INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA

Garden of Knowledge and Virtue

LAB REPORT 8: REMOTE TEMPERATURE MONITORING CONTROL

GROUP 1

MCTA 3203

SEMESTER 1 2024/2025

MECHATRONICS SYSTEM INTEGRATION

NO	NAME	MATRIC NO
1.	ABDUL A'LIM BIN MOHD RAJIB	2119687
2.	AHMAD HAZAMI BIN MOHD RAZIP	2211203
3.	ABDUL HADI BIN ZAWAWI	2210739
4.	AIN MAISARA BINTI ABDULLAH	2217856
5.	ADIBAH BINTI MOHD AZILAN	2212670

DATE OF SUBMISSION: 11/12/2024

ABSTRACT

In this project, the connection between Arduino microcontroller, a temperature sensor, and ThinkSpeak platform for real-time temperature monitoring is showcased. Through this connection, a temperature monitoring system is created, where temperature values are sensed by the sensor, and the data obtained is processed by using the Arduino. Finally, it is uploaded into ThinkSpeak for display and remote monitoring. Once the temperature is detected by the sensor, the data is sent to ThinkSpeak platform through Wi-Fi connectivity. This project focuses on the integration between hardware and cloud-based applications for efficient environmental monitoring.

TABLE OF CONTENTS

1. INTRODUCTION

The integration of Bluetooth and Wi-Fi technologies with microcontrollers and computer-based systems has revolutionized modern data processing and device interfacing. These wireless communication protocols enable seamless exchange of data between sensors, actuators, and computational systems, facilitating real-time monitoring and control in diverse applications. Bluetooth, known for its short-range and energy-efficient communication, is widely employed in portable devices and IoT applications. Wi-Fi, with its higher data throughput and extended range, is a preferred choice for systems requiring robust internet connectivity.

This report explores the interfacing of Bluetooth and Wi-Fi with microcontrollers, emphasizing their role in acquiring data from sensors, processing it, and triggering appropriate responses in actuators. By leveraging these wireless technologies, microcontroller-based systems achieve greater flexibility, scalability, and efficiency, making them essential in fields such as home automation, healthcare, and industrial control systems. The focus will also include the data processing mechanisms and the integration of these communication technologies into a cohesive system, ensuring optimal performance and reliability.

2. MATERIAL AND EQUIPMENT

- Arduino board with Wi-Fi capability (e.g., Arduino ESP8266, Arduino MKR1000, or an ESP32)

- Temperature sensor LM35

- Bluetooth module (e.g., HC-05 or HC-06)

- Smartphone with Bluetooth support

- Wi-Fi network and internet access

- Power supply for the Arduino

- Breadboard and jumper wires

3. EXPERIMENTAL SETUP
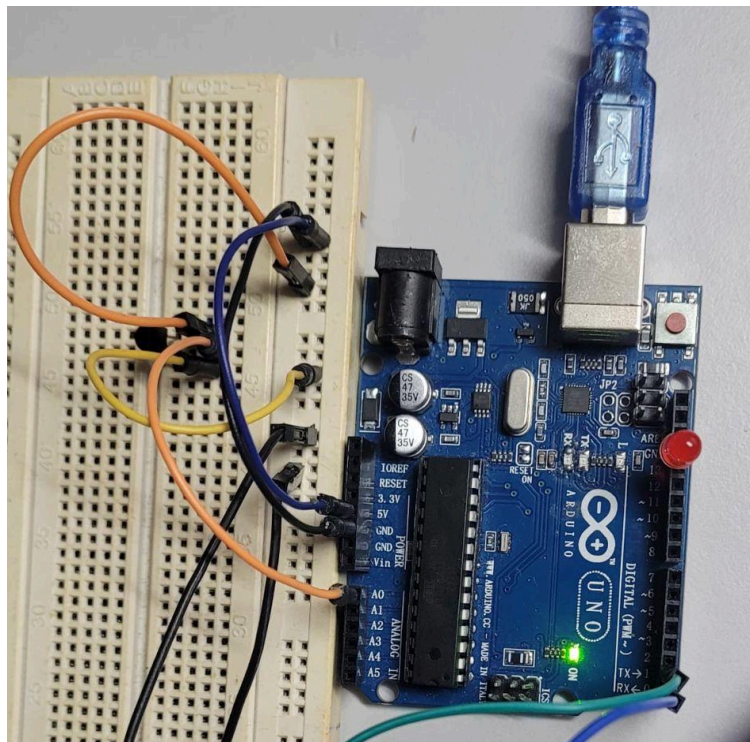
4. METHODOLOGY

PROCEDURES:

