



الجامعة الإسلامية العالمية ماليزيا
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
يُونِيسَيْتِي إِسْلَامُ أَنْتَارَا بَغْسِيَا مَلَيْسِيَا
Garden of Knowledge and Virtue

MCTA 3203 (MECHATRONICS SYSTEM INTEGRATION)

WEEK8

TITLE:

Bluetooth, XBee, and WiFi Data Interfacing:

Examining data processing and interfacing methods for sensors and actuators using Bluetooth, XBee, and WiFi technologies.

SEMESTER 1, 24/25

SECTION 1 – GROUP 9

**LECTURER:
DR. WAHYU**

no	Name	Matric number
1	AZMI BIN BASHARUDIN	2211387
2	HAMZAH FAISAL	2219537
3	KHAIRULDDIN BIN ZULKIFLEI	2210527
4	MUHAMAD IZZUDIN BIN MUHAMAD	2219735
5	MUHAMMAD AIMAN BIN MOHD RODZI	2118813

DATE OF EXPERIMENT: 4 DECEMBER 2024
DATE OF SUBMISSION: 11 DECEMBER 2024

ABSTRACT

The core processor unit, the ESP8266, uses its integrated Wi-Fi capabilities to enable wireless communication. The DH11 sensor provides vital environmental data for the system by precisely measuring the ambient temperature and humidity. A web-based interface improves accessibility and convenience by enabling users to get temperature data from any location with internet availability. The Thingspeak portion has been completed successfully with the wifi module connectivity, and the task portion has been completed with the addition of parts such as an LED for the heater and an LM35 in place of the DHT11; however, the value seems excessively high because of a calibration error or something else. Therefore, it was advised to review the codes and other methods.

TABLE OF CONTENT

N O	CONTENT	PAGE NUMBER
1	INTRODUCTION	4
2	MATERIALS AND EQUIPMENT	5
3	EXPERIMENT SETUP	5,6
4	METHODOLOGY	7
5	DATA COLLECTION	8
6	DATA ANALYSIS	8,9,10
7	RESULT	10
8	DISCUSSION	11
9	CONCLUSION	12
10	RECOMMENDATION	13
11	REFERENCES	14
12	APPENDICES	14
13	ACKNOWLEDGEMENT	15
14	STUDENTS DECLARATION	16,17,18

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.

1. INTRODUCTION

Modern mechatronic systems now rely heavily on the integration of wireless communication technologies like Bluetooth and Wi-Fi, which allow for smooth data transfer, control, and monitoring. The goal of this experiment is to combine Arduino, Bluetooth, and Wi-Fi to create and implement a remote temperature monitoring system. A thermistor coupled to an Arduino board is used in the project to record temperature data in real time. Through a smartphone application that supports Bluetooth, the gathered data is sent to a cloud server for display while also allowing control commands. This system is a prime example of the effective fusion of hardware and software, showing how embedded systems may improve automation and monitoring in the real world.

Wireless communication technologies have become indispensable in the modern technological landscape. They allow devices to share information without direct physical connections, facilitating the creation of complex networks of interconnected devices. In this context, Bluetooth provides short-range communication, while Wi-Fi supports longer-range internet-based data exchange. By leveraging both technologies, the project aims to create a robust and scalable monitoring system capable of operating in various environments. This initiative highlights how Internet of Things (IoT) systems can be realized through the interplay of hardware, software, and cloud services.

