

# National Textile University, Faisalabad



## Department of Computer Science

<b>Name:</b>	Arsh-E-Noor
<b>Class:</b>	BSCS-5th-A
<b>Registration No:</b>	23-NTU-CS-1018
<b>Assignment</b>	Homework1-After Mids
<b>Course Name:</b>	Embedded IOT
<b>Submitted To:</b>	Sir Nasir
<b>Date:</b>	17 <sup>th</sup> December , 2025

# Question – 01

## **ESP32 WEB SERVER**

### **Part A: Short Questions**

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

**Answer:**

“`WebServer server(80);`” creates an HTTP web server on the ESP32 that listens for incoming client requests.

Port **80** is the default port for HTTP communication , which allows users to access the ESP32 web server directly through a standard web browser without requiring any additional port number.

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

**Answer:**

This statement defines a route for the web server. When a user accesses the root URL “(/)” of the ESP32s IP address ,the `handleRoot()` function is executed. This function is responsible for generating and sending the HTML webpage that displays the temperature and humidity readings.

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

**Answer:**

“ `server.handleClient()` ” continuously checks for and processes incoming HTTP requests from clients. It is placed inside the `loop` function to ensure the web server remains responsive at all times.

If this function is removed , the ESP32 will not respond to browser requests , and the webpage will fail to load.

**4. In `handleRoot()`, explain the statement:**

```
server.send(200, "text/html", html);
```

**Answer:**

This statement sends an HTTP response from the ESP32 to the client.

- 200 indicates a successful HTTP request
- " text/html " specifies that the content being sent is an HTML webpage
- html contains the actual webpage content

As a result , the browser receives and displays the generated HTML page.

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

**Answer:**

Displaying last measured values uses previously stored sensor data , which improves webpage loading speed and reduces unnecessary sensor readings. Taking a fresh DHT reading inside `handleRoot()` provides realtime data on every webpage refresh but may cause delays and increase sensor workload. Using last measured values is generally more efficient.

