National Textile University, Faisalabad

Department of Computer Science

Name	Sarwat Fatima
Class	BSCS 5th
Section	5th – A
Registration No	23-NTU-CS-1092
Assignment no.	2
Course Name	Embedded IoT system
Submitted To	Sir Nasir Mehmod
Submission Date	17/12/2025

Homework-01

Embedded IOT Systems Fall2025

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?**

Purpose:

WebServer server(80):

This line creates a web server object named server. It directly runs on the ESP32. The web server library allows the ESP32 to serve web page to any device connected to the same WiFi.

Port 80:

Port 80 is the default port for the HTTP traffic. When you type an IP address assign to the ESP in a web browser without specifying the port, the browser will automatically uses port 80.

Example: if your ESP32 gets IP 200.162.10.1, when you open this in the web browser the browser will connect to port 80 by default.

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

Role / Explanation:

This tells the web server that when a client means browser requests the root URL / of the ESP32, it should execute the function handleRoot(). “/” represents the root path of the server. And the function generates and sends the HTML page to the browser.

When we open the ESP32’s IP in the browser. The browser sends the HTTP GET request to /. The web server sees the request matches “/”. The server calls handleRoot() to generate the HTML page. The page is sent back to browser. This connects the root web server address / to the function that serves the ESP32.

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

Why it’s in `loop()`:
This function checks the incoming HTTP requests from the client. If a client requests a page, it processes the request and calls the corresponding handler function. Since it is frequently called, usually it is placed inside the loop(), so the server can respond as soon as the request is made.

What happens if it is removed :
If it is not placed inside the loop then the server wouldn’t check for client requests regularly, and the web page will not load or update. The browser trying to access the ESP32’s IP will hang or fail to load the page, because no function is processing the incoming HTTP requests.

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

Explanation:
Server.send() is a method of the webserver object that sends a response to the client that made an HTTP request.

- 200 is HTTP status code, means OK. It means the request was successful and the server is sending the requested data.
- “test/html” means content type of the response, so the browser will render it properly.

- html is the string containing the HTML code generated earlier in handleRoot().

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

Last measured values:
Displaying last measured values mean that the most recent value of the DHT from already taken when the button was pressed from the memory is shown on the web page. Rather than asking the sensor for the value of temperature or humidity it displays the most recent values from the memory stored in lastTemp or lastHum.

Fresh DHT reading:
It means that when the web page is open the ESP32 reads the DHT sensor immediately and show that values on the web page. These are the most current values of the sensor like real-time display of values.

Part B: Long Question

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

- ESP32 Wi-Fi connection process and IP address assignment:
ESP32 has a built-in Wi-Fi functionality, using the Wi-Fi.h library and the name and password provided in the code it connects to the internet. The ESP32 is assigned a local IP. This IP address works as the address of the web server, that allow any device to access the browser across the same network. The IP is shown on the serial monitor and displayed on the OLED for better display.

- Web server initialization and request handling:
The ESP32 works as the webserver using the webserver library. It works as a HTTP server. Port 80 is the default port for the HTTP traffic. The browser communicates with the server using this port.

Then the object of the webserver library also handle the GET request using the function handleRoot and generates and serves the HTML page to the browser. All of this is placed inside the loop so that the server can listen to multiple requests all the time. This sets our ESP32 as server and handle the client requests.

- Button-based sensor reading and OLED update mechanism:

In this code a button is attached with the DHT sensor to activate the sensor to detect the values when the button is pressed and store them in memory. When the button is pressed the ESP32 detects the button press and if it is HIGH to LOW it means the button is pressed, and it runs the function readDHTValues() using the DHT.h library and also updates the display of the OLED and a debounce of 50 ms delay is added, so that it did not count the bouncing of the button as triggers.

- Dynamic HTML webpage generation:

When the web page is dynamically created using the handleRoot ()

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

and send to the browser using the Port 80. It contains temperature and humidity readings, instructions to press the button for updating the values, and the basic HTML formatting for readability.

The page will always display the latest readings stored in ESP32 memory. By updating the HTML string every time handleRoot() is called, the real-time data is present on the web page.

- Purpose of meta refresh in the webpage:

The HTML includes:

```
<meta  http-equiv='refresh'  content='5'>
```

It automatically refreshes the web page every 5 seconds. So that the page does not need to be refreshed by the user. Even when the button is pressed and new readings are taken the page will update after 5 seconds automatically.

