درس مبانی اینترنت اشیاء

گزارش تمرین ۱

فهرست

سوال ۱

الف) تفاوت‌ها و شباهت‌های کلیدی اینترنت اشیاء

۱) معماری و لایه‌ها:

در اینترنت سنتی عموماً ساختار مبتنی بر مدل کلاینت-سرور و لایه‌های شبکه TCP/IP است. تمرکز اصلی بر انتقال داده میان سرورها و کلاینت‌ها (کاربران انسانی) است.

در مقابل، معماری اینترنت اشیاء به دلیل وجود دستگاه‌های ناهمگون (حسگرها، عملگرها، کنترلرها و غیره) اغلب چندلایه‌تر و دارای بخش‌هایی مانند لایه حسگر (Sensing Layer)، لایه شبکه (Network Layer)، لایه پردازش (Processing/Edge Layer) و لایه کاربرد (Application Layer) است. همچنین وجود گیت‌وی‌ها (Gateway) برای مدیریت دستگاه‌های کم‌توان و پروتکل‌های متنوع، در IoT مرسوم است.

۲) هدف و رویکرد:

اینترنت سنتی بیشتر بر تبادل اطلاعات بین انسان‌ها و دسترسی به سرویس‌های تحت وب متمرکز است.

اینترنت اشیاء بر ارتباط و تبادل داده میان دستگاه‌ها (Machine to Machine - M2M) و تحلیل خودکار داده‌ی حسگرها با هدف نظارت بلادرنگ، بهینه‌سازی فرایندها و ایجاد هوشمندی در سامانه‌ها تمرکز دارد.

۳) عملکرد و محدودیت‌ها:

در اینترنت سنتی، دستگاه‌هایی که به شبکه متصل می‌شوند (نظیر کامپیوتر شخصی، گوشی هوشمند) معمولاً دارای منابع محاسباتی، توان مصرفی و پهنای باند نسبتا مناسبی هستند.

در اینترنت اشیاء، بسیاری از دستگاه‌ها (حسگرها و عملگرها) محدودیت‌های جدی در توان مصرفی، قدرت پردازشی و ظرفیت ذخیره‌سازی دارند. این مسئله باعث می‌شود راهکارهای امنیتی، ارتباطی و پردازشی ویژه‌ای برای آنها در نظر گرفته شود.

شباهت‌ها:

هر دو از زیرساخت‌های شبکه‌ای مشترک (نظیر پروتکل‌های TCP/IP) بهره می‌برند و بستری برای تبادل داده فراهم می‌کنند.

هر دو نیازمند معماری‌های مقیاس‌پذیر و انعطاف‌پذیر هستند تا بتوانند حجم رو به رشد تقاضا را پشتیبانی کنند.

منبع:

فصل ۱ کتاب Springer Handbook of Internet of Things

ب) نوآوری‌ها در تکامل اینترنت اشیاء

ارتباطات بی‌سیم پرتوان و کم‌توان:

پیشرفت در فناوری‌های ارتباطی نظیر LPWAN ،G۵ (مانند LoRa، Sigfox، NB-IoT) و Wi-Fi با مصرف انرژی کم، امکان برقراری ارتباط دستگاه‌های متعدد با پهنای باندهای مختلف را فراهم کرده است. این امر، اتصال تعداد زیاد حسگر و عملگر با هزینه پایین و برد بالا را میسر می‌کند.

همچنین توسعه پروتکل‌های ارتباطی سبک‌وزن به دلیل مصرف پهنای باند و توان کمتر، برای سناریوهای IoT ایده‌آل هستند.

Table 3.6 Comparison of IoT wireless technologies

Technology	Frequency	Data rate	Range	Mobility	Energy cons.
2G/3G	Cellular bands	10 Mb/s	Several km's	High	High
LTE Cat M1	Cellular bands	1–10 Mb/s	Several Km's	High	Medium
NB-IoT	Cellular bands	60 kb/s	Several Km's	Medium	Low
Bluetooth/BLE	2.4 GHz	1/2/3 Mb/s	<100 m	Low	Low
IEEE 802.15.4	2.4 GHz and Sub-GHz	40, 250 kb/s	<100 m	Low	Low
ZigBee	2.4 GHz and Sub-GHz	40, 250 kb/s	<100 m	Low	Low
WirelessHART	2.4 GHz	250 kb/s	<100 m	Low	Medium
ISA100.11a	2.4 GHz	250 kb/s	<100 m	Low	Medium
Z-Wave	Sub-GHz	40 kb/s	~30 m	Low	Low
IEEE 802.11	2.4 GHz, 5 GHz and Sub-GHz	0.1/54 Mb/s	<100 m	Low	Medium
LoRaWAN	Sub-GHz	<250 kb/s	~15 Km	Medium	Low
Sigfox	Sub-GHz	<1 kb/s	Several Km's	Medium	Low

Credit to https://blog.helium.com/802-15-4-wireless-for-internet-of-things-developers-1948fc313b2e

منبع:

فصل ۳ کتاب Intelligent Internet of Things

سیستم‌های نهفته و ریزکنترل‌گرهای پیشرفته:

پیشرفت در طراحی تراشه‌ها و ریزکنترل‌گرها (Microcontrollers) باعث شده است که توان محاسباتی و قابلیت‌های پردازشی بیشتری در ابعاد کوچک و با مصرف انرژی کمتر در دسترس باشد. این موضوع به دستگاه‌های IoT امکان می‌دهد وظایف پیچیده‌تری نظیر پردازش داده‌های محلی و اجرای الگوریتم‌های هوشمند را به‌صورت غیرمتمرکز انجام دهند.

فناوری‌های جدید در ساختار سیستم‌های نهفته با قابلیت مدیریت هوشمند توان (به‌عنوان مثال تکنیک‌های Sleep Mode پیشرفته) باعث افزایش طول عمر باتری در حسگرها و عملگرها می‌شود و کارکرد مداوم در محیط‌های متنوع را تسهیل می‌کند.

منبع:

فصل ۲.۴ و ۲.۵ کتاب Intelligent Internet of Things

سوال ۲

معماری ۳ لایه ای

معماری ۳ لایه ای شامل ۳ لایه‌های Things, Network, Cloud and application می‌شود.

