ESP32-S3 Web Server for AP and Station Mode

Objective:

The objective of this project is to develop a web server using an ESP32-S3 that functions in both Access Point (AP) and Station modes. The web server allows users to input RGB values through a web page, which then controls an RGB LED connected to the ESP32. Additionally, the web server reads humidity and temperature data from a DHT11 sensor and displays these values on the same web page. An OLED display is used to show relevant messages. Furthermore, an additional web page is designed to perform extra tasks on the ESP32.

Requirements

Web Interface

- Design a web page where users can input values for Red (R), Green (G), and Blue (B).
- Submit the RGB values from the web page to the ESP32.
- Display the same RGB values on an RGB LED connected to the ESP32.

Sensor Integration

- Use a DHT11 sensor to sense humidity and temperature.
- Display the sensed humidity and temperature values on the web page.

OLED Display

- Update the web page design to display messages on an OLED interfaced with the ESP32.

Additional Web Page

- Design another web page to perform additional tasks on the ESP32, chosen by the developer.

Styling

- Use CSS for proper formatting and an attractive interface for all web pages.

Implementation

- Run the code through Thonny and record the ESP32's performance based on the designed pages, in both AP and Station modes.
- Push the work to GitHub with individual commits for each sub-task.
- Ensure all team members understand the complete assignment and show individual contributions.
- All tasks must be executed by each group member on their own laptop while sharing the ESP32.
- Create a comprehensive PDF report for all tasks.
- Avoid direct copying from any source.
- Complete a viva before the Eid holidays.

ESP32-S3 Web Server Code

```
from machine import Pin, I2C

from ssd1306 import SSD1306_I2C

from time import sleep

import network

import socket

import json


# OLED setup

WIDTH = 128

HEIGHT = 64

i2c = I2C(0, scl=Pin(9), sda=Pin(8))
```

```
oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)
oled.fill(0)
oled.text('Starting AP...', 0, 0)
oled.show()

# AP mode setup
AP_SSID = "ESP32-S3_AP"
AP_PASSWORD = "12345678"
ap = network.WLAN(network.AP_IF)
ap.active(True)
ap.config(essid=AP_SSID, password=AP_PASSWORD)

while not ap.active():
    pass

ip = ap.ifconfig()[0]
print('AP Started. IP:', ip)
oled.fill(0)
oled.text('AP Mode Active', 0, 0)
oled.text(f'IP: {ip}', 0, 10)
oled.show()

# HTML page with Wi-Fi connect form
html = """
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>ESP32-S3 Control Panel</title>
</head>
<body>
  <h1>ESP32-S3 Control Panel</h1>
  <button onclick="scanNetworks()">Scan Networks</button>
  <div id="networks"></div>

  <h2>Connect to Wi-Fi</h2>
  <form onsubmit="connectWiFi(event)">
    <label for="ssid">Select Network:</label>
    <select id="ssid" name="ssid" required></select>
    <br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>
    <br>
    <button type="submit">Connect</button>
  </form>
  <div id="status"></div>
</body>
</html>
"""

# Scan Wi-Fi networks
```

```
def scan_networks():
    sta = network.WLAN(network.STA_IF)
    sta.active(True)
    networks = sta.scan()
    sta.active(False)

    net_list = []
    for net in networks:
        ssid = net[0].decode('utf-8', 'ignore')
        rssi = net[3]
        auth = net[4]
        sec_type = "Open" if auth == 0 else "WPA/WPA2" if auth in [2, 3] else "WPA3"
        net_list.append({"ssid": ssid, "rssi": rssi, "security": sec_type})

    return json.dumps(net_list)

# Connect to Wi-Fi network
def connect_to_wifi(ssid, password):
    sta = network.WLAN(network.STA_IF)
    sta.active(True)
    sta.connect(ssid, password)

    timeout = 10
    while not sta.isconnected() and timeout > 0:
        sleep(1)
        timeout -= 1
    if sta.isconnected():
        ip = sta.ifconfig()[0]
        oled.fill(0)
        oled.text('Connected to:', 0, 0)
        oled.text(ssid, 0, 10)
        oled.text('IP: ' + ip, 0, 20)
        oled.show()
        return f'Connected! IP: {ip}'
    else:
        oled.fill(0)
        oled.text('Failed to connect', 0, 0)
        oled.show()
        return 'Connection Failed'

# Web server
def web_server():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(('0.0.0.0', 80))
    s.listen(5)
    print('Web server running...')

    while True:
        conn, addr = s.accept()
        request = conn.recv(1024).decode('utf-8')
        if '/scan' in request:
            networks = scan_networks()
            conn.send('HTTP/1.1 200 OK\r\nContent-Type: application/json\r\n\r\n' + networks)

        elif '/connect' in request:
            data = json.loads(request.split('\r\n\r\n')[1])
            ssid = data.get('ssid')
            password = data.get('password')
            status = connect_to_wifi(ssid, password)
            conn.send('HTTP/1.1 200 OK\r\nContent-Type: text/plain\r\n\r\n' + status)

        else:
            conn.send('HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n' + html)

        conn.close()

web_server()
```