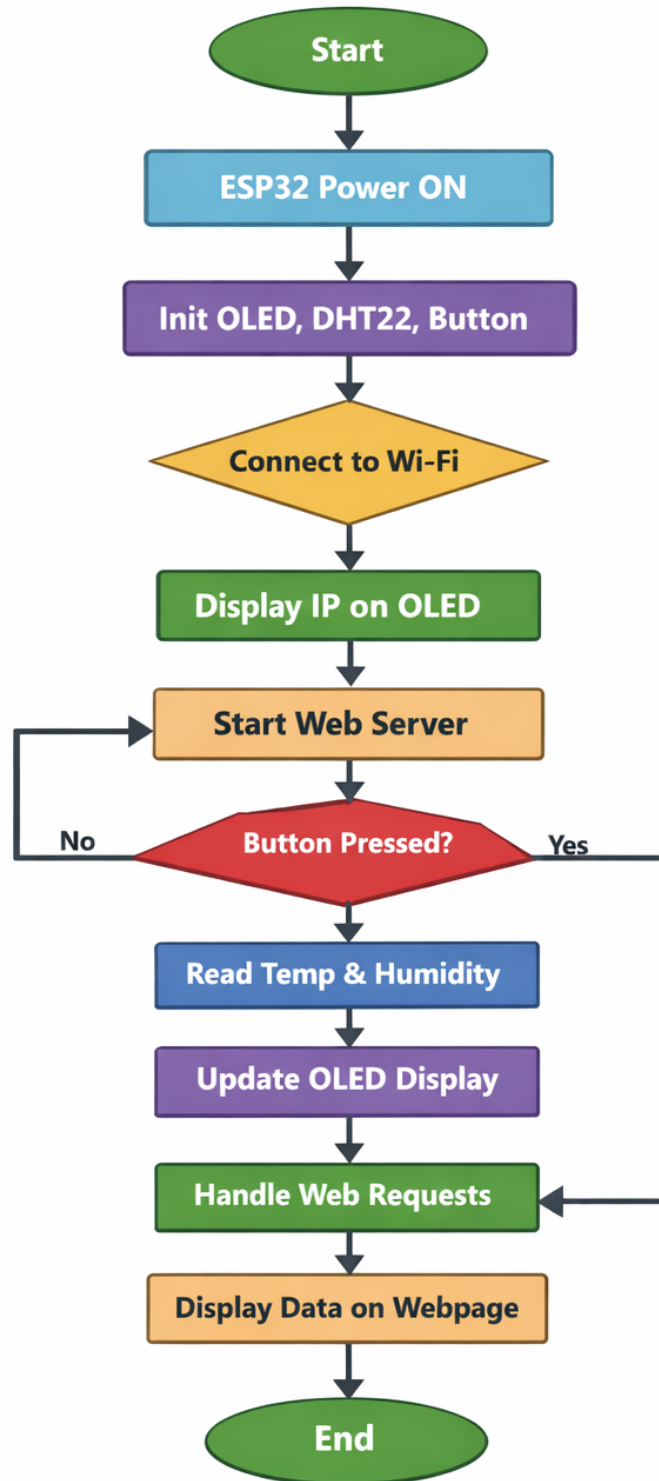
---

## **Part B: Long Question**

### **Complete Working of the ESP32 Webserver-Based Temperature and Humidity Monitoring System**

This system is designed to monitor temperature and humidity using a DHT22 sensor and display the data on both an OLED screen and a webbased interface using an ESP32 microcontroller.

## Flowchart:



*Figure 1: Flowchart of ESP32 webserver-based monitoring system*

## **ESP32 Wi-Fi Connection and IP Address Assignment**

When the system starts , the ESP32 connects to the specified Wi-Fi network using the provided SSID and password. Once the connection is established , the router assigns a unique IP address to the ESP32. This IP address is used to access the web server through a browser.

## **Web Server Initialization and Request Handling**

An HTTP web server is created on the ESP32 using the web server library on port 80 . The server listens for incoming client requests. When a request is made to the root URL , the server calls the `handleRoot()` function , which prepares and sends the HTML webpage containing the sensor readings.

## **Button Based Sensor Reading and OLED Update Mechanism**

A push button is connected to the ESP32 and configured using the internal pullup resistor. When the button is pressed , the ESP32 reads temperature and humidity data from the DHT22 sensor. These values are stored in variables and simultaneously displayed on the OLED screen. Button debounce logic ensures reliable and accurate detection of button presses.

## **Dynamic HTML Webpage Generation**

The webpage is dynamically generated within the `handleRoot()` function using HTML code. The page displays the latest available temperature and humidity values. If no valid sensor data is available , an appropriate message is shown to the user.

## **Purpose of Meta Refresh in the Webpage**

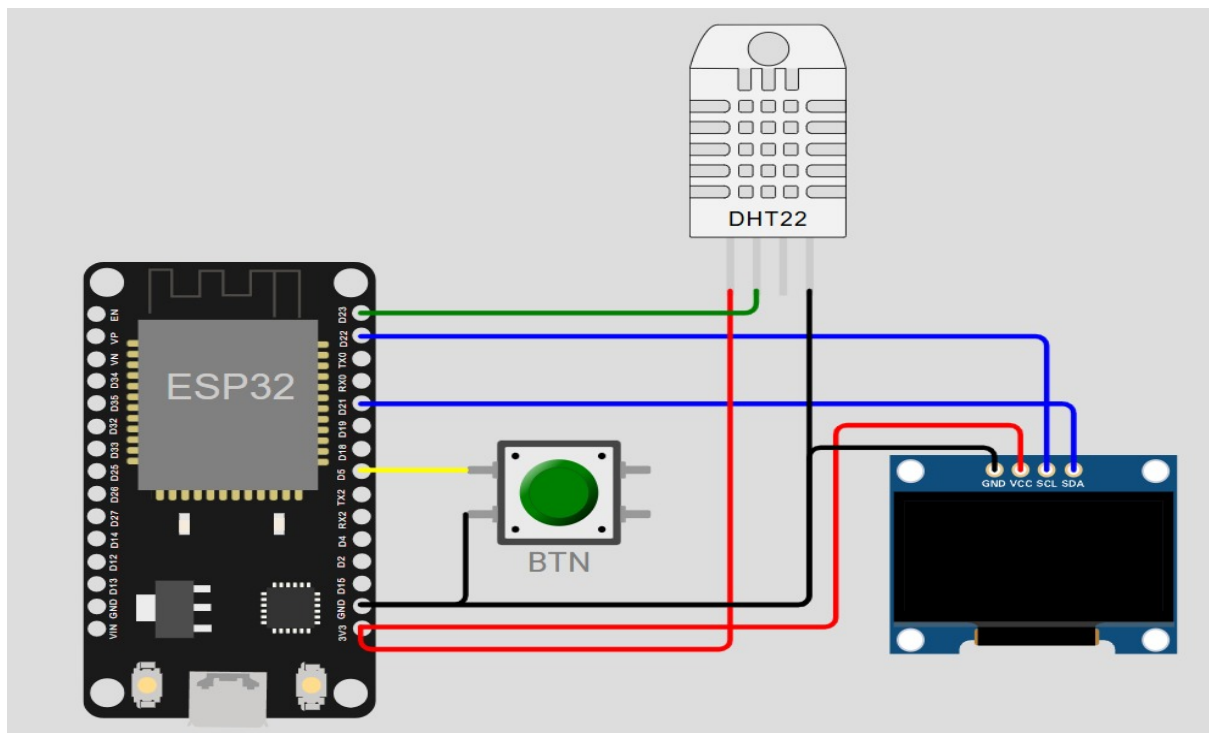
The HTML meta refresh tag is used to automatically reload the webpage at regular intervals (every 5 seconds). This ensures that updated sensor readings are displayed without requiring the user to manually refresh the page.

