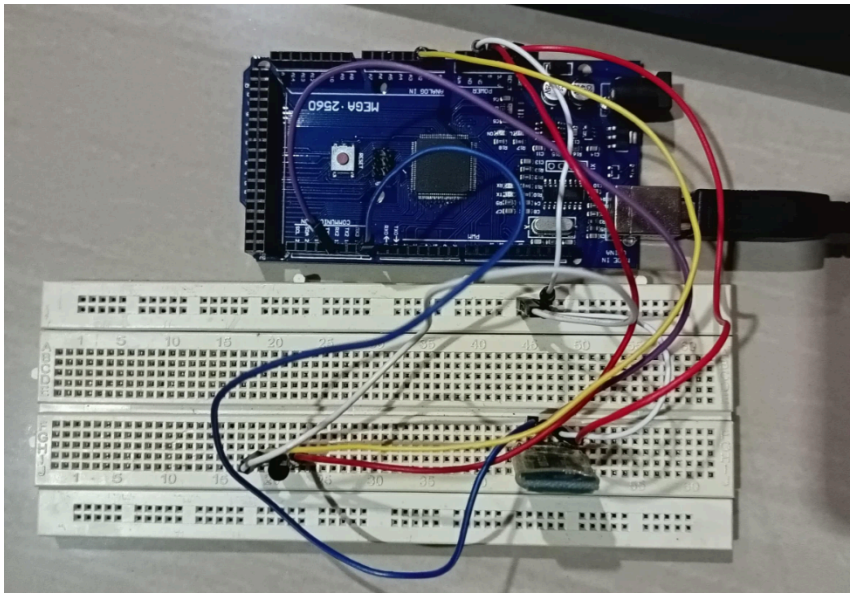


**Figure 1: Circuit Setup**

## 4.0 METHODOLOGY

Here are the steps followed during the experiment:

1) The circuit was built as shown in **Figure 1** and the Arduino Mega 2560 was set up.
2) An Arduino code was written and uploaded to read temperature data from the sensor at regular intervals and enable Bluetooth communication.
3) The code functionality was verified by observing the Serial Monitor and Bluetooth output.
4) A basic serial Bluetooth terminal app was installed on a smartphone.
5) The smartphone was paired with the HC-05 module using Bluetooth settings.
6) The HC-05 was connected via the app and verified incoming temperature readings
7) The "AUTO" command was sent from the smartphone to the Arduino.
8) The Arduino processed the received commands and switched the fan on or off accordingly based on temperature.
9) The python code was written to plot a graph of temperature over time.
10) Temperature readings were recorded over time to observe how environmental temperature changed and how control actions affected them.

**The Arduino code used:**

```
#include <SoftwareSerial.h>

const int analogPin = A0;       // LM35 sensor connected to A0

const int fanPin = 8;           // Fan connected to digital pin 8 (via
relay)

const float voltageRef = 5.0;   // Voltage reference (5V)

const int sensorMax = 1023;     // Max analog value for 10-bit ADC



bool manualOverride = false;    // Tracks if phone manually controls fan

bool fanState = false;          // Stores the fan's ON/OFF state



SoftwareSerial bluetooth(2, 3); // RX, TX (connect HC-05 TX to pin 2, RX
to pin 3)



void setup() {

  Serial.begin(9600);           // For Bluetooth (HC-05)

  bluetooth.begin(9600);        // Bluetooth communication

  pinMode(fanPin, OUTPUT);      // Set fan pin as output

  digitalWrite(fanPin, LOW);    // Fan is OFF initially

  Serial.println("Bluetooth Fan Control Ready");

}



void loop() {

  // Read temperature from LM35

  int sensorValue = analogRead(analogPin);

  float voltage = sensorValue * (voltageRef / sensorMax);

  float temperature = voltage * 100.0; // LM35 outputs 10mV/°C
```

```
// Send temperature to both Serial Monitor and Bluetooth
String tempMessage = "Temperature: " + String(temperature) + " °C";
Serial.println(tempMessage);
bluetooth.println(tempMessage);


// Auto fan control (only if not manually overridden)
if (!manualOverride) {
  if (temperature >= 30.0 && !fanState) {
    digitalWrite(fanPin, HIGH);  // Turn fan ON
    fanState = true;
    Serial.println("Auto: Fan ON");
    bluetooth.println("Auto: Fan ON");
  } else if (temperature < 30.0 && fanState) {
    digitalWrite(fanPin, LOW);   // Turn fan OFF
    fanState = false;
    Serial.println("Auto: Fan OFF");
    bluetooth.println("Auto: Fan OFF");
  }
}


// Check for incoming Bluetooth commands
if (bluetooth.available()) {
  String command = bluetooth.readStringUntil('\n');
  command.trim();


  if (command == "FAN_ON") {
    digitalWrite(fanPin, HIGH);  // Turn fan ON
```

```
      fanState = true;
      manualOverride = true;
      Serial.println("Manual: Fan ON");
      bluetooth.println("Manual: Fan ON");
    } else if (command == "FAN_OFF") {
      digitalWrite(fanPin, LOW);   // Turn fan OFF
      fanState = false;
      manualOverride = true;
      Serial.println("Manual: Fan OFF");
      bluetooth.println("Manual: Fan OFF");
    } else if (command == "AUTO") {
      manualOverride = false;
      Serial.println("Switched to AUTO mode");
      bluetooth.println("Switched to AUTO mode");
    }
  }

  delay(1000);  // 1 second delay between readings
}
```

