National Textile University, Faisalabad

Department of Computer Science

Name:	Essha Imran
Class:	BSCS-A
Registration No:	23-NTU-CS-1022
Assignment:	2

PART A – Short Questions

1. What is the purpose of WebServer server(80); and what does port 80 represent?

It creates the object of web server which is used for http requests. The port 80 is used for communicating as a web and it is default for http web communication. Whenever, we write the ip-address in a browser without mentioning port, it always uses the port 80 for web communication.

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

The "/" in server.on("/" , handleroot()) is used to tell the esp-32 that what actions should be taken when the client request the root URL "/". And handleroot() function is used to generate and send the web page when someone opens the IP address of esp-32 in a browser.

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

The server.handleClient() is used for continuously checking the incoming requests which is made from clients (browser). We keep it in a loop so the esp-32 respond immediately. If we do not keep it in a loop, the server will not respond to client request.

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

The server.send(200, "text/html", html) sends the response of web page to the client. The "200" shows the status of http is OK. The "text/html" shows the type of content so the browser understands it. The "html" shows that the browser actually receives the actual html content.

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

The display of last measured values provides

- The response faster.
- It is more stable than others.
- It also prevents us from sensor timing issues.

Taking fresh DHT reading inside handleroot() function provides

- The slower response.
- It does not provide stability.
- Provides incorrect readings or causes delays because sensor needs time in taking readings.

PART B – Long Questions

1. **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

WiFi Connection and IP Assignment

When powered on, the ESP32 connects to "Wokwi-GUEST" using WiFi.begin().

System wait until connection is built. Then the router assigns an IP address via DHCP. This IP is displayed on the OLED and Serial monitor. Without proper IP address the devices cannot communicate with each other.

Web Server Initialization

The server.on(“/”, handleRoot) command registers the handleRoot function for the root path. When server.begin() is called, the server starts listening.

In the main loop, server.handleClient() constantly watches for browser requests. When someone visits the IP address, it triggers handleRoot() to send the webpage.

Button and Sensor Reading

The button connected to GPIO 5 uses INPUT_PULLUP mode. When button is pressed, the code waits 50ms for debouncing. Then readDHTValues() reads the temperature and humidity, updating lastTemp and lastHum. The showOnOLED() function displays these new values on screen.

Dynamic HTML Webpage Generation

The handleRoot() function builds an HTML string that includes a title, responsive design, and conditional display.

If sensor data exists, it shows temperature and humidity to 1 decimal place. If no data yet, it prompts the user to press the button. The server sends this HTML to the browser for display.

Meta Refresh Purpose

The <meta http-equiv='refresh' content='5'> tag automatically reloads the webpage after every 5 seconds. This allows users see updated readings without manually refreshing. But right now in code readdhtvalues() function is commented so we can read data only when button is pressed.

Common Issues and Fixes

- **Issue :** Esp32 not connected to WiFi .
- **Fix :** Check SSID/password, Check WiFi module and make sure it is enable. Sometimes it properly work if we are using our own data by giving hotspot.

- **Issue :** DHT shows NAN.
- **Fix :** Check wiring, ensure proper power supply

- **Issue :** Button is not working.
- **Fix :** Confirm INPUT_PULLUP is set, Check if proper debouncing is done, check wiring to GND

- **Issue :** Web page is loading slowly.
- **Fix:** Avoid taking reading continuously, Check if server.handleClient() is present in the loop.

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?**

   Template ID links ESP32 to Blynk cloud Project. If ID does not match the device will not connect properly and our app will not show display.

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

   Template ID is same for all the devices we are using in our project. Auth Token is unique for every device.

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

DHT22 can measure in steps of 0.1°C but DHT11 only does 1°C steps. If we connect DHT11 but code has DHT22, it will produce incorrect readings.

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

Virtual Pins are basically software pins on the cloud, not the actual GPIO pins. They have range from V0 to V2555. They are preferred over physical GPIO pins for cloud communication because we never run out of them.

