<u>Homework-01 – After mid</u>

<u>Embedded IOT Systems Fall2025</u>

<u>AI 5th</u>

Name: Sabika Batool Zaidi.

Reg.No.: 23-NTU-CS-1282.

<u>Question-1</u>

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): This declares an instance of the WebServer class, by default from the ESP32WebServer library, and on ESP32 often simply WebServer. It instantiates the web server object named server and sets up it to listen for incoming HTTP requests on some port number.

Port 80: This is the default port for the Hypertext Transfer Protocol, or HTTP. When a web browser accesses an IP address without specifying what port it wants to connect to, for example, by simply typing ip address in the address bar, it attempts to connect to port 80 by default. By hosting your ESP32 web server on port 80, you can ensure the web server will be easily accessible from a common web browser.

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

Role: It is a request handler definition. It tells the server object what to do when it receives a client request for a specific URL path.

"/": It is the root path or the homepage of the web server. It's the path requested when a user simply types the ESP32's IP address (e.g., 192.168.1.10) into their browser.

HandleRoot: This is the callback function that will be executed whenever a request is received for the root path (/). The handleRoot() function has the logic to generate the HTML response.

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

is removed?

Server.handleClient(): It is constantly monitoring incoming web requests made by clients.

This needs to be placed within the loop() function so that the ESP32 is always able to handle any requests that may arise at any time.

If removed, the web server will not respond, and the webpage will not display.

4. In handleRoot(), explain the statement: server.send(200, "text/html", html);

The statement "**server.send(200, "text/html", html)"** sends the response from ESP32 to the browser.

In this statement 200 indicates the successful HTTP request.

Text/html indicates that the content is in HTML.

Html indicates the web page data.

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

The last recorded values for the sensors displayed on the screen reveal that data is being retrieved from global variables that had been updated previously, and this causes quicker loading of the web page. It is a non-blocking technique and helps the web server run efficiently.

While to obtain a fresh DHT reading inside the handleRoot() function, it will read values directly from the sensor, yielding real-time information. However, this approach is blocking and may end up slowing down the server if the page is being refreshed constantly.

Part B: Long Question

1. Describe the complete working of the ESP32 webserver-based temperature and humidity monitoring system.

- <u>ESP32 Wi-Fi Connection Process and IP Address Assignment</u>

ESP32 is connected to a Wi-Fi network using SSID and password with the WiFi.begin() function. The code continuously monitors the connection status until it gets connected to the ESP32. After connecting, it allocates an IP address to ESP32, which can be shown using WiFi.localIP().This IP address is utilized by the user to gain access to the web server using the browser.

- Initialization and Request Processing for Web Server

After creating Wi-Fi, the webpage server is initialized by server.begin(). ESP32 listens for HTTP requests on port 80.When the client connects with the IP address of the ESP32, routes like / are managed using the server.on() function. The method server.handleClient() is continuously running in the loop for handling client connections.

- Button-Based Sensor Reading and OLED Update Mechanism

A push button is employed to regulate when data from the DHT sensor should be read. Upon pressing the button, the ESP32 will read the temperature and humidity data from the DHT sensor. These values are then reflected on the OLED display for monitoring. This method helps prevent continuous readings in the sensor. This results in a more efficient system.

- Dynamic HTML Webpage

The ESP32 creates a dynamic HTML webpage from string variables in the code. Sensor values are incorporated into the HTML page before it is sent to the client computer. Upon page reloading by the user, new temperatures and humidity levels are displayed. This features real-time monitoring via web browser.

- The Use of Meta Refresh in the Webpage

Meta refresh tags in HTML help in refreshing the webpage after a fixed time period. Thus, it can be seen that it ensures that the values being shown for the sensors that are being displayed in the application It can be applied in real-time data monitoring where live data updates are needed.

- Common Issues in ESP32 Webserver Projects and Their Solutions

Frequently arise may include internet connectivity issues due to a lack of the right credentials for the Wi-Fi. This can be solved by ensuring that the SSID and the password are right. Another problem is when the webpage fails to load due to the lack of the function server.handleClient() in the loop. The wrong sensor type or too brief a delay between samples may cause incorrect or even NaN (Nil) values in sensor data.

Question-2

Blynk Cloud Interfacing (blynk.cpp)

Part-A: Short Questions

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

cloud template?

It identifies the cloud template that defines the device's interface: widgets, pins, and settings.It should exactly be the same as the cloud template so that the device syncs correctly with the Blynk app interface.

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

The Blynk Template ID informs your board which project template to use, including how the widgets and virtual pins are arranged. The “Blynk Auth Token” is unique to each board and is how it authenticates and allows connectivity to the Blynk Cloud. In short, the Template ID is like the blueprint for the project, and the Auth Token is like your key to get into the cloud.

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

   Mention one key difference between the two sensors.

Would give incorrect readings since the DHT11 and DHT22 have different timings and data format. **Key difference:** DHT22 is more accurate and supports higher resolution/temperature range than DHT11.

4. ***What are Virtual Pins in Blynk? Why are they preferred over physical GPIO pins for***

   cloud communication?

Virtual pins are software-defined cloud pins that don't correspond to physical GPIO pins. Preferred for cloud communication, as they allow flexible and scalable interactions between a device and an application without changing any wiring in the hardware.