## Common Issues in ESP32 Webserver Projects and Their Solutions

Common issues include ,

- Wi-Fi connection failures
- Unresponsive web pages
- Invalid sensor readings
- OLED display errors.

These issues can be resolved by verifying Wi-Fi credentials , ensuring `server.handleClient()` is executed in the loop , checking sensor wiring and timing, and confirming correct OLED I2C connections and addresses.



*Figure 2: Wokwi simulation of ESP32 with DHT22, OLED, and push button*

# Question – 02

## **Blynk Cloud Interfacing**

### **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?**

**Answer:**

The Blynk Template ID uniquely identifies a specific device template created on the Blynk Cloud. It links the ESP32 firmware with the correct cloud dashboard , widgets, and datastreams.

The Template ID must match the cloud template to ensure proper communication , otherwise, the device will fail to connect or send data to the intended Blynk project.

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

**Answer:**

The Blynk Template ID identifies the overall project structure and dashboard configuration on the Blynk Cloud.

The Blynk Auth Token is a unique security key that authenticates a specific device and allows it to communicate with the Blynk server.

In simple terms, the Template ID defines what the project is, while the Auth Token defines which device is allowed to connect.

- 3. Why does using DHT22 code with a DHT11 sensor produce incorrect readings?**

**Answer:**

Using DHT22 code with a DHT11 sensor causes incorrect readings because both sensors have different communication timing and data formats.

One key difference is that DHT22 provides higher accuracy and a wider temperature range , while DHT11 has lower accuracy and limited range.

Therefore , the sensor type must match the code configuration.

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

**Answer:**

Virtual Pins are logical pins in Blynk that allow data exchange between the ESP32 and the Blynk Cloud without directly using physical GPIO pins. They are preferred because they enable flexible data handling allow multiple widgets to be connected easily, and simplify cloud-based communication without hardware pin limitations.

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

**Answer:**

BlynkTimer allows tasks to run at specific time intervals without blocking the main program execution.

Unlike delay() , it keeps the ESP32 responsive , ensures smooth Blynk communication, and allows multiple operations (such as sensor reading and cloud updates) to run simultaneously.