1. Hardware Setup:

    a. Connect the temperature sensor (thermistor) to the Arduino.

    b. Connect the Bluetooth module to the Arduino.

    c. Connect the Arduino to your Wi-Fi network using the built-in Wi-Fi capabilities.

2. Arduino Programming:

    a. Write an Arduino sketch that reads temperature data from the sensor.

    b. Set up Wi-Fi connectivity to send temperature data to a cloud service like ThingSpeak, where you can create a simple dashboard to visualize the data.

3. Bluetooth Programming:

    a. Write an Arduino sketch to enable Bluetooth communication1.
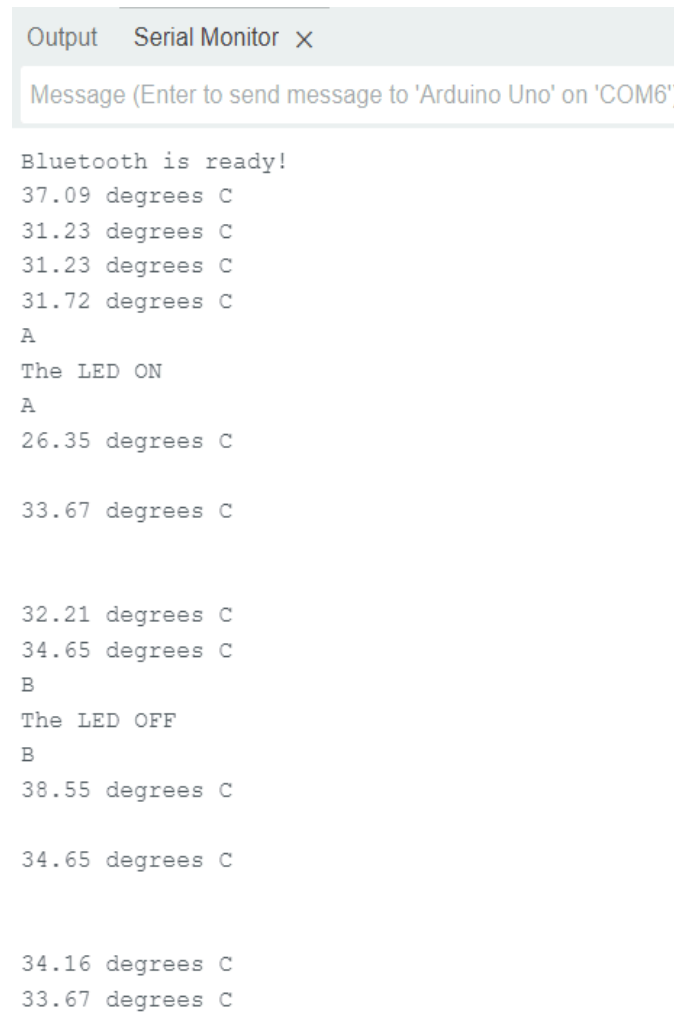
    b. Complete the task below

    Task: 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, like a fan or heater, based on the temperature data received from the Arduino (we used LED as an indicator).

4. Remote Monitoring:

    a. Access your ThingSpeak dashboard on your computer or smartphone to remotely monitor the temperature in real-time via the internet.

