PREMIER UNIVERSITY CHATTOGRAM

DEPARTMENT OF **COMPUTER SCIENCE AND ENGINEERING**

PROJECT REPORT

COURSE NAME	Microcontrollers Laboratory		
COURSE CODE	CSE 3816		
PROJECT NAME	**ESP32 Environmental Monitor.**		
DATE OF SUBMISSION	15-09-2024		
SUBMITTED TO			
MOHAMMED SAIFUDDIN MUNNA ***ASSISTANT PROFESSOR*** ***DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING***			
SUBMITTED BY			
NAME		**ID**	
MD SAKIB		0222220005101019	
SAMIRA SULTANA MYSHA		0222220005101022	
FARIHA RASHID NOHA		0222220005101035	
RIMJHIM DEY		0222220005101039	
SEMESTER	4th	**SECTION**	A

Introduction

In this project, our mission was to build a web server using the ESP32 Dev Kit to monitor and display temperature and humidity data from a DHT11 sensor. The primary goal was to leverage the ESP32 Dev Kit's capabilities to create a user-friendly web interface that provides real-time environmental data. By integrating the DHT11 sensor, we aimed to offer accurate and up-to-date information on temperature and humidity conditions. This project showcases how IoT (Internet of Things) technologies can be utilized to build practical and interactive systems for remote monitoring.

Objectives

- To set up a web server on the ESP32 Dev Kit.
- To interface the ESP32 Dev Kit with the DHT11 sensor for reading temperature and humidity data.
- To display this data in real-time on a web page served by the ESP32 Dev Kit.

Components Used

- **ESP32 Dev Kit**: Serves as the microcontroller and web server.
- **DHT11 Sensor:** Measures temperature and humidity.
- **Breadboard:** Used for prototyping and connecting components.
- **Wires:** For making electrical connections between components.
- **Power Supply:** Provides power to the ESP32 Dev Kit and other components.