- IOT Things Layer

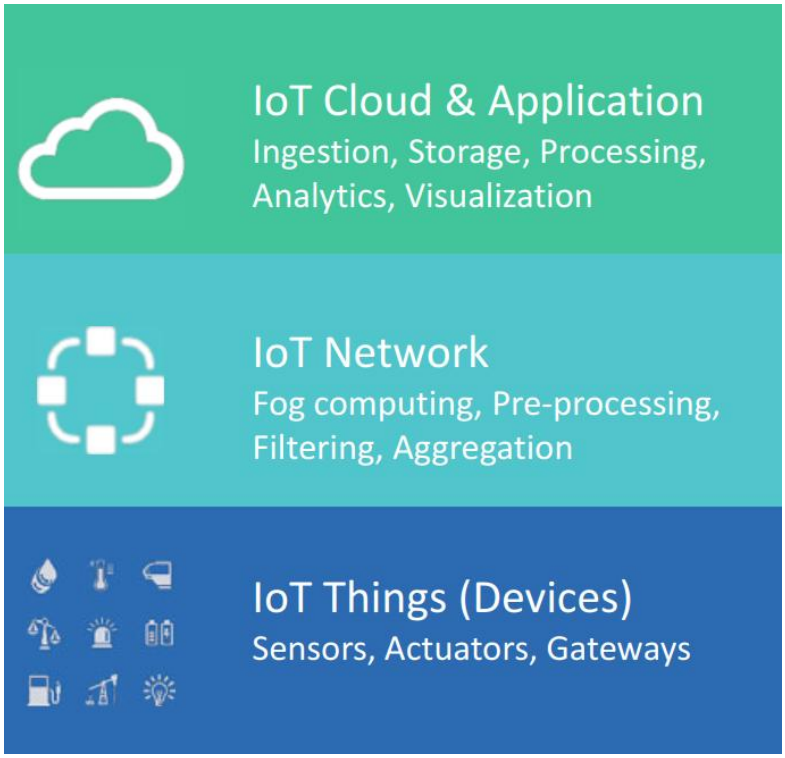
این لایه شامل سنسورها و محرک‌های اینترنت اشیاء می‌شود.

- IOT Network Layer

شامل اجزای شبکه مانند IOT Gateways, Switches, routers که وظیفه انتقال دادن داده ها به موقع و قابل اعتماد را دارند.

- IOT Cloud and Application Layer

این لایه مسئول مدیریت و پردازش دستگاه های اینترنت اشیاء و دو لایه دیگر را دارد. همچنین مسئول تفسیر داده ها از طریق برنامه های کاربردی نرم افزاری و همچنین ادغام با پلتفرم های دیگر برای بهبود ارزش تجاری است.



(Intelligent Internet of Things page 20)

معماری ۵ لایه ای

معماری ۵ لایه ای شامل ۵ لایه Perception Layer, Transport Layer, Processing Layer, Application Layer, Business Layer می‌باشد.

- Perception Layer

این لایه به عنوان لایه حسگر نیز شناخته می شود. مسئولیت شناسایی اشیا و جمع آوری اطلاعات از آنها را بر عهده دارد.

- Transport Layer

اطلاعات جمع آوری شده از اشیاء فیزیکی را از طریق حسگرها حمل و انتقال می دهد.

- Processing Layer

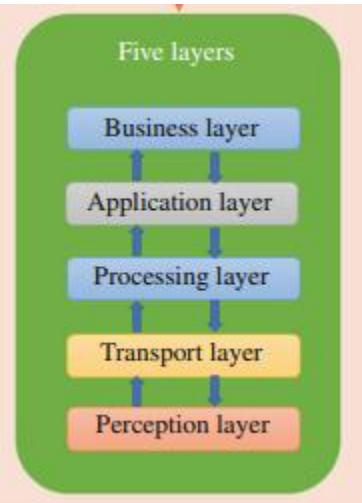
اطلاعات ارسال شده از لایه Transport را جمع آوری می کند. پردازش اطلاعات جمع آوری شده را انجام می دهد و وظیفه حذف اطلاعات اضافی بدون معنی و استخراج اطلاعات مفید را بر عهده دارد.

- Application Layer

این لایه شامل تمام برنامه هایی است که از فناوری اینترنت اشیا استفاده می کنند یا برای آنها اینترنت اشیا به کار گرفته شده است.

- Business Layer

این لایه مانند یک مدیر کل سیستم عمل می کند. مسئولیت مدیریت و کنترل مدل های کاربردی، تجاری و سود سیستم اینترنت اشیا را بر عهده دارد.



(Springer Handbook of Internet of Things page 298)

معماری ۷ لایه ای

معماری ۷ لایه‌ای شامل می‌شود از لایه‌های:

- Physical Devices and Controllers

این لایه شامل دستگاه‌های فیزیکی، حسگرها، محرک‌ها و کنترل‌کننده‌هایی می‌شود که اینترنت اشیا را تشکیل می‌دهند.

- Connectivity

این لایه ای است که به عنوان رسانه ای عمل می کند تا داده های حسگر را از دستگاه ها به لایه های بالایی برساند که در آن داده ها تمیز و تجزیه و تحلیل می شوند.

- Edge Computing

این لایه ای است که در آن پاکسازی، تجمیع و پردازش داده‌ها آغاز می شود. آماده سازی داده ها برای تجزیه و تحلیل و ذخیره سازی بر عهده این مرحله است.

- Data accumulation (storage)

این لایه ای است که داده ها را برای ذخیره در یک پایگاه داده، هر فرمتی که باشد آماده می کند.

- Data abstraction

در این لایه، داده ها سازگار، کامل و معتبر هستند. وظیفه این لایه این است که اطمینان حاصل کند که داده ها می توانند query شوند و یک نتیجه یکپارچه و قابل اعتماد بازگردانده شود.

- Application

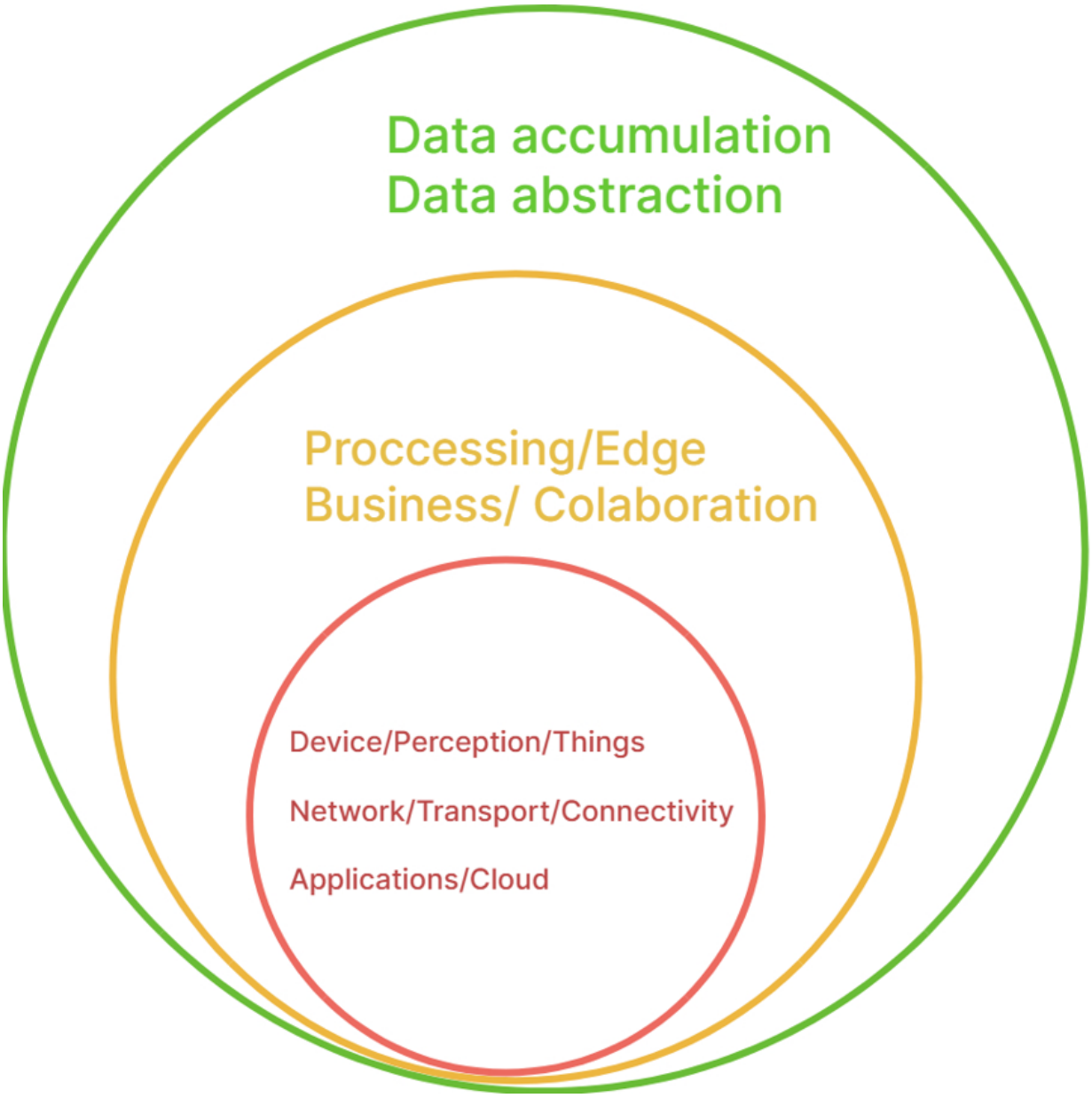
اینجاست که برنامه‌های نرم‌افزاری منحصر بفرد می‌توانند داده‌ها را برای انجام عملکردهای خاص، مانند گزارش‌دهی، نظارت، کنترل دستگاه‌ها، تجسم‌ها و تجزیه و تحلیل‌ها جستجو کنند.

