Name: AREEB FATIMA

Regno: 23-NTU-CS-1246

IoT Assignment :2

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 that listens for incoming HTTP requests.
   Port 80 is the default port for HTTP communication, allowing users to access the ESP32 web page directly through a browser without specifying a port number

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

   This line defines a route for the root URL (/).
   When a client opens the ESP32 IP address in a browser, the server automatically calls the handleRoot() function to generate and send the webpage containing temperature and humidity data

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

   server.handleClient(); continuously checks for incoming client requests and processes them.
   The ESP32 will stop responding to browser requests, and the web page will not load or update.

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

   The statement **server.send(200, "text/html", html);** sends the HTML webpage containing sensor data to the clients browser.

   - The parameter 200 represents HTTP success status code
   - Text/html defined content type

- html represents dynamically generated webpage

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

   - **Last measured values:** provides faster response , avoids repeated sensor reads and is more stable.
   - **Fresh reading inside handleRoot( ):** Provides real-time sensor data, but frequent access may introduce delays and can cause DHT timing or communication errors.

Part B: Long Question

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

The ESP32 connects to Wi-Fi using predefined credentials (configured SSID and password) and waits until a successful connection is established. Once connected, it is assigned an IP address by the router, which is used to access the ESP 32 webserver through a web browser.

A web server is initialized on port 80(default HTTP port). When a user enters the ESP32 IP address in a browser, the server handles the request using the handleRoot() function.

The system includes a push button connected to GPIO5. When the button is pressed, the ESP32 reads temperature and humidity values from the DHT11 or DHT22 sensor. These values are stored and immediately displayed on the OLED screen.

The web page is dynamically generated using HTML strings. It displays the latest temperature and humidity readings and automatically refreshes every 5 seconds using a meta refresh tag. This ensures updated data without manually reloading the page.

Purpose of Meta Refresh

The meta refresh tag automatically reloads the webpage after a fixed interval, allowing near real-time monitoring without user interaction.

Common Issues and Solutions

- Wi-Fi connection failure: Check SSID/password
- OLED not displaying: Verify I2C address and wiring

- DHT errors: Add delays and avoid frequent reads
- Webpage not loading: Ensure server.handleClient() is running

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?**

   The Template ID links the ESP32 device to a specific Blynk Cloud template.
   It must match the cloud template so the device can access predefined datastreams and dashboards.

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

- **Template ID**: Identifies the project structure in Blynk Cloud
- **Auth Token**: Authenticates a specific device and allows it to send data

3. **Why does using DHT22 code with a DHT11 sensor produce incorrect readings? Mention one key difference between the two sensors.**

   DHT11 and DHT22 have different data formats and accuracy.
   Using DHT22 code with DHT11 produces incorrect values because DHT22 has higher resolution and different timing.

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

   Virtual Pins are cloud-based data channels used to send or receive values.
   They are preferred because they are **hardware-independent** and designed for IoT communication.

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

BlynkTimer allows non-blocking periodic tasks.
Using delay() would block Wi-Fi and Blynk communication, causing disconnections

Part-B: Long Question

Explain the complete workflow of interfacing ESP32 with Blynk Cloud to display temperature and humidity values.

A Blynk template is first created on the Blynk Cloud. We add a new device named DHT11 or according to our project. Dashboard is created for the device and then we add datastreams for temperature and humidity. Each datastream is assigned a virtual pin eg V0 for temperature and V1 for humidity .

The ESP32 firmware includes the Template ID, Template Name, and Auth Token, allowing it to authenticate with the Blynk Cloud.

The DHT22 or DHT11 sensor is configured correctly to avoid incorrect readings. The ESP32 reads temperature and humidity values either periodically using BlynkTimer or manually through a button press.

Sensor values are sent to the Blynk Cloud using:

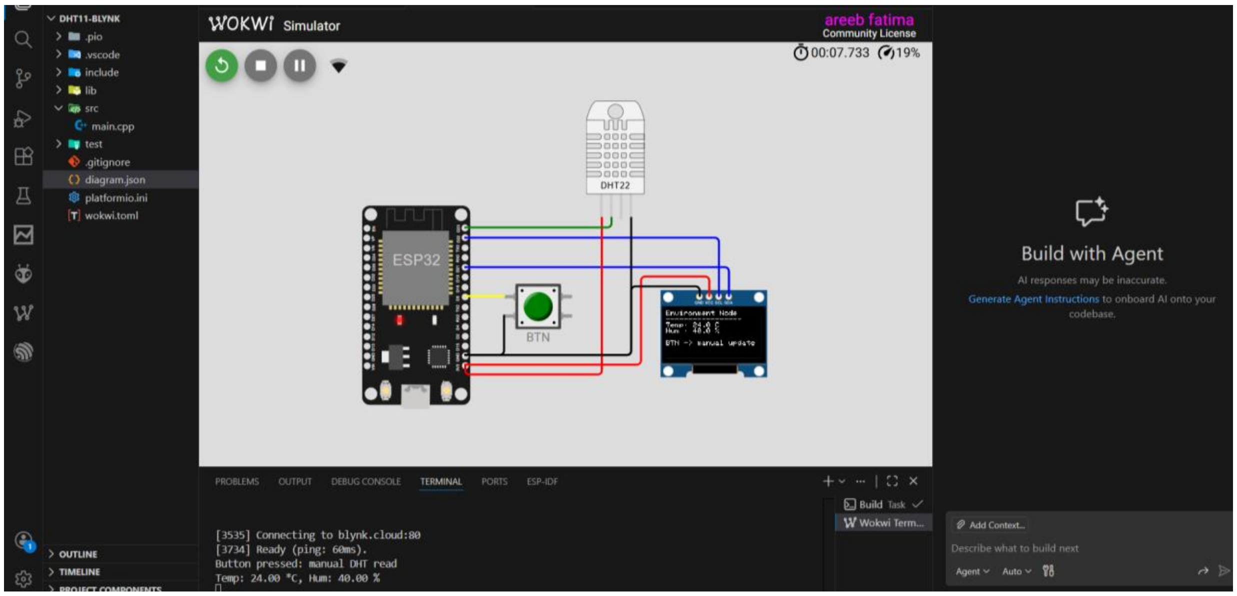
```
Blynk.virtualWrite(V0, temperature);

Blynk.virtualWrite(V1, humidity);
```

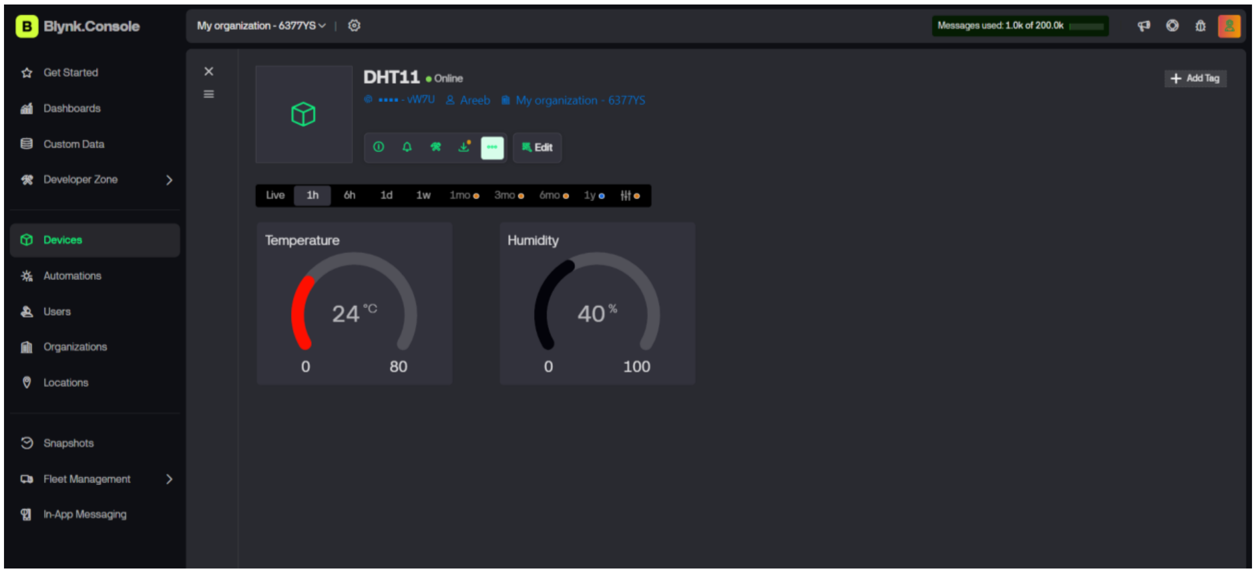
These values are displayed in real time on both the **Blynk web dashboard** and **mobile app**.

