National Textile University, Faisalabad

Department of Computer Science

Name:	Ayesha
Class:	CS 5th (A)
Registration No:	23-NTU-CS-1020
Assignment No:	02
Course Name:	Embedded IoT
Submitted To:	*Sir Nasir*

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?

Webserver server:

It generates a web server that pays attention to incoming HTTP requests.

Port 80:

Port 80 displays the default port utilized for HTTP traffic, so clients can access the server through a web browser without specifying a port number.

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

server.on("/", handleRoot); tells what the web server should do when a client approaches the root URL /.

It tells the server to call the handleRoot function and transfer its response when someone opens the main page of the server.

Example: entering the device IP in a browser.

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

It is put inside the loop () function so the microcontroller can continuously examine and proceed with incoming client requests.

If it is detached, the server will not reply to any HTTP requests. The web page will not be loaded because incoming connections are never controlled.

4-In handle Root (), explain the statement: server.send(200, "text/html", html).

server.send(200, "text/html", html) transmit a reply from the server to the client.

- 200: The request was successful.
- "text/html": Tells the browser what type of content to await
- html: the genuine HTML data forward to the browser

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

Aspect	Last measure Value	Fresh Reading
Response Time	Fast	Slow
User Experience	Fast Page Load	Slow Page Load
Concurrent Users	Handles many	One at a time
Server Load	Light	Heavy

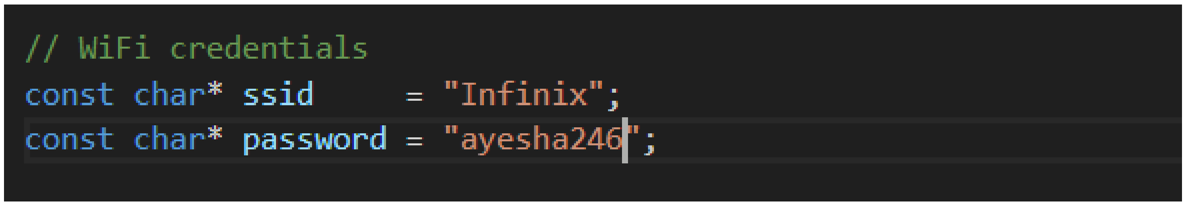
Long Question:

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

Answer:

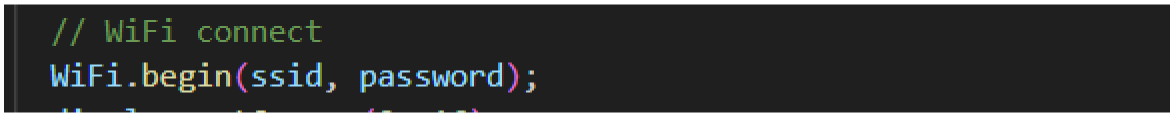
ESP32 Wi-Fi connection process and IP address assignment

In this system, esp32 connects to a WiFi to act as a webserver to display temperature and humidity reading. To connect with WiFi first we assign the SSID and password as follows:



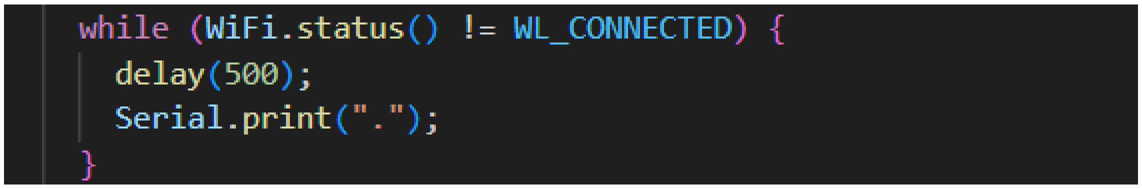
```
// WiFi credentials
const char* ssid     = "Infinix";
const char* password = "ayesha246";
```

In the setup function we initiated the Wi-Fi connection by calling the wifi.begin function.

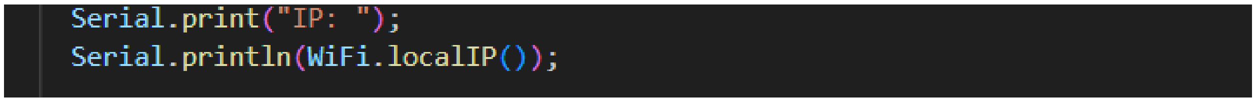


```
// WiFi connect
WiFi.begin(ssid, password);
```

Some parts of code continuously checks the status whether it is connected or not



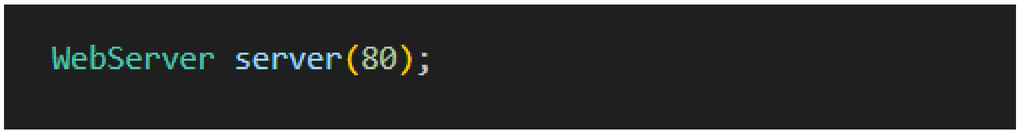
```
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
```


```
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());
```

Once esp32 is connected, router automatically assigns it an ip address.

