



الجامعة الإسلامية العالمية ماليزيا  
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA  
يُونِيسُوتِي إِسْلَامُ، أَنْتَ أَرَاغُثْسَا مِلْدُسِيَا

*Garden of Knowledge and Virtue*

**REPORT 6: BLUETOOTH AND WIFI DATA INTERFACING WITH  
MICROCONTROLLER AND COMPUTER BASED SYSTEM**

**GROUP 4**

**MCTA 3203**

**SEMESTER 1 2024/2025**

**MECHATRONICS SYSTEM INTEGRATION**

**DATE OF SUBMISSION: 11 DECEMBER 2024**

NO	NAME	MATRIC NUMBER
1.	IRDINA NABIHAH BINTI MOHD NAZRI	2214772
2.	KHALISAH AMANI BINTI YAHYA AZMI	2218184
3.	LUQMAN AZFAR BIN AZMI	2219857
4.	MOHAMAD NASRI BIN MOHAMAD NAZRI	2219879
5.	MUHAMAD NURHAKIMIE THAQIF BIN ABDULLAH	2213217

## TABLE OF CONTENT

NO.	CONTENT
1.	INTRODUCTION
2.	ABSTRACT
3.	MATERIALS AND EQUIPMENT
4.	EXPERIMENTAL SETUP
5.	METHODOLOGY
6.	RESULT
7.	DISCUSSION
8.	CONCLUSION
9.	RECOMMENDATIONS
10.	ACKNOWLEDGEMENTS
11.	STUDENT'S DECLARATION

## **INTRODUCTION**

In this generation, wireless communication technologies have transformed environmental monitoring systems, facilitating real-time data gathering and remote management features. In this experiment, we created a wireless temperature monitoring setup using an Arduino microcontroller, an LM35 temperature sensor, and an HC-01 Bluetooth module. The primary objective was to create a strong and adaptable system capable of wirelessly transmitting temperature data, showcasing the effective combination of embedded systems, sensor technologies, and communication protocols.

The experiment aimed at enabling smooth interaction among hardware parts and developing an intuitive interface for monitoring temperature. By connecting the LM35 temperature sensor to the Arduino and using the HC-01 Bluetooth module, we developed a setup that can gather accurate temperature measurements and send them to a mobile app. This method not only highlights the capabilities of IoT (Internet of Things) technologies in monitoring the environment but also offers a scalable framework for creating smart, interconnected sensing solutions in diverse areas like home automation, industrial process management, and environmental studies.

## **ABSTRACT**

This experiment focuses on developing a smart monitoring and control system utilizing the LM35 temperature sensor and HC-01 Bluetooth module, integrated with an Arduino microcontroller. The primary objective was to create a wireless communication framework capable of real-time temperature monitoring and environmental control. The LM35 sensor was employed for its precision and reliability in capturing temperature data, while the HC-01 module facilitated seamless wireless communication between the Arduino and a smartphone application. By leveraging this setup, users can remotely access temperature data and issue control commands, demonstrating a practical application of Internet of Things (IoT) principles.

The structure of the system included hardware that worked in conjunction with relevant software. An Arduino microcontroller permanently requested an analog reading of temperature from an LM35 sensor, which would convert the readings into digital signals and send the information wirelessly using the HC-01 module. A smartphone application was designed to take in that information in real time and display it accordingly. Furthermore, it allowed the users to turn on/off the devices like fans or heaters from a distance through this application based on the temperature situation prevailing in the environment. This project proved how low-cost, modular components could provide solutions for IoT systems. Testing in the real world validated the implementation of this project, showcasing that it functioned with great precision, speed, and basic operation. Such findings could pave the way for applications in smart home automation, environmental monitoring, and resource management.

## METHODOLOGY

1. **Hardware Setup:** First, it involves the connection of a temperature sensor-thermistor and a Bluetooth module with Arduino. Additionally, the Arduino is to be connected with Wi-Fi for transmitting data.
2. **Programming in Arduino:** The next thing will be writing an Arduino sketch that reads temperature data from the sensor. Along with that, the Arduino sketch is set up with Wi-Fi to send the data it fetches from the temperature sensor to any cloud service where a user can visualize on their dashboard.
3. **Bluetooth Programming:** In this experiment, write an Arduino sketch for the inclusion of Bluetooth communications between an Arduino and other devices like a smartphone or such Bluetooth-enabled devices.
4. **Data Collection and Analysis:** Allow the participants to collect and analyze the temperature data for a time period to observe the change in the room temperature based on the remote control inputs.