- Common issues in ESP32 webserver projects and their solutions:
The common issues for ESP32 webserver can be:

- Web page does not load.
- DHT22 does not read values.
- OLED does not display anything
- Button multiple triggers read.
- Web page updates slowly.

There solutions can be:

- Check Wi-Fi.status () and serial monitor IP.
- Ensure correct wiring, add dht.begin().
- Check parameters for display function and ensure driver is added.
- Use short debounce delay.
- Use last stored values and meta-refresh instead of reading sensor every page load.

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?

Blynk Template ID in ESP32 uniquely identifies the device created on the blynk cloud. It includes datastreams (V0, V1), widgets and device structure.

It must match the cloud template because the ESP32 and Blynk cloud connects with each other using the Template ID. If it is not correct then the values of the app will not match the real values.

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

Blynk Template ID Blynk Auth Token

Blynk template ID identifies the device template, but it is same for all the devices that are using the same template. It basically links ESP32 firmware to the dashboard of the design.

Blynk Auth Token authenticates the specific device, it is unique for each device even if they use the same template/ it allows devices to connect securely.

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

DHT11 and DHT22 are different types of sensors DHT11 is lower accuracy and integer values but the other is higher in accuracy and gives values in decimal values. The incorrect reading will occur due to wrong protocol. When the ESP32 will try to read the values and handle them in code and if we define DHT11 instead of DHT22 will give incorrect result due to wrong protocol.

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

Virtual pins:

Virtual pins are like built-in software-based pins for cloud communication. It does not react to real GPIO pins of ESP32. They are like communication link between ESP32 and Blynk cloud.
These pins are independent of hardware, allow flexible data mapping to widgets, enable cloud-based control and monitoring and reduce physical pins conflicts.

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

It is non-blocking, that is one of the biggest advantages of BlynkTimer. It provides a timed execution, keeps Wi-Fi and Blynk connection alive. It ensures multitasking, reliable cloud communication and periodic sensor updates. The rest of the code keeps running without being affected.

Part-B: Long Question

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

- Creation of Blynk Template and Datastreams:
First of all a Blynk Template is created on the blynk cloud and two datastreams are defined.

e.g.:
V0 – Temperature
V1 – Humidity

Widgets like gauges or values display are linked to these datastreams. These datastreams recaive sensor data sent from ESP32.

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

Template ID: it connects firmware to the correct cloud template.

Template Name: it is for the user to read name of the template, user specify it.

Auth Token: it authenticates the ESP32 device to Blynk cloud.

If all three are correct means that the cloud connection is created successfully.

- ESP32 Wi-Fi and Blynk Cloud Connection:

```
Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
```

This line connects the ESP32 to Wi-Fi and create secure connection with Blyno Cloud, and the device appears online on the Blynk dashboard.

- Sensor configuration issues (DHT11 vs DHT22):

```
#define DHTPIN 23

#define DHTTYPE DHT22
```

DHT22 sensor is highly accurate and give perfect values and it also have protocols to send detected value, so, if we use DHT11 instead of DHT22 protocols will generate the wrong protocol message.

- Button-Based and Timer-Based Data Acquisition:

This code also include reading the values when the button id pressed and automatically after every 5 seconds.

- Sending data using Blynk.virtualWrite():

```
Blynk.virtualWrite(V0, t);

Blynk.virtualWrite(V1, h);
```

After these two lines the ESP32 sends the read values to the Blynk cloud which updates the values on the dashboard, no physical wiring is needed for this. Not only this but the values are also being shown on the OLED display. The values will update there as well.