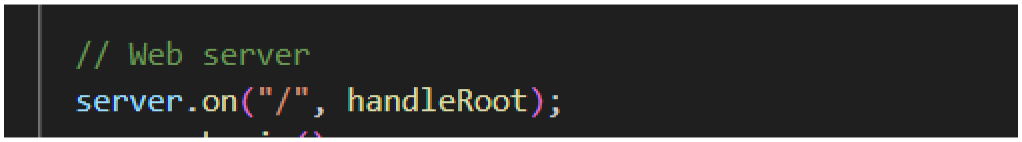
Web server initialization and request handling

After esp32 connect to Wi-Fi and give us an IP address. In the next step it starts the web browser to respond to the client requests.

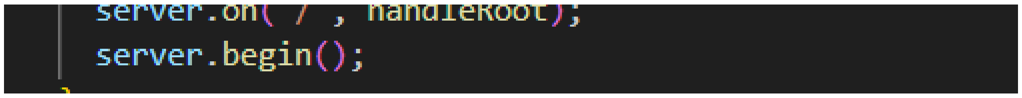

```
WebServer server(80);
```

We use port 80 as default port for HTTP. It allows us to access web servers by entering the Ip.

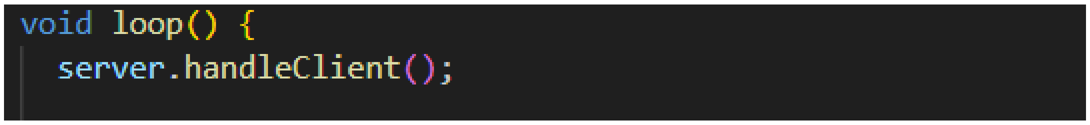
In setup () function we define the path as

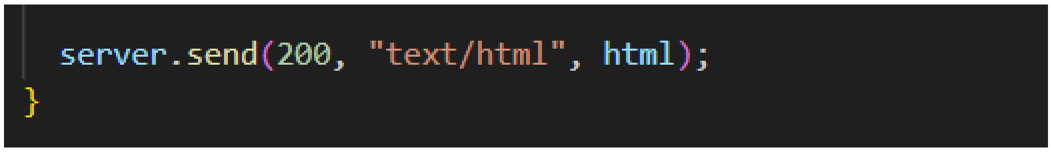

```
  // Web server
  server.on("/", handleRoot);
```

To activate the web server, we enter the command:


```
  server.on("/", handleRoot);
  server.begin();
```

We write the function below to handle the client requests

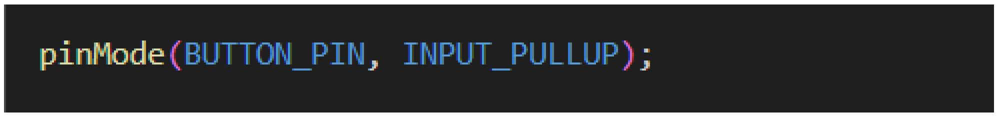

```
void loop() {
  server.handleClient();
```


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

In this code, 200 shows the request was successful, "text/html" indicate the content type, and html displays the webpage material.

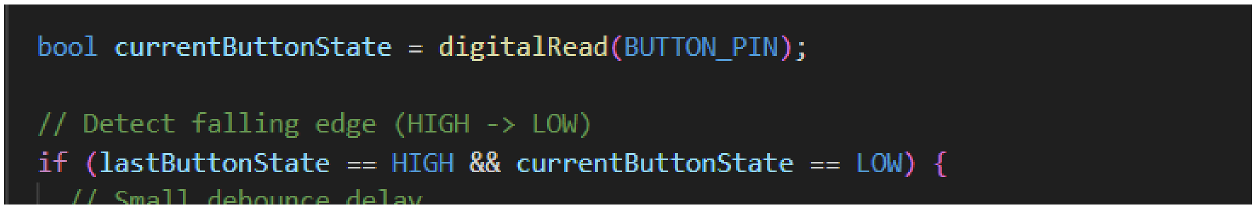
Button-based sensor reading and OLED update mechanism

In this system we used a push button to change the reading of temperature and humidity from OLED and on the server.



```
pinMode(BUTTON_PIN, INPUT_PULLUP);
```

With INPUT_PULLUP, the button pin remains **HIGH** when not pressed and becomes **LOW** when the button is pressed.

