# EXPERIMENT NO - 9

# Aim: Interfacing LM35 temperature sensor with ESP32 Web Server to display sensor readings.

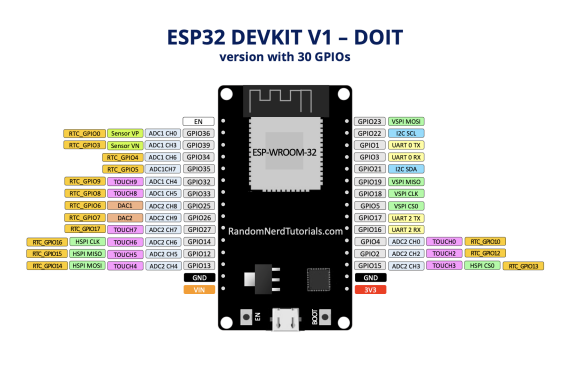
# Apparatus:

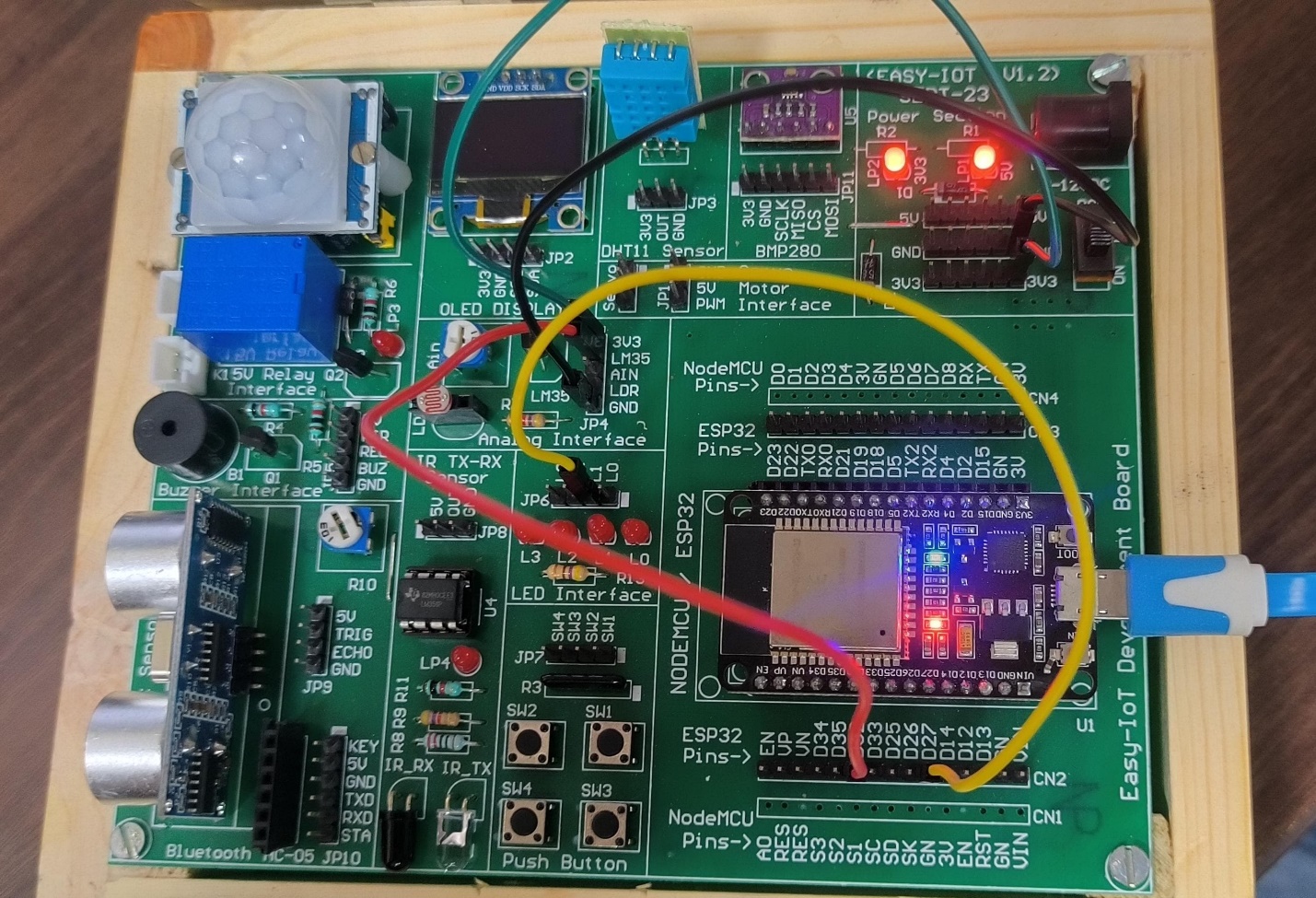
- ESP32 Development Board  
- Jumper Wires  
- Breadboard  
- LM35 temperature sensor  
- USB Cable (for connecting ESP32 to laptop/PC)