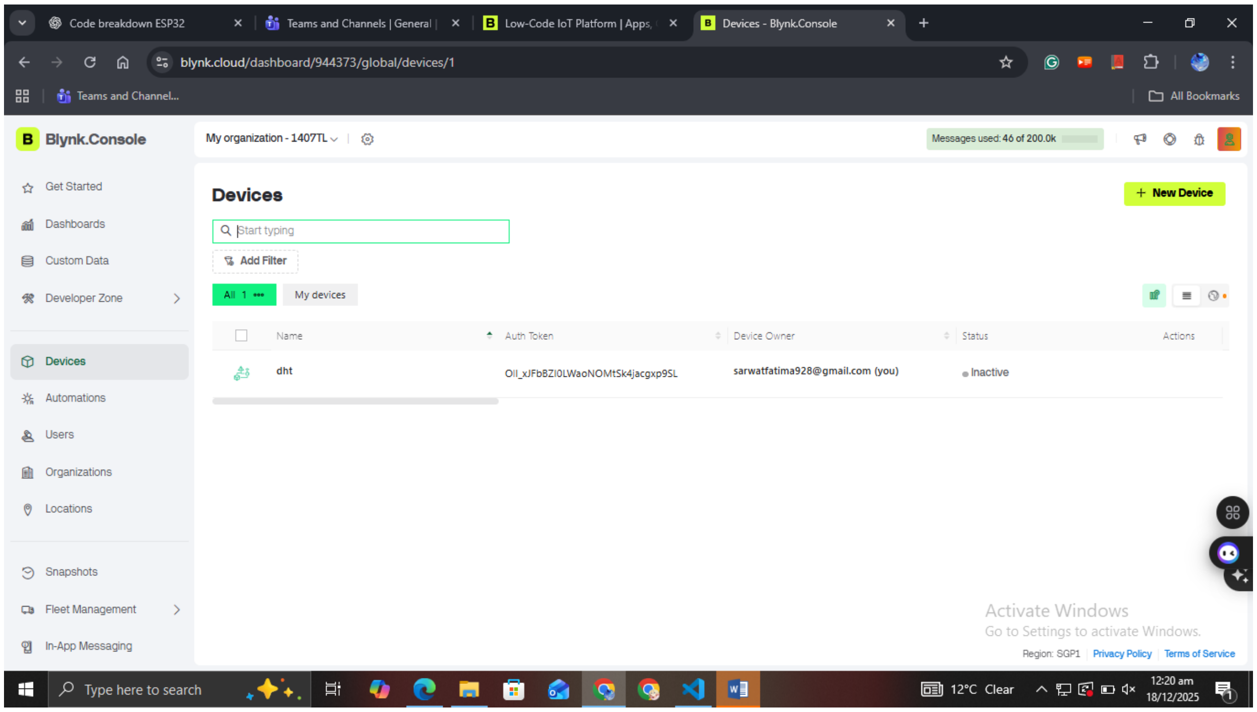
- Common problems faced during configuration and their solutions:

- Device is offline.
- There is no data shown on the widgets
- NAN readings is shown.
- The wi-fi connection is unstable.
- OLED is not working.

Few solutions for these problems are:

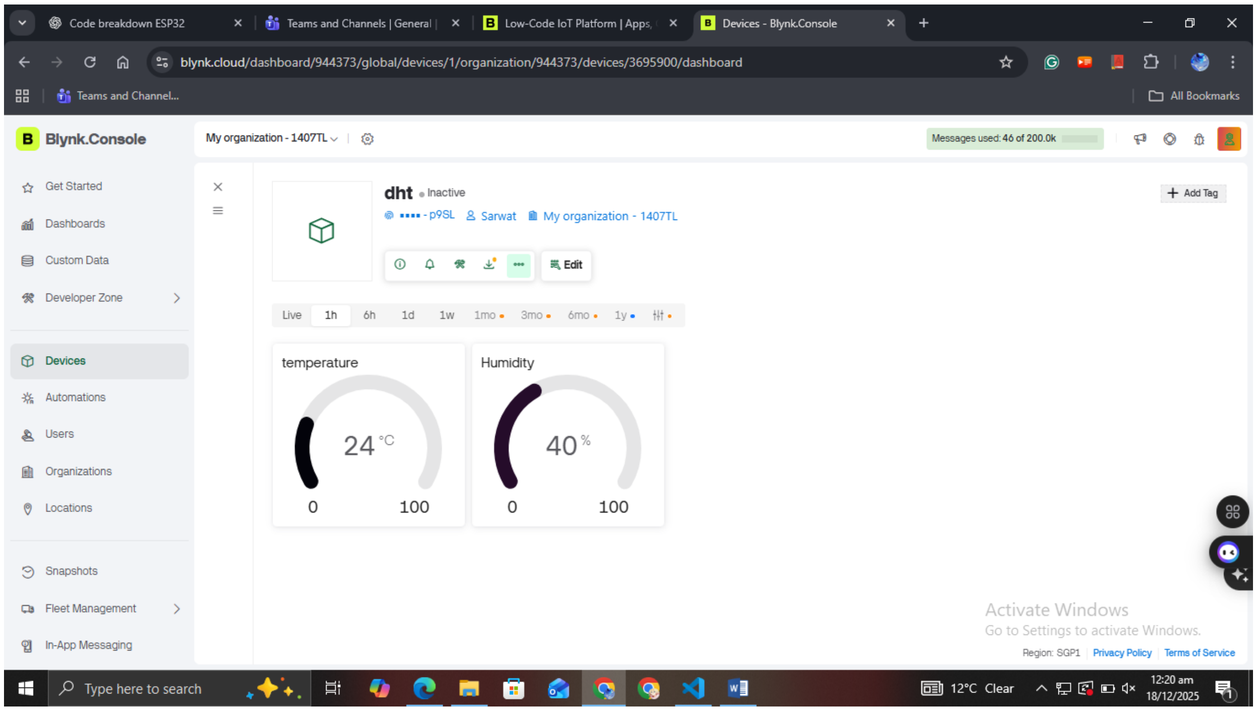
- Use correct token.
- Match datastreams.
- Correct DHT type
- Use BlynkTimer
- Use correct SDA/SCL

