National Textile University, Faisalabad

Department of Computer Science

Name:	Muhammad Umar Mushtaq
Class:	BSCS-B 5th
Registration No:	23-NTU-CS-1077
Assignment:	Iot and Embedded System
Submitted To:	Sir Nasir Mehmood
Submission Date:	17-12-2025

Question-1: ESP32 Webserver

Part-A: Short Questions

1. Purpose of WebServer server(80); and port 80

This line creates a web server on the ESP32.
Port **80** is the normal port used for websites, so the browser can open the ESP32 page easily.

2. Role of server.on("/", handleRoot);

This line tells the ESP32 what to do when someone opens its IP address.
When the home page is opened, the function handleRoot() runs.

3. Why server.handleClient(); is in loop()? What if removed?

It checks again and again if a browser is requesting the page.
If it is removed, the ESP32 will not respond and the webpage will stop working.

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

This sends the webpage to the browser:

- **200** means everything is OK
- **text/html** tells the browser it is a web page
- **html** contains the page content

5. Difference between last value and fresh DHT reading

Showing the last saved values is quick and stable.
Taking a new DHT reading every time can slow the page and sometimes give wrong readings.

Part-B: Long Question

Working of ESP32 webserver-based temperature and humidity monitoring system

1. ESP32 Wi-Fi connection and IP address assignment

First, the ESP32 connects to the Wi-Fi network using the given SSID and password in the code.
When the connection is successful, the router assigns an IP address to the ESP32.
This IP address is important because it is used in the browser to open the ESP32 web page.

2. Web server initialization and request handling

After Wi-Fi is connected, the web server is started on port 80.
The ESP32 keeps listening for incoming requests from a browser.
When a user opens the IP address, the server detects the request and runs the assigned function to respond.

3. Button-based sensor reading and OLED update mechanism

A push button is used to control sensor reading instead of reading continuously.
When the button is pressed, the ESP32 takes temperature and humidity values from the DHT sensor.
These values are then displayed on the OLED screen so the user can see the readings directly on the device.

4. Dynamic HTML webpage generation

The webpage is created using HTML code inside the ESP32 program.
Live sensor values are added to the HTML content before sending it to the browser.
This makes the webpage dynamic because the values change according to the latest readings.

5. Purpose of meta refresh in the webpage

Meta refresh is used to reload the webpage automatically after a fixed time.
This helps in showing updated temperature and humidity values without manually refreshing the page.
It also keeps the webpage simple and user-friendly.

6. Common issues in ESP32 webserver projects and their solutions

Common problems include Wi-Fi disconnection, wrong IP address, and sensor errors.
These issues are usually solved by checking Wi-Fi credentials, using the correct sensor type, and avoiding frequent sensor reads.
Proper delay handling and clean code help improve system stability.

Question-2: Blynk Cloud Interfacing

Part-A: Short Questions

1. Role of Blynk Template ID

Template ID connects ESP32 code with the correct Blynk project.
If it does not match, the device will not connect.

2. Difference between Template ID and Auth Token

- **Template ID:** Identifies the project
- **Auth Token:** Allows the ESP32 to send data to Blynk

3. Why DHT22 code gives wrong result with DHT11

DHT11 and DHT22 work differently.
Using wrong sensor code causes incorrect temperature and humidity values.

4. What are Virtual Pins in Blynk?

Virtual pins are online pins used to send data.
They are better than real pins because they work through the internet.

5. Why use BlynkTimer instead of delay()

BlynkTimer keeps the system running smoothly.
delay() stops everything and can break cloud connection.

Part-B: Long Question

Complete workflow of interfacing ESP32 with Blynk Cloud

1. Creation of Blynk Template and Datastreams

First, a template is created in the Blynk Cloud dashboard.
Datastreams for temperature and humidity are added and linked with virtual pins.
These datastreams define how data will be shown on the Blynk app.

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

The Template ID connects the ESP32 firmware with the correct cloud project.
Template Name helps in identifying the project easily on the dashboard.
The Auth Token is used to authenticate the ESP32 and allow secure data transfer.

3. Sensor configuration issues (DHT11 vs DHT22)

DHT11 and DHT22 sensors work differently and have different data formats.
Using the wrong sensor definition in the code leads to incorrect readings.
Selecting the correct sensor type ensures stable and accurate values.

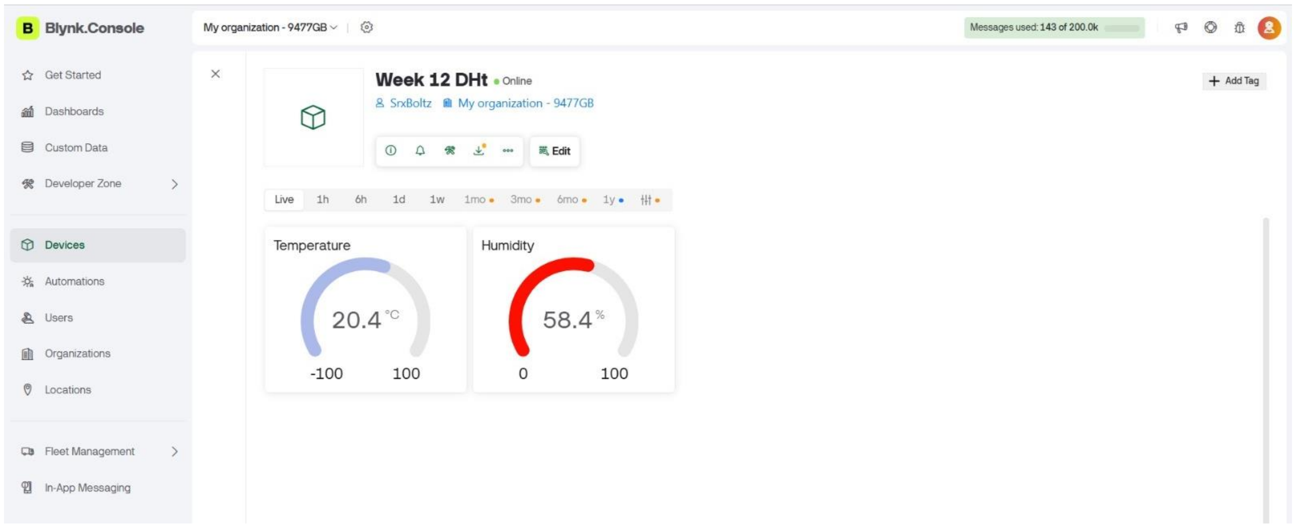
4. Sending data using Blynk.virtualWrite()