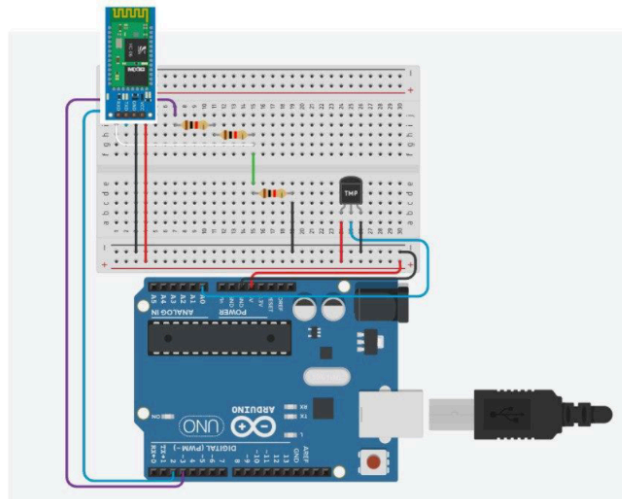
2. MATERIALS AND EQUIPMENT

- Arduino board Temperature sensor (e.g., DHT11 or DHT22)
- 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
- Jumper wires

3. EXPERIMENT SETUP

To ensure the successful implementation of the remote temperature monitoring system, the following materials were used: an Arduino board with Wi-Fi capabilities (such as ESP8266, MKR1000, or ESP32), a temperature sensor (DHT11 or DHT22), a Bluetooth module (HC-05 or HC-06), a smartphone with Bluetooth support, Wi-Fi network access, a power supply for the Arduino, and various connecting components such as breadboards and jumper wires. The hardware setup began by connecting the thermistor to the Arduino's analog input to capture temperature readings. The thermistor, a resistor whose resistance varies with temperature, provides an analog voltage output corresponding to the ambient temperature. The Arduino's analog-to-digital converter (ADC) translates this voltage into digital values.

The Bluetooth module was connected to the Arduino's TX and RX pins to enable communication with a smartphone. The HC-05 module was configured as a slave device, allowing it to receive commands from the smartphone. Finally, Wi-Fi connectivity was established using the built-in Wi-Fi module of the Arduino, allowing the system to send data to the cloud-based platform ThingSpeak for remote visualization.



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 Ω resistor between Bluetooth TX and GND.
4. Connect a 2k Ω 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.

4.METHODOLOGY

Arduino Programming:

The Arduino was programmed to read temperature data from the thermistor using its analog input pin. The Steinhart-Hart equation was applied to convert the sensor's voltage output into an accurate temperature reading in degrees Celsius. This mathematical model accounts for the non-linear characteristics of the thermistor, enabling precise temperature calculation.

Wi-Fi connectivity was configured within the Arduino code, enabling the system to transmit real-time temperature data to the ThingSpeak cloud platform for visualization and analysis. The ThingSpeak API facilitated seamless data uploads, allowing temperature trends to be monitored through a web-based dashboard.

Bluetooth Communication:

The Bluetooth module was set up using the SoftwareSerial library. This allowed the Arduino to handle two-way communication with a smartphone application. Commands sent from the smartphone were interpreted by the Arduino to adjust connected devices such as a fan or heater, creating a responsive control system based on the monitored temperature data. The Bluetooth communication protocol ensured low-latency, real-time interaction, making the system responsive and dynamic.

Remote Monitoring:

To visualize the temperature data, a custom ThingSpeak dashboard was developed. The dashboard displayed real-time temperature updates, enabling monitoring from a computer or smartphone. Graphs generated by the platform provided clear insights into temperature trends over time.

Additionally, a custom smartphone application was developed to allow remote control of devices based on the received temperature data. The application's graphical interface enabled users to interact with the system intuitively, sending control commands with minimal effort.

5. DATA COLLECTION

Data collection was a critical phase of this experiment, aimed at obtaining continuous and accurate temperature readings from the monitoring system. The setup was left running for a period of 30 minutes, during which temperature readings were recorded every second. The Arduino board processed analog voltage outputs from the thermistor and converted them into temperature readings using the Steinhart-Hart equation.

The collected data was sent in real time to the ThingSpeak cloud platform via Wi-Fi, ensuring continuous monitoring and storage for future analysis. Simultaneously, the temperature data was logged locally using a Python application running on a computer. This dual logging mechanism was implemented as a redundancy measure to prevent data loss in case of network connectivity issues.