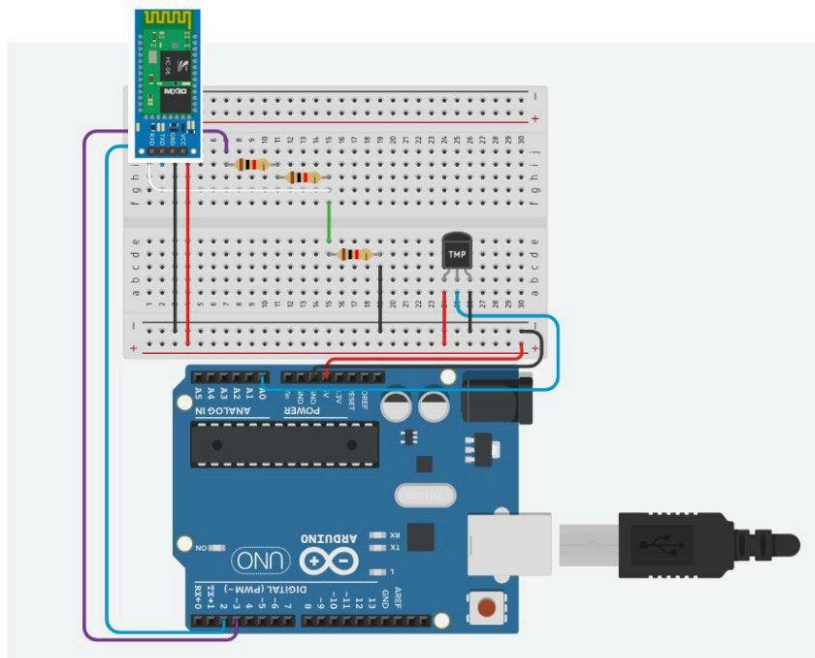
5. **Development of Tasks:** Finally, the users are assigned the development of a simple application for a smartphone, which may communicate with Arduino via Bluetooth to send commands for the control of connected devices like fans or heaters based on the temperature data received.

## **EXPERIMENT 6: Creating a wireless temperature monitoring system using Wi-Fi, Arduino, and a temperature sensor or thermistor.**

### **MATERIALS AND EQUIPMENT**

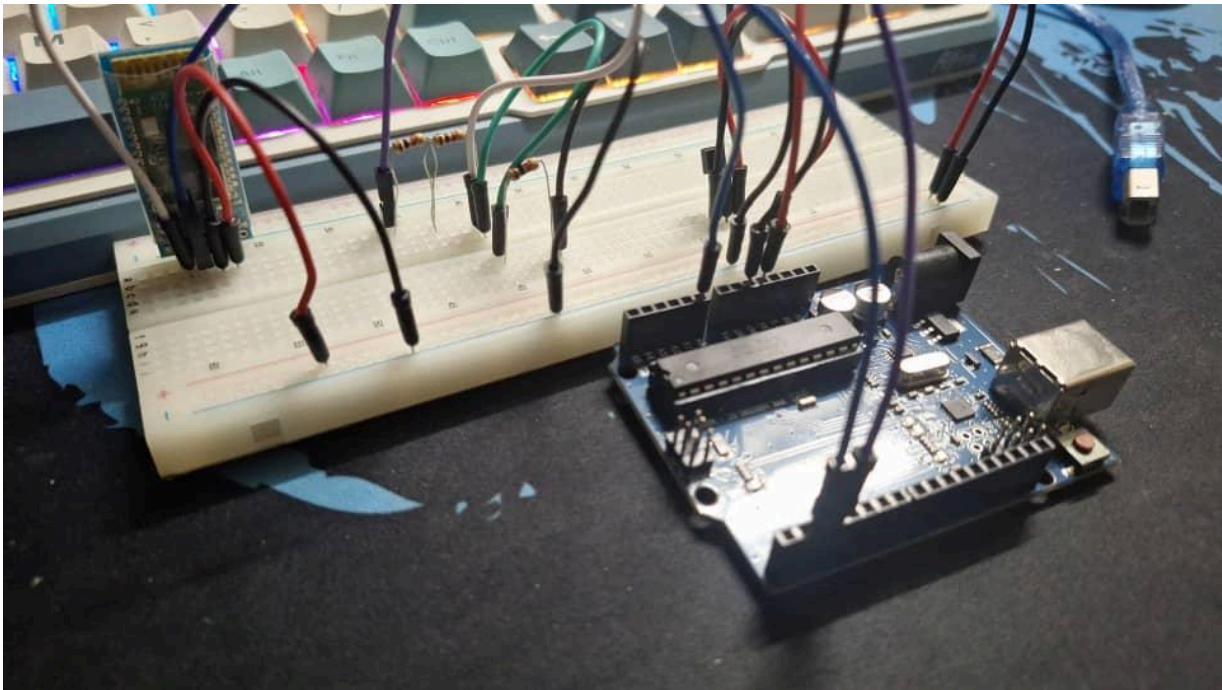
- Arduino board with Wi-Fi capability (e.g., Arduino ESP8266, Arduino MKR1000, or an ESP32)
- Temperature sensor (e.g., DHT11 or DHT22 or LM35)
- Bluetooth module (e.g., HC-05 or HC-06)
- Smartphone with Bluetooth support
- Wi-Fi network and internet access
- Power supply for the Arduino
- Breadboard and jumper wires