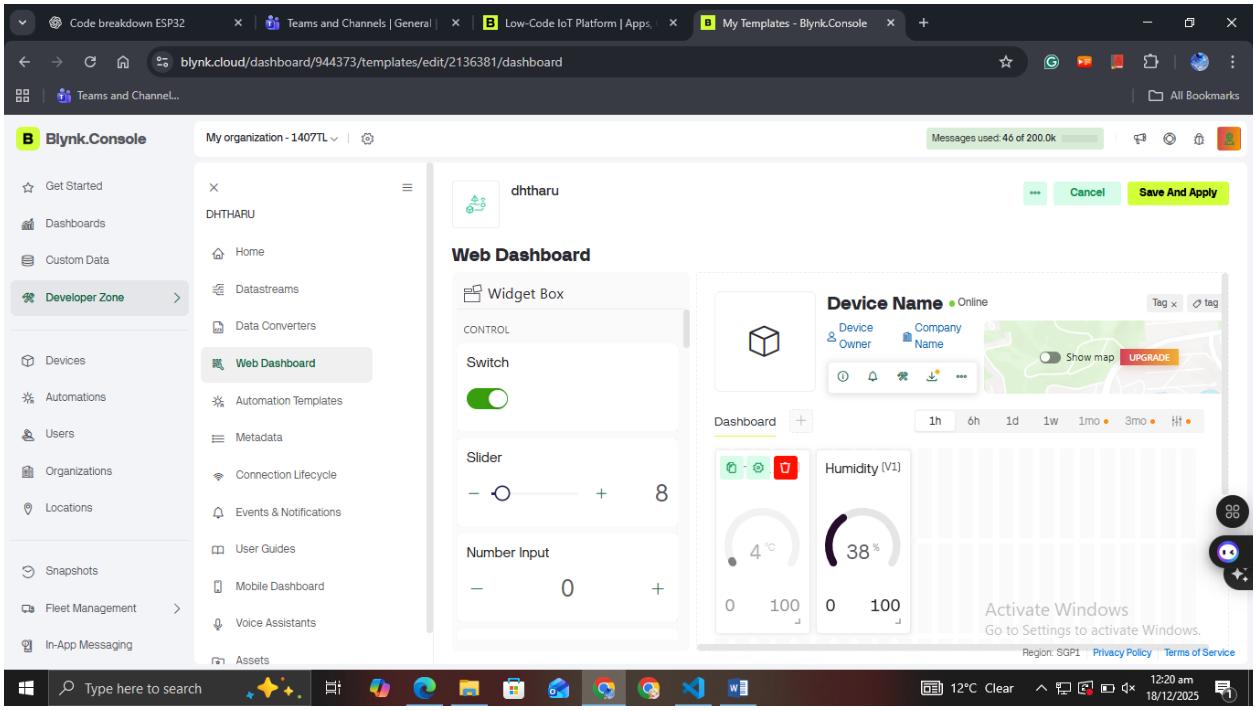
Device(Web) Blynk Dashboard Screenshot:

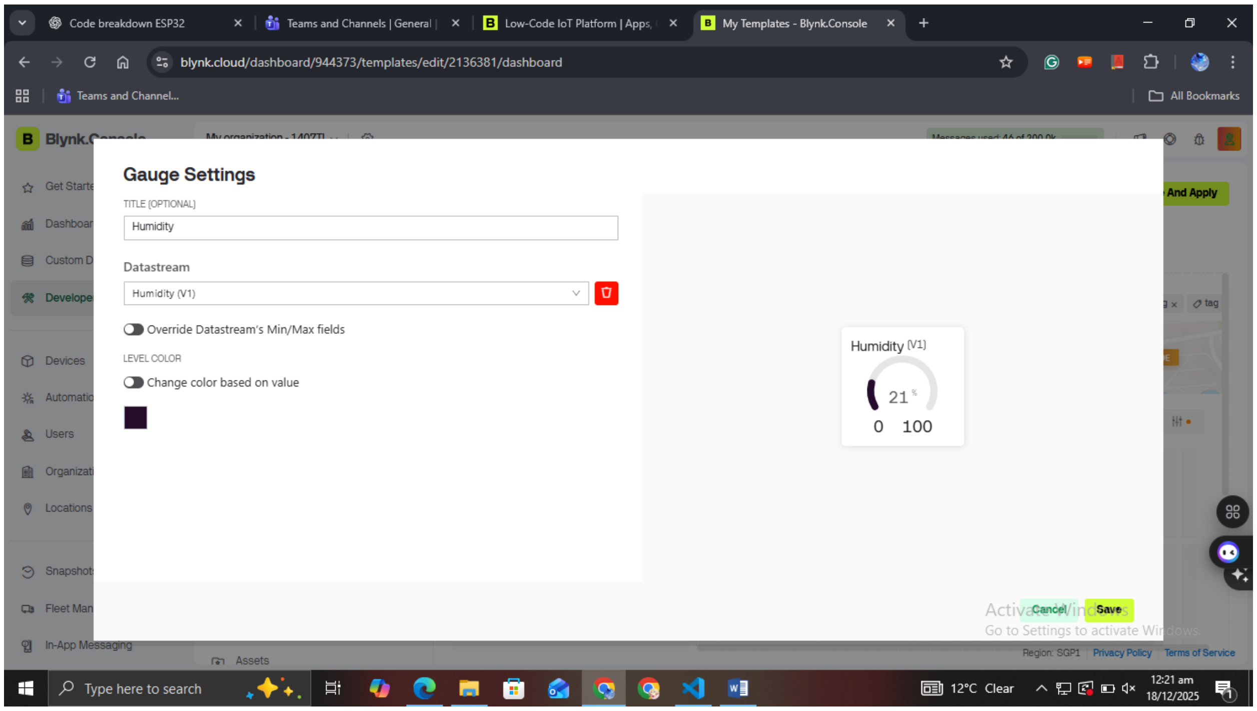


Devices

Name	Auth Token	Device Owner	Status	Actions
dht	OIl_xJFbBZI0LWaoNOMtSk4jacgxp9SL	sarwatfatima928@gmail.com (you)	Inactive	


Code breakdown ESP32
Teams and Channels | General |
Low-Code IoT Platform | Apps,
Devices - Blynk.Console
blynk.cloud/dashboard/944373/global/devices/1/organization/944373/devices/3695900/dashboard
Blynk.Console
My organization - 1407TL
Messages used: 46 of 200.0k
Get Started
Dashboards
Custom Data
Developer Zone
Devices
Automations
Users
Organizations
Locations
Snapshots
Fleet Management
In-App Messaging
dht
Inactive
•••• - p9SL
Sarwat
My organization - 1407TL
Edit
Add Tag
Live
1h
6h
1d
1w
1mo
3mo
6mo
1y
temperature
24 °C
0
100
Humidity
40 %
0
100
Activate Windows
Go to Settings to activate Windows.
Region: SGP1
Privacy Policy
Terms of Service
Type here to search
12°C Clear
12:20 am
18/12/2025


