Muhammad Abdullah
23-NTU-CS-1056

National Textile University, Faisalabad

Department of Computer Science

Name:	Muhammad Abdullah
Class:	BSCS-B 5th
Registration No:	23-NTU-CS-1056
Assignment:	Embedded Iot Systems
Submitted To:	Sir Nasir Mehmood
Submission Date:	16-12-2025

Muhammad Abdullah
23-NTU-CS-1056

Question No. 01

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 on the ESP32. Port 80 is the default port used for HTTP web communication.

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

This line defines what happens when the root URL (/) is requested. It calls the handleRoot() function to send the main web page.

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

It continuously checks for incoming client requests.
If removed, the ESP32 will not respond to browser requests.

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

It sends a response to the browser. 200 means request successful, "text/html" defines the content type, and html contains the webpage data.

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

Using last values gives fast response and avoids sensor delay. Taking fresh readings may slow the webpage and cause DHT errors.

Part(B)-Long Question

Muhammad Abdullah
23-NTU-CS-1056

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

Answer:

1. The ESP32 first connects to the Wi-Fi network using the SSID and password provided in the code. After successful connection, the router assigns an IP address to the ESP32, which is used to access the web server from a browser.
2. Once Wi-Fi is connected, the web server is initialized on port 80. The ESP32 continuously listens for incoming HTTP requests from the client. When a user enters the ESP32 IP address in a browser, the request is received and processed by the web server.
3. A push button is used to trigger sensor readings. When the button is pressed, the ESP32 reads temperature and humidity values from the DHT sensor. These values are then displayed on the OLED screen so the user can see the updated readings locally.
4. The web page is generated dynamically using HTML code stored in the program. Sensor values are embedded inside the HTML content and sent to the browser when requested. This allows real-time monitoring through a web interface.
5. Meta refresh is used in the web page so it automatically reloads after a fixed time. This helps in updating the displayed sensor values without manually refreshing the browser.
6. Common issues in ESP32 webserver projects include Wi-Fi disconnection, incorrect IP address, slow page loading, and DHT sensor timing errors. These problems can be solved by using stable Wi-Fi, checking network credentials, avoiding frequent sensor reads, and handling requests properly inside the loop.

Question No.02

Muhammad Abdullah
23-NTU-CS-1056

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?

Template ID links the ESP32 project with the correct Blynk cloud template. It must match to allow proper communication.

2. Differentiate between Blynk Template ID and Blynk Auth Token

Template ID identifies the project template. Auth Token authenticates and connects the specific device to Blynk Cloud.

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 timing and data formats. Using incorrect code causes wrong sensor readings.

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

Virtual Pins are cloud-based pins used for data transfer. They are preferred because they do not depend on ESP32 physical pins.

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

BlynkTimer allows non-blocking code execution. It keeps the ESP32 responsive while sending data to the cloud.

Part(B)-Long Question

Muhammad Abdullah
23-NTU-CS-1056

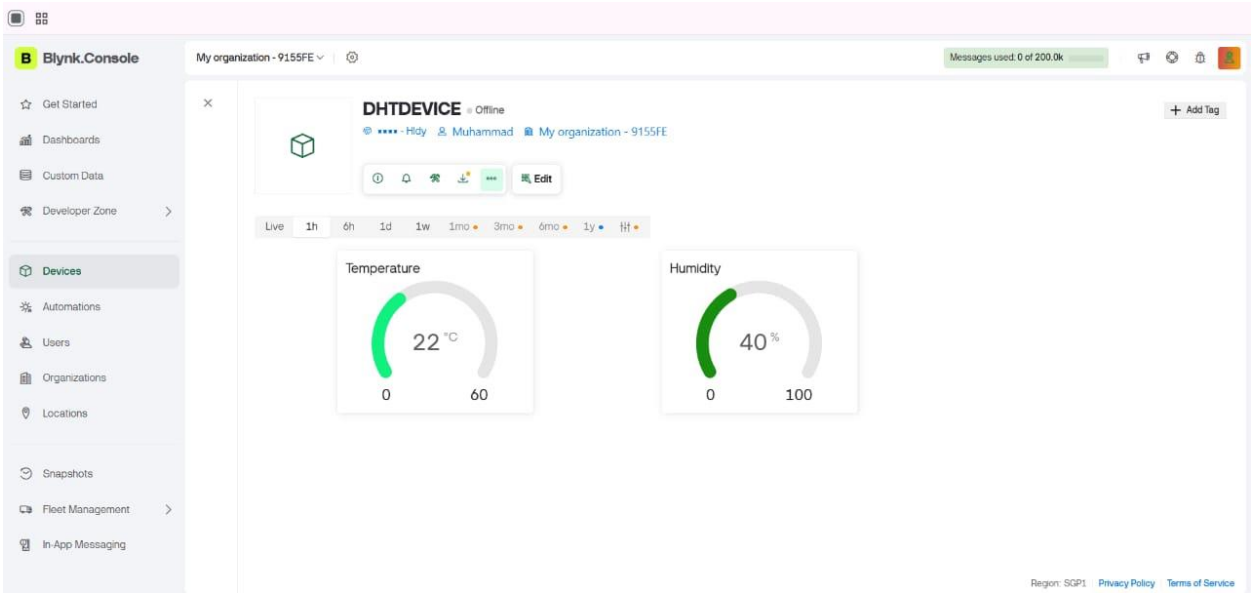
Explain the complete workflow of interfacing ESP32 with Blynk Cloud to display temperature and humidity values.
Your answer should include:
- Creation of Blynk Template and Datastreams
- Role of Template ID, Template Name, and Auth Token
- Sensor configuration issues (DHT11 vs DHT22)
- Sending data using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