- **The Python code:**

```
import serial
import matplotlib.pyplot as plt
import re
import time

# CONFIGURATION
port = 'COM4'  # Replace with your actual COM port
baud_rate = 9600
timeout = 2
duration = 30  # Duration in seconds to record data

# Connect to serial
try:
    ser = serial.Serial(port, baud_rate, timeout=timeout)
    print(f"Connected to {port} at {baud_rate} baud.")
except Exception as e:
    print(f"Connection failed: {e}")
    exit()

temperatures = []
timestamps = []

start_time = time.time()

print(f"\nCollecting data for {duration} seconds...\n")

try:
    while True:
        current_time = time.time()
        elapsed = current_time - start_time

        if elapsed > duration:
            print("Finished data collection.\n")
            break

        raw = ser.readline().decode('utf-8').strip()
        if raw:
            match = re.search(r"([-+]?\d*\.\d+|\d+)", raw)
            if match:
```

```
                temperature = float(match.group(1))
                temperatures.append(temperature)
                timestamps.append(elapsed)
                print(f"{elapsed:.1f}s -> {temperature:.2f} °C")
            else:
                print(f"Ignored: {raw}")
        else:
            print("No data received.")

finally:
    ser.close()
    if temperatures:
        plt.plot(timestamps, temperatures, marker='o', linestyle='-')
        plt.title("Temperature Over 30 Seconds")
        plt.xlabel("Time (s)")
        plt.ylabel("Temperature (°C)")
        plt.grid(True)
        plt.tight_layout()
        plt.show(block=True)
    else:
        print("No temperature data to plot.")
```

## 5.0 DATA COLLECTION

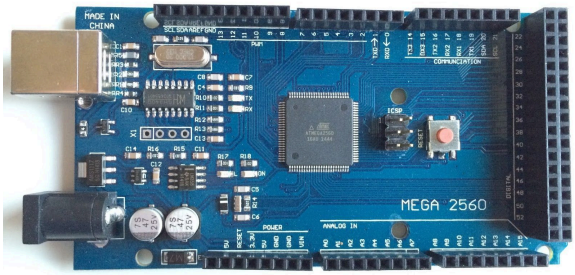
Temperature was measured using the LM35 sensor connected to analog pin A0 of the Arduino Mega 2560. The LM35 outputs 10mV per °C and uses the Arduino's 5V reference. Data was wirelessly transmitted via an HC-05 Bluetooth module to a computer or smartphone. The Arduino continuously read and converted the sensor's output using its 10-bit ADC, then sent the temperature data to both Bluetooth and the Serial Monitor. Python recorded this data every second for 30 seconds and logged the time and temperature values.