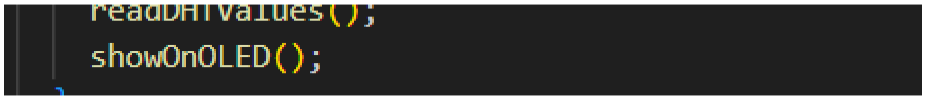


```
bool currentButtonState = digitalRead(BUTTON_PIN);

// Detect falling edge (HIGH -> LOW)
if (lastButtonState == HIGH && currentButtonState == LOW) {
  // Small debounce delay
```

To detect a valid button press we use

- lastButtonState load the last button state
- currentButtonState load the latest button reading

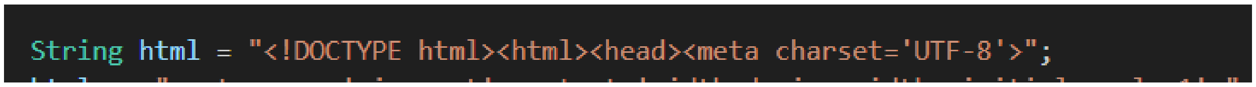


```
    readDHTValues();
    showOnOLED();
```

This function updates the OLED display.

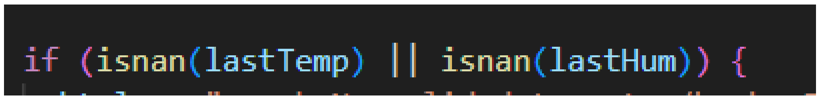
Dynamic HTML webpage generation

In this ESP32-based system, the webpage displaying temperature and humidity values is generated.



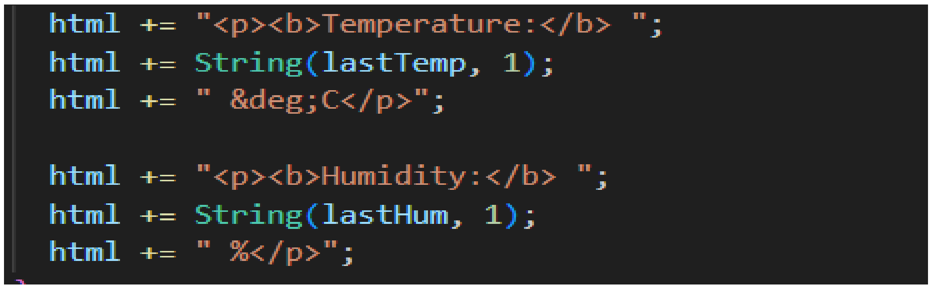
```
String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'>";
```

The HTML content is then built step by step by appending tags and text.



```
if (isnan(lastTemp) || isnan(lastHum)) {
```

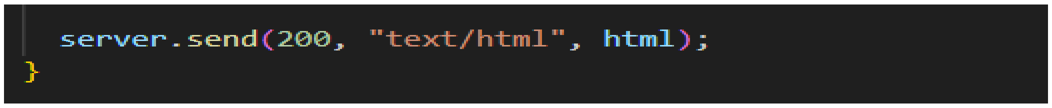
If the correct data did not exist it shows message for the user to enter the correct data. If the valid data exists webpage automatically insert these values at right place by:



```
  html += "<p><b>Temperature:</b> ";
  html += String(lastTemp, 1);
  html += " &deg;C</p>";

  html += "<p><b>Humidity:</b> ";
  html += String(lastHum, 1);
  html += " %</p>";
```

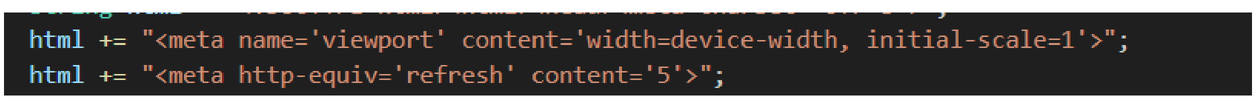
After the HTML page is created, it sends to the user browser using:



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

Purpose of meta refresh in the webpage:

Meta refresh is used to automatically refresh the page meta refresh is inserted inside the HTML section:



```
html += "<meta name='viewport' content='width=device-width, initial-scale=1'>";
html += "<meta http-equiv='refresh' content='5'>";
```

This refreshes the webpage within every 5 seconds.