5. ***What is the purpose of using BlynkTimer instead of delay() in ESP32 IoT***

   applications?

It can be used for non-blocking periodic tasks, which keeps the ESP32 responsive. This delay() will block any code execution and may freeze the communication of your device in the cloud, or any sensor reading.

Part-B: Long Question

1. ***Explain the complete workflow of interfacing ESP32 with Blynk Cloud to display***

temperature and humidity values.

Hardware Setup: Connect DHT11/DHT22 sensor to ESP32 as follows: VCC to 3.3V, GND to GND, Data to a GPIO digital pin.

Blynk Project Creation: Launch the Blynk application, create new template involving ESP32 with WiFi, then add datastreams of temperature and humidity, then add devices and generate the Template Name, Template ID and Auth Token.

Widget Configuration: Create widgets such as gauges or value display widgets in the Blynk app and set up virtual pins for temperature and humidity.

ESP32 Code:This code can also be implemented by including the

Explain WiFi credentials, Blynk Auth Token, and DHT pin/type.

WiFi, Blynk, DHT sensor initialized in setup().

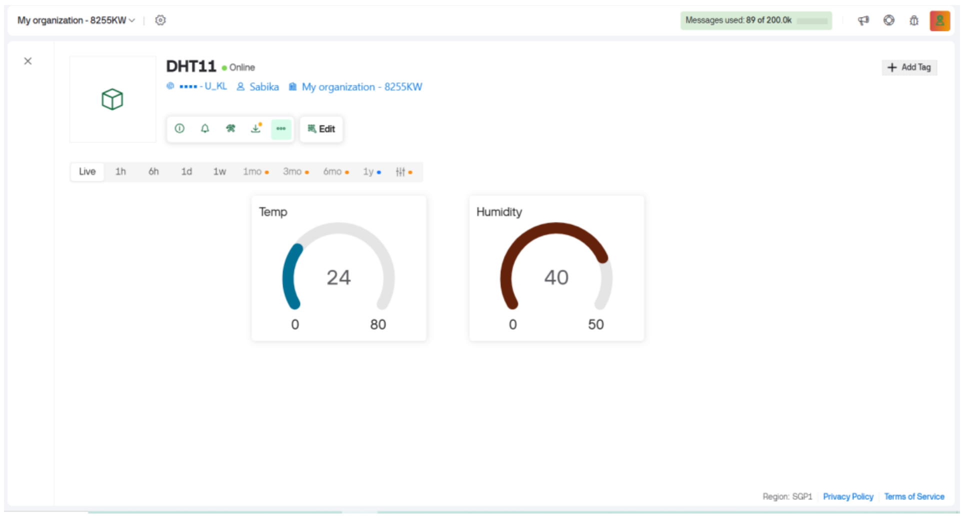
Start the function loop() and inside this function call its counterpart in the other sketch using Blynk.run(), then update the readings from the sensors and write them into the

Upload and Test: The code needs to be uploaded to ESP32.

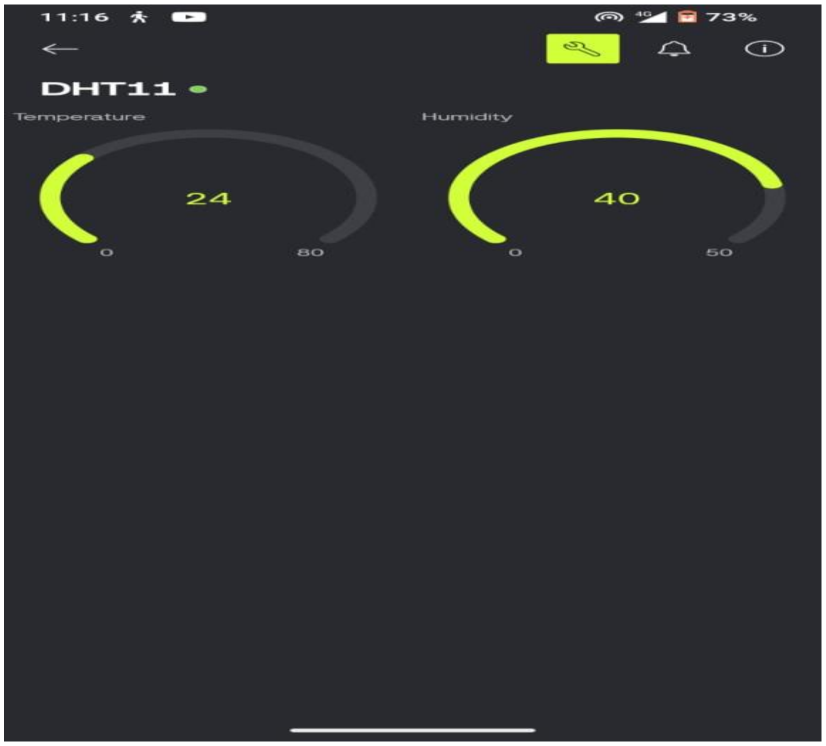
Once connected, readings of temperature and humidity are displayed live in real time on the Blynk app.

SCREENSHOTS:

BLYNK WEB DASHBOARD:



BLYNK MOBILE INTERFACE:



GITHUB LINK:

https://github.com/Sabika-Batool-Zaidi/IOT-practical