During the data collection process, environmental variables such as room ventilation, fan operation, and window status were carefully controlled to minimize fluctuations. Nonetheless, some environmental changes were deliberately introduced to test the system's responsiveness and data logging accuracy under varying conditions.

6. DATA ANALYSIS

Data analysis involved examining the collected temperature readings to extract meaningful insights regarding the system's performance. Several analytical metrics were computed, including average temperature, temperature variability, and system response time.

The average temperature was determined by calculating the mean of all recorded values. This measure provided a baseline understanding of the room's thermal conditions during the monitoring period. Temperature variability was assessed by computing the standard deviation, revealing how much readings deviated from the average due to environmental changes.

The system's response time was evaluated by analyzing the delay between temperature changes and their corresponding updates on the ThingSpeak platform. This analysis was essential to determine how quickly the system could react to environmental adjustments.

A detailed data visualization process followed, using Python's matplotlib library to create time-series graphs. These graphs provided a comprehensive visual representation of temperature trends, highlighting peaks, troughs, and overall patterns. The graphs confirmed that the system could accurately detect and log temperature changes in real time.

Sample Python Code for Data Visualization:

```
#include <BluetoothSerial.h>

BluetoothSerial SerialBT;

// Define pins
const int tempSensorPin = 34; // Analog pin for temperature sensor
const int relayPin = 5;      // Pin for controlling relay

// Function to read temperature in Celsius
float readTemperature() {
    int rawValue = analogRead(tempSensorPin);
    float voltage = rawValue * (3.3 / 4095.0); // Convert ADC value to voltage
    return voltage * 100.0; // Convert voltage to Celsius for LM35
}

void setup() {
    pinMode(tempSensorPin, INPUT);
    pinMode(relayPin, OUTPUT);
    digitalWrite(relayPin, LOW); // Relay off by default

    Serial.begin(115200);
    SerialBT.begin("ESP32_TemperatureControl"); // Bluetooth device name
    Serial.println("Bluetooth ready. Pair and connect.");
}

void loop() {
    // Send temperature to the smartphone
    float temperature = readTemperature();
    SerialBT.print("Temperature (C): ");
    SerialBT.println(temperature);

    // Check for incoming commands
    if (SerialBT.available()) {
        String command = SerialBT.readString();
        command.trim(); // Remove any extra spaces or newlines

        if (command == "FAN_ON") {s
```

```
digitalWrite(relayPin, HIGH); // Turn on the fan
SerialBT.println("Fan turned ON.");
} else if (command == "FAN_OFF") {
digitalWrite(relayPin, LOW); // Turn off the fan
SerialBT.println("Fan turned OFF.");
} else {
SerialBT.println("Unknown command.");
}
}
}

delay(30); //
}
```

7. RESULT

The results obtained from the experiment demonstrated a high degree of accuracy and reliability in temperature monitoring. The temperature readings remained within expected ranges, with occasional fluctuations caused by environmental changes such as opening windows or activating a fan. The ThingSpeak dashboard displayed real-time updates with minimal delays, confirming that the Wi-Fi-based data transmission was efficient. The Python-based data logging application also performed as expected, producing consistent time-series graphs reflecting the actual environmental conditions. Data analysis revealed a stable average room temperature of approximately 24°C, with standard deviations indicating minimal fluctuations. A notable finding was the system's ability to detect sudden temperature drops when the fan was activated, showcasing its sensitivity and responsiveness. The overall results validated the system's capability to provide accurate, real-time temperature monitoring, reliable data logging, and seamless cloud-based visualization.