- Collaboration and processes

این لایه ای است که از خروجی های برنامه های نرم افزاری لایه قبلی استفاده می کند.

(Intelligent Internet of Things page 19)

الف) نمودار ون اشتراکات و تفاوت‌ها



ب) معماری اینترنت اشیاء در کاربرد کشاورزی

در حوزه کشاورزی هوشمند (Smart Agriculture)، معمولاً ویژگی‌ها و نیازهای زیر را در نظر می‌گیرند:

- تعداد زیاد حسگرهای ساده (اندازه‌گیری دما، رطوبت، نور، رطوبت خاک و ...) با محدودیت توان و ارتباط بی‌سیم.
- گستره جغرافیایی بزرگ (مزارع وسیع، باغ‌ها، گلخانه‌ها).
- نیاز به ترکیب داده و پردازش در محل و گاهی ارسال به زیرساخت ابری.
- سادگی در مدیریت و استقرار (کشاورزان یا اپراتورهای محلی لزوماً متخصص شبکه‌های پیچیده نیستند).

در این شرایط معماری ۵ لایه مناسب‌ترین گزینه است.

لایه ادراک: برای جمع‌آوری داده‌های محیطی مانند دما، رطوبت، نور، خاک و...

لایه شبکه: انتقال این داده‌ها از مزرعه به مراکز تحلیل

لایه پردازش: تحلیل بلادرنگ داده‌ها (مثلاً تشخیص نیاز به آبیاری یا آفت‌زدایی)

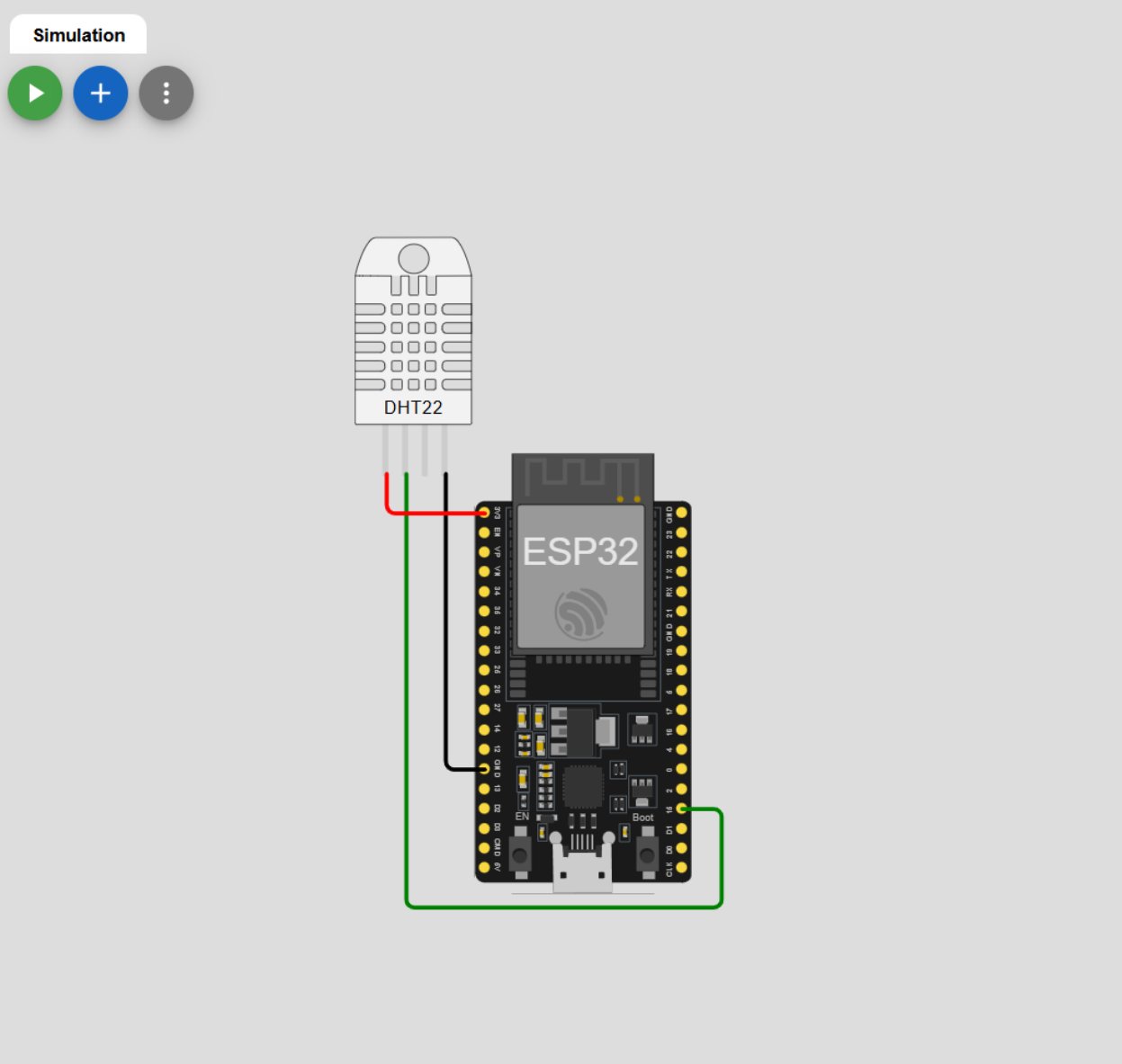
لایه کاربرد: ارائه اطلاعات به کشاورز از طریق اپلیکیشن یا داشبورد

لایه کسب‌وکار: تصمیم‌گیری بر اساس تحلیل‌ها (مثلاً برنامه‌ریزی برای کوددهی یا برداشت)

معماری ۷ لایه ممکن است در محیط‌های صنعتی پیشرفته کاربرد داشته باشد، اما در کشاورزی (به‌ویژه در کشورهایی با زیرساخت محدود) ساده‌سازی مهم است، و معماری ۵ لایه تعادلی مناسب میان پیچیدگی و کارایی ارائه می‌دهد.

