به نام خدا

امیرحسین راعقی
400522373
مرتضی ملکی نژاد شوشتری
400522211

پروژه درس IOT

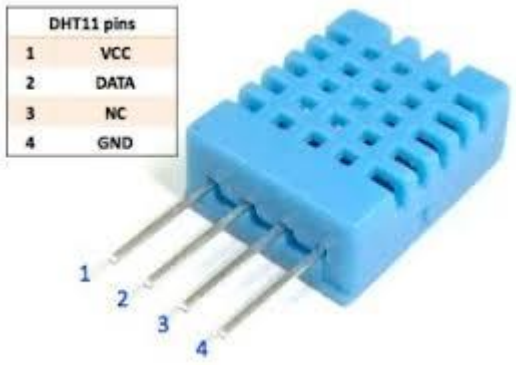
پایش دمای محیط و ارسال داده با MQTT به سرور

مقدمه

در این پروژه ما داده سنسور DHT22 را از طریق دیتا کالکتور و گیت وی ESP32 با پروتکل MQTT به یک بروکر آنلاین ارسال کرده و آن داده را در داشبورد تحت وب نمایش می دهیم.

پیاده سازی

- در ابتدا دما را از سنسور دریافت کرده و نمایش می دهیم لازم به ذکر است که پایه های پین سنسور DHT به شکل مقابل می باشد:



- کد بخش دریافت دما از سنسور و نمایش آن به صورت سریال مانیتور:

```
#include "DHT.h"

#define DHTPIN 13
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println(F("Show DHT Temp And Humidity On Monitor:"));

  dht.begin();
}

void loop() {
  delay(2000);

  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  float temperatureFarenhait = dht.readTemperature(true);

  if (isnan(humidity) || isnan(temperature) ||
isnan(temperatureFarenhait)){
    Serial.println(F("Failed To Read From Sensor"));
    return;
  }

  float heatIndexCelsius = dht.computeHeatIndex(temperature,
humidity, false);

  Serial.print(F("Humidity: "));
  Serial.print(humidity);
  Serial.print(F("%  Temperature: "));
  Serial.print(temperature);
  Serial.print(F("°C "));
  Serial.print(temperatureFarenhait);
  Serial.print(F("°F  Heat index: "));
  Serial.print(heatIndexCelsius);
  Serial.print(F("°C \r\n"));

}

```

- در این بخش ما داده های دما، رطوبت و شاخص گرما را نمایش می دهیم. نمایش آن به شکل زیر می شود:

```
17:32:06.540 -> Humidity: 48.70%  Temperature: 28.40°C 83.12°F
Heat index: 28.78°C
17:32:08.560 -> Humidity: 48.80%  Temperature: 28.40°C 83.12°F
Heat index: 28.79°C
17:32:10.528 -> Humidity: 48.80%  Temperature: 28.40°C 83.12°F
Heat index: 28.79°C
17:32:12.562 -> Humidity: 48.80%  Temperature: 28.40°C 83.12°F
Heat index: 28.79°C
17:32:14.547 -> Humidity: 48.70%  Temperature: 28.40°C 83.12°F
Heat index: 28.78°C
17:32:16.573 -> Humidity: 48.70%  Temperature: 28.40°C 83.12°F
Heat index: 28.78°C
17:32:18.552 -> Humidity: 48.60%  Temperature: 28.40°C 83.12°F
Heat index: 28.77°C
```