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

If we use delay() in our code, the whole program just stops and waits for that time. So, we use BlynkTimer so our Timer can schedule things while the program keeps running in background.

PART-B - Long Question

1. 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

Creating the Blynk Template and Datastreams

When we start a new Blynk project, we first need to create a template on the Blynk. Then name it, like I gave "dhot by aleeha" Inside that template, create datastream. I created one datastream for temperature and another for humidity. I linked each data stream to a virtual pin, so temperature goes to V0 and humidity goes to V1. Then I added widgets for temperature and humidity. Those gauge widgets display reading on web and mobile dashboard.

Role of Template ID, Template Name, and Auth Token

We include Template ID in our code, so the Blynk server knows which project template we are trying to connect. The Template Name is just a label we give it, so we remember project by name. The Auth Token is unique to each device. It proves Blynk that our device has permission to use that template. So, our ESP32 sends the Template ID and Token to Blynk, the server checks them and if we are authorized. After confirmation we are connected to that project.

Sensor configuration issues (DHT11 vs DHT22)

The code we have is specifically written for DHT22 sensors. If we connect to DHT11 and don't change the code, it will give incorrect readings. The problem is that these two sensors talk to the microcontroller using different electrical signals and timing. DHT22 gives us more precise readings for eg 25.3°C but DHT11 only gives us 25°C.

Sending data using Blynk.virtualWrite()

In our readAndDisplayAndSend() function, we first read the current temperature and humidity from the DHT sensor.

Then we use Blynk.virtualWrite(V0, t) to push that temperature value to the cloud on virtual pin V0, and Blynk.virtualWrite(V1, h) to send humidity to V1. Once the data reaches Blynk's servers, it gets sent to our mobile app and we see the live readings update.

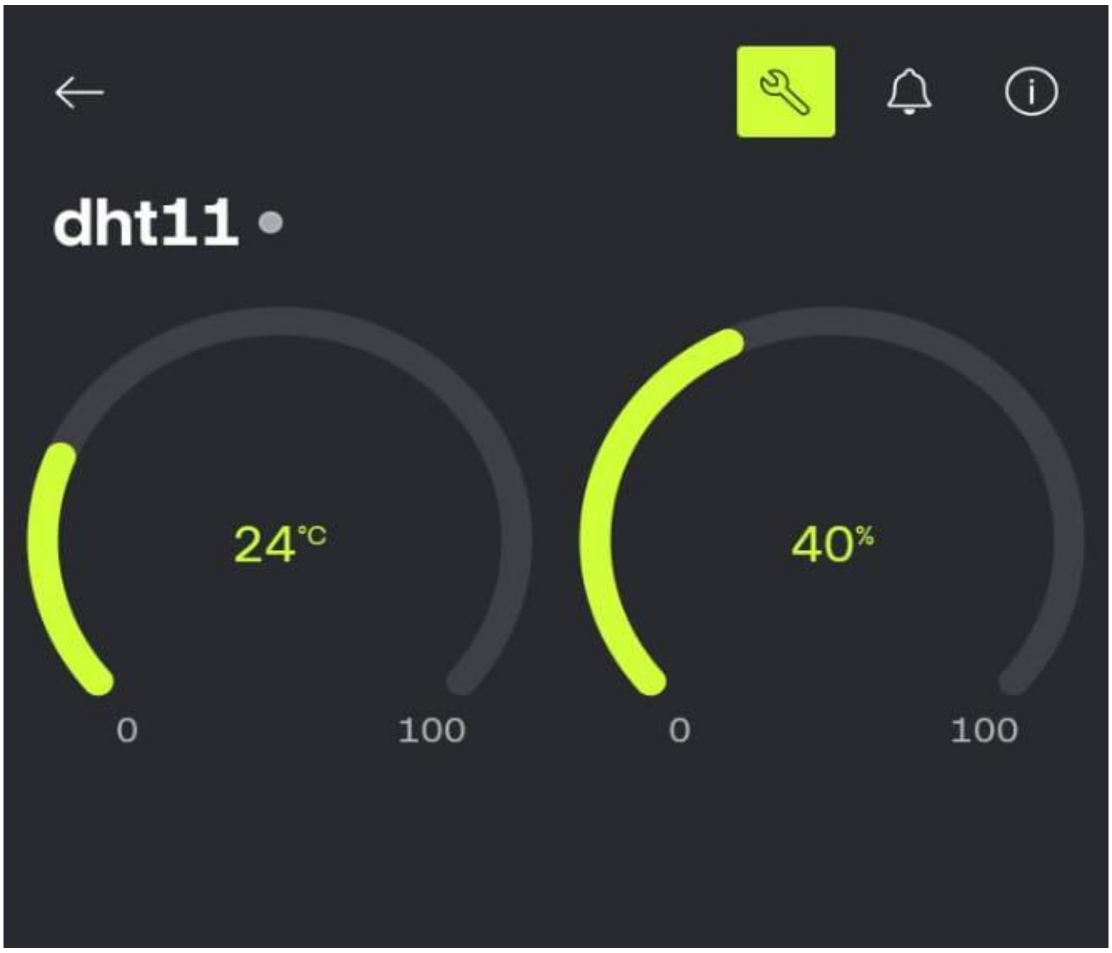
Common problems faced during configuration and their solutions

