MECHATRONICS SYSTEM INTEGRATION (MCTA 3203)

SECTION 2, SEMESTER 1, 2024/2025

ACTIVITY REPORT

Week 8:

Bluetooth and Wifi data interfacing with microcontroller and computer based system:

Data processing, sensors and actuators.

No	Name (Group 6)	Matric No.
1	MUHAMMAD BASIL BIN ABDUL HAKIM	2215315
2	MUHAMMAD SYAFIQ BIN NOR AZMAN	2213187
3	MUHAMMAD HAFIZ HAMZAH BIN FANSURI	2212803
4	MUHAMMAD AMMAR ZUHAIR BIN NOR AZMAN SHAH	2110259
5	MUHAMMAD RAZIQ BIN KAHARUDDIN	2120225

Table of content

Abstract

In this experiment, we interface data to a microcontroller and computer-based system via Bluetooth and Wi-Fi. The objective is to create a remote temperature monitoring and control system using a microcontroller ESP32 (this microcontroller has embedded Bluetooth and Wi-Fi modules), and a temperature sensor LM35. The outputs or temperature readings will be displayed on a smartphone with Bluetooth and Wi-fi capabilities on an app. Using this method, we can monitor our data remotely as long as the system is connected to the User Interface application.

In addition, the data we receive then can be stored in a cloud system for future reference or to train our system to increase its capabilities, but in this experiment we are not focusing on the data storage on any cloud rather we control the feedback from the temperature output signal to switch ON or OFF the new inputs such as a cooling fan or a heater (2 LEDs are being used as substitute).

Material and equipment

- 1, ESP32

- 1, LM 35

- 2, LEDs

- 1, Breadboard

- Jumper wires

- Smartphone with Android OS

- Software for interface (named Serial Bluetooth)

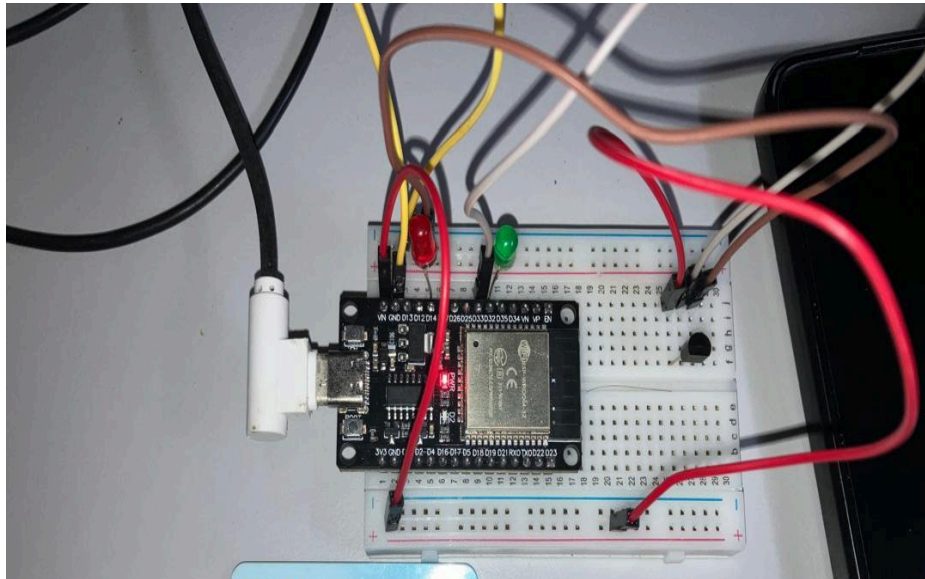- Arduino IDE

Experimental Setup and Methodology

Figure 1: Experimental setup

1. The circuit is set up as shown in the figure above, with the ESP32 connected to the LM35 sensor and two LEDs.

2. The ESP32 is powered by a USB connection to a computer.

3. The program is written in the Arduino IDE and uploaded to the ESP32 for temperature monitoring, Bluetooth communication, and LED control.

4. The Serial Bluetooth app is used to pair with the ESP32 for receiving temperature data and sending commands to control the LEDs.

Results

The experiment demonstrated successful temperature monitoring and control using the ESP32 and LM35 sensor. When the temperature exceeded 21°C, the LED connected to pin 14 turned on, and it turned off when the temperature was below this threshold. The measured temperature was displayed on both the serial monitor and the Bluetooth Serial Terminal app, confirming accurate functionality.