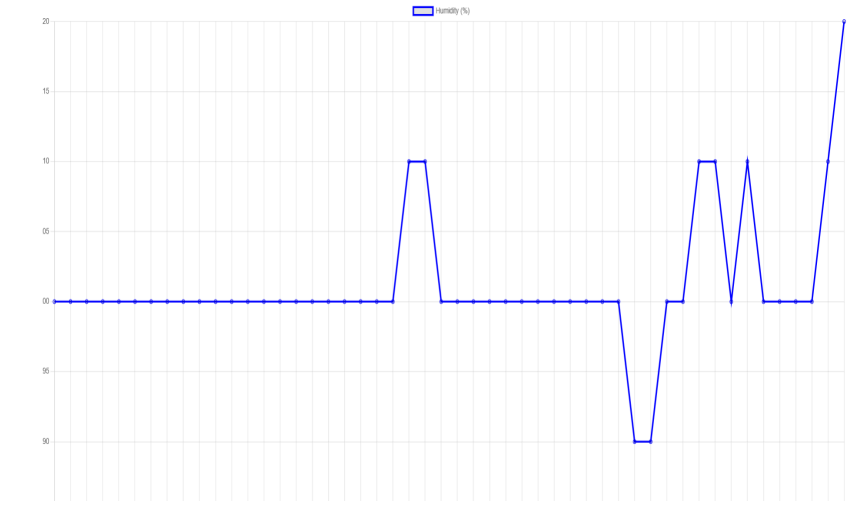
- در مرحله بعدی ما داده های دریافتی را به یک Topic تحت یک پروکر آنلاین ارسال می کنیم. در ادامه نمایشی از قطعه کد مورد نظر و نمایش خروجی آن نشان میدهیم:

```ino
#include "DHT.h"
#include <WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>

#define DHTPIN 13
#define DHTTYPE DHT22

const char* ssid = "TestNetwork";
const char* password = "TestNetwork1234";
const char* mqttServer = "broker.emqx.io";

DHT dht(DHTPIN, DHTTYPE);
WiFiClient espClient;
PubSubClient client(espClient);
StaticJsonDocument<200> doc;
void setupWifi(){
  delay(10);
  WiFi.begin(ssid, password);
  Serial.print(F("Connecting to Wifi"));
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
}
void connectMQTT() {
  if (!client.connected()) {
    Serial.print("Try To Connect MQTT... ");

    if (client.connect("ESP32ClientARMM")) {
      Serial.println("Connected!");
    } else {
      Serial.print("Failed, rc=");
      Serial.println(client.state());
      Serial.println(" Will retry in next loop iteration.");
    }
  }
}
void setup() {
  Serial.begin(9600);
  Serial.println(F("Show DHT Temp And Humidity On Monitor:"));
  setupWifi();
  client.setServer(mqttServer, 1883);
  dht.begin();
}

void loop() {
  client.loop();
  delay(2000);
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  float temperatureFarenhait = dht.readTemperature(true);

  if (isnan(humidity) || isnan(temperature) ||
isnan(temperatureFarenhait)) {
    Serial.println(F("Failed To Read From Sensor"));
    return;
  }

  float heatIndexCelsius = dht.computeHeatIndex(temperature,
humidity, false);

  doc["humidity"] = humidity;
  doc["temperature"] = temperature;
  doc["temperatureFarenhait"] = temperatureFarenhait;
  doc["heatIndexCelsius"] = heatIndexCelsius;
  doc["timeStamp"] =  millis();
  //Serial.print(F("Humidity: "));
  //Serial.print(humidity);
  //Serial.print(F("%  Temperature: "));
  //erial.print(temperature);
  //Serial.print(F("°C "));
  //Serial.print(temperatureFarenhait);
  //Serial.print(F("°F  Heat index: "));
  //Serial.print(heatIndexCelsius);
  //Serial.print(F("°C \r\n"));
  //Serial.print("===");
  //Serial.print("\n\r Connected To WiFi.");
  //Serial.print("\n\r IP Address:");
  //Serial.print(WiFi.localIP());
  //Serial.print("\n\r === \r\n");
  if (!client.connected()){
    connectMQTT();
  }
  if (client.connected()){
    char jsonBuffer[256];
    serializeJson(doc, jsonBuffer);
    client.publish("IOTProject4032ARMM", jsonBuffer);
    Serial.println("JSON published: ");
    Serial.println(jsonBuffer);
  }
}

```

- در این مرحله با کمک Flask در داشبورد تحت وب نمایش می دهیم. با قطعه کد زیر هم از MQTT داده های سنسور را دریافت کرده و نمایش می دهیم:

```
from flask import Flask, jsonify, render_template_string
import paho.mqtt.client as mqtt
import threading
import time
import json

app = Flask(__name__)
data_points = []

MQTT_BROKER = "broker.emqx.io"
MQTT_PORT = 1883
MQTT_TOPIC = "IOTProject4032ARMM"

def on_message(client, userdata, msg):
    try:
        payload = json.loads(msg.payload.decode())
        data_points.append({
            "time": time.strftime("%H:%M:%S"),
            "temp": payload.get("temperature"),
            "hum": payload.get("humidity"),
            "tempF": payload.get("temperatureFarenhait"),
            "heat": payload.get("heatIndexCelsius")
        })
        if len(data_points) > 50:
            data_points.pop(0)
    except:
        pass

def mqtt_thread():
    client = mqtt.Client()
    client.on_message = on_message
    client.connect(MQTT_BROKER, MQTT_PORT, 60)
    client.subscribe(MQTT_TOPIC)
    client.loop_forever()

threading.Thread(target=mqtt_thread, daemon=True).start()

@app.route("/")
def index():
    template = """
    <!DOCTYPE html>
    <html>
    <head>
      <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    </head>
    <body>
      <h2>ESP32 DHT Dashboard</h2>
      <canvas id="tempChart" height="150"></canvas>
      <canvas id="humChart" height="150"></canvas>
      <script>
        let tempChart, humChart;

        async function fetchData() {
          const resp = await fetch('/data');
          const dataPoints = await resp.json();
          const labels = dataPoints.map(d => d.time);

          const tempData = {
            labels: labels,
            datasets: [
              {label: 'Temperature (°C)', data: dataPoints.map(d=>d.temp),
borderColor:'red', fill:false},
              {label: 'Heat Index (°C)', data: dataPoints.map(d=>d.heat),
borderColor:'orange', fill:false}
            ]
          };

          const humData = {
            labels: labels,
            datasets: [
              {label: 'Humidity (%)', data: dataPoints.map(d=>d.hum),
borderColor:'blue', fill:false}
            ]
          };

          if (!tempChart) {
            tempChart = new Chart(document.getElementById('tempChart'),
{type:'line', data:tempData});
            humChart = new Chart(document.getElementById('humChart'),
{type:'line', data:humData});
          } else {
            tempChart.data = tempData;
            humChart.data = humData;
            tempChart.update();
            humChart.update();
          }
        }

        // Fetch every 3 seconds
        setInterval(fetchData, 3000);
        fetchData(); // initial fetch
      </script>
    </body>
    </html>
    """
    return render_template_string(template)

@app.route("/data")
def data():
    return jsonify(data_points)

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)