- **Issue:** Can't connect to WiFi at all
  Fix: We need to check our SSID and password are correct.

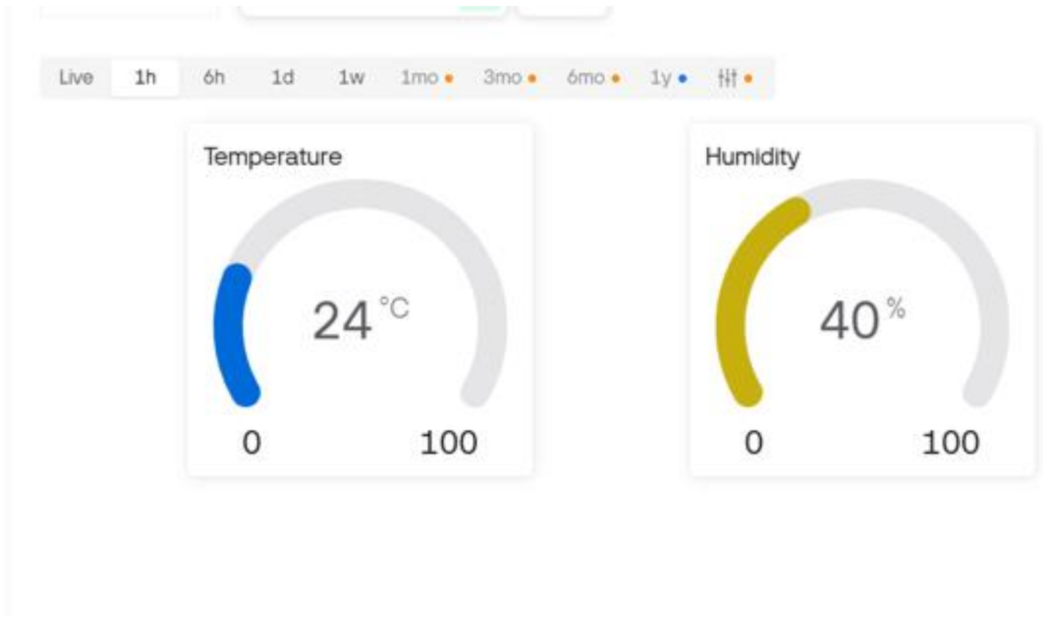
- **Issue:** Data is not updating on the phone app.
  Fix: We need to verify that Blynk.run() is actually in our main loop function.

- **Issue:** Button press is not doing anything.
  Fix: We might need a debounce delay, or we should verify the button is wired correctly with one side going to GPIO5 and the other side to ground.

Blynk Mobile app dashboard



Blynk Cloud web


Live
1h
6h
1d
1w
1mo
3mo
6mo
1y
Temperature
24 °C
0
100
Humidity
40 %
0
100