بخش ۳

طراحی پروژه در wokwi



رطوبت و دماسنج DH22 را به ESP32 متصل می‌کنیم و پین ۱۵ کنترلر را به عنوان پین داده در نظر گرفته شده است.

```
#include <DHT.h>

#define DHTPIN          15
#define DHTTYPE         DHT22

DHT dht(DHTPIN, DHTTYPE);
```

حال کافی است هر ۵ ثانیه مقادیر خوانده‌شده دما و رطوبت را به بروکر ارسال کنیم.

بروکر MQTT

برای اتصال به بروکر و انتشار مقادیر در یک topic، از کتابخانه PubSubClient استفاده شده است. همچنین به منظور دسترسی به اینترنت از کتابخانه WiFi استفاده شده است.

```
#include <WiFi.h>
#include <PubSubClient.h>

#define WIFI_SSID       "Wokwi-GUEST"
#define WIFI_PASSWORD   ""
#define WIFI_CHANNEL    6

#define MQTT_CLIENT_ID  "wokwi"
#define MQTT_BROKER     "test.mosquitto.org"
#define MQTT_PORT       1883
#define MQTT_TOPIC      "/g5/sensor/data"

void callback(char* topic, byte* payload, unsigned int length);
void reconnect();

WiFiClient espClient;
PubSubClient client(espClient);

void setup(void) {
  Serial.begin(115200);
  dht.begin();

  Serial.println(F("DHT22 begin."));

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD, WIFI_CHANNEL);
  Serial.print("Connecting to WiFi ");
  Serial.print(WIFI_SSID);

  while (WiFi.status() != WL_CONNECTED) {
    delay(100);
    Serial.print(".");
  }
  Serial.println(" Connected!");

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

```
  client.setServer(MQTT_BROKER, MQTT_PORT);
  client.setCallback(callback);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  String payload = String(t) + "," + String(h);

  Serial.print(F("Temperature: "));
  Serial.print(t);
  Serial.print(F("°C, Humidity: "));
  Serial.print(h);
  Serial.println(F("%"));

  client.publish(MQTT_TOPIC, payload.c_str());

  delay(5000);
}


void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i=0;i<length;i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}

void reconnect() {
  while (!client.connected()) {
```

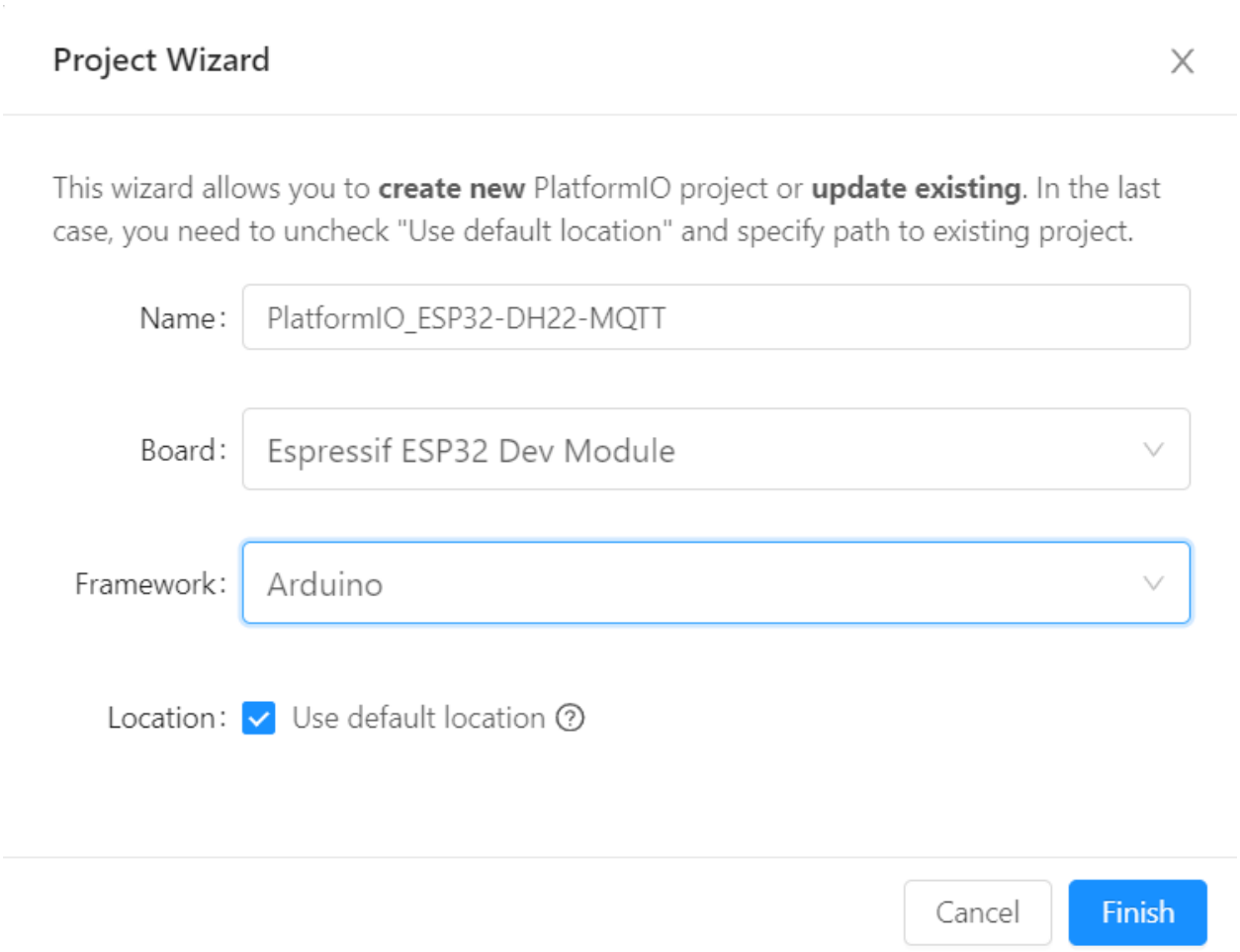
```
    Serial.print("Attempting MQTT connection...");

    String clientId = MQTT_CLIENT_ID;
    clientId += String(random(0xffff), HEX);