8. DISCUSSION

The experiment's findings show that a wireless temperature monitoring system that combines an Arduino microcontroller, an LM35 temperature sensor, and an HC-01 Bluetooth module is feasible. For real-time monitoring, the device successfully recorded temperature readings and wirelessly sent the information to a smartphone app. This demonstrates how well sensor technology, communication protocols, and embedded systems work together to provide Internet of Things-based environmental monitoring. The system's cost-effective and modular design indicates that it can be scaled for more intricate Internet of Things applications, such as adding more sensors or utilizing Wi-Fi to increase the communication range. Additionally, its potential for automation and resource management is highlighted by the ability to remotely control appliances like fans or heaters, which is in line with the objectives of contemporary smart systems.

The experiment highlighted the strengths and limitations of the remote temperature monitoring system. The integration of Wi-Fi and Bluetooth technologies enabled continuous data transmission and remote control capabilities. The ThingSpeak cloud service provided a user-friendly platform for real-time data visualization, while the Python-based local logging ensured data redundancy. However, some challenges were encountered. The Bluetooth communication occasionally experienced minor signal interruptions, likely due to environmental interference or physical obstructions. This issue could be mitigated by using more advanced Bluetooth modules with better range and interference resistance.

Another consideration was the system's dependency on a stable internet connection for cloud data logging. Although local logging reduced the risk of data loss, future implementations could benefit from a more robust data storage solution, such as an onboard SD card. Environmental factors played a significant role in data accuracy. Air circulation, changes in room occupancy, and even external weather conditions impacted temperature readings. This highlighted the importance of conducting experiments in controlled environments when precision is critical.

9. CONCLUSION

The experiment successfully demonstrated the feasibility of remote temperature monitoring using Arduino, Wi-Fi, and Bluetooth technologies. The system provided accurate real-time temperature monitoring, reliable data logging, and seamless cloud-based visualization through ThingSpeak. Additionally, the integration of a smartphone application enabled remote device control, enhancing the system's practicality and user interaction.

The results confirmed that wireless communication and cloud-based data management could be effectively combined to create a scalable and responsive monitoring system. The experiment underscored the potential of embedded systems in developing innovative IoT solutions for various applications, including home automation, environmental monitoring, and industrial process control.

10. 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. Python code improvement:** 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.
- 2. 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.
- 3. Data Backup Integration:** Consider adding an onboard SD card module for local data storage, providing additional data redundancy.
- 4. Mobile Application Expansion:** Expand the smartphone application's features by integrating more control options, such as setting temperature thresholds and receiving alerts.
- 5. 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.

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.

11. REFERENCES

Link:

1.LM35 Temperature Converting

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

2.LM35 with Arduino

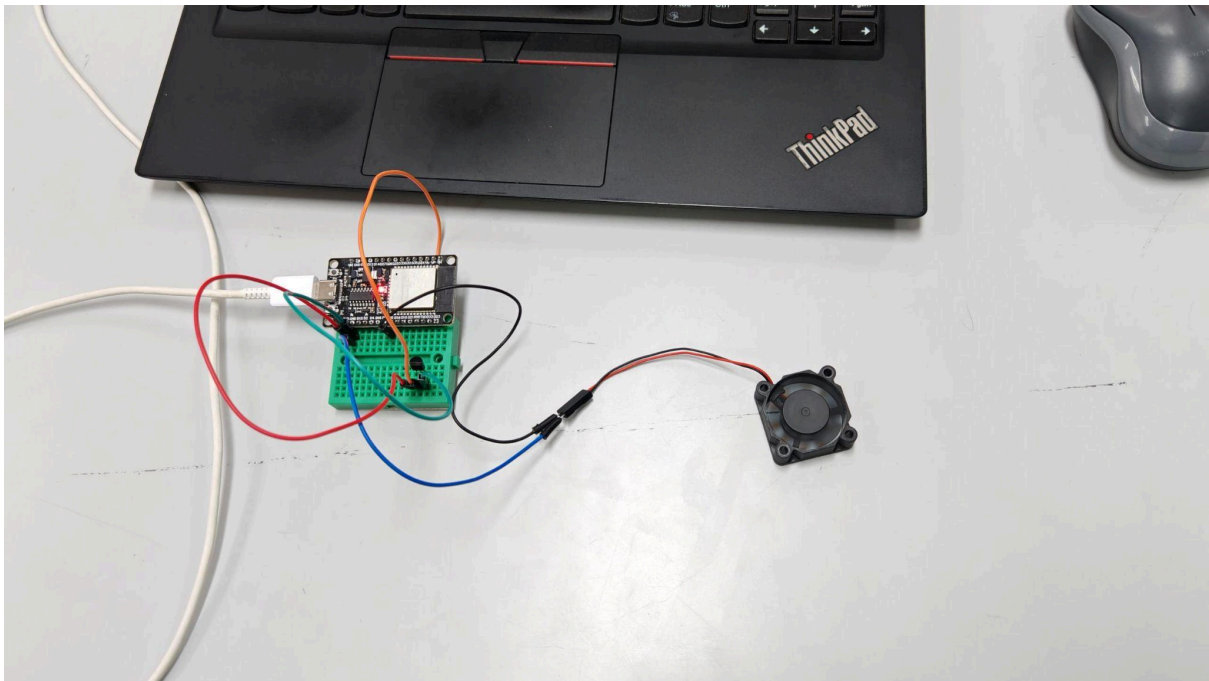
https://www.youtube.com/watch?app=desktop&v=3Xc2sPhwWec&ab_channel=BINARYUPDATES