EXPERIMENT WORKFLOW:

1. Place the temperature sensor in a room or area you want to monitor and control the temperature.

2. Connect the Arduino to a power source and ensure it's connected to your Wi-Fi network.

3. Monitor the room's temperature in real-time using the ThingSpeak dashboard.

5. DATA COLLECTION


```
Output    Serial Monitor ×

Message (Enter to send message to 'Arduino Uno' on 'COM6'

Bluetooth is ready!
37.09 degrees C
31.23 degrees C
31.23 degrees C
31.72 degrees C
A
The LED ON
A
26.35 degrees C

33.67 degrees C


32.21 degrees C
34.65 degrees C
B
The LED OFF
B
38.55 degrees C

34.65 degrees C


34.16 degrees C
33.67 degrees C
```
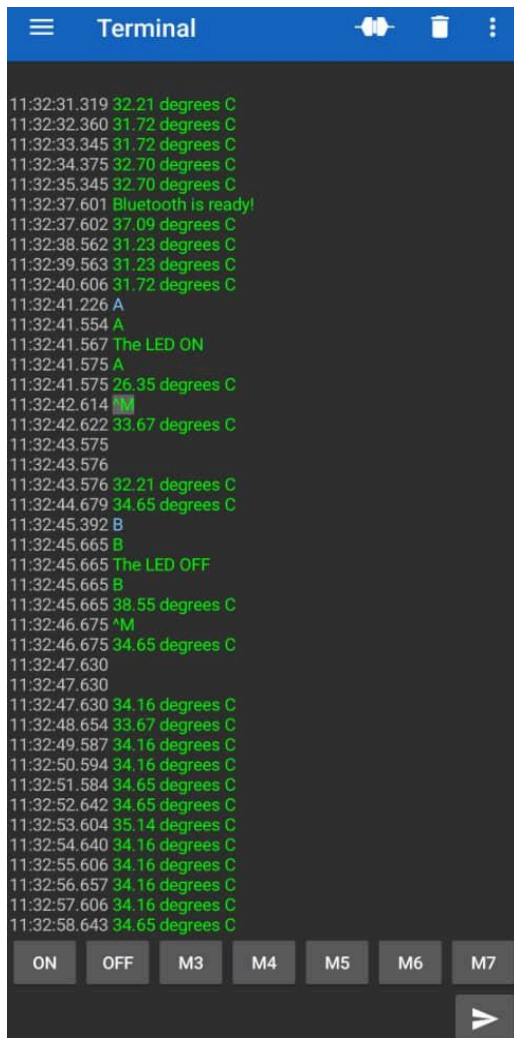
6. DATA ANALYSIS

The data presented showcases a functional Bluetooth-based system that effectively integrates temperature sensing and LED control. The system initializes successfully, indicated by the message "Bluetooth is ready!" and consistently logs temperature readings in degrees Celsius, which range between 26°C to 38°C. This demonstrates the system's ability to capture live environmental data. Simultaneously, Bluetooth commands ("A" and "B") are sent to control the LED, with responses confirming the actions—"A" turns the LED ON, and "B" turns it OFF. Despite these commands, the system continues to log temperature data without interruptions, highlighting its multitasking capabilities.
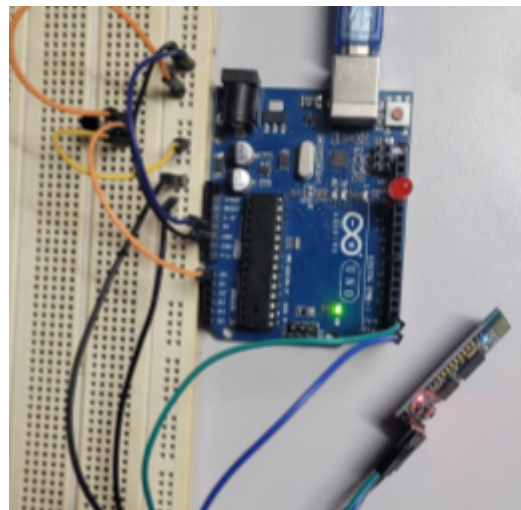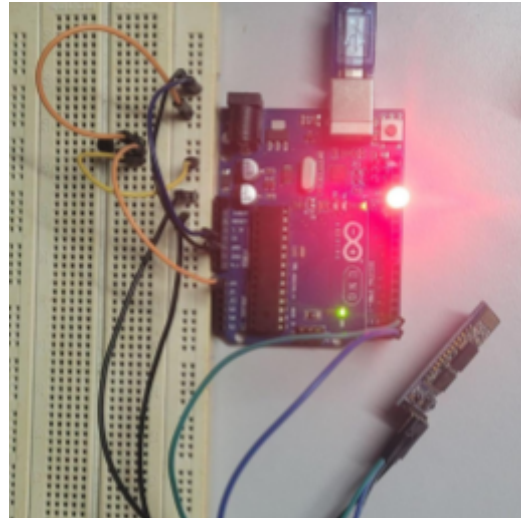
The terminal interface mirrors the serial monitor's output and offers additional functionality with labeled buttons such as "ON," "OFF," and others, suggesting an intuitive and user-friendly design for real-time monitoring and control. The synchronization between the microcontroller's output and the terminal display ensures consistency and reliability. However, there is variability in the temperature data, likely reflecting real-time environmental changes or the sensor's precision limits. For improved analysis, adding timestamps to the readings would provide better context, and implementing error detection mechanisms could enhance system reliability. Overall, the system demonstrates robust performance in wireless communication, data acquisition, and device control.