    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");

      delay(5000);
    }
  }
}
```

پروژه PlatformIO



به منظور سهولت اجرا، برنامه در قالب یک پروژه PlatformIO تعریف شده است. برای انجام شبیه‌سازی به دو فایل زیر نیاز است.

محتوای فایل diagram.json از پروژه تحت وب wokwi برداشته شده است. محتوای فایل wokwi.toml شامل آدرس فایل‌های اجرایی برنامه است.

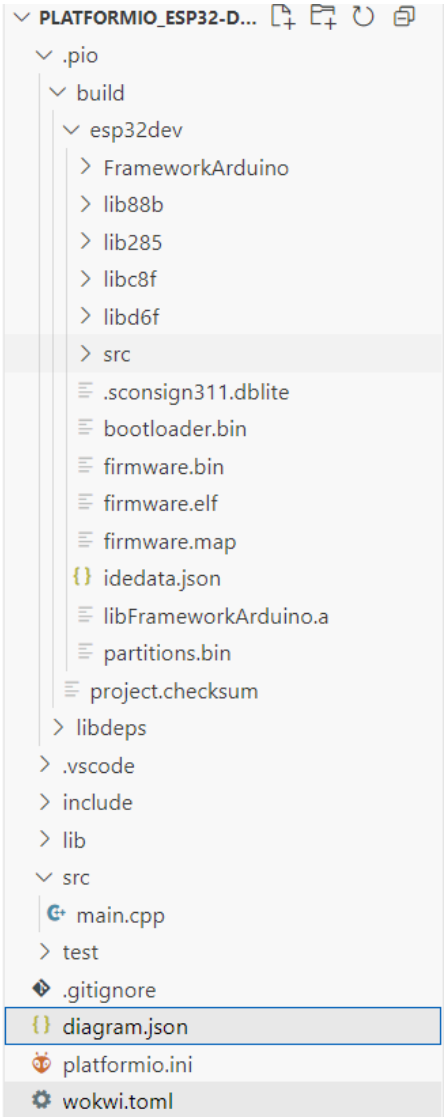


diagram.json

```
{
    "version": 1,
    "author": "Erfan Abedi",
    "editor": "wokwi",
    "parts": [
      { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": -9.6, "left":
14.44, "attrs": {} },
```

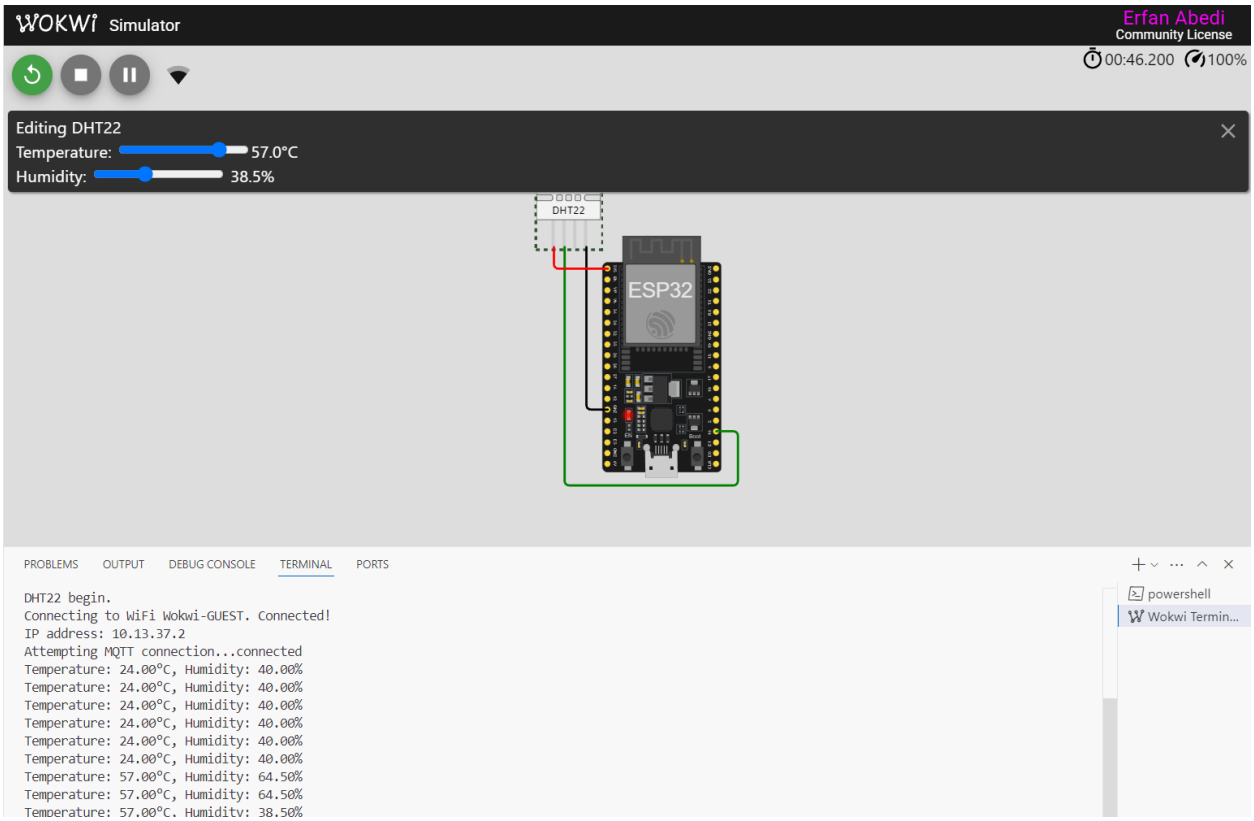
```
      { "type": "wokwi-dht22", "id": "dht1", "top": -114.9, "left": -43.8,
"attrs": {} }
    ],
    "connections": [
      [ "esp:TX", "$serialMonitor:RX", "", [] ],
      [ "esp:RX", "$serialMonitor:TX", "", [] ],
      [ "dht1:SDA", "esp:15", "green", [ "v211.2", "h153.7", "v-48" ] ],
      [ "dht1:GND", "esp:GND.1", "black", [ "v0" ] ],
      [ "dht1:VCC", "esp:3V3", "red", [ "v0" ] ]
    ],
    "dependencies": {}
}
```

wokwi.toml