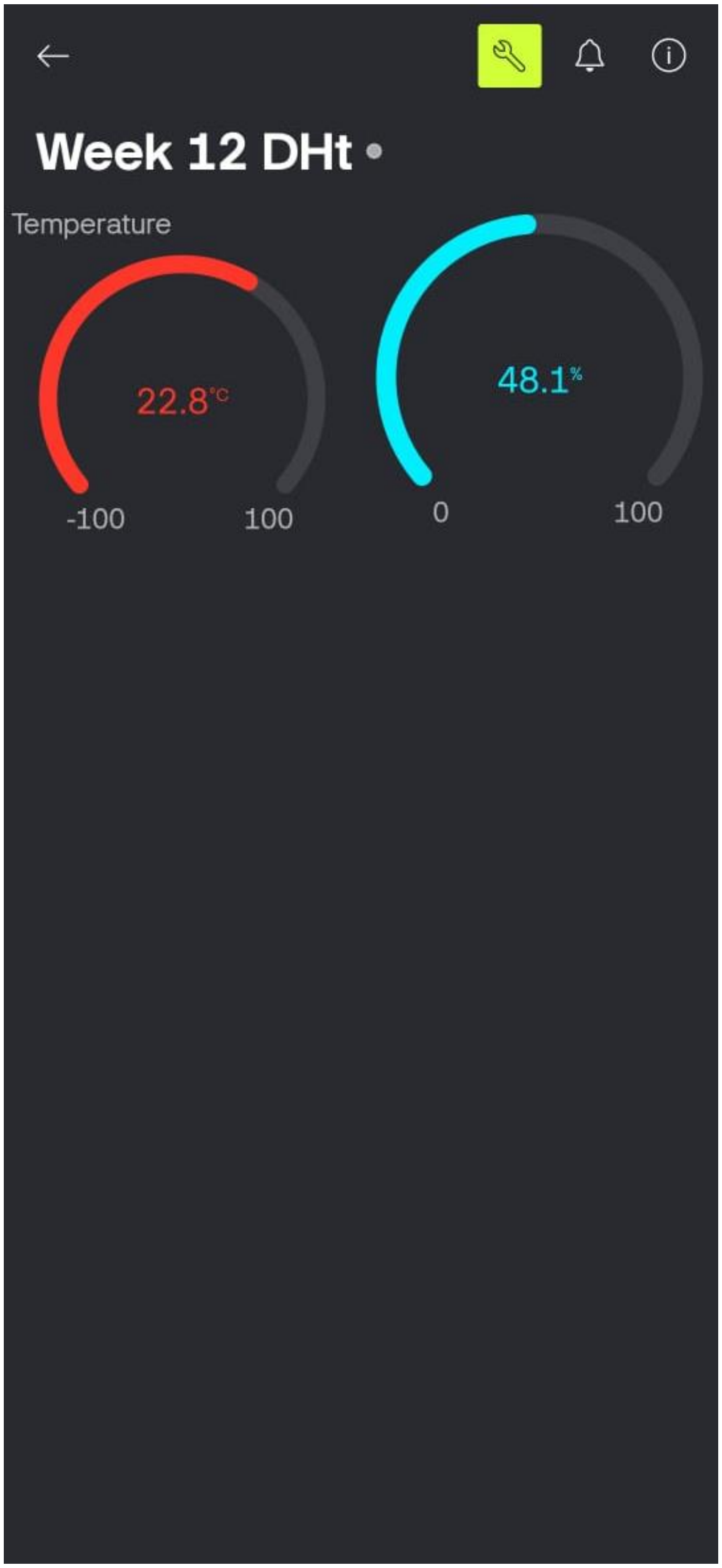
After reading the sensor, the ESP32 sends data to Blynk Cloud using virtual pins.
The Blynk.virtualWrite() function updates the values on widgets like labels and gauges.
This allows real-time monitoring from anywhere.

5. Common problems faced and their solutions

Common issues include device going offline, data not updating, or frequent disconnections.
These problems are usually caused by wrong tokens, incorrect virtual pins, or using delay().
Using BlynkTimer and proper configuration helps keep the system stable.

ScreenShots:


Blynk.Console
My organization - 9477GB
Messages used: 143 of 200.0k
Get Started
Dashboards
Custom Data
Developer Zone
Devices
Automations
Users
Organizations
Locations
Fleet Management
In-App Messaging
Week 12 DHt
Online
SrxBoltz
My organization - 9477GB
Edit
Add Tag
Live
1h
6h
1d
1w
1mo
3mo
6mo
1y
Temperature
20.4 °C
-100
100
Humidity
58.4 %
0
100


Week 12 DHt
Temperature
22.8°C
-100
100
48.1%
0
100