7. RESULTS

8. DISCUSSION

The project highlights the importance of integrating Bluetooth communication for short-range, energy-efficient control and data transfer. The temperature sensor's ability to provide real-time readings complements the system's capability to handle simultaneous data acquisition and actuator control. The system's multitasking performance, as evidenced by its ability to process commands without interrupting temperature logging, demonstrates its effectiveness for real-world applications. However, variability in the temperature readings raises questions about sensor accuracy or potential environmental fluctuations. The clear

feedback provided by the interface, both in the serial monitor and the terminal, ensures smooth user interaction and system control.

The project could further benefit from exploring additional features such as Wi-Fi integration to extend the system's range and data throughput capabilities. This would enable remote monitoring and control, which is crucial for applications requiring scalability. Furthermore, the current interface design, while functional, could incorporate timestamps and graphical data visualization for better insights into temperature trends and system performance. The use of error-handling mechanisms would also enhance the system's reliability, especially in scenarios involving unexpected command inputs or sensor malfunctions.

9. CONCLUSION

The integration of Bluetooth and Wi-Fi with microcontrollers for data processing, sensor monitoring, and actuator control proves to be an efficient and versatile solution for real-time applications. The system demonstrates robust functionality by continuously logging temperature data while processing Bluetooth commands to control the LED. The seamless synchronization between the serial monitor and the terminal interface highlights its reliability and user-friendly design. This project showcases the potential of wireless communication technologies in creating scalable and efficient systems for various domains, including home automation, industrial control, and IoT applications.

10. RECOMMENDATION

To further enhance the system's functionality and reliability, several improvements are recommended. First, incorporating timestamps into the temperature readings would provide a clearer understanding of data trends and allow for better time-based analysis. Enhancing the accuracy of the temperature sensor through calibration and implementing algorithms to filter out anomalies or noise can ensure more reliable data. Additionally, integrating Wi-Fi alongside Bluetooth would significantly expand the system's capabilities, enabling remote monitoring and control for greater scalability in IoT applications. Introducing error-handling mechanisms to detect and address invalid commands or sensor faults would strengthen the system's robustness. Furthermore, the interface can be enhanced with graphical visualization of temperature data, making it more user-friendly and efficient for monitoring. Lastly, the additional buttons available in the terminal interface, such as M3 and M4, could be programmed to control other devices or trigger specific functions, further testing the system's multitasking capabilities. These improvements would make the system more versatile, reliable, and suitable for advanced real-world applications.

11. REFERENCES

[1] *LM35 Interfacing with Arduino UNO | Arduino*. (n.d.). © 2018 ElectronicWings.

https://www.electronicwings.com/arduino/lm35-interfacing-with-arduino-uno

[2] *Reading an Analog Temperature Sensor | Onion Omega2 Arduino Dock Starter Kit*.