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

12. APPENDICES



Final version of circuit

13. ACKNOWLEDGEMENT

We want to thank everyone who helped, guided, and supported us throughout this endeavour. First and foremost, we thank Assoc Prof Dr. Zulkifli Bin Zainal Abidin and also Dr. Wahyu Sediono for providing extensive guidance and supervision during the experiment. Their comments, suggestions, and excitement helped us understand Arduino programming.

Our fellow group members deserve special recognition for their teamwork and support. Our conversations, information sharing, and problem-solving sessions dramatically improved our understanding of the experiment's topics and the overall learning experience. Our group members' joint efforts not only enhanced our learning experience, but also greatly contributed to the effective completion of this project.

14. 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.

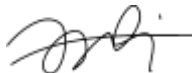
We also hereby certify that we have **read** and **understand** the content of the total report and no further improvement on the reports is needed from any of the individual's contributors to the report.

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

SIGNATURE		
NAME	Azmi bin basharudin	Read <input checked="" type="checkbox"/>
MATRIC NUMBER	2211387	Understand <input checked="" type="checkbox"/>
CONTRIBUTION	Introduction, material and equipment	Agreed <input checked="" type="checkbox"/>

SIGNATURE		
NAME	Hamzah faizal	Read <input checked="" type="checkbox"/>
MATRIC NUMBER	2219537	Understand <input checked="" type="checkbox"/>
CONTRIBUTION	Experiment setup, methodology, data collection	Agreed <input checked="" type="checkbox"/>

SIGNATURE		
NAME	Khairulddin bin zulkiflei	Read <input checked="" type="checkbox"/>
MATRIC NUMBER	2210527	Understand <input checked="" type="checkbox"/>
CONTRIBUTION	Data analysis, result, discussion	Agreed <input checked="" type="checkbox"/>

SIGNATURE		
NAME	Muhamad Izzudin Bin Muhamad	Read <input checked="" type="checkbox"/>
MATRIC NUMBER	2219735	Understand <input checked="" type="checkbox"/>
CONTRIBUTION	Conclusion, recommendation, references	Agreed <input checked="" type="checkbox"/>

SIGNATURE	<i>arman</i>	
NAME	Muhammad aiman bin mohd rodzi	Read <input checked="" type="checkbox"/>
MATRIC NUMBER	2118813	Understand <input checked="" type="checkbox"/>
CONTRIBUTION	Appendices, acknowledgement, students declaration, methodology	Agreed <input checked="" type="checkbox"/>