Name: Amna Sajid

Reg no: 23_ntu_cs_1245

Asssignment 02:

Embedded IOT

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?**

Ans: WebServer server(80); is used to make the ESP32 work as a web server. Port 80 is the default port for web communication. When we type the ESP32 IP address in a browser, it connects using port 80. This helps the ESP32 to display a web page**.**

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

Ans: The statement server.on("/", handleRoot); is used to define what happens when the home page of the website is opened. The symbol / represents the main page. When a user opens this page in the browser, the handleRoot() function is called. This function sends the web page data to the browser.

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

Ans: server.handleClient(); is placed inside the loop() function so that the ESP32 can continuously check for new browser requests. The loop runs again and again, so the server stays active. If this line is removed, the ESP32 will not respond to any client request. As a result, the web page will not load.

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

Ans: In handleRoot(), the statement server.send(200, "text/html", html); is used to send the web page to the browser. The value 200 means the request was successful. "text/html" tells the browser that the data is in HTML format. The variable html contains the web page content.

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

Ans: Displaying last measured sensor values means showing the previously stored DHT readings. Taking a fresh DHT reading inside handleRoot() means the sensor is read again when the page is opened. Fresh readings are more accurate and updated. However, they can make the web page respond slowly.

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?**

Ans: The Blynk Template ID is used to connect the ESP32 project with a specific template in the Blynk Cloud. It tells the cloud which dashboard and widgets belong to this device. It must match the cloud template so that the correct layout and data points are used. If it does not match, the device will not connect properly.

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

Ans: The Blynk Template ID identifies the project template in the Blynk Cloud. The Blynk Auth Token is a unique key that allows a specific device to connect to the cloud. Template ID is same for all devices using that template. Auth Token is different for each device.

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

Ans: DHT22 code gives wrong readings when used with a DHT11 sensor because both sensors work differently. DHT11 has lower accuracy and a smaller temperature range. DHT22 is more accurate and supports wider range values. Using the wrong code causes incorrect data.

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

Ans: Virtual Pins in Blynk are software-based pins used to send data between ESP32 and the Blynk Cloud. They are not real hardware pins. They are preferred because cloud communication does not directly control GPIO pins. Virtual pins make data handling easier and more flexible.

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

Ans: BlynkTimer is used to run tasks at specific time intervals without stopping the program. It allows the ESP32 to do multiple tasks at the same time. Using delay() stops the whole program and blocks communication. That is why BlynkTimer is better for IoT applications.

Question 1 part (B):

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

Your answer should include:
- ESP32 Wi-Fi connection process and IP address assignment
- Web server initialization and request handling
- Button-based sensor reading and OLED update mechanism
- Dynamic HTML webpage generation

- Purpose of meta refresh in the webpage
- Common issues in ESP32 webserver projects and their solutions

The ESP32 first connects to the WiFi network via SSID and password.After a successful connection, the router assigns an IP address to the ESP32.This IP address is used in the browser to open the website. Wi-Fi connection is required for cloud and web communication.The web server runs on port 80 with WebServer (80); begin. It processes browser requests using server.on() and server.handleClient().When a button is pressed, the ESP32 reads the temperature and humidity from the DHT sensor and updates the OLED display.The same sensor values are also prepared for the website.The web page is created with dynamic HTML and sent to the browser.A refresh meta tag is used to automatically refresh the page and update the values.Some of the most common issues include losing Wi-Fi connection, lagging freezes, and unresponsive server.These problems are solved by correct Wi-Fi settings, avoiding lags () and processing requests correctly.

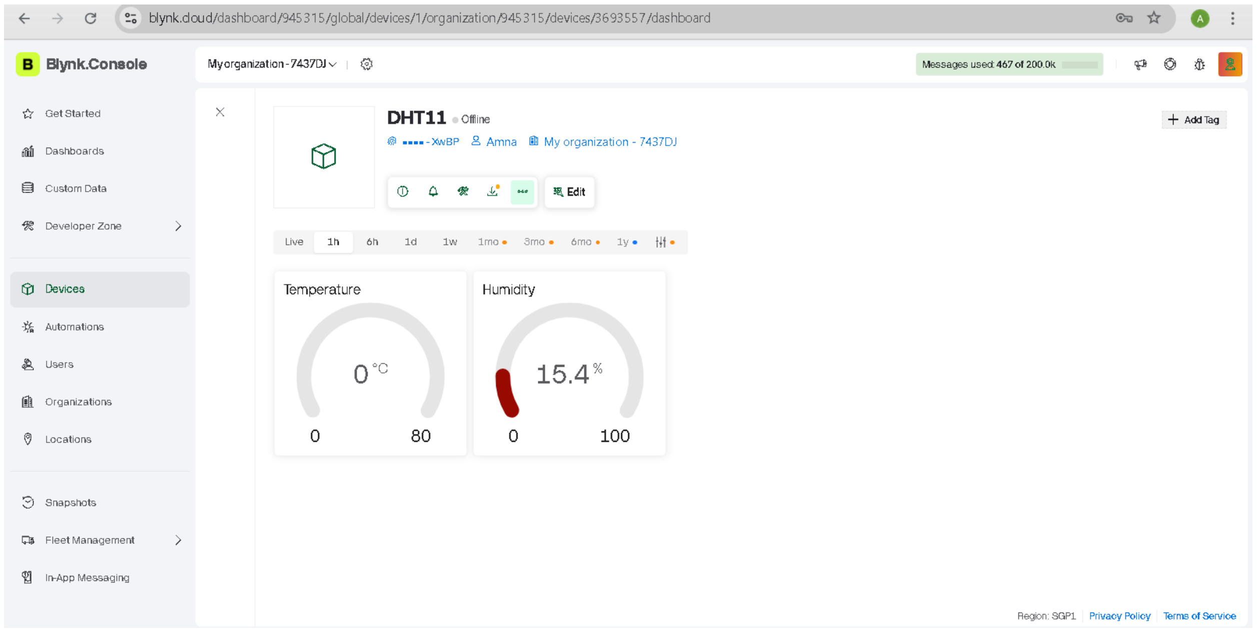
Question 2 part(B):