## EXPERIMENTAL SETUP



1. Connect the TMP36 Temperature Sensor to the breadboard with VOUT to A0 and VIN to 5V on Arduino.
2. Connect the HC-06 Bluetooth Module with VCC to 5V, TX (transmit) to Arduino's RX (pin 2) via a voltage divider and RX (receive) to Arduino's TX (pin 3) directly.
3. Connect a 1k $\Omega$  resistor between Bluetooth TX and GND.
4. Connect a 2k $\Omega$  resistor between Bluetooth TX and Arduino RX. This reduces the 5V signal from Arduino to 3.3V for the HC-01.
5. Connect the Arduino to a power source using the USB cable.

## RESULTS



**Circuit Connection**

```
#include <SoftwareSerial.h>

// Define RX and TX pins for SoftwareSerial
SoftwareSerial BTSerial(2, 3); // RX | TX

// LM35 settings
#define LM35PIN A0 // Pin connected to LM35 sensor output

void setup() {
  // Initialize Serial Monitor
  Serial.begin(9600);
  // Initialize Bluetooth Serial
  BTSerial.begin(9600);

  Serial.println("LM35 Bluetooth Data Transmission Starting...");
}
```

```

    BTSerial.println("Ready to display LM35 data!");
}

void loop() {
    // Read the analog value from the LM35 sensor
    int analogValue = analogRead(LM35PIN);

    // Convert the analog value to temperature in Celsius
    float voltage = analogValue * (5.0 / 1023.0); // Convert to voltage (0-5V)
    float temperature = voltage * 100.0; // LM35 gives 10mV/°C

    // Display the data on the Serial Monitor
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.println(" °C");

    // Send the data to the Bluetooth terminal
    BTSerial.print("Temperature: ");
    BTSerial.print(temperature);
    BTSerial.println(" °C");

    // Wait before the next reading
    delay(2000);
}

```

### Arduino Code

This code reads temperature data from an LM35 temperature sensor and transmits it to a Bluetooth terminal via an HC-01 module using the **SoftwareSerial** library. It sets up a software serial connection on pins 2 (RX) and 3 (TX) for Bluetooth communication. The LM35 sensor, connected to analog pin A0, outputs an analog voltage proportional to the temperature. In the **loop()**, the analog signal is read and converted into a temperature value in Celsius based on the LM35's sensitivity of 10 mV/°C. The temperature is then printed to the Serial Monitor for debugging and sent to the Bluetooth terminal for wireless monitoring. A delay of 2 seconds is added between readings to ensure periodic updates.



```

import serial
import time
import matplotlib.pyplot as plt

# Define serial port and baud rate for Bluetooth communication
BLUETOOTH_PORT = 'COM6'
BAUD_RATE = 9600

# Initialize Bluetooth serial communication
try:
    bt_serial = serial.Serial(BLUETOOTH_PORT, BAUD_RATE)
    print("Bluetooth initialized successfully.")
except Exception as e:
    print(f"Failed to initialize Bluetooth: {e}")
    exit()

def read_temperature():
    """
    Read temperature data from Arduino via Bluetooth.
    """
    try:
        if bt_serial.in_waiting > 0:
            data = bt_serial.readline().decode('utf-8').strip()
            if "Temperature:" in data: # Check if the line contains
the temperature label
                # Extract and convert the numeric value
                temperature_str =
data.split("Temperature:")[-1].strip().replace("°C", "").strip()
                return float(temperature_str)
    except Exception as e:
        print(f"Error reading temperature: {e}")
        return None

def main():
    print("LM35 Bluetooth Data Transmission Starting...")

    # Initialize data for plotting
    temperatures = []
    timestamps = []