| Time (s) | Temperature (℃) | Time (s) | Temperature (℃) |
|---|---|---|---|
| 1.0 | 28.35 | 16.5 | 27.86 |
| 2.0 | 28.35 | 17.5 | 28.35 |
| 3.0 | 28.84 | 18.5 | 27.86 |
| 4.1 | 28.35 | 19.6 | 28.35 |
| 5.1 | 28.35 | 20.6 | 29.33 |
| 6.1 | 28.84 | 21.6 | 27.86 |
| 7.2 | 28.35 | 22.7 | 27.86 |
| 8.2 | 28.35 | 23.7 | 28.35 |
| 9.2 | 28.35 | 24.8 | 27.86 |
| 10.3 | 28.35 | 25.8 | 27.86 |
| 11.3 | 28.35 | 26.8 | 27.86 |
| 12.3 | 28.35 | 27.9 | 27.86 |
| 13.4 | 28.35 | 28.9 | 27.86 |
| 14.4 | 27.86 | 29.9 | 27.86 |
| 15.4 | 28.35 | | |

**Table 1: Data Collection for 30 seconds**

Instruments used for data acquisition:

1) Arduino Mega 2560: A microcontroller board based on the ATmega2560, used to process read temperature and sends the data via HC-05 to the smartphone.




2) Bluetooth HC-05 : Enables wireless communication between the Arduino and smartphone.

3) LM 35 sensor: Measures the room's temperature.

4) Jumper wires: Used to establish electrical connections between components on the breadboard.
5) Breadboard: A prototyping platform used to connect components without soldering, allowing for easy circuit modifications.

## 6.0 DATA ANALYSIS

The temperature values observed over a time interval of 30 seconds were measured to obtain the trend of variability in indoor thermal conditions. The values, ranging from around 27.9 °C to 29.3 °C, fluctuated slightly in accordance with usual stable indoor environments.

To quantify such variations, average temperature was measured. Based on the data measured, the average temperature was around 28.5 °C, and its range (minimum-maximum difference) was 1.4 °C. These small fluctuations reflect that the environment was thermally stable throughout the experiment without any considerable changes.

No advanced mathematical models were employed here since the aim was to monitor live developments and check their response as measured by the sensors. Graphing the data did, however, prove helpful in recording trends, for example, a brief spike at the 20-second point followed by a phase of slow-down.

This test confirms that the LM35 sensor was able to accurately measure small but real temperature changes, confirming the intention of testing real-time temperature sensing and fan control logic within the system. The data collected supports the experiment's objective of monitoring real-time temperature and controlling a fan based on defined thresholds. While the environment did not heat up sufficiently to test the fan's automatic behavior, the experiment validates the foundational structure of the system and its readiness for further testing under more variable conditions.

## 7.0 RESULTS

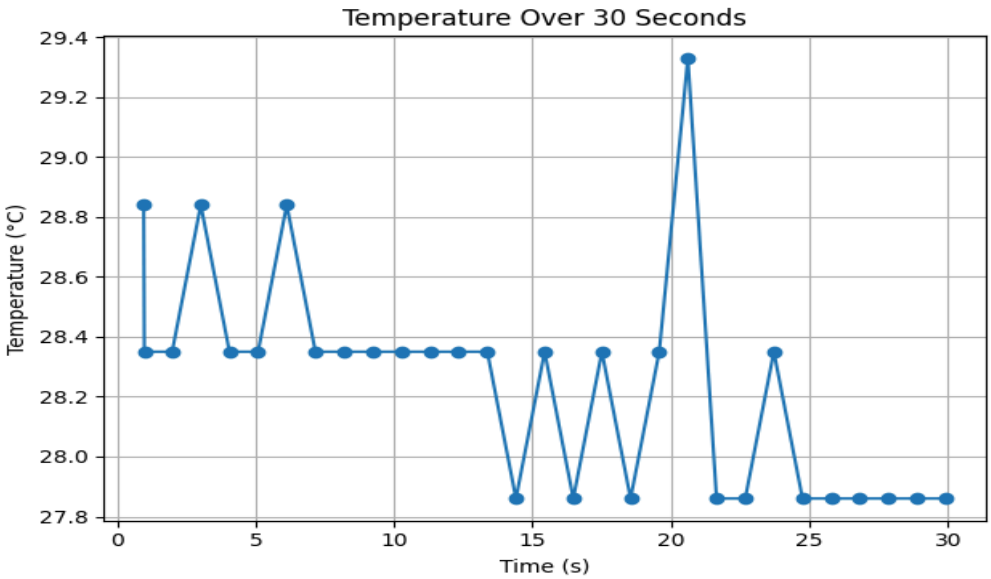



**Figure 2: Graph of Temperature vs Time**

The experiment successfully demonstrated the functionality of a Bluetooth-enabled temperature monitoring and fan control system using an Arduino microcontroller, an LM35 temperature sensor, and an HC-05 Bluetooth module.