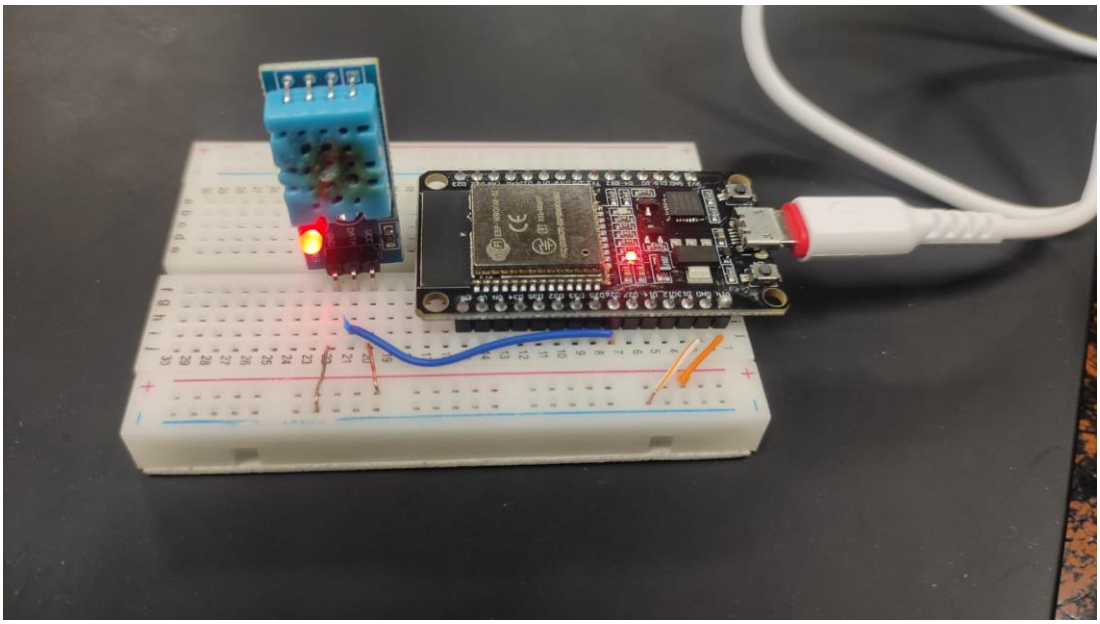
Programming Environment

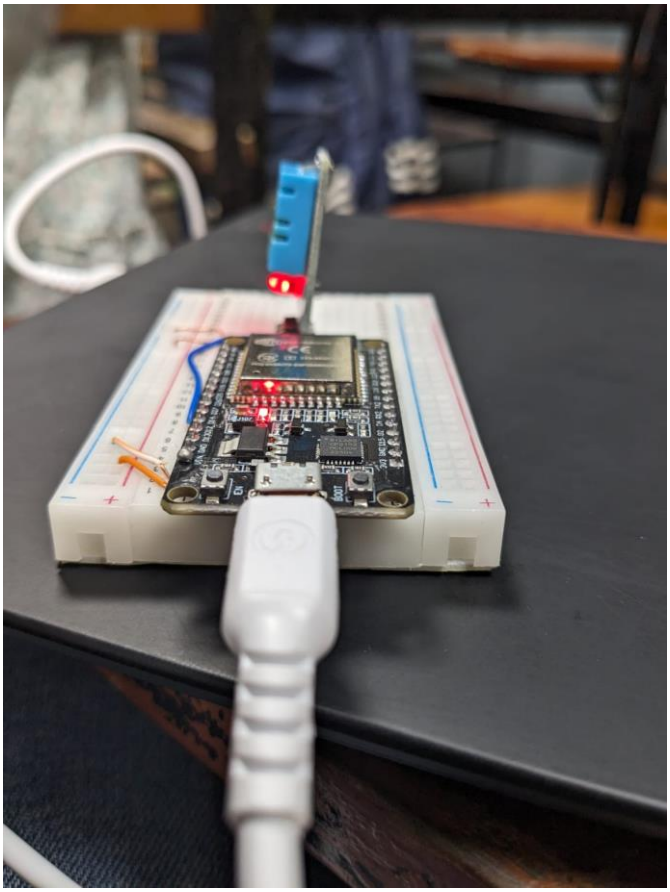
IDE: Arduino IDE with ESP32 board support installed

Libraries Used:

- DHT library for interacting with the DHT11 sensor
- WiFi library for network connectivity

Circuit Design





Code Implementation

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <DHT.h>

const char *ssid = "a";
const char *password = "bbbbbbbb";

WebServer server(80);
DHT dht(26, DHT11);

void handleRoot() {
  char msg[1500];

  snprintf(msg, 1500,
           "<html>\
  <head>\
    <meta http-equiv='refresh' content='4'/>\
    <meta name='viewport' content='width=device-width, initial-scale=1'>\
    <link rel='stylesheet'
href='https://use.fontawesome.com/releases/v5.7.2/css/all.css'
integrity='sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr'
crossorigin='anonymous'>\
    <title>ESP32 DHT Server</title>\
    <style>\
    html { font-family: Arial; display: inline-block; margin: 0px auto; text-
align: center;}\
    h2 { font-size: 3.0rem; }\
    p { font-size: 3.0rem; }\
    .units { font-size: 1.2rem; }\
    .dht-labels{ font-size: 1.5rem; vertical-align:middle; padding-bottom:
15px;}\
    </style>\
  </head>\
  <body>\
      <h2>ESP32 DHT Server!</h2>\
      <p>\
        <i class='fas fa-thermometer-half' style='color:#ca3517;'></i>\
        <span class='dht-labels'>Temperature</span>\
        <span>%.2f</span>\
        <sup class='units'>&deg;C</sup>\
      </p>\
      <p>\
```

```
        <i class='fas fa-tint' style='color:#00add6;'></i>\
        <span class='dht-labels'>Humidity</span>\
        <span>%.2f</span>\
        <sup class='units'>&percnt;</sup>\
      </p>\
  </body>\
</html>",
           readDHTTemperature(), readDHTHumidity()
          );
  server.send(200, "text/html", msg);
}

void setup(void) {

  Serial.begin(115200);
  dht.begin();

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  if (MDNS.begin("esp32")) {
    Serial.println("MDNS responder started");
  }
  server.on("/", handleRoot);

  server.begin();
  Serial.println("HTTP server started");
}

void loop(void) {
  server.handleClient();
  delay(2);//allow the cpu to switch to other tasks
}
```