Answer:

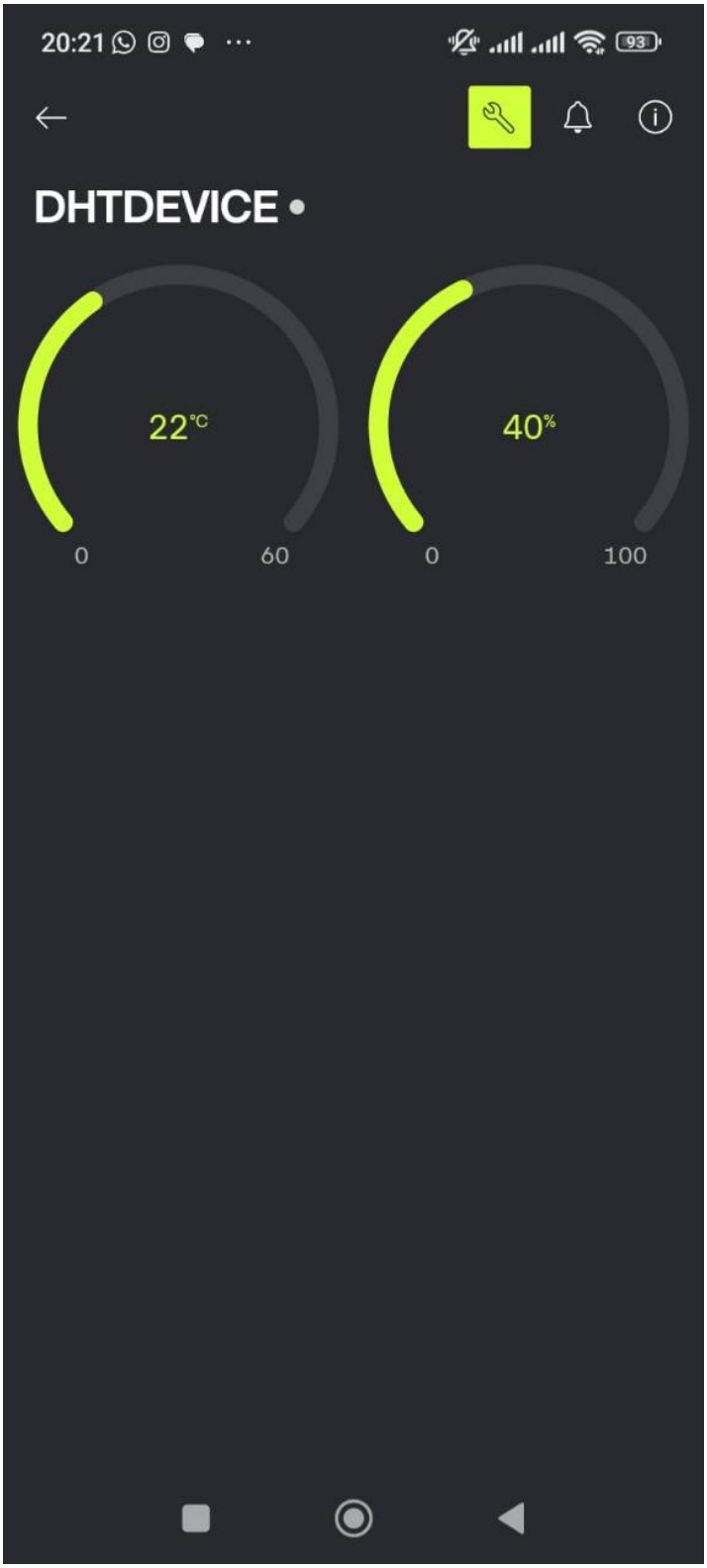
1. First, a Blynk template is created on the Blynk Cloud dashboard. Datastreams are added to the template for temperature and humidity values, and each datastream is assigned a virtual pin.
2. The Template ID and Template Name are used to identify the project, while the Auth Token is used to authenticate the ESP32 device with the Blynk Cloud. These credentials are added to the ESP32 code to establish a secure connection.
3. The ESP32 connects to Wi-Fi and then connects to the Blynk Cloud server. Once connected, the device is ready to send sensor data to the cloud. The correct DHT sensor type must be selected in the code, as using DHT11 and DHT22 interchangeably can cause incorrect readings.
4. Temperature and humidity values are read from the sensor and sent to the Blynk app using Blynk.virtualWrite(). Virtual pins are used because they allow easy and flexible data communication between the device and the cloud.
5. Common problems include wrong template credentials, incorrect virtual pin configuration, sensor mismatch, and Wi-Fi connectivity issues. These problems can be solved by double-checking template details, selecting the correct sensor type, and ensuring a stable internet connection.

Screenshots

Muhammad Abdullah
23-NTU-CS-1056



Muhammad Abdullah
23-NTU-CS-1056



Muhammad Abdullah
23-NTU-CS-1056

Github Repository Link:

https://github.com/abdullah-061317/CS-B-5th-23-NTU-CS-1056-Embedded-IoT