The experiment recorded the room temperature for 30 seconds using Arduino Mega 2560 and LM35 sensor, and transmitted both to the Serial Monitor and a smartphone application.. As can be seen from **Figure 2**, the temperature was in the close range of between 27.86°C and 29.33°C with little fluctuation throughout. The readings started steady at approximately 28.8°C. Halfway, the temperature dipped to 27.86°C, rising to a high of 29.33°C around the 20-second interval. From that point, it decreased steadily again and hovered at approximately 27.86°C for the rest.

The system's automatic control logic, which is designed to turn the fan ON when the temperature reaches or exceeds 30 °C, was not triggered during the observation period due to the relatively stable and low ambient temperature. The fan remained OFF in automatic mode throughout the test. Additionally, Bluetooth commands such as "FAN_ON," "FAN_OFF," and "AUTO" were successfully used to manually control the fan, confirming that the override mechanism functioned correctly.

The temperature data was plotted using a Python script to visualize trends over time. The graph revealed minimal temperature variation, supporting the statistical analysis that showed a mean temperature of 28.28 °C. These results align with expectations for a stable indoor environment. No abrupt spikes or sensor malfunctions were observed. The successful transmission and reception of temperature data via Bluetooth further verified the integrity of the communication link between the Arduino and the smartphone interface. Overall, the LM35 sensor was effective at sensing minute changes in temperature, showing that it's sensitive and reliable for short-term monitoring.

Here's the link for the video:
https://github.com/amalinblqs/MCTA3203_GROUP_9/blob/9fb233b1ec6d42754f144859b79bcaa7cb856458/Lab%208/TutoW8.mp4

## 8.0 DISCUSSION

The results indicate that the LM35 temperature sensor provided consistent and responsive temperature readings within the duration of monitoring as 30 seconds. The slight fluctuations and the small spike evident on or around the 20th second prove sensitivity to minimal changes in surrounding environments, hence the reliability of the sensor in real-time monitoring applications such as automated fan control. The results obtained from the experiment align with the expected behavior of the system under normal room temperature conditions. Since the LM35 sensor detected temperatures consistently below the 30 °C threshold, the fan did not activate in automatic mode. This confirms that the temperature sensing and decision-making logic implemented in the Arduino code is functioning correctly.

Ideally, in a stable room environment, the temperature was expected to remain relatively constant. The sharp rise near the 20-second mark was slightly unexpected and could be attributed to brief external factors such as human presence near the sensor, airflow changes, or heat from nearby electronics. This highlights the LM35's responsiveness but also suggests it may react to transient environmental changes not related to room temperature itself.

One notable observation was the lack of temperature fluctuation significant enough to trigger automatic fan activation. While this confirms system stability, it limits the opportunity to test the full range of automatic control behavior. Introducing a heat source near the LM35 sensor during testing could have provided a broader dataset, including instances where the fan turned ON and OFF in response to temperature thresholds. This would have strengthened the validation of the automatic control system.

Possible sources of error include analog noise in sensor data, power supply voltage variations, or small timing differences in the data acquisition due to the use of serial communication. The experiment was run for only 30 seconds, so longer-term trends and control responses, such as the fan turning on may not be fully represented in this short time. Additionally, the LM35 sensor, while simple and reliable, can be slightly influenced by ambient airflow and placement, potentially affecting precision. Noise in temperature readings was minimal but could be further reduced with averaging or digital filtering techniques.

Overall, the experiment revealed that the system can be relied upon to collect and read real-time temperature data, but with longer observation periods and more controlled situations, data accuracy and interpretation can be improved.