```
float readDHTTemperature() {

  // Sensor readings may also be up to 2 seconds
  // Read temperature as Celsius (the default)

  float t = dht.readTemperature();
  if (isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return -1;
  }
  else {
    Serial.println(t);
    return t;
  }
}

float readDHTHumidity() {
  // Sensor readings may also be up to 2 seconds
  float h = dht.readHumidity();
  if (isnan(h)) {
    Serial.println("Failed to read from DHT sensor!");
    return -1;
  }
  else {
    Serial.println(h);
    return h;
  }
}
```

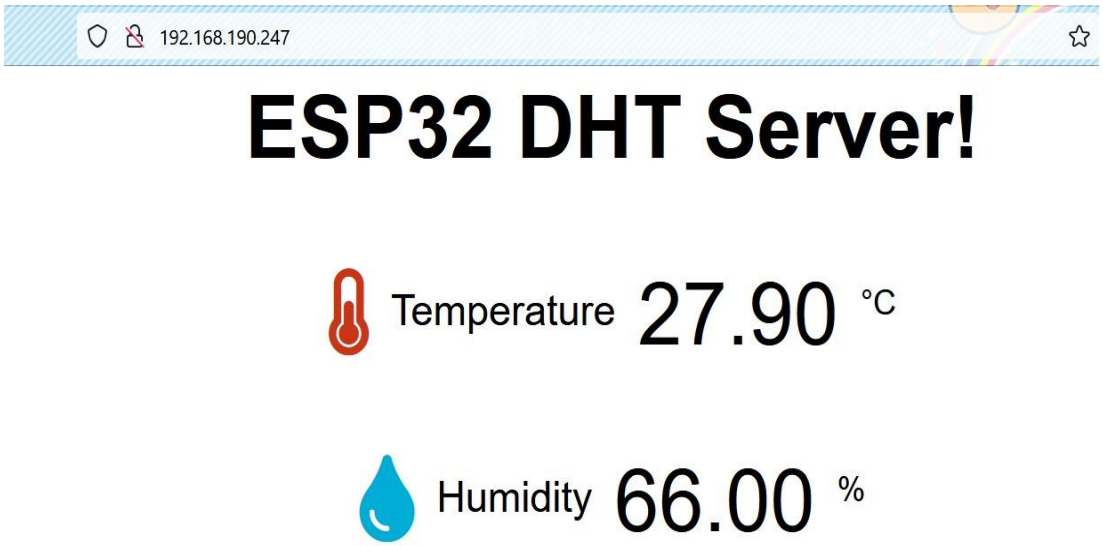
Results

Web Interface: The ESP32 Dev Kit hosts a web page that displays real-time temperature and humidity data. Accessing the ESP32's IP address through a web browser shows this data clearly and promptly.

Data Accuracy: The DHT11 sensor provides reliable temperature and humidity readings, suitable for general monitoring purposes.

Performance: The ESP32 Dev Kit handles web server duties effectively, with minimal delay in data transmission and response to client requests.

A screenshot of the web interface is provided below, showcasing the real-time temperature and humidity data as displayed on the ESP32 Dev Kit's web server. This visual evidence highlights the effectiveness of the server in delivering accurate environmental data.



Challenges

Sensor Issues: One significant challenge was dealing with the DHT11 sensor, which initially provided inconsistent and inaccurate readings. We had to replace the original sensor with a new one to obtain reliable data. This issue required troubleshooting and verification to ensure the sensor provided accurate measurements.

Wi-Fi Connectivity: Establishing and maintaining a stable Wi-Fi connection between the ESP32 Dev Kit and the network was challenging. We faced intermittent connectivity issues that required careful network configuration and adjustments to ensure consistent communication.

Server Connection Stability: Maintaining a reliable connection to the web server was another hurdle. We had to ensure that the server could handle client requests smoothly and without interruption, which involved refining the server setup and optimizing performance.

Code Optimization: Refining the code to efficiently handle sensor data and server responses was essential. Ensuring that the code was optimized for performance and reliability involved iterative testing and adjustments to address issues such as latency and data handling.

Conclusion

The ESP32 Dev Kit DHT Web Server project effectively demonstrated the capability of the ESP32 Dev Kit to operate as a web server and interface with sensors. The resulting system provides real-time environmental data through an intuitive web interface, making it a practical tool for remote monitoring.

Future Work

Enhanced Sensor Accuracy: Consider integrating more advanced sensors for improved data accuracy and reliability.

Data Logging: Implement features to log historical data and analyze trends over time.

User Interface Improvements: Develop a more sophisticated web interface with additional features such as charts and graphs.