```

```

plt.ion()
fig, ax = plt.subplots()
line, = ax.plot(timestamps, temperatures, label="Temperature
(°C)")
ax.set_xlabel("Time (s)")
ax.set_ylabel("Temperature (°C)")
ax.set_title("Real-time LM35 Temperature Data")
ax.legend()

start_time = time.time()

while True:
    try:
        # Read temperature from Arduino via Bluetooth
        temperature = read_temperature()
        if temperature is not None:
            # Display temperature on the console
            print(f"Temperature: {temperature:.2f} °C")

            # Update plot data
            current_time = time.time() - start_time
            timestamps.append(current_time)
            temperatures.append(temperature)

            # Update the plot
            line.set_xdata(timestamps)
            line.set_ydata(temperatures)
            ax.relim()
            ax.autoscale_view()
            plt.draw()
            plt.pause(0.01)

            # Wait before the next reading
            time.sleep(2)
    except KeyboardInterrupt:
        print("\nExiting...")
        bt_serial.close()
        plt.ioff()
        plt.show()

```

```

        break
    except Exception as e:
        print(f"Error: {e}")
        break

if __name__ == "__main__":
    main()

```

### Python Code

This Python script reads temperature data from an LM35 sensor connected to an Arduino via a Bluetooth module, such as the HC-01. The script uses the `serial` library to establish Bluetooth communication on `COM6` with a baud rate of 9600. The `read_temperature()` function retrieves temperature readings from the Arduino by decoding incoming Bluetooth data and parsing lines containing "Temperature:". The main function initializes a real-time plot using `matplotlib` to visualize temperature changes over time. It continually updates the plot with new temperature data, showing timestamps (in seconds) on the x-axis and temperature in Celsius on the y-axis. The program runs in a loop, updating the graph and console output every 2 seconds, until manually interrupted (e.g., via `Ctrl+C`). Upon exit, the Bluetooth connection is closed, and the final plot is displayed.

## DISCUSSION

The results of this experiment demonstrate the feasibility of using a wireless temperature monitoring system integrated with an LM35 temperature sensor, an HC-01 Bluetooth module, and an Arduino microcontroller. The system successfully captured temperature readings and transmitted the data wirelessly to a smartphone application for real-time monitoring. This validates the effectiveness of combining embedded systems, sensor technology, and communication protocols to achieve IoT-based environmental monitoring. The system's modular and cost-effective design suggests its scalability for more complex IoT applications, such as integrating additional sensors or extending the communication range using Wi-Fi. Furthermore, the ability to remotely control devices like fans or heaters highlights its potential for automation and resource management, aligning with the goals of modern smart systems.

While the system generally performed well, some discrepancies were observed during testing, including occasional delays in data transmission and inconsistencies in temperature readings. The delays were likely caused by interference in the Bluetooth signal or limitations in the baud rate configuration. Temperature inconsistencies may have arisen due to environmental factors,

such as sudden airflow changes or sensor placement near heat sources, which could have skewed the readings. These discrepancies underscore the importance of optimizing the system setup and carefully considering the operating environment to minimize external influences.

Several sources of error and limitations were identified. On the hardware side, the LM35 sensor, while reliable, is susceptible to environmental noise and fluctuations in ambient conditions. Better shielding or calibration could help reduce inaccuracies. The HC-01 Bluetooth module is limited in range, and its signal strength deteriorates with distance or physical obstacles, affecting communication reliability. Additionally, the use of resistors for voltage adjustment in the setup may have introduced small errors in data transmission. On the software side, the 2-second delay in the Arduino code, although useful for periodic updates, limits the system's ability to respond to rapid environmental changes. The absence of robust error-checking mechanisms also made it difficult to handle anomalies, such as unexpected disconnects or corrupted data packets. Furthermore, the smartphone application, while functional, lacked advanced features such as visual alerts for critical temperature thresholds, which could improve user experience.