```
[wokwi]
version = 1
'firmware = '.pio/build/esp32dev/firmware.bin
'elf = '.pio/build/esp32dev/firmware.elf
```

پس از اجرای دستور build، فایل diagram.json را باز و شبیه‌سازی را آغاز می‌کنیم.

نتایج انتشار مقادیر دما و رطوبت

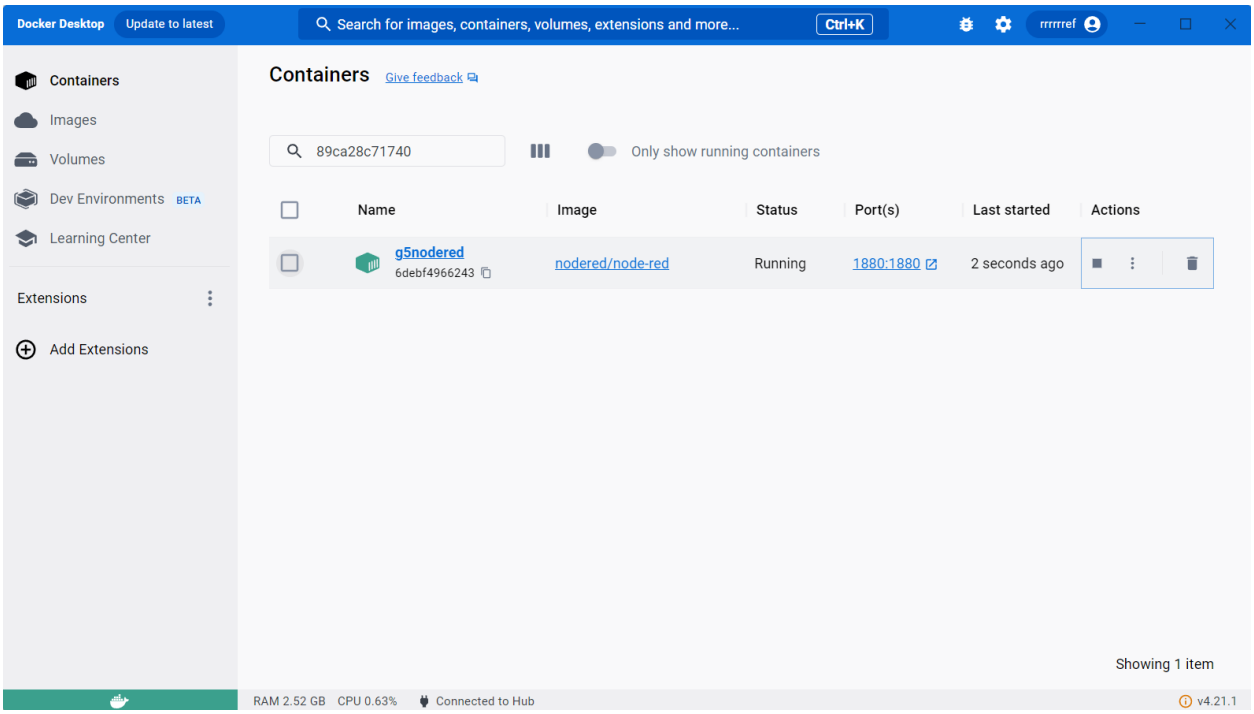


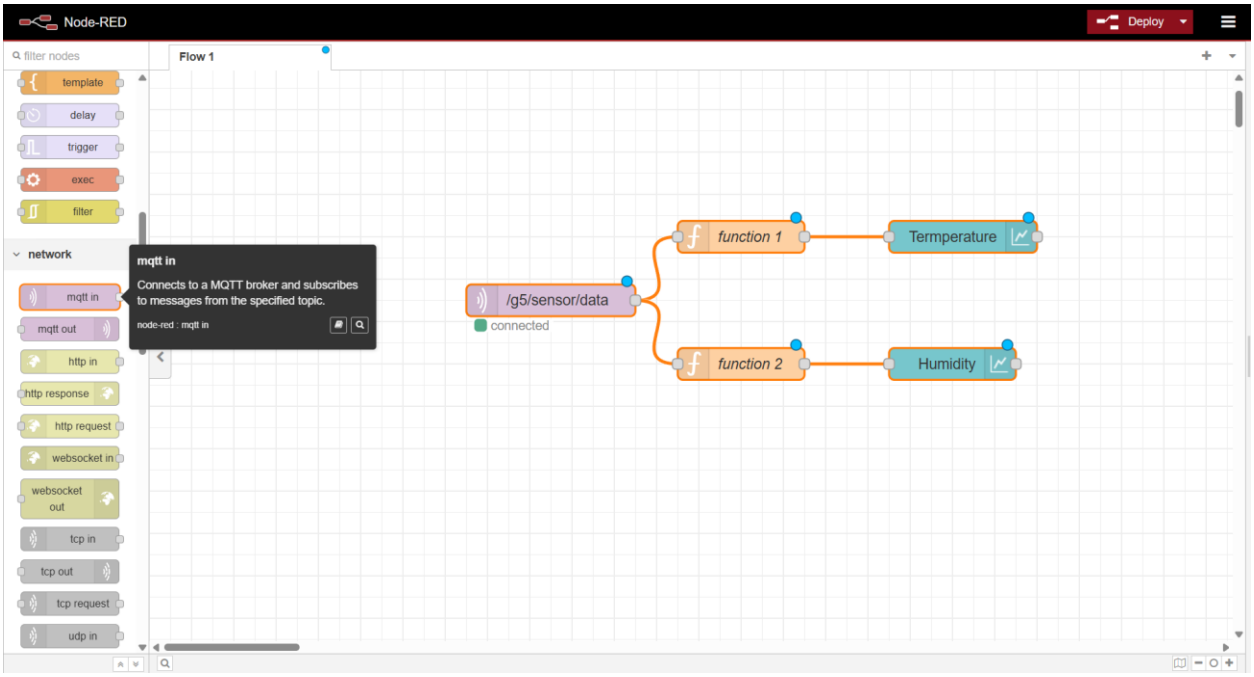
برای مشاهده پیام‌ها با استفاده از کلاینت mosquitto در تاپیک بالا عضو می‌شویم.