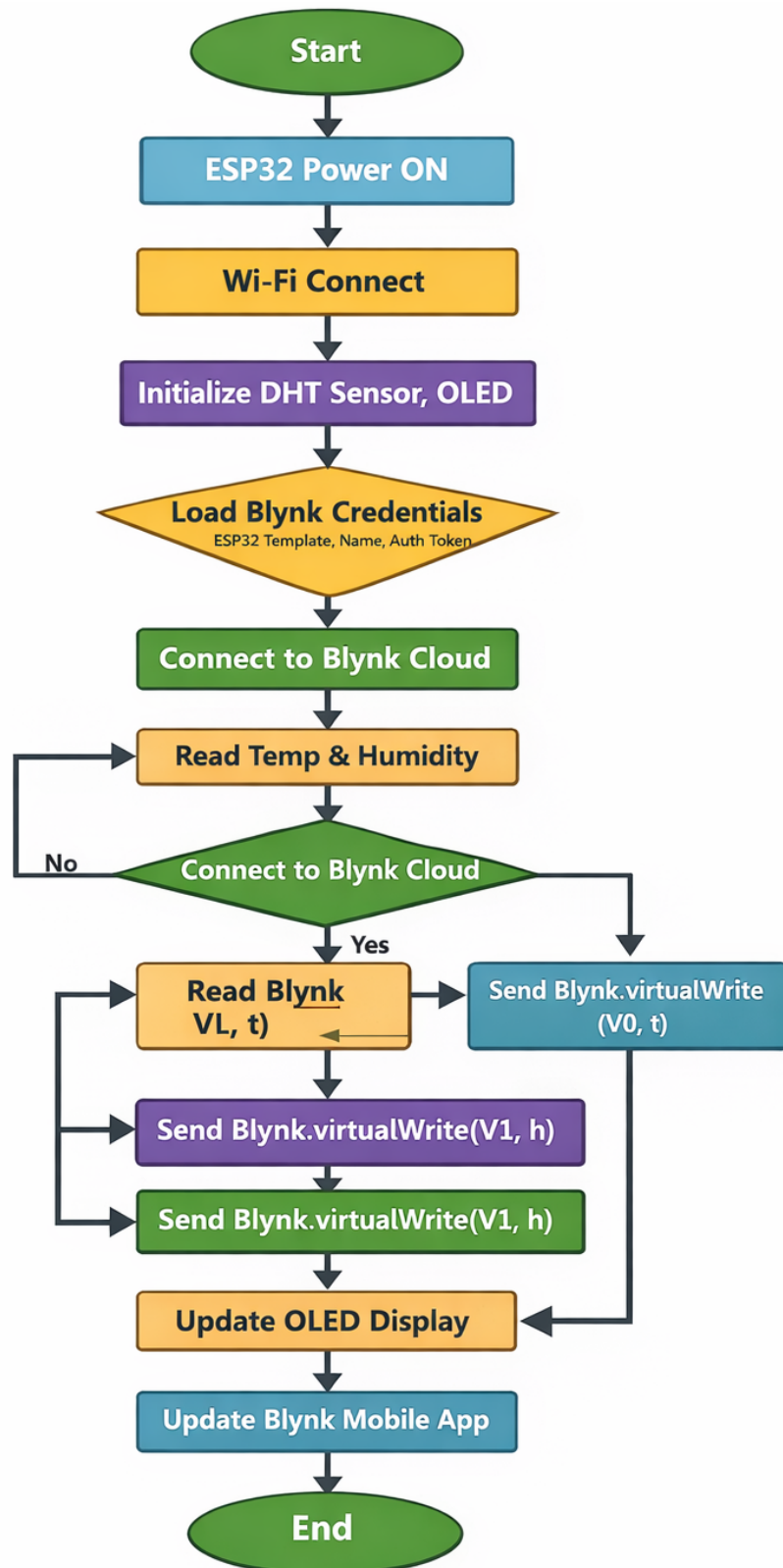
---

## **Part B: Long Question**

### **Complete Workflow of Interfacing ESP32 with Blynk Cloud**

This project demonstrates the integration of an ESP32 with the Blynk Cloud to monitor temperature and humidity values using a DHT sensor and display the data on both an OLED screen and the Blynk mobile application.

## Flowchart:



*Figure : Flowchart of ESP32 interfacing with Blynk Cloud*

### **Creation of Blynk Template and Datastreams**

A device template is first created on the Blynk Cloud platform. Datastreams are configured using Virtual Pins to receive temperature and humidity values. These datastreams are later linked to widgets in the Blynk mobile dashboard.

### **Role of Template ID, Template Name, and Auth Token**

The **Template ID** connects the ESP32 firmware to the correct Blynk Cloud project.

The **Template Name** provides a human-readable identification for the project.

The **Auth Token** authenticates the ESP32 device, ensuring secure communication between the hardware and the Blynk Cloud.

### **Sensor Configuration Issues (DHT11 vs DHT22)**

Correct sensor configuration is essential for accurate readings.

Using DHT22 code with a DHT11 sensor can result in invalid data due to differences in timing , resolution , and operating ranges. Selecting the correct sensor type in the code ensures reliable temperature and humidity measurements.

### **Sending Data Using Blynk.virtualWrite()**

Sensor readings are sent to the Blynk Cloud using the Blynk.virtualWrite() function.

Temperature and humidity values are mapped to specific virtual Pins , allowing them to be displayed on the Blynk dashboard widgets in real time.