Common issues in ESP32 webserver projects and their solutions

1. Wi-Fi Connection Failure

Issue:
The ESP32 fails to connect to the Wi-Fi network

Solution:

- Ensure that the SSID and password must be correct.
- Make sure that Wi-Fi is connected
- Add a serial monitor message to detect the situation

2. Web Page is not loaded

Issue:
The browser cannot access the ESP32 web page.

Solution:

- Make sure esp32 is connected to Wi-Fi.
- Check that the IP you use must be correct.

3. DHT Sensor Errors

Issue:
The DHT22 sensor fails to provide value

Solution:

- Ensure that your DHT is working correctly.

- Must add delay during sensor readings.
- Make sure that wiring is correct.

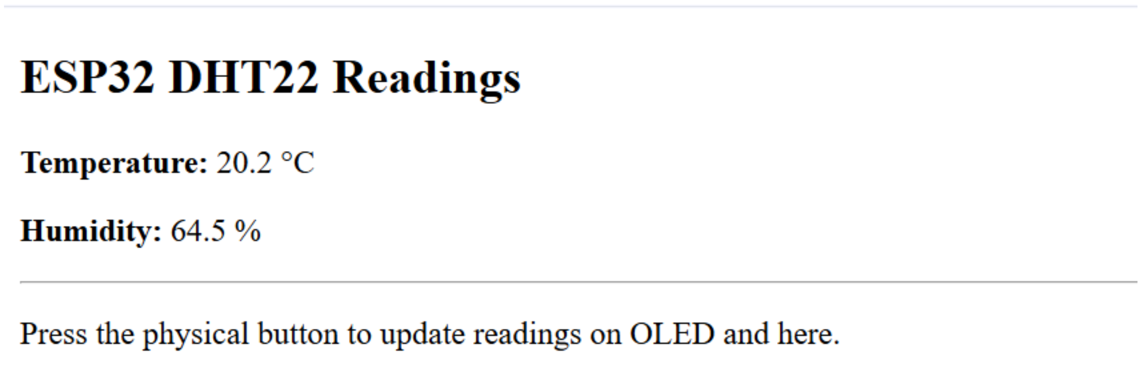
4. Web Page is Not Updating

Issue:
The webpage shows old data until we refresh it.

Solution:

- Use a meta refresh so it refreshes automatically

Webserver screenshot:



ESP32 DHT22 Readings

Temperature: 20.2 °C

Humidity: 64.5 %

Press the physical button to update readings on OLED and here.

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?**
   The Blynk Template ID connects the ESP32 to a particular device in the Blynk Cloud. It must match so the cloud determines which widgets, data streams, and settings related to that specific device. If it does not match, the device cannot associate or transfer data precisely.

2. **Differentiate between Blynk Template ID and Blynk Auth Token.**
   Template ID: Determine the template in Blynk Cloud. Similar for all devices utilizing that template.
   Auth Token: A distinctive key that authenticates a particular device. It is different for every device.

3. **Why does using DHT22 code with a DHT11 sensor produce incorrect readings? Mention one key difference between the two sensors.**
   DHT11 and DHT22 utilize contrasting data formats and accuracy ranges**.**
   Difference: DHT22 supply higher accuracy and extensive temperature range**,** while DHT11 has low-resolution. Utilizing incorrect code causes incorrect decoding of sensor data.

4. **What are Virtual Pins in Blynk? Why are they preferred over physical GPIO pins for cloud communication?**
   Virtual Pins are software-defined pins utilize to transfer and gain data between ESP32 and Blynk Cloud.
   They are preferred as they are independent of hardware GPIO, flexible, and perfect for cloud-based transmission.
5. **What is the purpose of using BlynkTimer instead of delay () in ESP32 IoT applications?**
   BlynkTimer permits non-blocking, scheduled tasks while keeping Wi-Fi and Blynk connections operative.
   Using delay () blocks implementation and can produce disconnection or missed updates.

Part-B: Long Question

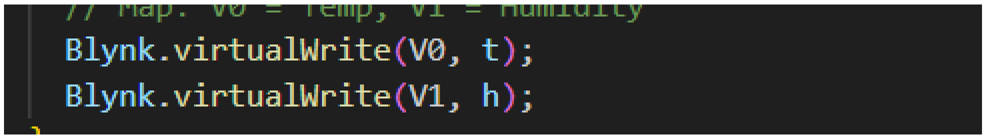
Explain the complete workflow of interfacing ESP32 with Blynk Cloud to display temperature and humidity values.