**Theory -**In this experiment, we are interfacing an LM35 temperature sensor with an ESP32 microcontroller to read temperature data and display it on a web server. The LM35 is a commonly used temperature sensor that provides an analog output proportional to temperature in Celsius. Its accuracy, linearity, and ease of use make it ideal for various applications in IoT, weather monitoring, and home automation.

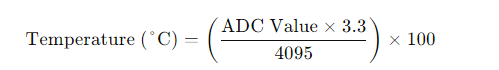
The LM35 is a precision integrated-circuit temperature sensor with an output voltage that is linearly proportional to the Celsius temperature.It operates within a temperature range of -55°C to 150°C and has an accuracy of ±0.5°C at room temperature.The output of the LM35 increases by 10 mV per degree Celsius, which means 250 mV corresponds to 25°C.Since the LM35 does not require external calibration, it is highly suitable for embedded systems and remote data monitoring.  
The ESP32 is a low-cost microcontroller with integrated Wi-Fi and Bluetooth, which makes it an ideal choice for IoT applications.It has multiple analog-to-digital converter (ADC) pins, allowing it to read analog signals from sensors like the LM35.The Wi-Fi capabilities of the ESP32 enable it to act as a web server, where real-time data from the LM35 sensor can be displayed on a webpage.

The ESP32’s Wi-Fi module allows it to serve as a web server, making it possible to display sensor readings in real time.The web server is set up using the ESP32’s IP address, which users can access via a web browser on a local network.HTML code is embedded within the ESP32 code to create a webpage interface for displaying the temperature data**.**

**ESP32 required pin diagram-**  
  
  
**CONNECTION DIAGRAM -**

****

**Procedure:**

1. Connect the LM35 Sensor: Connect the Vcc and GND of the LM35 to 3.3V and GND on the ESP32, and the LM35 output pin to an ADC pin on the ESP32 (e.g., GPIO 34).
2. Setup Wi-Fi in Code: Open Arduino IDE, set the ESP32 board, and enter your Wi-Fi SSID and password to connect the ESP32 to the network.
3. Read Temperature Data: Read the analog output from the LM35 through the ADC pin and convert it to Celsius using the formula:  
    
4. Create a Web Server: Initialize the web server on the ESP32 and write HTML to display the temperature, refreshing data periodically using JavaScript.
5. Upload and Test: Upload the code to the ESP32, check its IP address in Serial Monitor, and visit that address in a browser to view real-time temperature readings.

**CODE-**

// Load Wi-Fi library

#include <WiFi.h>

// Replace with your network credentials

const char\* ssid = "Emb\_Lab";

const char\* password = "emb@1234";

// Set web server port number to 80

WiFiServer server(80);

N \ // Variable to store the HTTP request

String header;

// Assign output variables to GPIO pins

const int inputpin = 32;

void setup() {

Serial.begin(9600);

// Initialize the output variables as outputs

pinMode(inputpin, INPUT);

// Connect to Wi-Fi network with SSID and password

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

// Print local IP address and start web server

Serial.println("");

Serial.println("WiFi connected.");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

server.begin();

