Aditya Vishe BT22CIV091

Experiment No. 09

Interfacing LM35 Temperature Sensor with ESP32 Web Server

Aim

To interface an LM35 temperature sensor with an ESP32 microcontroller and display temperature readings on a web server.

Apparatus

- ESP32 microcontroller
- LM35 temperature sensor
- WiFi network
- Jumper wires
- Breadboard (optional)
- Computer with Arduino IDE for programming

Theory

The LM35 is an analog temperature sensor with a linear output proportional to temperature in Celsius. It produces a voltage output of 10 mV per degree Celsius. When interfaced with an ESP32 microcontroller, the analog reading from the LM35 is processed and converted into a temperature value, which can then be displayed on a web server hosted by the ESP32. This enables remote monitoring of temperature data.

The ESP32 can connect to a WiFi network to serve web pages, allowing temperature data to be accessed over the network from any web-enabled device.

Procedure

1. Circuit Setup:

- - Connect the LM35 sensor to the ESP32:
  - Connect the VCC pin of the LM35 to the 3.3V pin of the ESP32.
  - Connect the GND pin of the LM35 to the GND pin of the ESP32.

- Connect the output pin of the LM35 to the GPIO 34 (analog input) of the ESP32.

2. Code Explanation and Uploading:

- Connect the ESP32 to your computer and open the Arduino IDE.

- Paste the following code into the Arduino IDE:

```
// Load Wi-Fi library
#include <WiFi.h>
#include <Wire.h>

// Replace with your network credentials
const char* ssid     = "Emb_Lab";
const char* password = "emb@1234";

const int lmPin = 34;

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {
 Serial.begin(115200);
 // 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(".");
 }
 Serial.println("WiFi connected.");
 Serial.println("IP address: ");
```

```
  Serial.println(WiFi.localIP());
  server.begin();
  pinMode(lmPin, OUTPUT);
}

void loop(){
  WiFiClient client = server.available();
  if (client) {
    currentTime = millis();
    previousTime = currentTime;
    String currentLine = "";
    while (client.connected() && currentTime - previousTime <= timeoutTime) {
      currentTime = millis();
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        header += c;
        if (c == '\n') {
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name='viewport' content='width=device-width, initial-
scale=1'>");
            client.println("<style>body { text-align: center; font-family: 'Trebuchet MS', Arial;}");
            client.println("table { border-collapse: collapse; width:35%; margin-left:auto; margin-
right:auto; }");
            client.println("th { padding: 12px; background-color: #0043af; color: white; }");
            client.println("td { border: none; padding: 12px; }");
            client.println("</style></head><body><h1>ESP32 with LM35</h1>");
            client.println("<table><tr><th>MEASUREMENT</th><th>VALUE</th></ tr>");
            client.println("<tr><td>Temp. Celsius</td><td>" +
String(analogRead(lmPin)*110/4095) + " *C</td></tr>");
            client.println("</body></html>");
            client.println();
            break;
          } else {
            currentLine = "";
          }
        } else if (c != '\r') {
          currentLine += c;
```

```
      }
    }
   }
   header = "";
   client.stop();
 }
}
```

- Upload the code to the ESP32.

3. Running the Program:

-
  - After uploading the code, open the Serial Monitor in the Arduino IDE to view connection status.
  - Once connected, enter the ESP32 IP address in a web browser to view the temperature data.