Creation of Blynk Template and Datastreams

In the Blynk Cloud, a Device Template is generated with two datastreams:

- V0 → Temperature
- V1 → Humidity

These datastreams are utilized in the code when transferring sensor values:

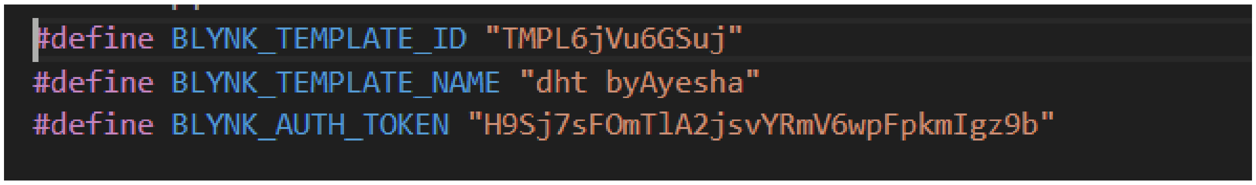


```
  // Map: V0 = Temp, V1 = Humidity
  Blynk.virtualWrite(V0, t);
  Blynk.virtualWrite(V1, h);
```

This makes sure that temperature and humidity values show on the correct Blynk dashboard.

- Role of Template ID, Template Name, and Auth Token

At the top of the code, in the Blynk template ID, Name and Auth Token are defined:

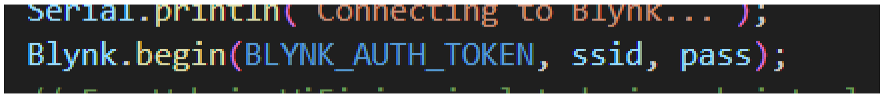


```
#define BLYNK_TEMPLATE_ID "TMPL6jVu6GSuj"
#define BLYNK_TEMPLATE_NAME "dht byAyesha"
#define BLYNK_AUTH_TOKEN "H9Sj7sFOmTlA2jsvYRmV6wpFpkmIgz9b"
```

- Template ID links the ESP32 to the accurate cloud template.
- Template Name determines the template.
- Auth Token verifies this specific ESP32 device.

These values are same as the Blynk Cloud values, however the ESP32 will not connect.

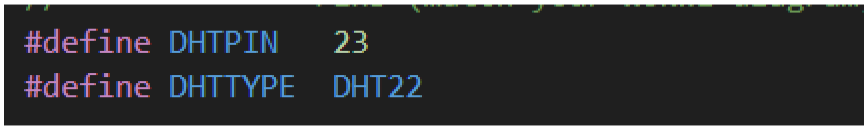
The connection is created in setup ():



```
Serial.println("Connecting to Blynk...");
Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
```

- Sensor configuration issues (DHT11 vs DHT22)

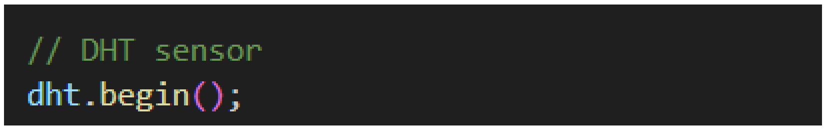
The sensor type is dht22 and pin is 23 which is mention in the code as:



```
#define DHTPIN   23
#define DHTTYPE  DHT22
```

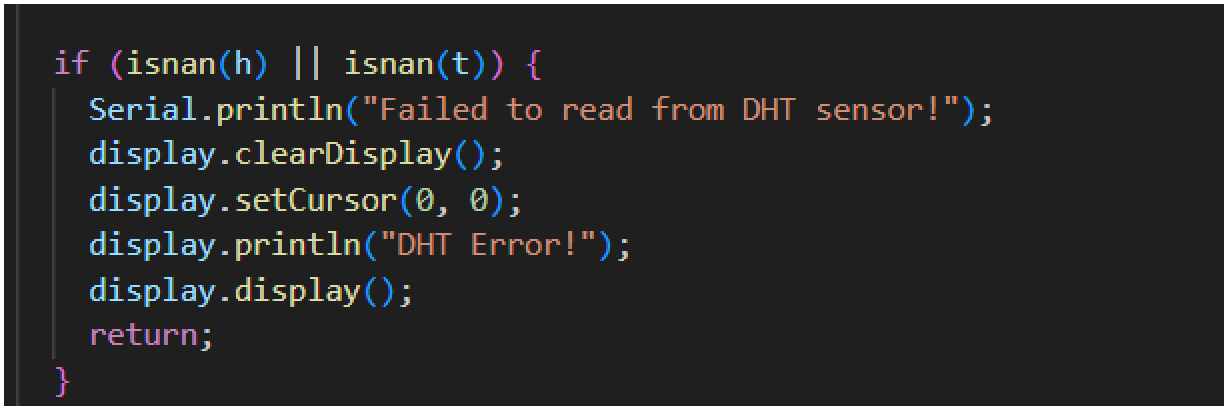
The code is written for DHT22. If a DHT11 is used instead, readings become inappropriate because both sensors utilize different data formats and accuracy levels.

