Department of Computer Science

Name:	Muhammad Hanzala Zahid
Class:	BSCS 5th-B
Registration No:	23-NTU-CS-1067
Assignment No:	02
Course Name:	Embedded Iot system
Submitted To:	Sir Nasir
Submission Date:	20-12-2025

Assignment 02:

Question 01:

Part A: Short questions:

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

This line creates a web server object on the ESP device. Port 80 is the standard port used for HTTP web traffic. When a browser connects to the ESP's IP address, it automatically communicates through port 80. This allows the ESP to serve web pages without specifying a port number.

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

This statement defines what happens when a user opens the main page of the web server. The symbol `/` represents the home URL of the ESP web server. When this URL is accessed, the function is executed to send the web page content. It links a web request to a specific function.

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

The loop () function runs continuously, so placing server.handleClient() inside it allows the ESP to check for new client requests all the time. It listens and responds to incoming web requests. If it is removed, the server will stop responding and the web page will not load. The ESP will appear disconnected.

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

This line sends a response from the ESP web server to the client's browser. The code 200 indicates a successful request. "text/html" tells the browser that the data is a web page. The `html` variable contains the actual webpage content to display.

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

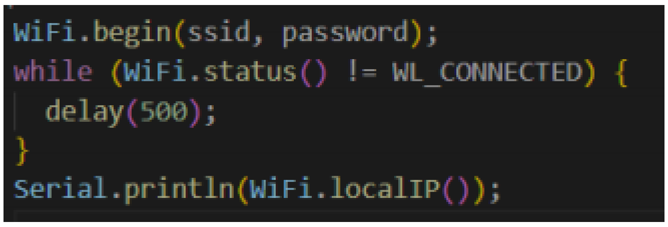
Displaying last values is faster and avoids sensor delay. Taking a fresh DHT reading inside handleRoot() can slow down page loading and may cause sensor errors if refreshed too often.

Part B: Long Questions:

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

ESP32 Wi-Fi Connection & IP Address Assignment:

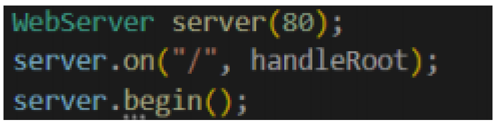
The ESP32 connects to a Wi-Fi network using SSID and password. After successful connection, the router assigns an IP address, which is used to access the web page.



```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
}
Serial.println(WiFi.localIP());
```

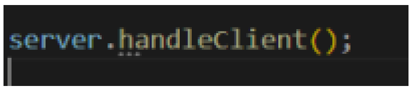
Web Server Initialization & Request Handling:

A web server is created on port 80. The root URL (/) is linked with a function that sends the webpage to the browser.



```
WebServer server(80);
server.on("/", handleRoot);
server.begin();
```

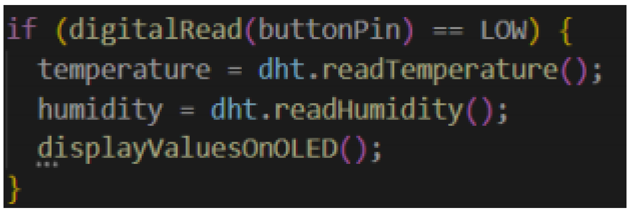
Client requests are handled continuously inside the loop.



```
server.handleClient();
```

Button-Based Sensor Reading & OLED Update:

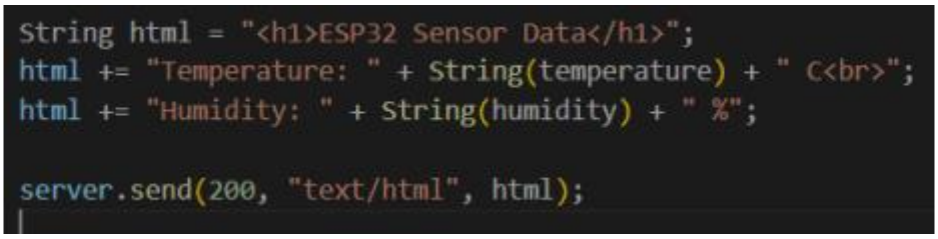
A button is used to take sensor readings only when needed. When pressed, the ESP32 reads temperature and humidity from the DHT sensor and updates the OLED display.



```
if (digitalRead(buttonPin) == LOW) {
  temperature = dht.readTemperature();
  humidity = dht.readHumidity();
  displayValuesOnOLED();
}
```