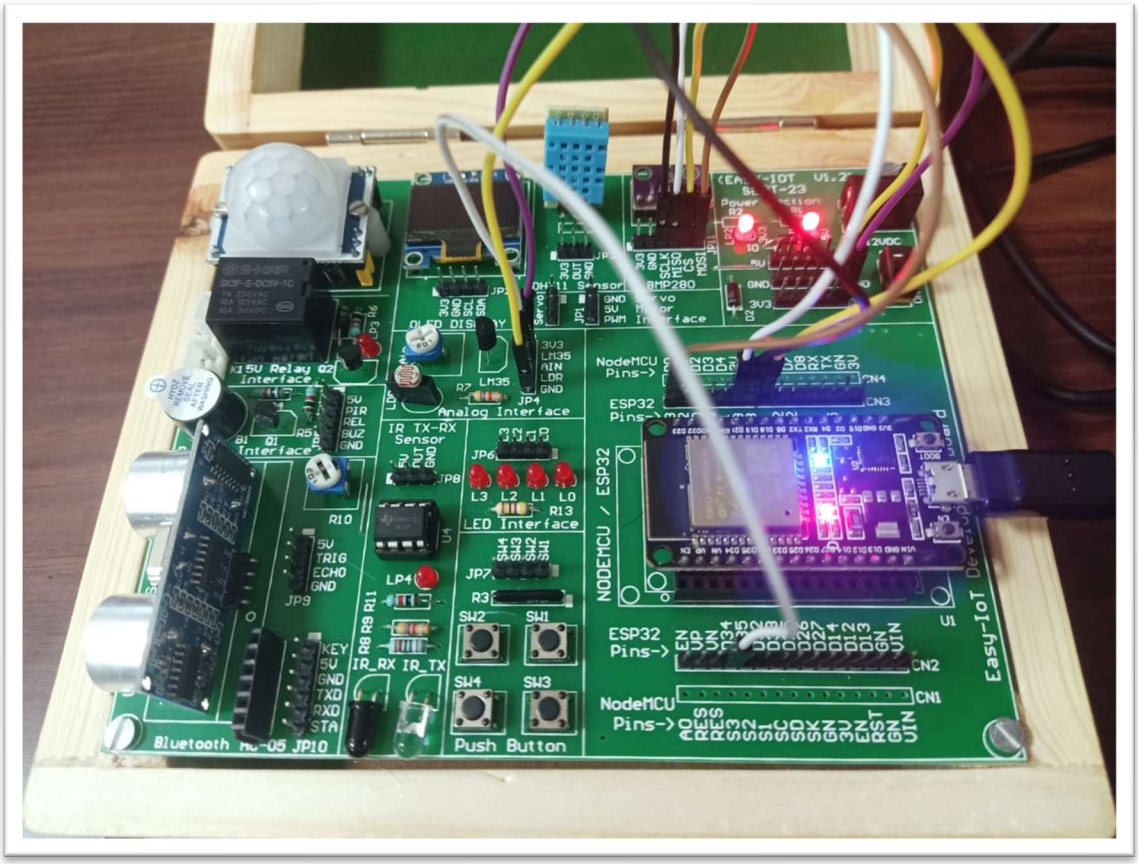
Observations

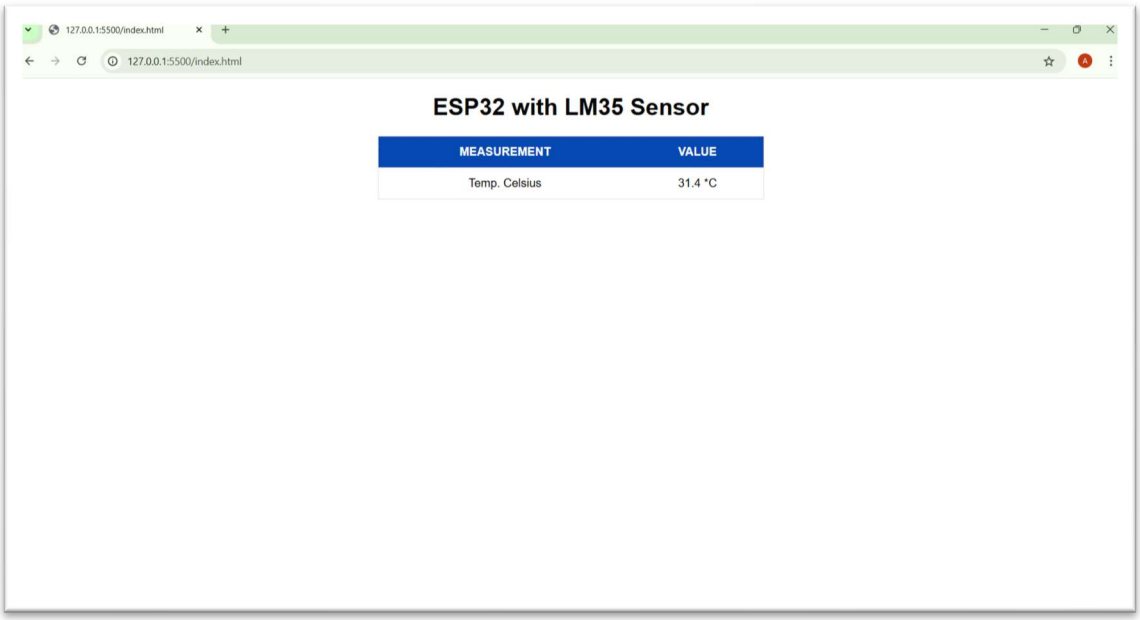
- The ESP32 web server displays temperature readings from the LM35 sensor in real-time.
- Any changes in temperature around the sensor are reflected promptly on the web page, demonstrating the sensor's accuracy and responsiveness.
- The readings remain consistent within a narrow range, suggesting stable sensor output, but slight fluctuations may be noticed based on the ambient environment.
- Observing the behavior over an extended period can reveal trends or patterns in temperature changes, if present.

Conclusion

The successful interfacing of the LM35 temperature sensor with the ESP32 and the display of readings on a web server demonstrate the practical applications of IoT in environmental monitoring.
This setup provides a convenient way to monitor temperature remotely, with potential applications in home automation, industrial monitoring, and scientific data collection. The ESP32's ability to act as a web server enables real-time monitoring, and the LM35 sensor's accuracy in temperature measurement makes it suitable for such applications. Overall, this experiment shows the effectiveness of integrating sensors with microcontrollers for real-time data collection and accessibility.

Result


5V Relay
Interface
IR TX-RX
Sensor
Analog Interface
LED Interface
Push Button
NODEMCU / ESP32
ESP32
Pins->
NodeMCU
Pins->
Easy-IoT


127.0.0.1:5500/index.html
ESP32 with LM35 Sensor
MEASUREMENT
VALUE
Temp. Celsius
31.4 °C