Laiba Taseen
23-NTU-CS-FL-1258

Homework-01 – After Mid

Embedded IoT Systems – Fall 2025

Question-1: ESP32 Webserver (webserver.cpp)

Part-A: Short Questions

1. What is the purpose of WebServer server(80); and what does port 80 represent?

WebServer server(80); creates a web server object on the ESP32.
It allows the ESP32 to listen for incoming HTTP requests from a web browser.

Port **80** is the **default HTTP port**, which means when a user types the ESP32 IP address in a browser, the request automatically goes to port 80. This enables the ESP32 to act like a small web server.

2. Explain the role of server.on("/", handleRoot); in this program.

This statement defines the **route handling** for the web server.

- / represents the **home page**
- handleRoot is the function that will execute when the home page is requested

When a user opens the ESP32 IP address in a browser, the handleRoot() function is called to send the webpage content.

3. Why is server.handleClient(); placed inside the loop() function? What will happen if it is removed?

server.handleClient(); continuously checks for incoming client (browser) requests.

It is placed inside loop() because loop() runs repeatedly, allowing the ESP32 to respond to web requests at any time.

If it is removed:

- ESP32 will not respond to browser requests
- The webpage will not load or refresh

Laiba Taseen
23-NTU-CS-FL-1258

4. In handleRoot(), explain the statement:

server.send(200, "text/html", html);

This line sends the HTTP response to the client.

- 200 → HTTP status code meaning **OK**
- "text/html" → specifies the content type as a web page
- html → contains the webpage content

This command displays the webpage on the browser.

5. What is the difference between displaying last measured sensor values and taking a fresh DHT reading inside handleRoot()?

Displaying last measured values uses already stored sensor data, which is faster and more stable.

Taking a fresh DHT reading inside handleRoot():

- Causes delay in webpage loading
- May lead to sensor errors
- Can make the web server unresponsive

Therefore, it is better to read the sensor separately and only display stored values on the webpage.

Part-B: Long Question

Complete Working of ESP32 Webserver-Based Temperature and Humidity Monitoring System

1. ESP32 Wi-Fi Connection and IP Address Assignment

The ESP32 connects to a Wi-Fi network using the SSID and password.
Once connected, the router assigns an IP address to the ESP32.
This IP address is used by the user to access the web server through a browser.

2. Web Server Initialization and Request Handling

A web server is created on port 80 using the WebServer library.
Routes are defined using server.on().

Laiba Taseen
23-NTU-CS-FL-1258

The server is started using server.begin().
Incoming browser requests are handled continuously using server.handleClient().

3. Button-Based Sensor Reading and OLED Update Mechanism

A push button is used to manually trigger sensor readings.
When the button is pressed:

- DHT sensor values are read
- OLED display updates with new temperature and humidity values

This approach avoids unnecessary continuous sensor readings.

4. Dynamic HTML Webpage Generation

HTML content is generated dynamically by inserting the latest temperature and humidity values into the webpage.
This allows real-time monitoring without reprogramming the ESP32.

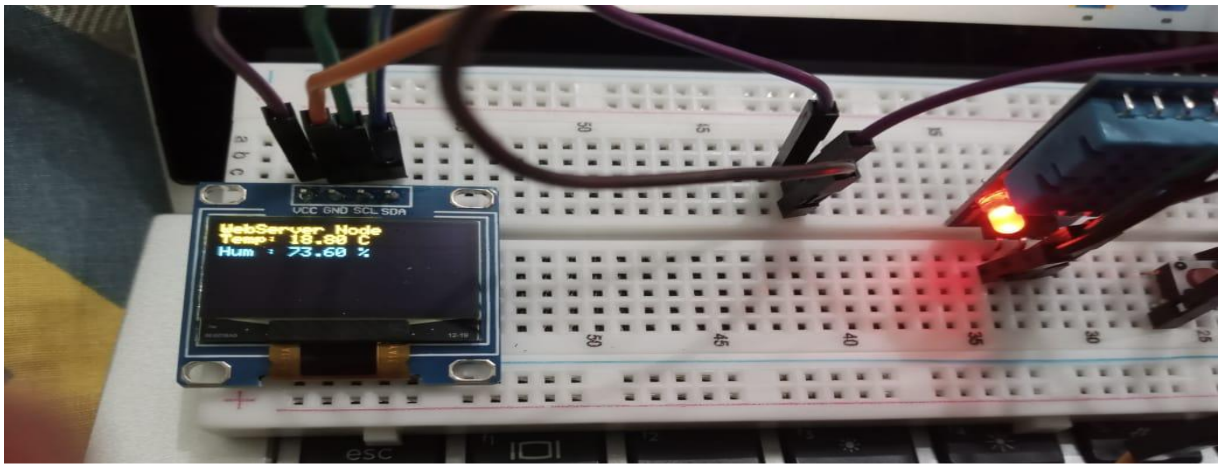
5. Purpose of Meta Refresh in the Webpage

The meta refresh tag automatically reloads the webpage after a fixed interval (e.g., 5 seconds).
This ensures updated sensor values are shown without manually refreshing the browser.