Dynamic HTML Webpage Generation:

The webpage is created using a string. Sensor values are added dynamically so updated data appears on the webpage.



```
String html = "<h1>ESP32 Sensor Data</h1>";
html += "Temperature: " + String(temperature) + " C<br>";
html += "Humidity: " + String(humidity) + " %";

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

Purpose of Meta Refresh:

Meta refresh automatically reloads the webpage after a few seconds, showing updated sensor values without manual refresh. This improves real-time monitoring.

Common Issues & Their Solutions:

• **Wi-Fi not connecting:** Check SSID/password and signal strength

• **Page not loading:** Ensure server.handleClient() is running

• **DHT sensor errors:** Avoid frequent readings and add delays

• **Slow response:** Do not read sensors inside every page request

Question 02:

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 links the ESP32 firmware to a specific project template in the Blynk cloud. It defines the dashboard layout and widgets used by the device. If it does not match the cloud template, the device will fail to connect properly. Matching ensures correct data exchange

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

The Template ID identifies the project design in the Blynk cloud. The Auth Token uniquely identifies and authenticates a specific device. One connects the firmware to the template, while the other authorizes the device to communicate. Both are required for successful operation.

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

DHT11 and DHT22 use different data formats and measurement ranges. Using DHT22 code with a DHT11 causes incorrect data interpretation. A key difference is that DHT22 supports decimal values, while DHT11 provides only integer readings. This mismatch leads to faulty results.

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

Virtual Pins are software-based pins used to send and receive data between ESP32 and Blynk widgets. They are preferred because they are not tied to physical GPIO pins and work easily with cloud data.

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

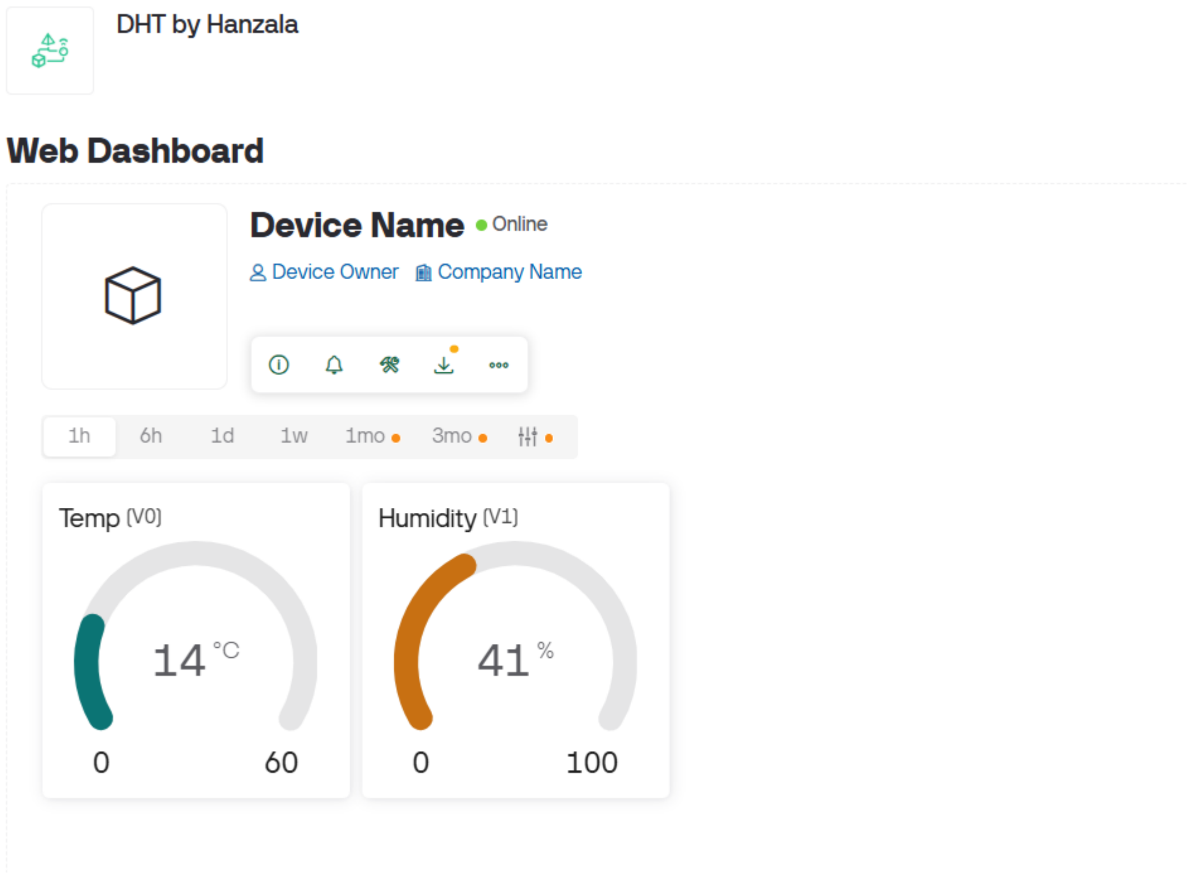
BlynkTimer allows tasks to run at intervals without blocking the program.Using delay() stops Wi-Fi and cloud communication, while BlynkTimer keeps the system responsive.

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:

First, a Blynk template is created in the cloud and datastreams are added for temperature and humidity using virtual pins. These datastreams define how data will appear on the dashboard.



This is dashboard and these datastreams show how data is flowing.

Template ID, Template Name, and Auth Token:

The Template ID and Template Name link the ESP32 code to the correct cloud project, while the Auth Token authorizes the specific device. All credentials must match the Blynk Cloud for a successful connection.

Sensor Configuration (DHT11 vs DHT22):

The sensor type must be correctly defined in the code. Using DHT22 configuration with a DHT11 sensor results in incorrect readings due to differences in accuracy and data format.