## 9.0 CONCLUSION

This experiment successfully demonstrated the design and implementation of a Bluetooth-based temperature monitoring and fan control system using an Arduino microcontroller, LM35 temperature sensor, and HC-05 Bluetooth module. The system effectively measured ambient temperature in real time and transmitted the data to both a serial interface and a smartphone application. The temperature trend observed was consistent with anticipated indoor conditions, thereby confirming the hardware configuration as well as the data logging method. While the recorded temperatures remained below the 30 °C threshold, preventing automatic fan activation, the system's response to manual Bluetooth commands confirmed the correct functioning of both manual and automatic control modes. These findings validate the core functionality and reliability of the system under stable temperature conditions.

The device was successful in capturing the change in temperature over a 30-second time frame with small but consistent changes between 27.86°C and 29.33°C. Such findings proved that the LM35 sensor is sensitive and has the capability of detecting small variations in temperature.

The results support the initial hypothesis that a smartphone-controlled fan could be regulated based on real-time temperature data transmitted via Bluetooth. Although automatic fan activation was not triggered during this test due to the consistently low temperatures, the logical control structure and Bluetooth communication proved to be accurate and responsive.

The broader significance of this experiment lies in its demonstration of how low-cost microcontroller platforms can be integrated with mobile technology to create effective remote environmental monitoring and automation systems. This system has wider applications in residential automation in smart homes, where precise and timely temperature data can be used to

control fans or air conditioning systems. This system can also be employed for server room applications, greenhouses, or other settings requiring stable thermal conditions.

## 10.0 RECOMMENDATIONS

To enhance the depth and usefulness of future experiments, it is recommended to conduct testing in a more dynamic thermal environment, perhaps by introducing a controlled heat source near the temperature sensor. This would allow for testing of the automatic fan activation feature under real conditions. For long-term optimization, adding a data logging feature to capture temperature readings on a cloud service like ThingSpeak would enable long-term monitoring and analysis. A digital temperature sensor, such as the DHT22 would also provide more accurate and stable measurements for fluctuating conditions.

In addition, protecting the sensor from direct air flow or sunlight exposure might enhance reliability by removing noise in measurements. Also, increasing the data collection period and sampling rate could reveal more detailed temperature trends.

Future iterations could benefit from expanding the functionality of the smartphone application, such as including real-time graphing, push notifications, or adjustable temperature thresholds. From a learning perspective, students are encouraged to pay close attention to sensor calibration, Bluetooth pairing stability, and the design of intuitive user interfaces. This project offers valuable hands-on experience in embedded systems, communication protocols, and practical IoT applications, making it an ideal foundation for more advanced automation and smart home projects. Future students are also encouraged to thoroughly test serial communication between devices beforehand, and to label all hardware connections clearly. Using plotting

libraries like matplotlib proved very useful for visual analysis, and documenting the code and setup process step-by-step makes troubleshooting much easier.