(2024, December 10).

https://docs.onion.io/omega2-arduino-dock-starter-kit/arduino-kit-reading-a-temp-sensor.html

APPENDICES

```
1   #include <SoftwareSerial.h>
2   SoftwareSerial bluetooth(0, 1); // RX, TX
3   const int lm35_pin = A0; // Analog pin for thermistor
4   int led = 13;
5   int sensorValue = 0;
6   double temperature;
7   char command;
8   char data = 0;
9
10  void setup() {
11   Serial.begin(9600);
12   Serial.println("Bluetooth is ready!");
13   bluetooth.begin(9600);
14   pinMode(13, OUTPUT);
15  }
16
17  void loop() {
18    if(Serial.available() > 0){
19     command = Serial.read();
20     Serial.println(command);
21
22    }
23    if(command == 'A' && temperature >= 30)
24    {
25      digitalWrite(13, HIGH);
26      Serial.println("The LED ON");
27    }
28    if(command == 'B' )
29    {
30      digitalWrite(13, LOW);
31      Serial.println("The LED OFF");
32    }
33
34    while (bluetooth.available() > 0) {
35    char data = bluetooth.read();
36    Serial.print(data);
37    }
38    sensorValue = analogRead(lm35_pin);
39    temperature = (sensorValue * 4.88) / 10 ;
40    Serial.print(temperature);
41    Serial.println(" degrees C");
42
43    delay(1000);
44  }
```

```python
import serial
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Set up the serial connection (adjust COM port and baud rate as needed)
ser = serial.Serial('COM6', 9600)

# Lists to store temperature data and time points
temperatures = []
time_stamps = []

# Initialize the plot
plt.style.use('classic')
fig, ax = plt.subplots()
line, = ax.plot([], [], marker='o', label='Temperature (°C)')
ax.set_title("Real-Time Temperature Monitoring")
ax.set_xlabel("Time (s)")
ax.set_ylabel("Temperature (°C)")
ax.legend(loc="upper left")

# Update function for the real-time plot
def update(frame):
    global temperatures, time_stamps

    try:
        # Read and decode data from the serial port
        data = ser.readline().decode('utf-8').strip()
        temperature = float(data)

        # Append new data
        temperatures.append(temperature)
        time_stamps.append(len(time_stamps))  # Use the length of the list as a simple time counter

        # Update the plot
        line.set_data(time_stamps, temperatures)
        ax.relim()
        ax.autoscale_view()

        return line,

    except ValueError:
        # Handle any data conversion errors
        print("Received invalid data:", data)
        return line,

# Set up the animation
ani = FuncAnimation(fig, update, interval=1000)  # Update every 1 second

try:
    plt.show()  # Display the plot
except KeyboardInterrupt:
    print("Exiting...")

finally:
    ser.close()  # Ensure the serial port is closed
```

python

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to our lecturers, Dr. Wahju Sediono and Dr. Zulkifli bin Zainal Abidin for their invaluable guidance in this experiment. Their expertise in the mechatronics field and encouragement have been a big support in the successful integration of our system. Not forgetting our teaching assistants in the lab for helping us indirectly.

To add, we also wish to extend a special thank you to our fellow team members for their collaboration and hard work during the course of this project. Their contributions were pivotal in driving our efforts forward and enhancing the overall quality of our work.

STUDENTS' 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 that no further improvement on the reports is needed from any of the individual 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: A'LIM	Read	/
Name: ABDUL A'LIM BIN MOHD RAJIB	**Understand**	/
Matric No: 2119687	**Agree**	/

Signature: *hazami*	Read	/
Name: AHMAD HAZAMI BIN MOHD RAZIP	**Understand**	/
Matric No: 2211203	**Agree**	/

Signature:	Read	/
Name: ABDUL HADI BIN ZAWAWI	**Understand**	/
Matric No: 2210739	**Agree**	/

Signature:	Read	/
Name: AIN MAISARA BT. ABDULLAH	**Understand**	/
Matric No: 2217856	**Agree**	/

Signature:	Read	/
Name: ADIBAH BINTI MOHD AZILAN	**Understand**	/
Matric No: 2212670	**Agree**	/