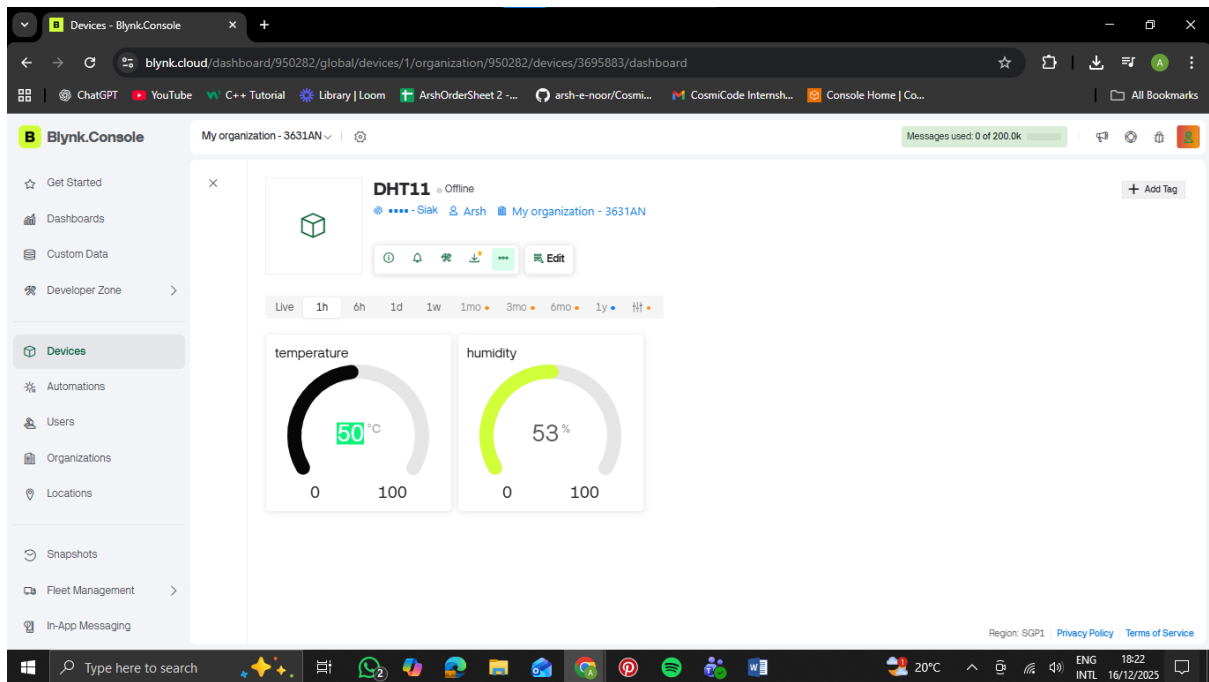
### **Common Problems Faced During Configuration and Their Solutions**

Common issues include

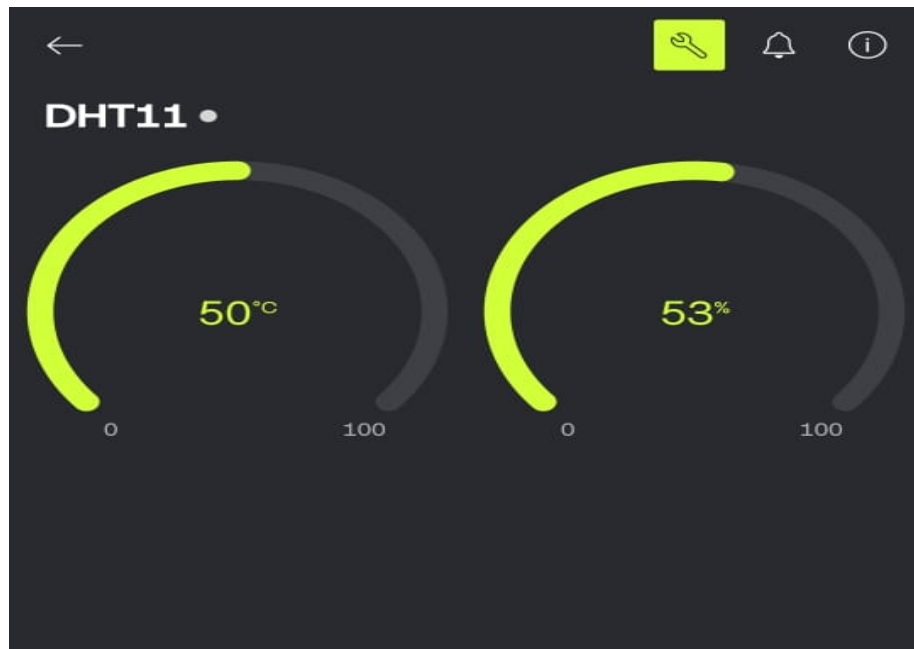
- incorrect Template ID or Auth Token
- Wi-Fi connectivity failures
- Wrong sensor selection
- Unresponsive dashboards

These problems can be resolved by verifying Blynk credentials , ensuring stable Wi-Fi connectivity , selecting the correct sensor type and properly mapping virtual Pins to datastreams.

## Screenshots:



*Figure :Blynk Cloud (web)*



*Figure :Blynk mobile app*