Sensor initialization occurs in setup():



```
// DHT sensor
dht.begin();
```

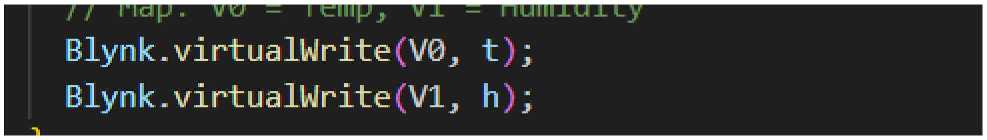
Error handling is written in code to determine the configuration:



```
if (isnan(h) || isnan(t)) {
  Serial.println("Failed to read from DHT sensor!");
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("DHT Error!");
  display.display();
  return;
}
```

- Sending data using Blynk.virtualWrite()

After reading the sensor values, data is transferred to the Blynk Cloud:



```
  // Map: V0 = Temp, V1 = Humidity
  Blynk.virtualWrite(V0, t);
  Blynk.virtualWrite(V1, h);
```

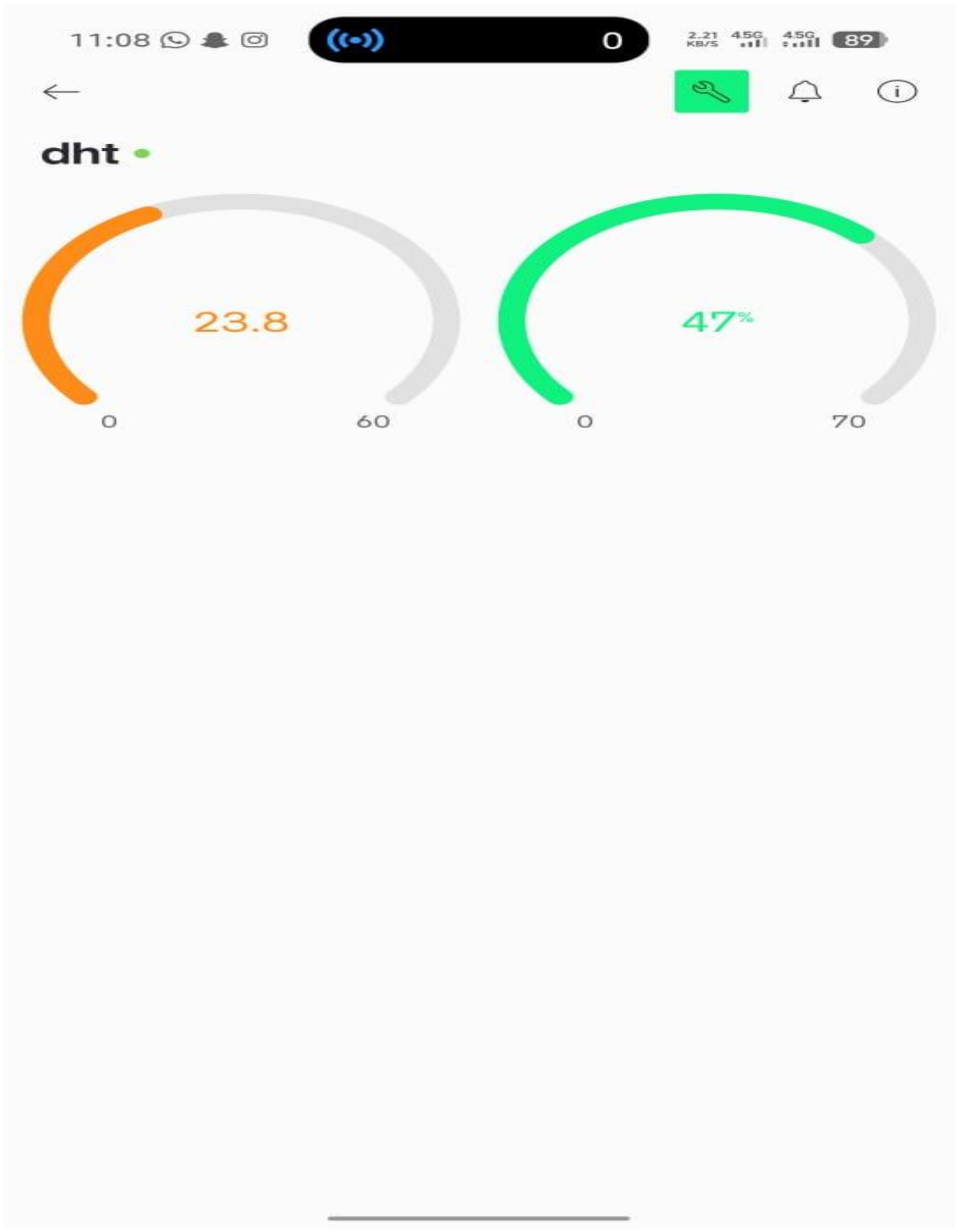
Virtual pins are not dependent on ESP32 physical GPIO pins.