The Bluetooth module allowed remote control of the LED connected to pin 32(green LED) . Command (1) turned the LED on, while (2) turned it off, verifying reliable bidirectional communication. The system effectively combined temperature-based automation and Bluetooth-based remote control, showcasing its potential for smart device applications.
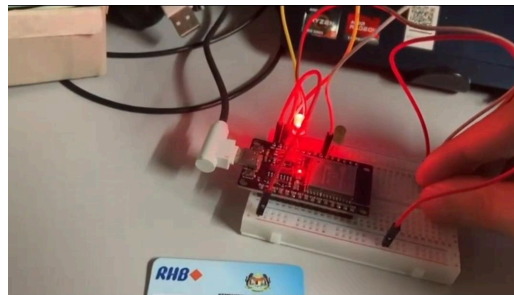

Figure 2: The red LED turns on when temperature exceeding 21°C

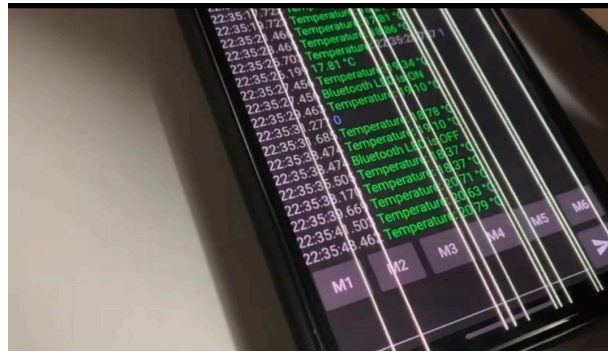Figure 3: The green LED turns on when it receive input (1) from app

Figure 4: The serial monitor readings from LM 35

Discussions

Hardware Discussion

Using the ESP32 for this experiment makes it easier and more efficient compared to using the Arduino UNO.

The ESP32 has built-in Bluetooth, so there's no need for extra modules, making the setup more compact and cheaper. It also reads temperature data more accurately thanks to its higher resolution. The faster processor allows it to handle tasks like Bluetooth communication and sensor readings smoothly at the same time. Additionally, the ESP32 has Wi-Fi, which could be useful for future projects or improvisation, and uses less power, making it better for battery-powered setups.

Overall, the ESP32 is a better choice because it's more powerful, flexible, and convenient to use.

Component	ESP32 Pin	Details
LM35 Sensor	GPIO33	Reads analog voltage from the sensor.
Temperature LED	GPIO14	Turns ON/OFF based on temperature.
Bluetooth LED	GPIO32	Controlled by Bluetooth commands.
LM35 Power	3.3V, GND	Provides power to the LM35 sensor.
LED Power	GPIO14/32 + Resistor	Current-limiting resistors to LEDs.

*GPIO stands for General Purpose Input/Output. It is a standard term used in microcontrollers and development boards like the ESP32.

Software Discussion

ARDUINO CODE

```cpp
#include <BluetoothSerial.h>

// LM35 sensor
int LM35Pin = 33;  // Analog pin where LM35 is connected
float Temperature;  // Store the temperature value in Celsius

// LED pins
int ledPinTemp = 14;  // LED for temperature control
int ledPinBluetooth = 32;  // LED controlled by Bluetooth

// BluetoothSerial object
BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  delay(100);

  pinMode(LM35Pin, INPUT);
  pinMode(ledPinTemp, OUTPUT);
```

```
  pinMode(ledPinBluetooth, OUTPUT);

  // Initialize Bluetooth
  SerialBT.begin("ESP32_Bluetooth");  // Name of the Bluetooth device
  Serial.println("Bluetooth device is ready to pair");
  SerialBT.println("Bluetooth device is ready to pair");

  // Ensure both LEDs are off initially
  digitalWrite(ledPinTemp, LOW);
  digitalWrite(ledPinBluetooth, LOW);
}

void loop() {
  // Read the analog value from the LM35 sensor
  int analogValue = analogRead(LM35Pin);

  // Convert the analog value to temperature in Celsius
  Temperature = (analogValue * 3.3 * 100) / 4095;  // Temperature in
Celsius

  // Control LED based on temperature
  if (Temperature >= 21.0) {
    digitalWrite(ledPinTemp, HIGH);  // Turn on LED when temperature
reaches 21°C
  } else {
    digitalWrite(ledPinTemp, LOW);  // Turn off LED if temperature is
less than 21°C
  }

  // Send temperature to Serial Bluetooth Terminal app
  SerialBT.print("Temperature: ");
  SerialBT.print(Temperature);
  SerialBT.println(" °C");

  // Check if data is available from Bluetooth app
  if (SerialBT.available()) {
    char receivedChar = SerialBT.read();
    if (receivedChar == '1') {
      digitalWrite(ledPinBluetooth, HIGH);  // Turn on LED when
receiving '1' via Bluetooth
      SerialBT.println("Bluetooth LED is ON");
    } else if (receivedChar == '0') {
```

```
      digitalWrite(ledPinBluetooth, LOW);   // Turn off LED when
receiving '0' via Bluetooth
      SerialBT.println("Bluetooth LED is OFF");
    }
  }

  // Print temperature to the standard serial monitor for debugging
  Serial.print("Temperature: ");
  Serial.print(Temperature);
  Serial.println(" °C");

  delay(2000);   // Wait for 2 seconds before the next reading
}
```

This Arduino code implements a system where an ESP32 microcontroller communicates with a smartphone application (named Serial Bluetooth) via Bluetooth. The setup includes an LM35 temperature sensor and two LEDs: one for temperature control and one for Bluetooth interaction.