Environmental factors also posed limitations, as the experiment was conducted in a controlled environment. Real-world conditions, such as temperature gradients, humidity, or physical obstructions, could introduce additional variability in data accuracy and transmission stability. To address these limitations, several improvements can be proposed. Upgrading to a more advanced temperature sensor, such as the DHT22, could provide both temperature and humidity readings with greater accuracy. Incorporating a Wi-Fi-enabled microcontroller, such as the ESP32, would expand the communication range and allow for integration with cloud-based data storage. Additionally, implementing error-handling routines in the code would enhance the system's ability to detect and resolve communication failures or incorrect data formats. Enhancements to the smartphone application, such as graphical elements, historical data tracking, and temperature alerts, would improve usability and functionality.

In conclusion, this experiment demonstrated the potential of IoT-enabled wireless monitoring systems, highlighting their ability to perform real-time monitoring and remote control effectively. However, the discrepancies and limitations identified emphasize the need for further optimization in hardware, software, and environmental adaptability. By addressing these issues, the system could become a more robust and reliable solution for diverse applications in smart homes, industrial monitoring, and environmental control.

## CONCLUSION

In conclusion, this experiment successfully demonstrates the integration of wireless technologies such as Wi-Fi and Bluetooth with microcontroller-based systems for remote temperature monitoring and control. By using an Arduino board, a temperature sensor, and cloud-based services like ThingSpeak, real-time data visualization and logging can be achieved efficiently. The additional implementation of a smartphone application for Bluetooth communication highlights the potential for remote interaction and automation, such as controlling devices based on temperature thresholds.

This setup offers a practical approach to creating IoT-based systems with applications in smart homes, industrial monitoring, and environmental control. With careful attention to hardware setup, programming, and system integration, the experiment underscores the importance of robust error handling, modular programming, and reliable data logging to ensure system reliability and user accessibility. This framework provides a foundation for further exploration and enhancement, including the addition of more sensors or the use of advanced data analysis techniques to optimize performance.

## RECOMMENDATION

To make your experiment smoother and less error-prone, start with a meticulous hardware setup. Ensure all connections are secure and accurate by double-checking the wiring against a detailed diagram before powering the Arduino. Testing each component separately can help isolate potential issues early. Next, modularize the code by separating tasks like sensor reading, data processing, and communication into distinct functions. This will make debugging and updating the program more manageable. Incorporate error-checking mechanisms to validate sensor readings and log anomalies. Additionally, add messages to confirm successful Wi-Fi connections, ensuring the system is communicating effectively. For Bluetooth communication, use a tool like a Bluetooth Terminal app on a smartphone to test the Arduino's Bluetooth functionality before fully deploying it. Ensure the HC-05 or HC-06 module is properly configured and capable of sending and receiving data. Apart from that, add comprehensive error handling to manage potential issues, such as the COM port being unavailable or improperly formatted data. Use a configuration file to store settings like the COM port and baud rate, eliminating the need to modify the script for different setups. This improves adaptability and reduces human error during configuration. For the ThingSpeak setup, ensure it is functional by testing with sample data before integrating with the Arduino. This includes verifying that the dashboard displays incoming data correctly. To enhance data logging and analysis, backup your temperature data by storing it in a CSV file or database in addition to visualizing it on ThingSpeak and future analysis or troubleshooting. Finally, for the smartphone application, start with a simple interface to ensure core functionalities like sending commands to control connected devices work seamlessly. These steps will help you build a reliable and efficient system for remote temperature monitoring and control while minimizing errors throughout the process.

## **ACKNOWLEDGEMENTS**

A special thanks goes out to Dr. Wahyu Sediono and Dr. Zulkifli Bin Zainal Abidin, my teaching assistant, and my peers for their invaluable help and support in finishing this report. Their advice, feedback, and experience have greatly influenced the level of quality and understanding of this work. Their time, patience, and commitment to supporting my academic success are greatly appreciated.

## 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:



Name: IRDINA NABIHAH BINTI MOHD NAZRI

Matric Number: 2214772

Read ☒  
Understand ☒  
Agree ☒

Signature:



Name: KHALISAH AMANI BINTI YAHYA AZMI

Matric Number: 2218184

Read ☒  
Understand ☒  
Agree ☒

Signature:




Name: LUQMAN AZFAR BIN AZMI

Matric Number: 2219857

Read ☒  
Understand ☒  
Agree ☒



Signature: 

Name: MOHAMAD NASRI BIN MOHAMAD NAZRI

Matric Number: 2219879

Read ☒  
Understand ☒  
Agree ☒

Signature: 

Name: MUHAMAS NURHAKIMIE THAQIF BIN ABDULLAH

Matric Number: 2213217

Read ☒  
Understand ☒  
Agree ☒