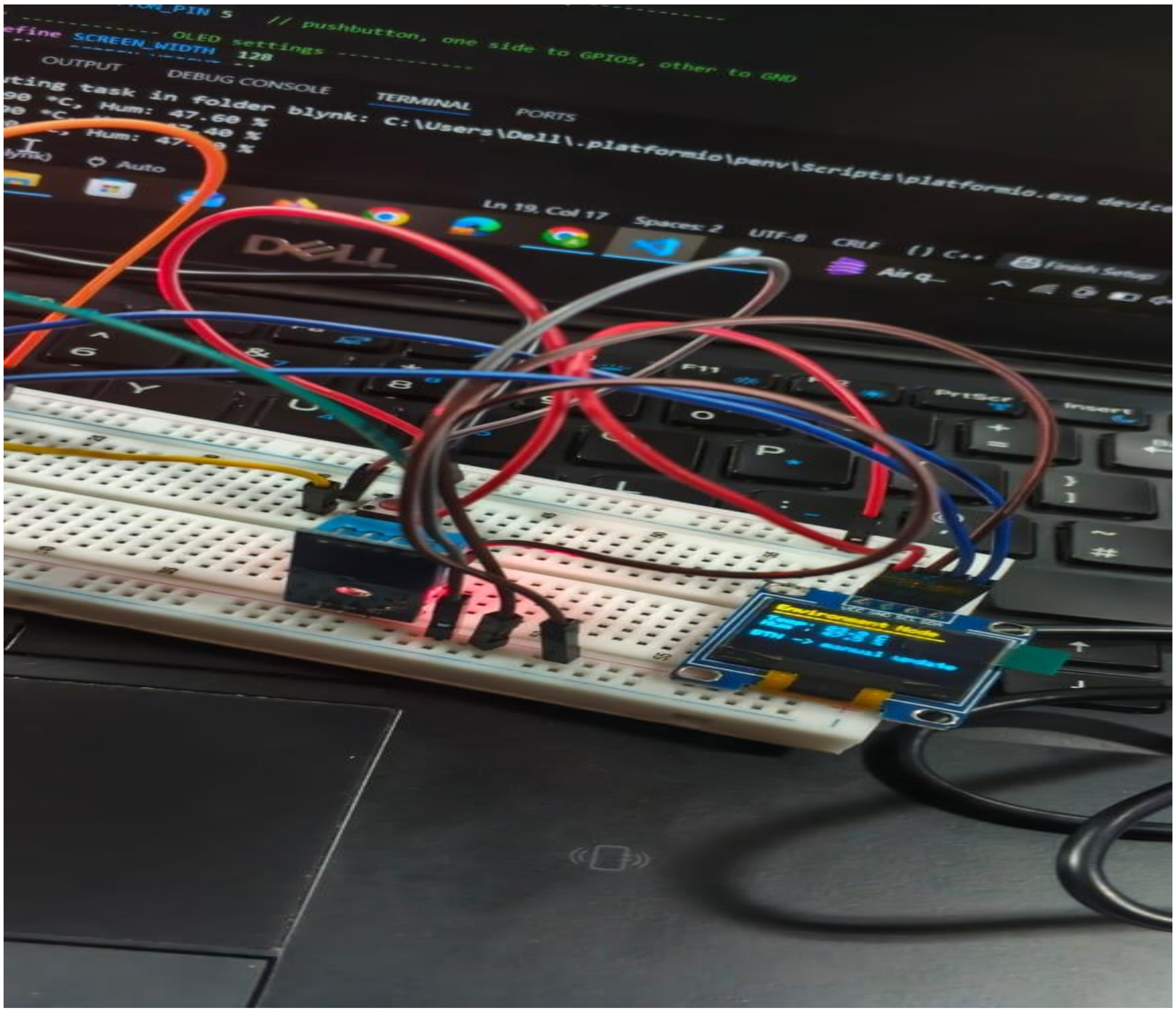
It allows cloud communication without hardware dependency.