```
#define DHTTYPE DHT11
```

Data Transmission Using Virtual Pins:

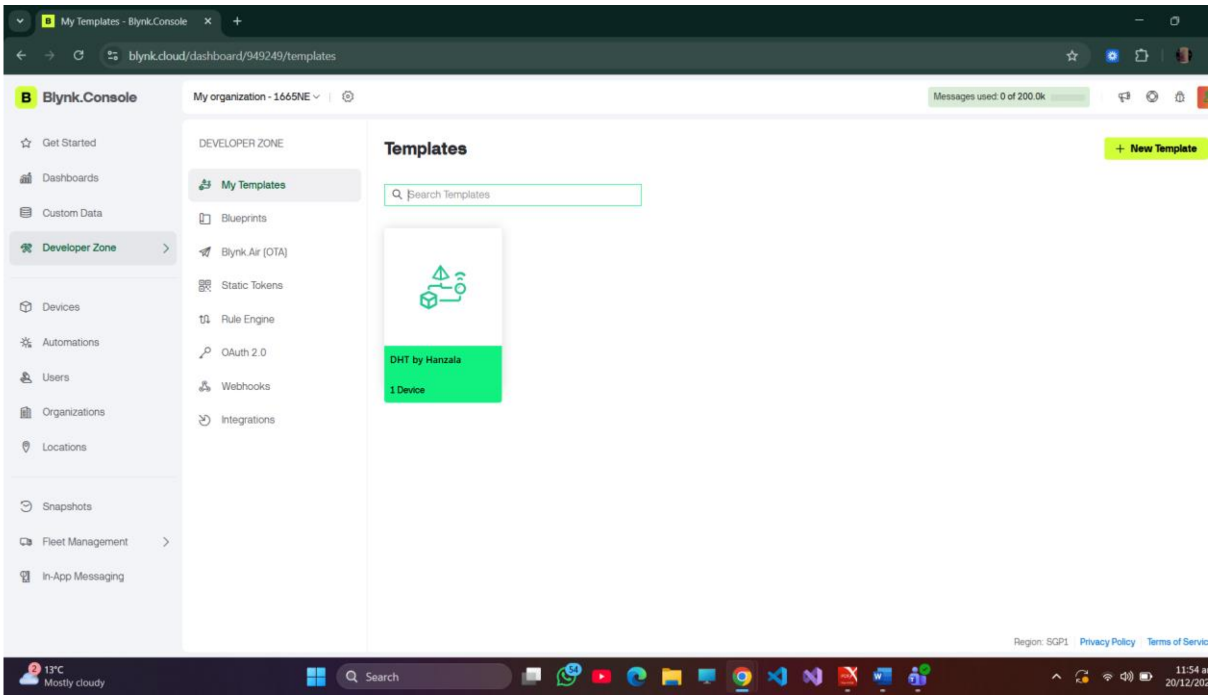
Temperature and humidity values are read from the sensor and transmitted to Blynk Cloud using Virtual Pins.

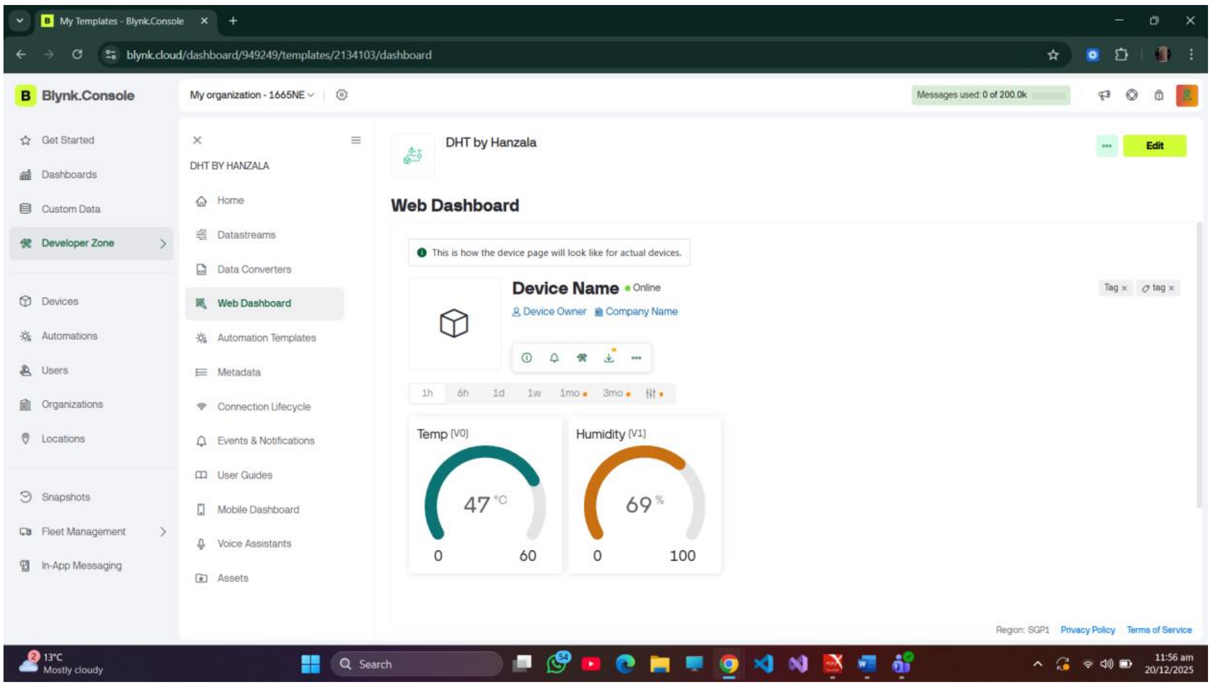
```
Blynk.virtualWrite(V0, temperature);