The code start with an initialization where:

- The BluetoothSerial library is included to enable Bluetooth communication.

- The LM35 sensor pin "LM35Pin", temperature LED pin "ledPinTemp", and Bluetooth LED pin "ledPinBluetooth" are defined and initialized.

- The "SerialBT" object sets up the ESP32 as a Bluetooth device named "ESP32_Bluetooth". This allows pairing with a smartphone.

As for the temperature Reading and LED Control:

- The LM35 temperature sensor outputs an analog signal proportional to the temperature. The signal is read using analogRead(), and the voltage value is converted to Celsius using the formula:

Temperature = (Analog Value x 3.3 x 100) / 4095

where 3.3 is the ESP32's ADC reference voltage, and 4095 is the resolution of the ADC (12-bit).

● The LED connected to ledPinTemp is used as a temperature indicator:

    ○ If the temperature is **21°C or higher**, the LED is turned ON.

    ○ Otherwise, the LED remains OFF.

For the Bluetooth Communication:

● The ESP32 sends the current temperature value to the smartphone app using SerialBT.print(). This allows the user to monitor the temperature on the app in real time.

● The ESP32 also listens for commands from the smartphone via Bluetooth:

-      Receiving **'1'** turns ON the second LED (ledPinBluetooth).

   -   Receiving **'0'** turns OFF the second LED.

● Status messages "Bluetooth LED is ON/OFF" are sent back to the app as confirmation.

Debugging:

● For debugging purposes, the temperature is also printed to the serial monitor using Serial.print(). This helps verify the sensor readings during development.

Delay:

- A delay of 2 seconds (delay(2000)) is added at the end of the loop to control the frequency of sensor readings and communication.

This code establishes a functional and interactive system where:

- Temperature data is read from an LM35 sensor and used to control an LED.
- The ESP32 communicates the temperature to a smartphone app via Bluetooth.
- Commands from the app control a second LED, demonstrating bidirectional communication.

The system is simple and efficient, providing both real-time monitoring and manual control via Bluetooth.

Recommendation

Recommendation

To improve the system's functionality and user experience, it is recommended to develop a dedicated smartphone application for real-time monitoring and control. This application should include features such as temperature graphs and notifications, enabling users to make informed decisions based on live data. Additionally, expanding the system to include additional sensors, such as humidity or motion detectors, could provide a more

comprehensive environmental monitoring solution, improving its adaptability to various conditions.

Furthermore, Upgrading the communication protocol from basic Bluetooth to Wi-Fi or Bluetooth Low Energy (BLE) would allow for a broader range and reduced power consumption, making the system more suitable for smart home integration. Furthermore, rigorous testing in diverse environments is essential to ensure the system's reliability and accuracy under varying conditions.

Finally, scalability can be achieved by incorporating advanced features such as voice control or compatibility with IoT platforms like Google Assistant or Amazon Alexa. These improvements would not only enhance user interaction but also transform the system into a more robust, versatile, and future-proof solution for modern applications.

Conclusion

In conclusion, we have interface data to a microcontroller and computer-based system via Bluetooth and Wi-Fi. The outcome of the experiment is reliable as our system manages to display temperature outputs wirelessly on our phones. Some errors may be caused by old pieces of equipment or weak connections between the microcontroller and our device. This however can be fixed by using better equipment. The LM35 can also be replaced by other temperature sensors such as thermistors to get a more accurate and reliable reading.

Student Declaration

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 taken or done by unspecified sources or person.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 have been verified by us.

| NAME: Muhammad Basil bin Abdul Hakim | READ | ✔ |
| MATRIC NO: 2215315 | UNDERSTAND | ✔ |

	AGREE	✔
SIGNATURE:		

NAME: Muhammad Syafiq Bin Nor Azman	READ	✔
MATRIC NO: 2213187	UNDERSTAND	✔
SIGNATURE	AGREE	✔

NAME:Muhammad Hafiz Hamzah Bin Fansuri	READ	✔
MATRIC NO:2212803	UNDERSTAND	✔
SIGNATURE :	AGREE	✔

NAME: Muhammad Ammar Zuhair Bon Nor Azman Shah	READ	✔
MATRIC NO: 2110259	UNDERSTAND	✔
SIGNATURE:	AGREE	✔

NAME: MUHAMMAD RAZIQ BIN KAHARUDDIN	READ	✔
MATRIC NO: 2120225	UNDERSTAND	✔
SIGNATURE	AGREE	✔