}

void loop(){

WiFiClient client = server.available(); // Listen for incoming clients

if (client) { // If a new client connects,

Serial.println("New Client."); // print a message out in the serial port

// Display the HTML web page

client.println("<!DOCTYPE html><html>");

client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

client.println("<link rel=\"icon\" href=\"data:,\">");

// CSS to style the on/off buttons

// Feel free to change the background-color and font-size attributes to fit your preferences

client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");

client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px;");

client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");

client.println(".button2 {background-color: #555555;}</style></head>");

// Web Page Heading

client.println("<body><h1>LM35 Temperature Sensor</h1>");

// Read analog value from LM35

int temp\_adc\_val = analogRead(inputpin);

// Convert ADC reading to voltage (in mV)

float voltage = (temp\_adc\_val / 4095.0) \* 1100;

// Convert voltage to temperature (Celsius)

float temp\_val = voltage / 10.0; // 10 mV per °c

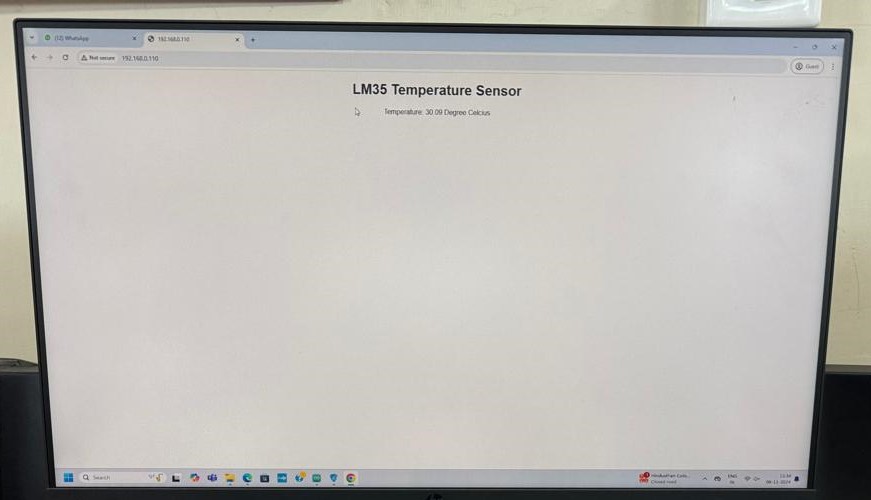
client.println("<p>Temperature: " + String(temp\_val) + " Degree Celcius</p>");

client.println("</body></html>");

delay(2000);

  }

}

**Photos of results :  
  
**

**Observations:**

* Wi-Fi Connection: The ESP32 successfully connects to the local Wi-Fi network, allowing it to serve as a web server accessible from other devices on the same network. The IP address displayed in the Serial Monitor is crucial for accessing the web server on any browser.
* HTML and CSS Styling: The webpage displayed on the client device uses embedded HTML and CSS for styling, with Google Fonts included to enhance readability. A yellow background with a shadowed temperature box creates a visually clean layout, while button styling and hover effects provide an interactive user experience.
* Temperature Reading: The code reads the temperature by capturing the LM35 sensor’s analog output voltage and converting it to Celsius using a formula. This reading is based on the ESP32’s 12-bit ADC, which translates the sensor output into a digital format for precise temperature measurements.
* Button-Triggered Readings: The webpage includes a button labeled “Get Temperature.” When clicked, the button triggers the temperature reading, retrieving the latest value from the sensor. The code checks for a specific character ('G') to initiate the temperature read-and-display cycle.
* Real-Time Display: The temperature is displayed dynamically on the webpage each time the button is clicked, allowing for real-time monitoring. The temperature value updates within a styled box, making it prominent and easy to view

# Conclusion:

This experiment successfully demonstrated how to interface the LM35 temperature sensor with an ESP32 microcontroller and create a web server to display temperature readings. By using WiFi to serve data, this setup allows for remote temperature monitoring, showcasing the ESP32’s potential for IoT applications where environmental data monitoring is needed.