Code breakdown ESP32
Teams and Channels | General |
Low-Code IoT Platform | Apps,
My Templates - Blynk.Console
blynk.cloud/dashboard/944373/templates/edit/2136381/dashboard
Blynk.Console
My organization - 1407TL
Messages used: 46 of 200.0k
DHTHARU
Home
Datastreams
Data Converters
Web Dashboard
Automation Templates
Metadata
Connection Lifecycle
Events & Notifications
User Guides
Mobile Dashboard
Voice Assistants
Assets
dhtharu
Cancel
Save And Apply
Web Dashboard
Widget Box
CONTROL
Switch
Slider
8
Number Input
0
Device Name
Online
Device Owner
Company Name
Show map
UPGRADE
Tag
tag
Dashboard
1h
6h
1d
1w
1mo
3mo
4 °C
0
100
Humidity (V1)
38 %
0
100
Activate Windows
Go to Settings to activate Windows.
Region: SGP1
Privacy Policy
Terms of Service
Type here to search
12°C Clear
12:20 am
18/12/2025



Gauge Settings

TITLE (OPTIONAL)

Humidity

Datastream

Humidity (V1)

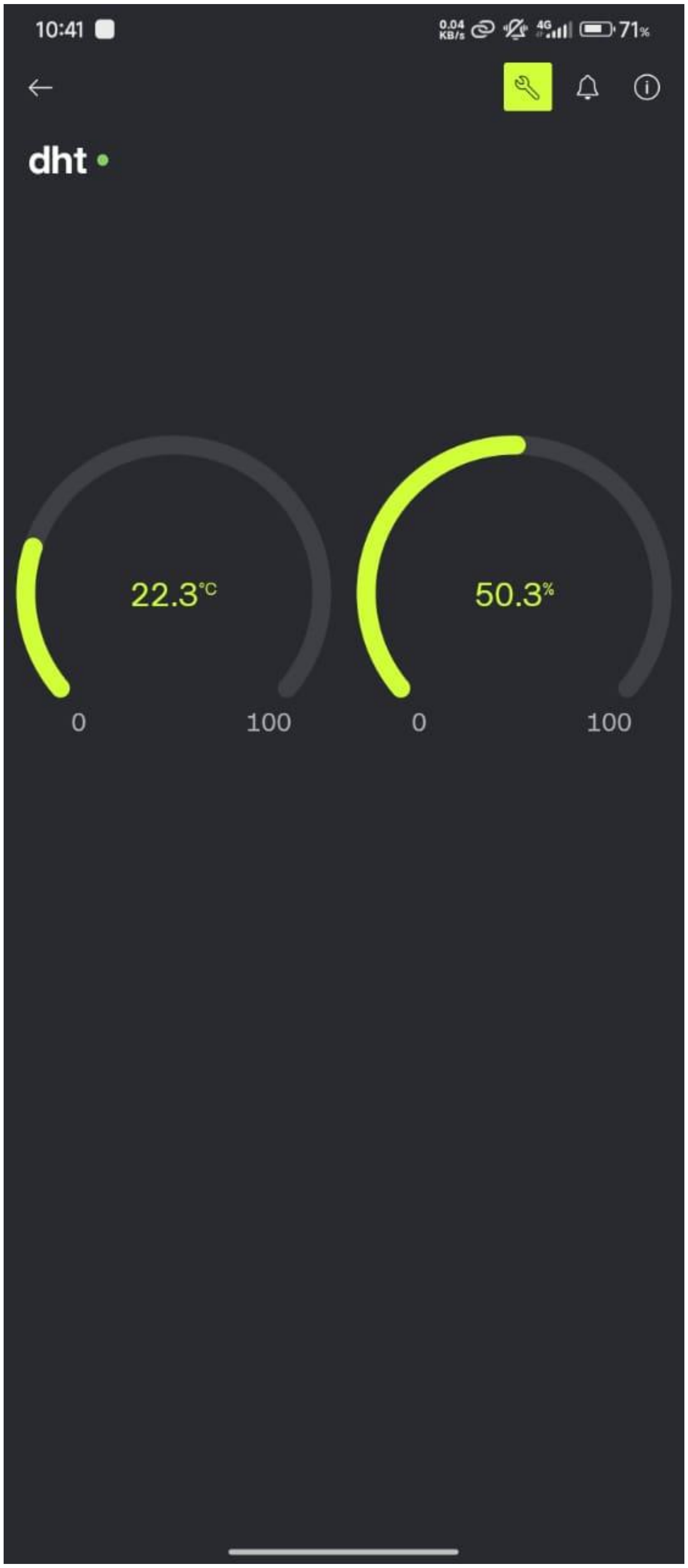
Override Datastream's Min/Max fields

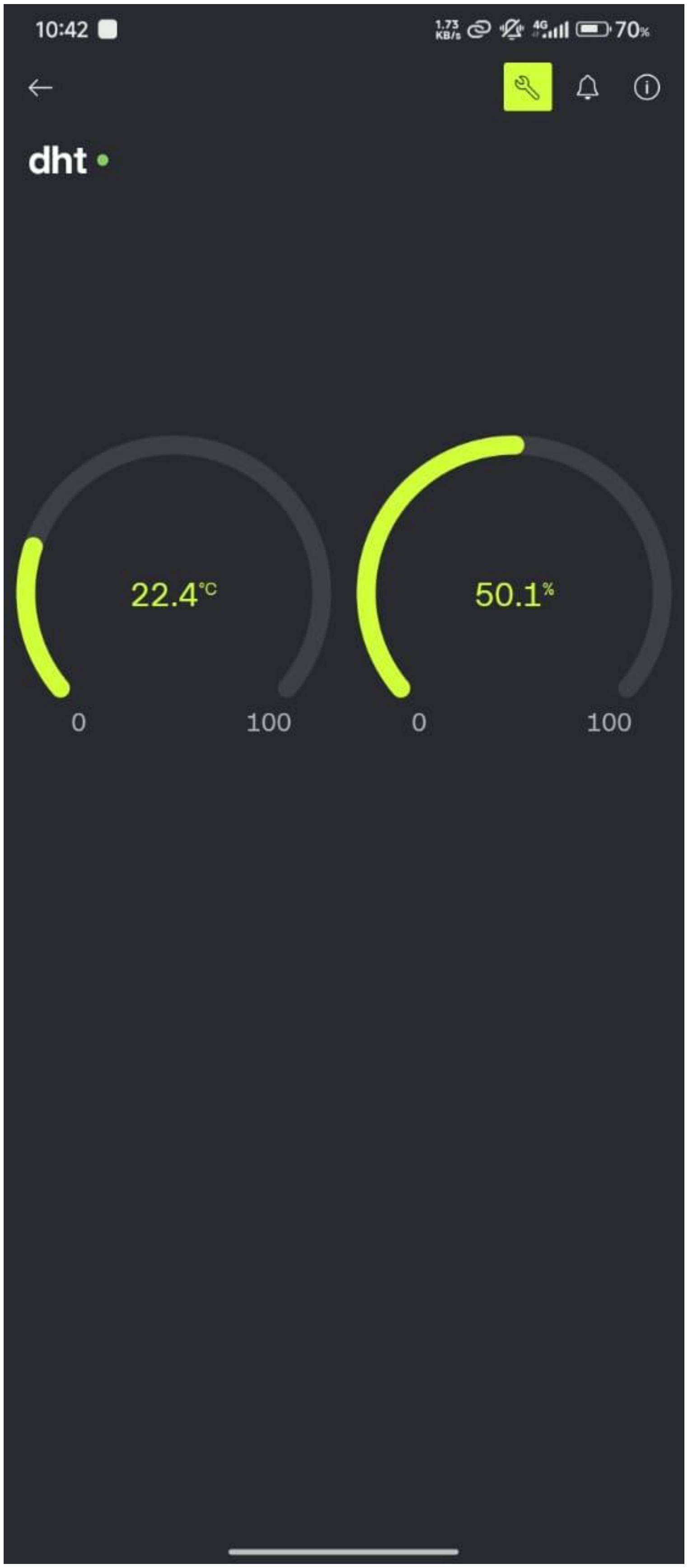