Blynk.virtualWrite(V1, humidity);
```

Common Configuration Issues and Solutions:

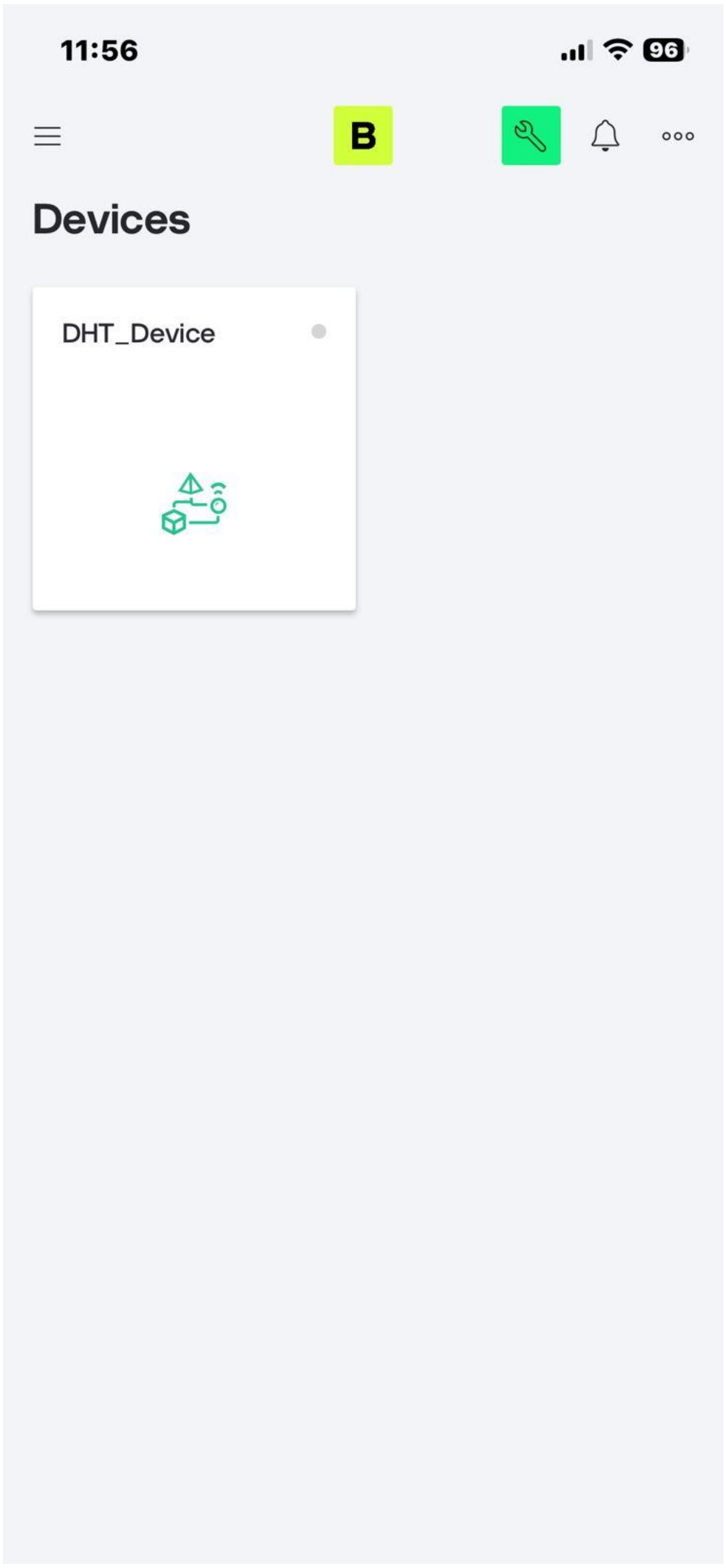
Common issues include Wi-Fi connection failure, wrong credentials, and incorrect sensor selection. These problems are solved by checking network settings, Blynk credentials, and sensor type configuration.