```
PS C:\Users\erfuu> mosquitto_sub -h test.mosquitto.org -t "/g5/sensor/data" -v
/g5/sensor/data 24.00,40.00
/g5/sensor/data 24.00,40.00
/g5/sensor/data 24.00,40.00
/g5/sensor/data 57.00,64.50
/g5/sensor/data 57.00,64.50
/g5/sensor/data 57.00,38.50
/g5/sensor/data 57.00,38.50
```

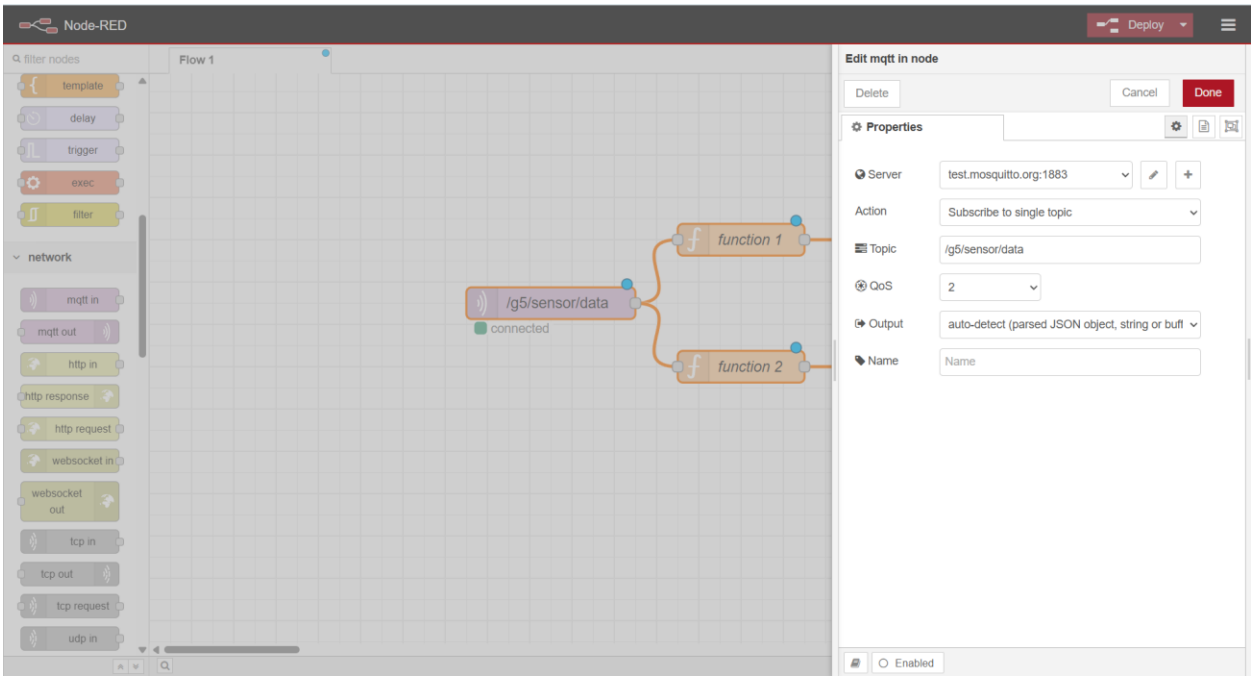
نمایش نمودار در محیط Node-RED

به منظور نمایش مقادیر بر روی یک نمودار در یک داشبورد تحت وب از برنامه داشبورد ساز Node-RED بر بستر داکر استفاده شده است.

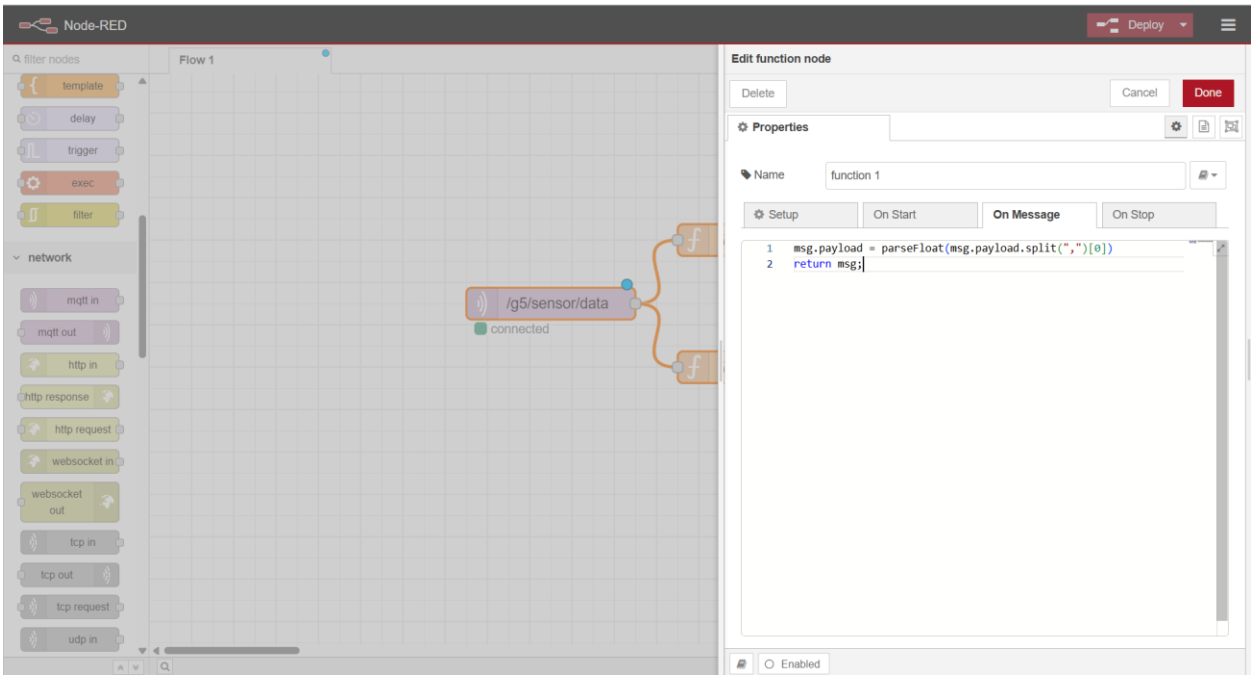




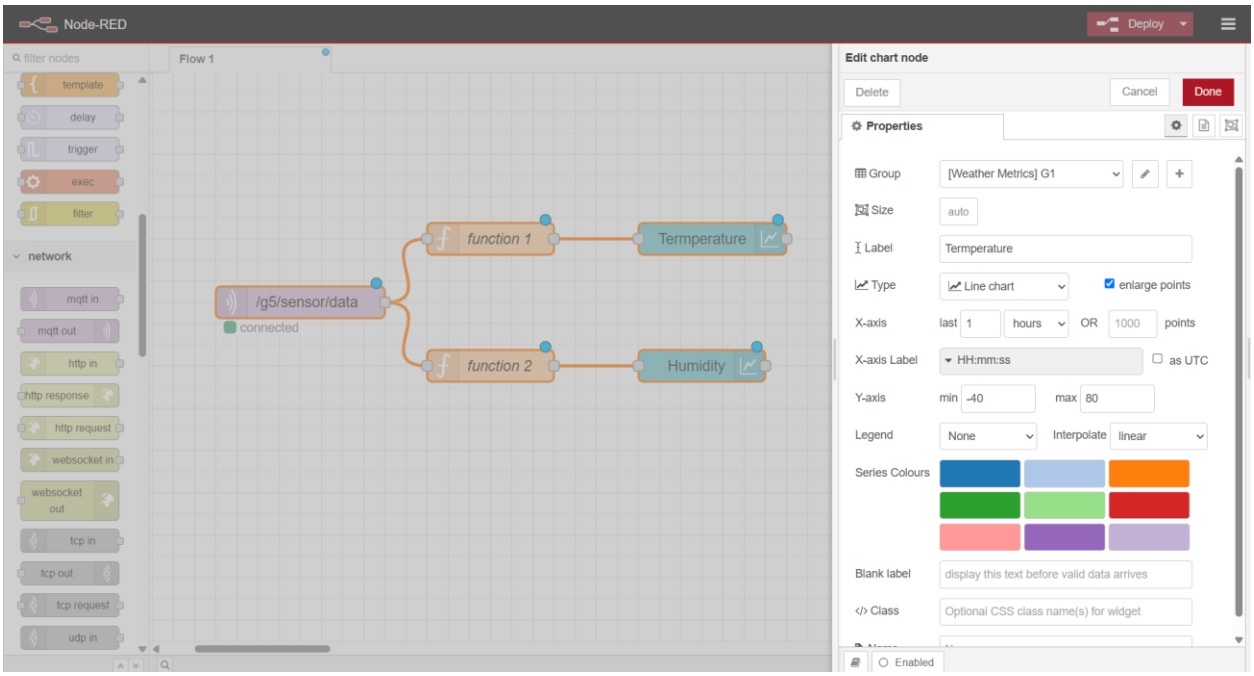
برای اتصال به بروکر، از یک بلوک mqtt in استفاده شده است.

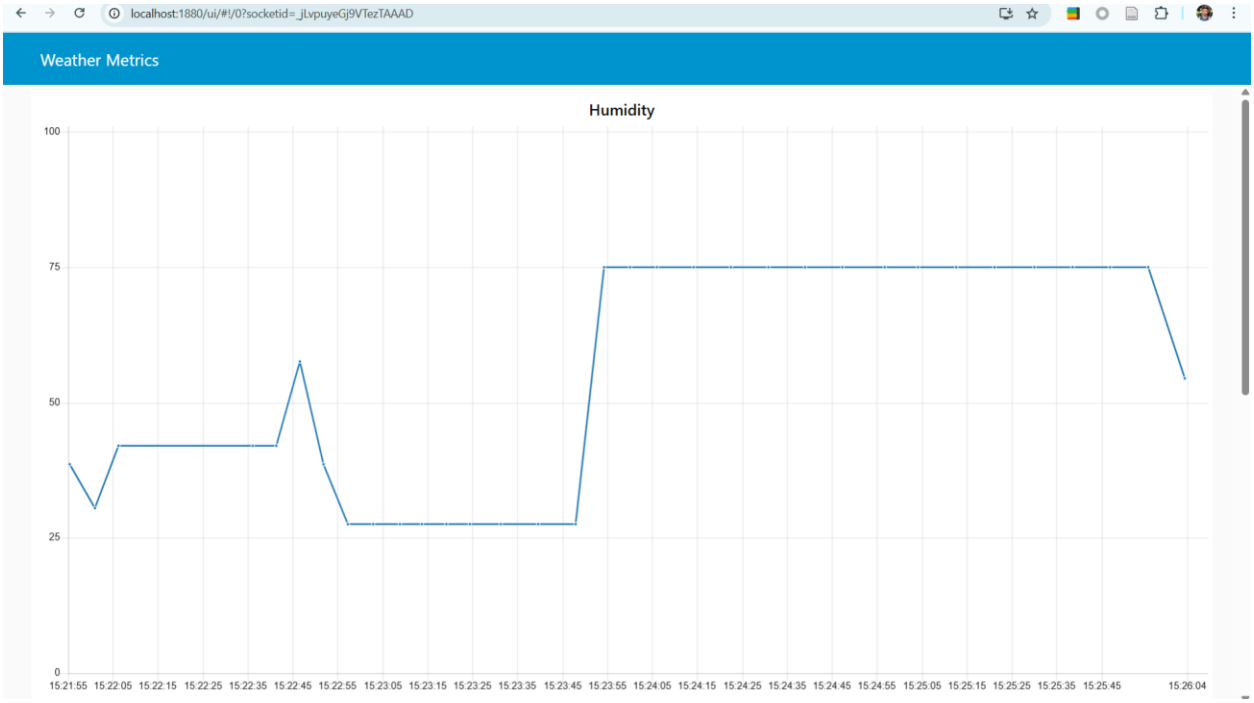


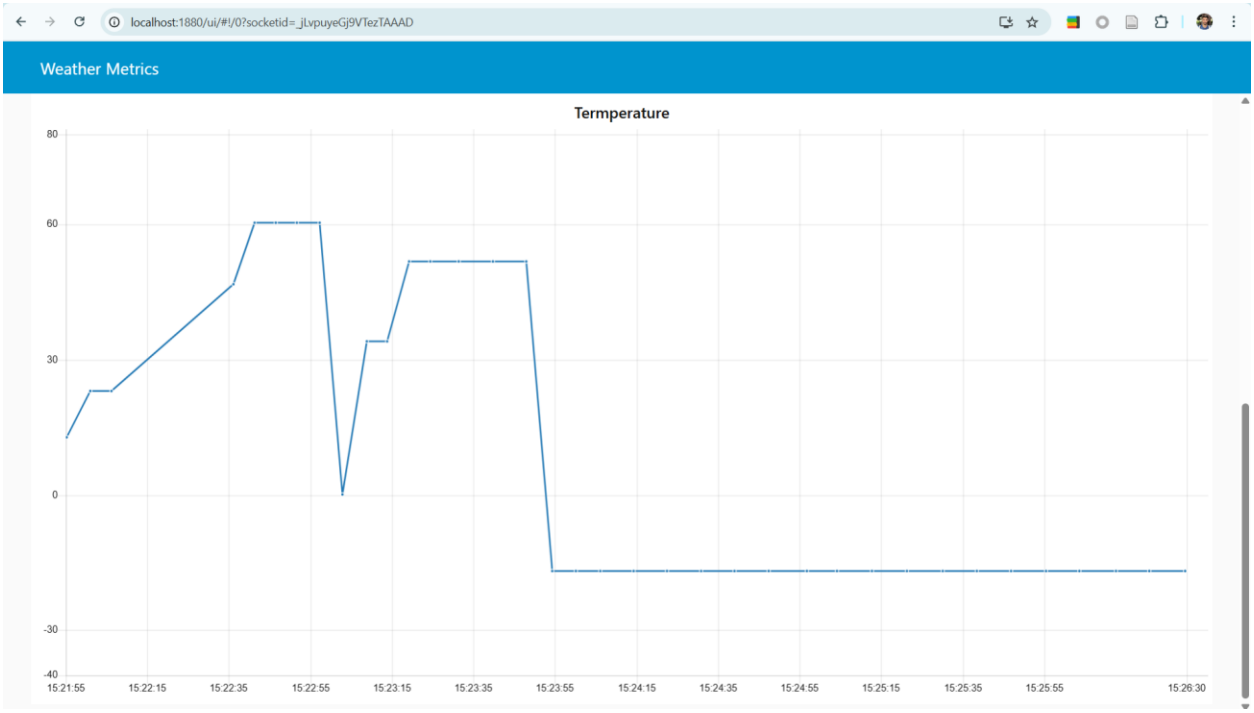
برای جداسازی مقادیر دما و رطوبت از پیام دریافت شده، از بلوک function استفاده شده است.



و در نهایت برای نمایش آن بر روی نمودار از بلوک chart استفاده شده است.




localhost:1880/ui/#!/0?socketid=_jLvpuyeGj9VTezTAAAD
Weather Metrics
Humidity
100
75
50
25
0
15:21:55 15:22:05 15:22:15 15:22:25 15:22:35 15:22:45 15:22:55 15:23:05 15:23:15 15:23:25 15:23:35 15:23:45 15:23:55 15:24:05 15:24:15 15:24:25 15:24:35 15:24:45 15:24:55 15:25:05 15:25:15 15:25:25 15:25:35 15:25:45 15:26:04


localhost:1880/ui/#!/0?socketid=_jLvpuyeGj9VTezTAAAD
Weather Metrics
Termperature
80
60
30
0
-30
-40
15:21:55 15:22:15 15:22:35 15:22:55 15:23:15 15:23:35 15:23:55 15:24:15 15:24:35 15:24:55 15:25:15 15:25:35 15:25:55 15:26:30