– Common problems faced during configuration and their solutions

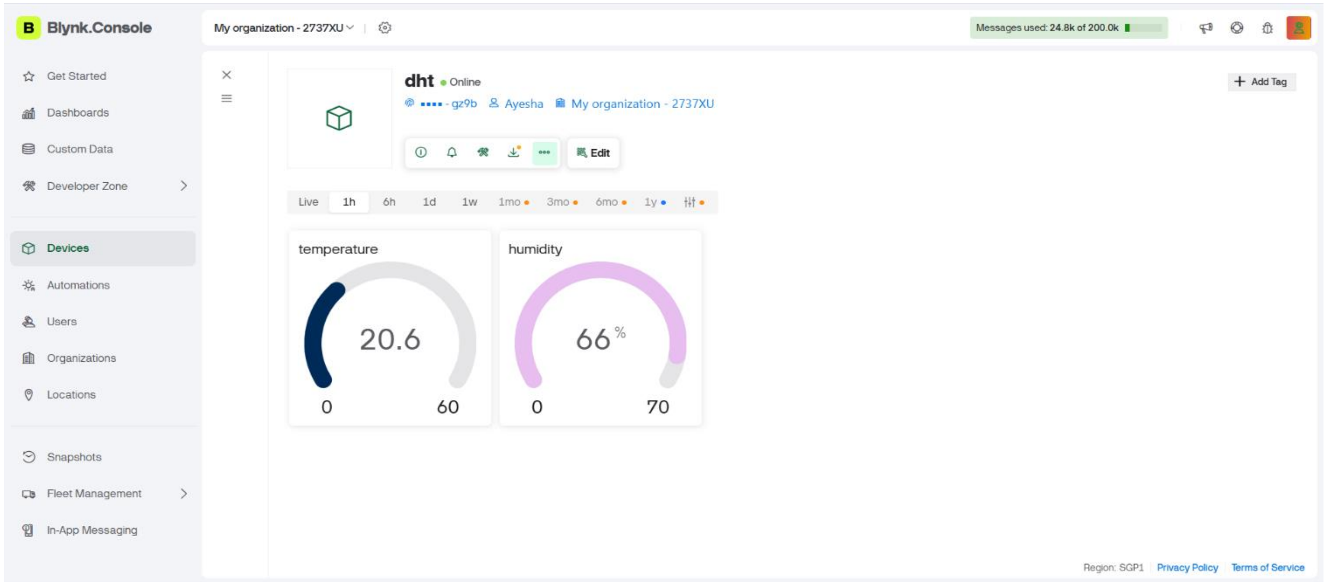
Problem 1:
The Blynk configuration didn't match our device.
Solution:
We modify the three Blynk lines (Template ID, Template Name, and Auth Token) from our own project.
Problem 2:
The code was set for a DHT22 sensor, but we are using a DHT11, causing incorrect readings.
Solution:
We changed the sensor type in the code from DHT22 to DHT11.
Problem 3:
The ESP32 couldn't connect to WiFi because the default SSID and password were incorrect.
Solution:
We replaced the SSID and PASS with our own internet credentials.
Problem 4:
Lastly Update the platform.ini with:
Solution:
blynkkk/Blynk@^1.3.2

Screenshot:





```
_PIN 5      // pushbutton, one side to GPIO5, other to GND
----------- OLED settings -----------
efine SCREEN_WIDTH 128
OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
ting task in folder blynk: C:\Users\Dell\.platformio\penv\Scripts\platformio.exe device
90 °C, Hum: 47.60 %
90 °C, Hum: 47.40 %
```



dht • Online

•••• - gz9b   Ayesha   My organization - 2737XU

temperature: 20.6 (0 – 60)

humidity: 66 % (0 – 70)