Dashboard of blynk Web application:





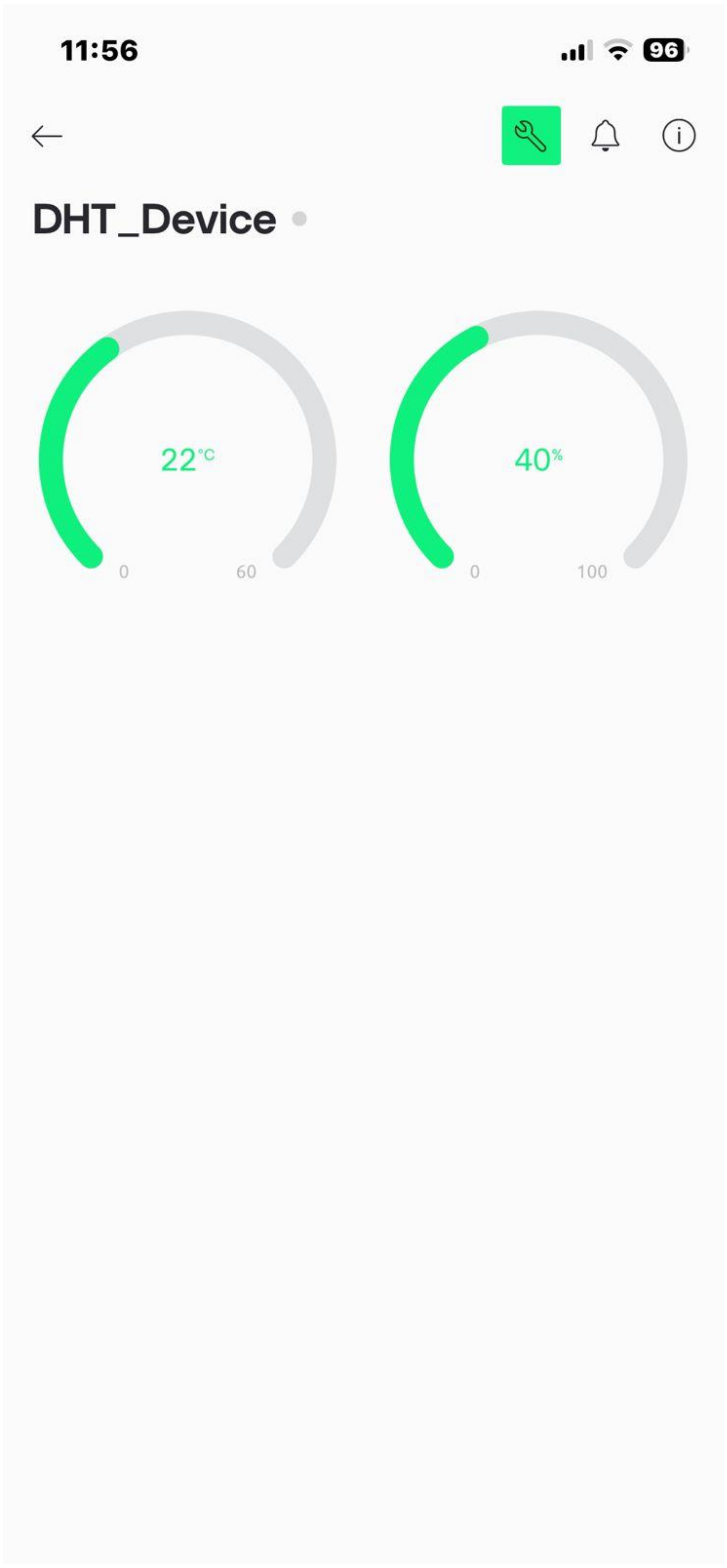
Also I have dashboard in mobile app and all the readings and data flow is also show at mobile app.



11:56

Devices

DHT_Device


11:56
96
DHT_Device
22°C
0
60
40%
0
100

This is my mobile app dashboard.