6. Common Issues and Their Solutions

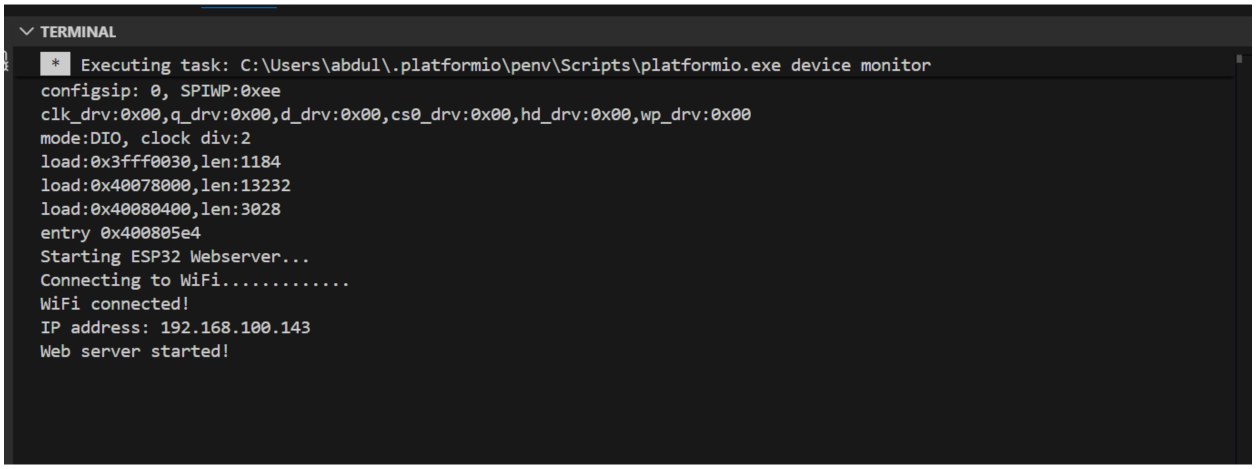
Issue	Solution
Webpage not loading	Check IP address and Wi-Fi
DHT sensor gives NaN	Correct sensor type and delay
ESP32 hangs	Avoid sensor reads inside request handler
OLED not working	Verify I2C address and pins

ScreenShot:



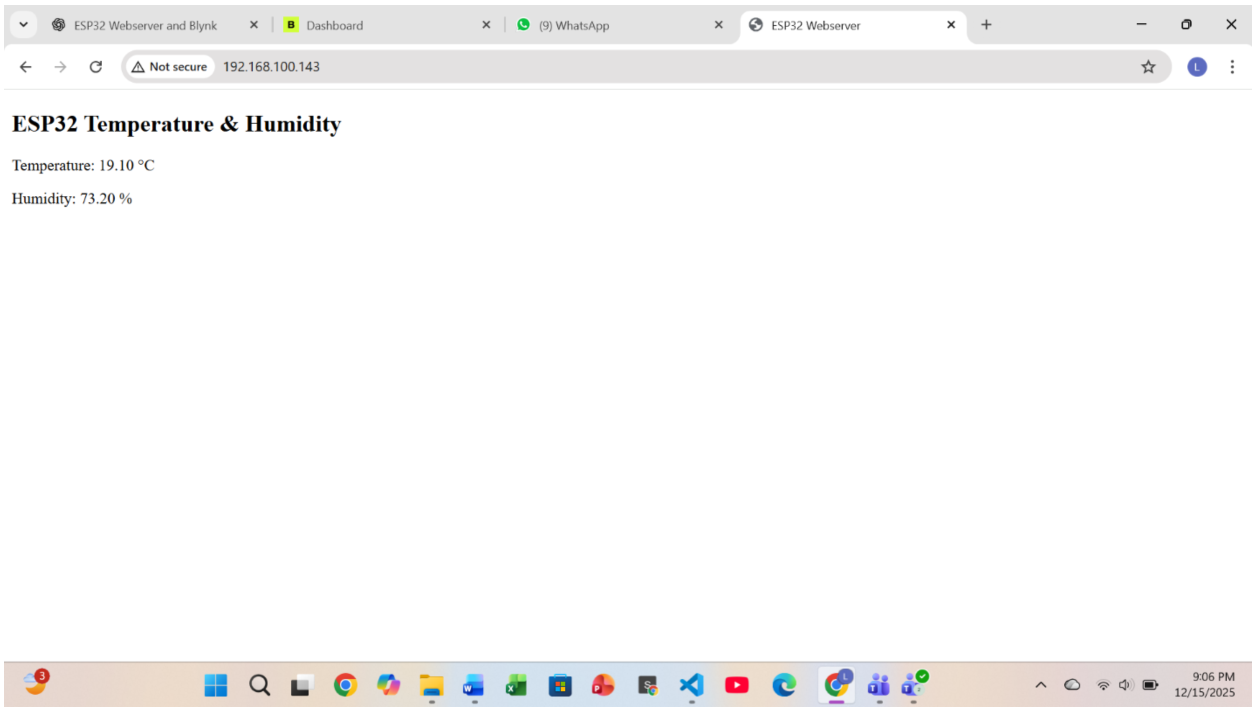
Laiba Taseen
23-NTU-CS-FL-1258

Serial Monitor:



```
 *  Executing task: C:\Users\abdul\.platformio\penv\Scripts\platformio.exe device monitor
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1184
load:0x40078000,len:13232
load:0x40080400,len:3028
entry 0x400805e4
Starting ESP32 Webserver...
Connecting to WiFi.............
WiFi connected!
IP address: 192.168.100.143
Web server started!
```

Web Dashboard:



ESP32 Temperature & Humidity

Temperature: 19.10 °C

Humidity: 73.20 %

Question-2: Blynk Cloud Interfacing (blynk.cpp)

Part-A: Short Questions

Laiba Taseen
23-NTU-CS-FL-1258

1. What is the role of Blynk Template ID in an ESP32 IoT project? Why must it match the cloud template?

The Blynk Template ID uniquely identifies the project on Blynk Cloud.
It must match the cloud template so that the ESP32 connects to the correct project configuration.

2. Differentiate between Blynk Template ID and Blynk Auth Token.

Template ID	Auth Token
Identifies project	Identifies device
Same for all devices	Unique for each device
Created once	Generated per device

3. Why does using DHT22 code with a DHT11 sensor produce incorrect readings?

DHT11 and DHT22 have different data timing, accuracy, and ranges.
Using DHT22 code with a DHT11 sensor results in incorrect or unstable readings.

4. What are Virtual Pins in Blynk? Why are they preferred?

Virtual Pins are software-based pins used for cloud communication in Blynk.
They are preferred because:

- They are flexible
- Independent of physical GPIO pins
- Ideal for cloud data transfer

5. What is the purpose of using BlynkTimer instead of delay()?

delay() blocks program execution and disrupts Wi-Fi and Blynk communication.
BlynkTimer allows non-blocking periodic tasks and keeps the system responsive.

Part-B: Long Question

Complete Workflow of Interfacing ESP32 with Blynk Cloud

1. Creation of Blynk Template and Datastreams

A template is created on Blynk Cloud specifying hardware and connection type.
Datastreams (V0, V1) are added for temperature and humidity.

2. Role of Template ID, Template Name, and Auth Token

Laiba Taseen
23-NTU-CS-FL-1258

- Template ID links firmware to the cloud project
- Template Name is for user identification
- Auth Token connects a specific ESP32 device to Blynk Cloud

3. Sensor Configuration Issues (DHT11 vs DHT22)

Incorrect sensor selection leads to wrong data.
Correct sensor type, pin configuration, and timing are required for accurate readings.

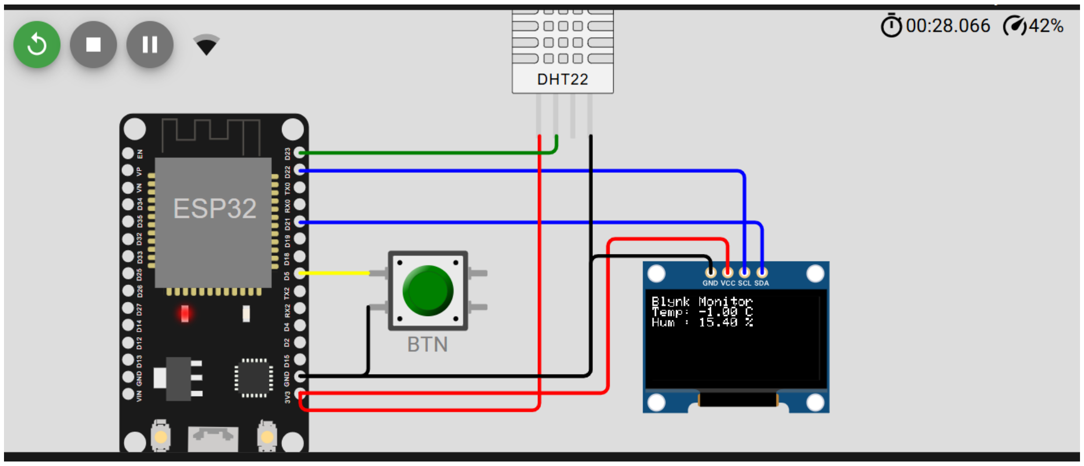
4. Sending Data Using Blynk.virtualWrite()

Sensor values are sent to Blynk using virtual pins.
These values are displayed on mobile app widgets in real time.