Explain the complete workflow of interfacing ESP32 with Blynk Cloud to display temperature and humidity values.
Your answer should include:
- Creation of Blynk Template and Datastreams
- Role of Template ID, Template Name, and Auth Token
- Sensor configuration issues (DHT11 vs DHT22)
- Sending data using Blynk.virtualWrite()
- Common problems faced during configuration and their solutions

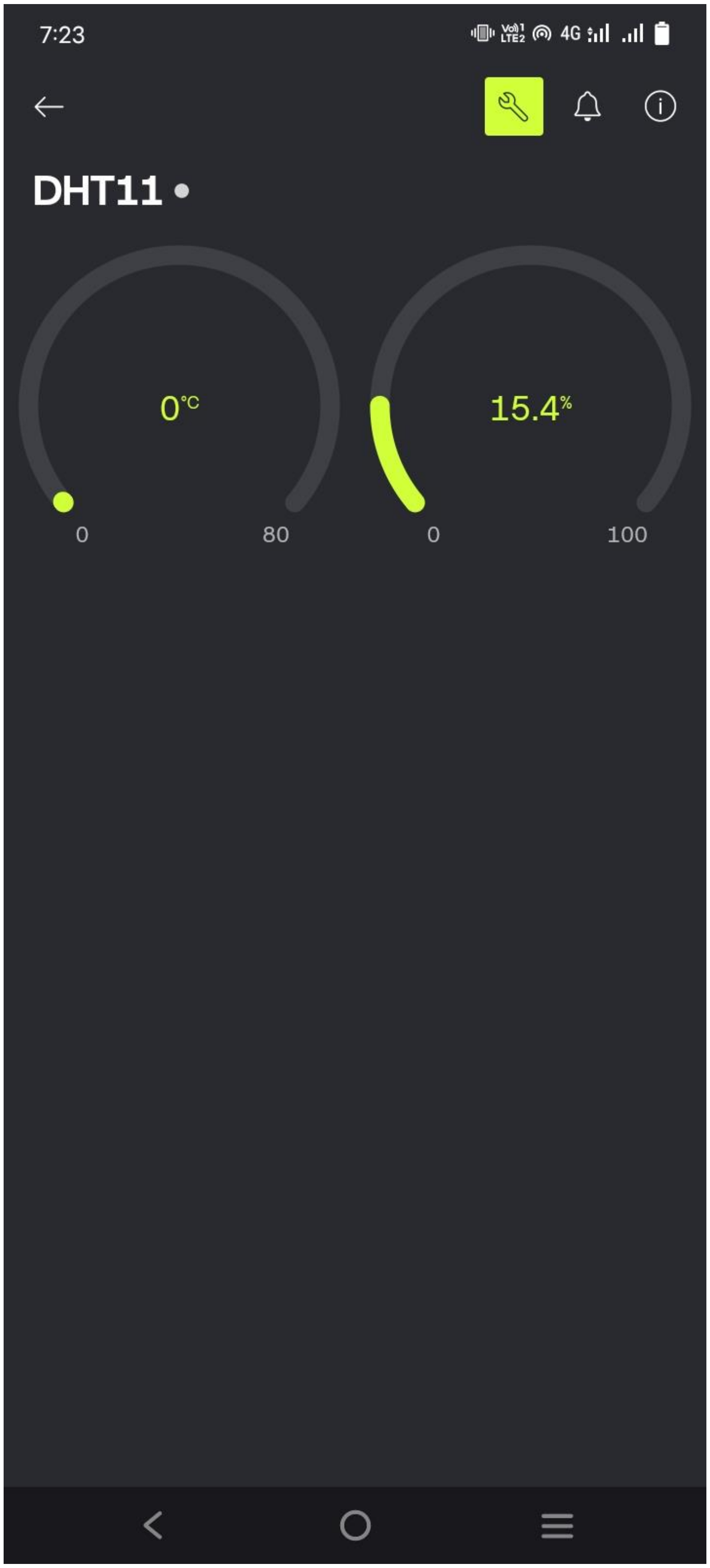
The workflow starts by creating a template in the Blynk Cloud. In the template, datastreams are created for temperature and moisture using virtual legs. These datastreams define how data will be entered and displayed on the Blynk dashboard. contraptions like markers or needles are also linked to these datastreams to show detector values. Each Blynk design uses a Template ID, Template Name, and Auth Token. The Template ID and Template Name connect the ESP32 law to the correct pall template. The Auth Token is a unique key that allows the ESP32 device to authenticate with the Blynk Cloud. However, the device wo n't connect to the pall, If any of these values are incorrect. The DHT detector is connected to the ESP32 to measure temperature and moisture. Using DHT22 law with a DHT11 detector causes wrong readings because both detectors have different delicacy and ranges. Correct detector selection in law is necessary. Detector values are transferred to the pall using Blynk.virtualWrite() with virtual legs. Common problems include wrong Auth Token, mismatched template details, internet issues, and use of detention(). These problems are answered by checking credentials, using BlynkTimer, and proper detector configuration.

SCREENSHOT:

(Cloud):



(mobile):


7:23
DHT11
0°C
0
80
15.4%
0
100