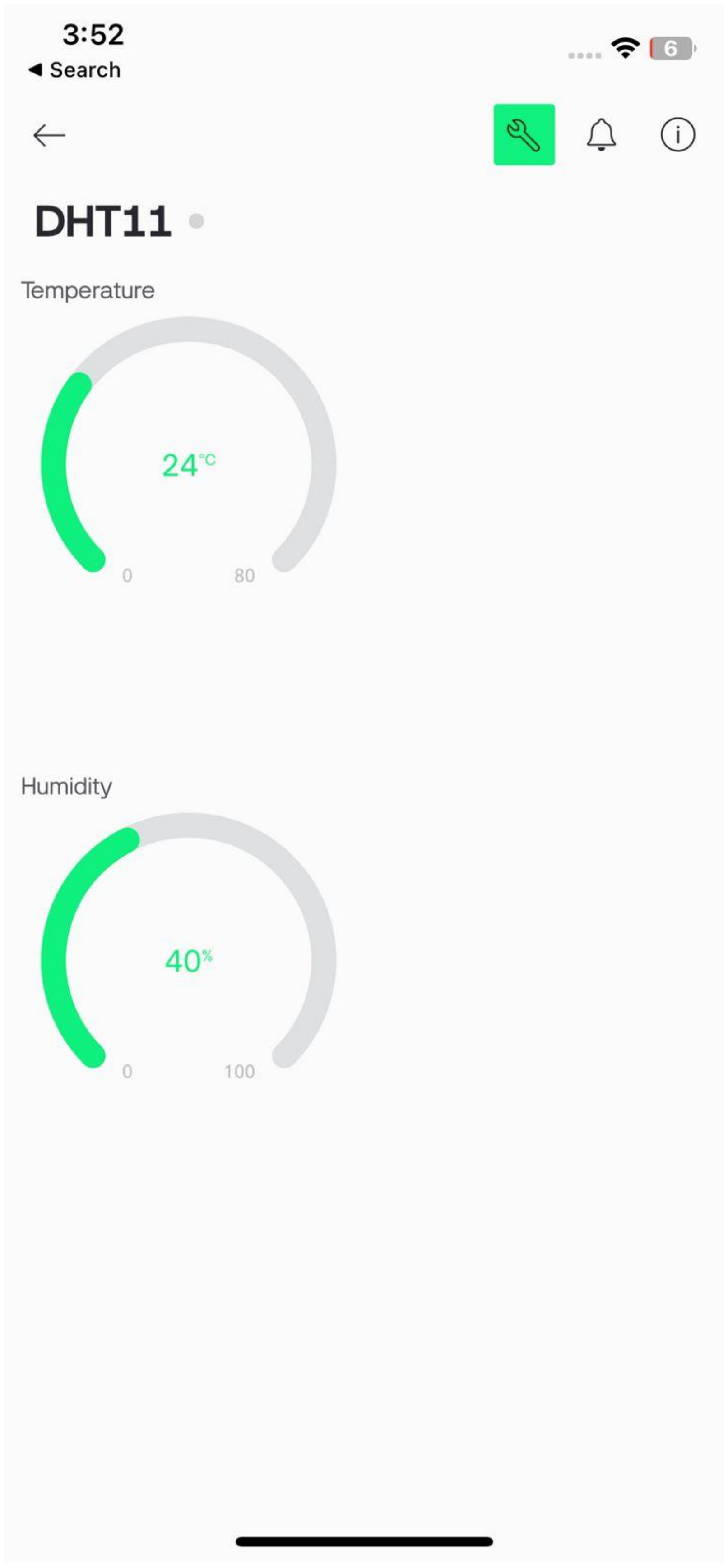
Common Problems and Solutions

- **Device offline:** Check Auth Token
- **No data displayed:** Verify virtual pin mapping
- **Wrong sensor values:** Ensure correct DHT type
- **Frequent disconnections:** Avoid delay() and use BlynkTimer



```
[3535] Connecting to blynk.cloud:80
[3734] Ready (ping: 60ms).
Button pressed: manual DHT read
Temp: 24.00 *C, Hum: 40.00 %
```




3:52
◀ Search
DHT11
Temperature
24°C
0
80
Humidity
40%
0
100

GitHub Link

https://github.com/fatima-Areeb/1246-IoT.git