LEVEL COLOR

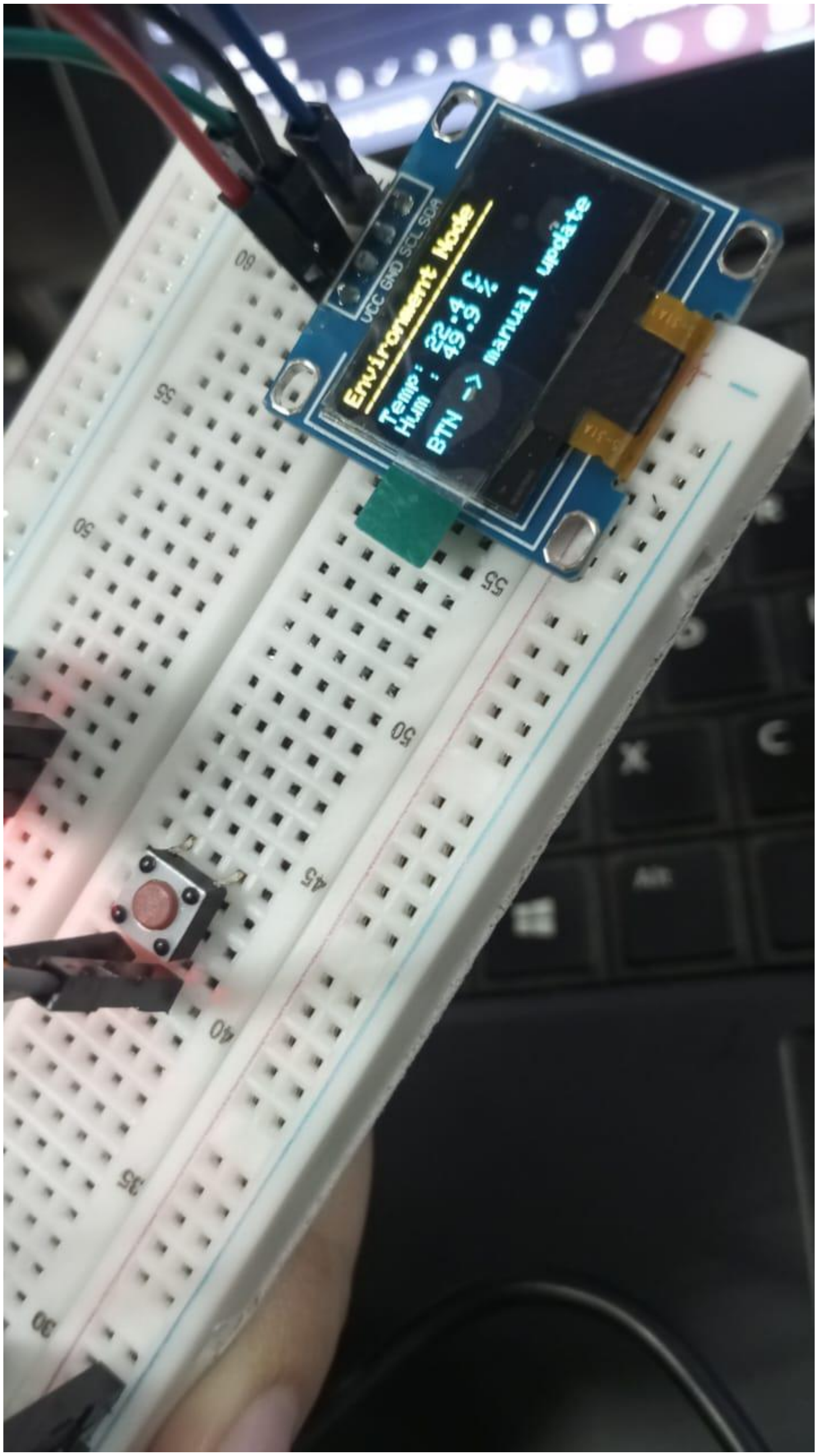
Change color based on value

Humidity (V1)
21
0
100
Cancel
Save
blynk.cloud/dashboard/944373/templates/edit/2136381/dashboard
Code breakdown ESP32
Teams and Channels | General |
Low-Code IoT Platform | Apps,
My Templates - Blynk.Console

Mobile Blynk Dashboard Screenshot:


10:41
0.04 KB/s
71%
dht
22.3°C
0
100
50.3%
0
100


10:42
1.73 KB/s
70%
dht
22.4°C
0
100
50.1%
0
100



```
Environment Mode
Temp: 22.4 C
Hum : 49.9 %
BTN -> manual update
```