5. Common Problems and Their Solutions

Problem	Solution
Device offline	Check Auth Token
No data update	Verify virtual pin mapping
Wrong values	Correct sensor type
App not updating	Check internet and timer

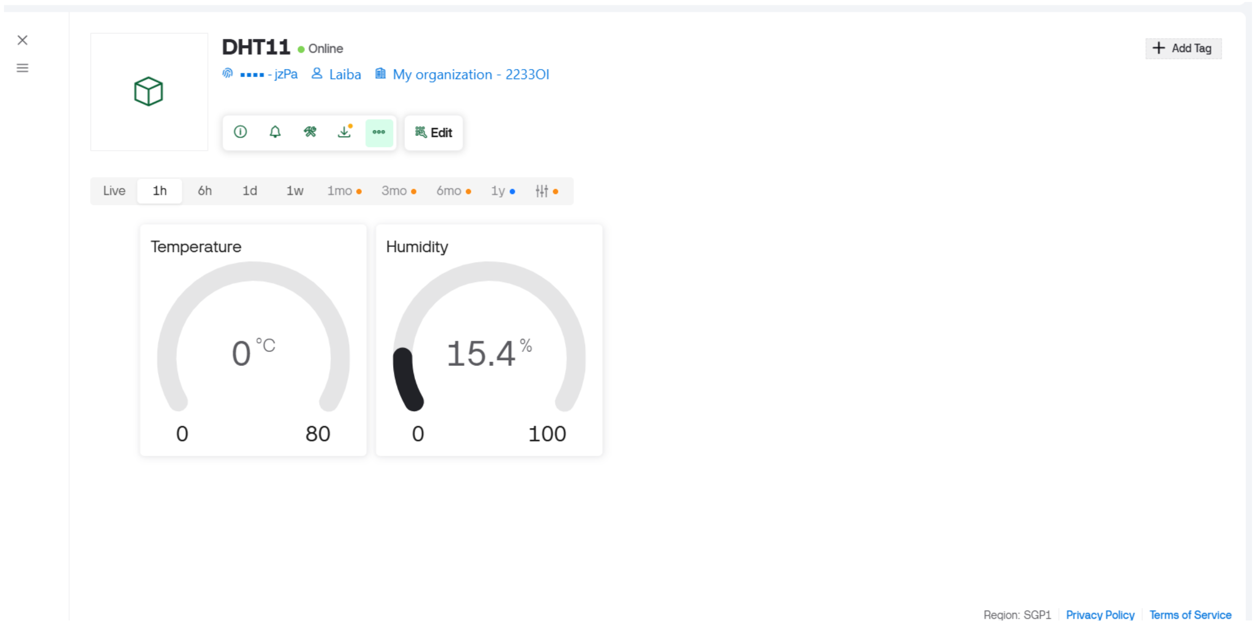
ScreenShot:



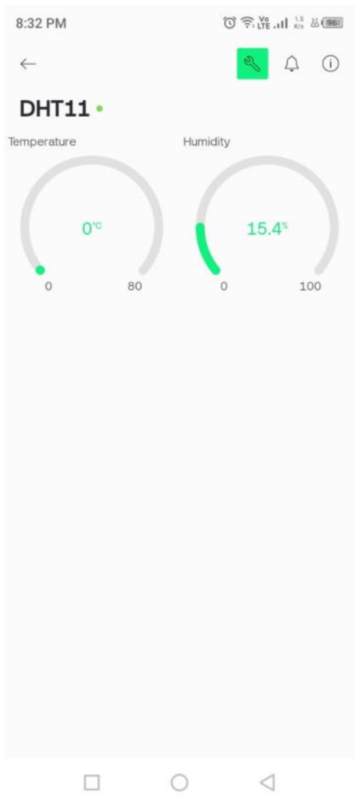
```
Blynk Monitor
Temp: -1.00 C
Hum : 15.40 %
```

Web Dashboard:

Laiba Taseen
23-NTU-CS-FL-1258


DHT11 ● Online
•••• - jzPa   Laiba   My organization - 2233OI
+ Add Tag
Edit
Live 1h 6h 1d 1w 1mo 3mo 6mo 1y
Temperature
0 °C
0 80
Humidity
15.4 %
0 100
Region: SGP1 Privacy Policy Terms of Service

Blynk Mobile App:


8:32 PM
DHT11
Temperature
0°C
0 80
Humidity
15.4%
0 100