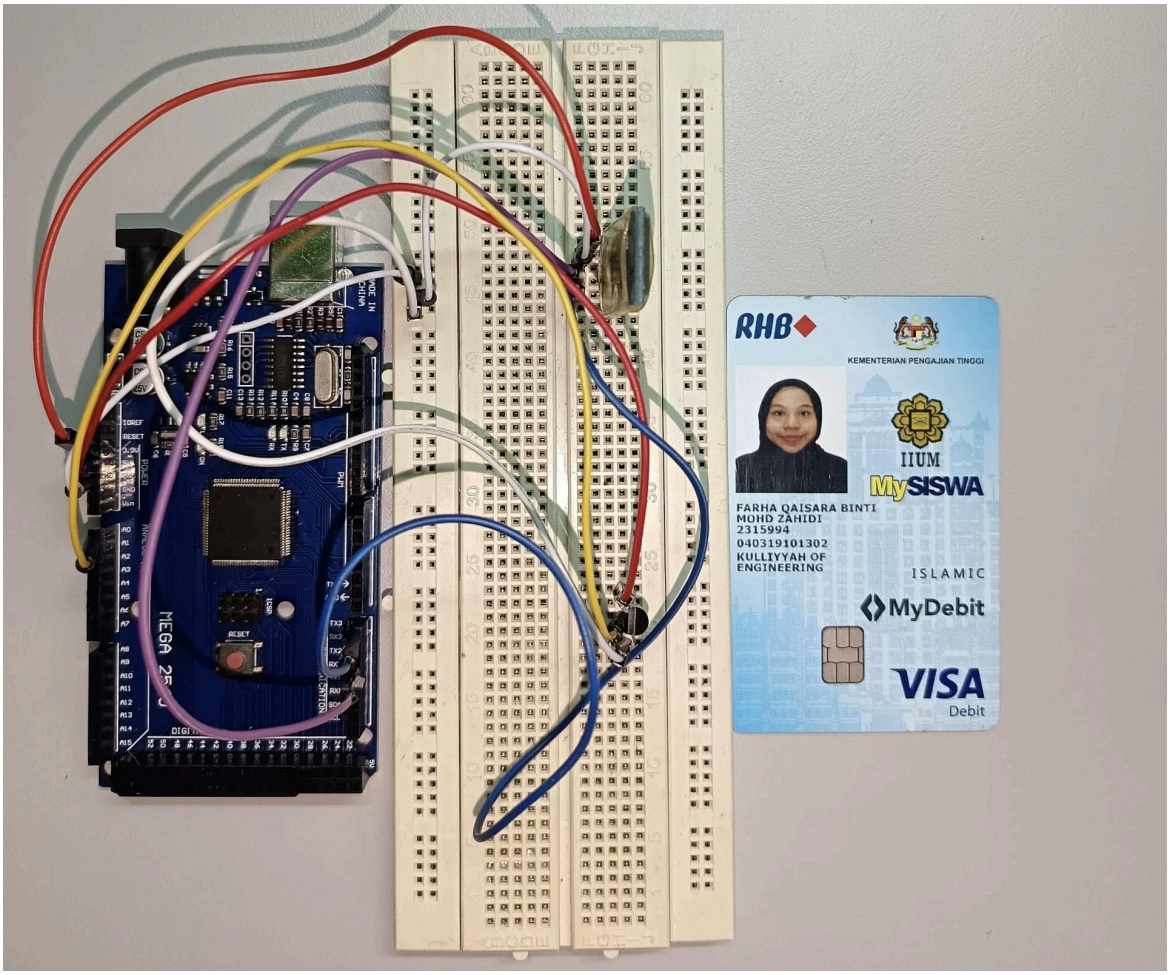
## 11.0 REFERENCES


How2Electronics. (2024, January). *Digital thermometer with Arduino & LM35 temperature sensor*. https://how2electronics.com/digital-thermometer-arduino-lm35-temperature-sensor/How2Electronics

Programming Boss. (2021, March). *LM-35 temperature sensor and Arduino-temperature monitor using LCD display*. https://www.programmingboss.com/2021/03/lm-35-temperature-sensor-and-arduino.htmlprogrammingboss.com

**APPENDICES**

Lab tutorial:
https://github.com/amalinblqs/MCTA3203_GROUP_9/blob/9fb233b1ec6d42754f144859b79bcaa7cb856458/Lab%208/TutoW8.mp4



**ACKNOWLEDGMENTS**

First of all, we would like to express our deepest appreciation to all those who provided us the possibility to complete this report. We would like to express our sincere gratitude to our lecturer, Assoc. Prof. Eur. Ing. Ir. Ts. Gs. Inv. Dr. Zulkifli Bin Zainal Abidin and Dr. Wahju Sediono for their invaluable guidance and support throughout this lab. Their insightful explanations and encouragement greatly enhanced our understanding of the concepts covered.

Next, a special thanks goes to our own teammates, Wafdi, Amalin and Farha, who helped to build up the circuit and set up the programming code using Arduino IDE, and also successfully completed this experiment. Last but not the least, we would like to thank everyone, especially our classmates who are willingly helping us out in the lab experiment directly or indirectly.

**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 acknowledgement, and that the original work contained herein have 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's contributors 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: *Wafdi* Read [ / ]
Name: Amer Wafdi bin Zainizam Understand [ / ]
Matric Number: 2228703 Agree [ / ]
Contribution: Introduction, materials and equipment, methodology & recommendations

Signature: *Amalin* Read [ / ]
Name: Amalin Balqis binti Ruyusni Understand [ / ]
Matric Number: 2314684 Agree [ / ]
Contribution: Abstract, experimental setup & conclusion

Signature: *farha* Read [ / ]
Name: Farha Qaisara binti Mohd Zahidi Understand [ / ]
Matric Number: 2315994 Agree [ / ]
Contribution: Data collection, data analysis, results & discussion