```

- که نمایش آن به شکل مقابل می شود:



ESP32 DHT Dashboard

Disconnected!

ESP32 DHT Dashboard

Connected

توضیحات دقیق کد سخت افزاری و نرم افزاری در این پروژه به شکل زیر است:

نرم افزار:

نصب کتابخانه‌ها

برای اجرای برنامه نیاز داریم:

```
pip install flask paho-mqtt
```

- **Flask**: فریمورک سبک Python برای ایجاد وب‌سرور.
- **paho-mqtt**: کتابخانه Python برای اتصال و دریافت پیام‌های MQTT.

تعریف متغیرهای اصلی

```python
data_points = []
last_msg_time = 0  # زمان دریافت آخرین پیام (سرور)
MQTT_BROKER = "broker.emqx.io"
MQTT_PORT = 1883
MQTT_TOPIC = "IOTProject4032ARMM"
DISCONNECT_THRESHOLD = 10  # ثانیه برای تشخیص قطع اتصال

```

- data_points: آرایه‌ای برای ذخیره آخرین ۵۰ پیام دریافتی.
- last_msg_time: زمان دریافت آخرین پیام، برای تشخیص قطع اتصال.

- DISCONNECT_THRESHOLD: اگر بیش از این زمان پیام جدید نمیاد، وضعیت قطع اتصال اعلام می‌شود.

دریافت پیام‌های MQTT

Callback برای پیام‌های MQTT:

```python
def on_message(client, userdata, msg):
    global last_msg_time
    try:
        payload = json.loads(msg.payload.decode())
        server_time = time.time()  # زمان دریافت روی سرور
        last_msg_time = server_time

        esp_time = payload.get("timeStamp", 0)  # timestamp از ESP32
        latency = server_time*1000 - esp_time  # تأخیر (ms)

        freq = None
        if data_points:
            freq = esp_time - data_points[-1]['esp_time']  # فاصله بین پیام‌ها

        data_points.append({
            "time": time.strftime("%H:%M:%S"),
            "temp": payload.get("temperature"),
            "hum": payload.get("humidity"),
            "tempF": payload.get("temperatureFarenhait"),
            "heat": payload.get("heatIndexCelsius"),
            "esp_time": esp_time,
            "latency": latency,
            "freq": freq,
            "server_time": server_time
        })

        if len(data_points) > 50:
            data_points.pop(0)  # فقط آخرین ۵۰ پیام را نگه می‌دارد
    except:
        pass

```

توضیح فیلدها:

فیلد	توضیح
time	زمان دریافت روی سرور به صورت ساعت دقیقه ثانیه
temp	دمای محیط به صورت سانتی گراد
hum	رطوبت محیط
tempF	دمای محیط به صورت فارنهایت
heat	شاخص حرارتی
esp_time	زمان ارسال از ESP
latency	اختلاف زمان سرور و ESP
freq	فاصله بین 2 پیام متوالی
server_time	زمان دریافت پیام روی سرور

راه‌اندازی MQTT در Thread جداگانه

```
def mqtt_thread():
    client = mqtt.Client()
    client.on_message = on_message
    client.connect(MQTT_BROKER, MQTT_PORT, 60)
    client.subscribe(MQTT_TOPIC)
    client.loop_forever()

threading.Thread(target=mqtt_thread, daemon=True).start()

```

اجرای MQTT در **Thread جداگانه** تا Flask بتواند همزمان وب‌سرور را اجرا کند.

پیام‌ها به صورت خودکار دریافت و در data_points ذخیره می‌شوند.

مسیر Flask برای داشبورد

مسیر اصلی /:

- **وظیفه:** نمایش داشبورد با ۴ نمودار و وضعیت اتصال.

```
@app.route("/")
def index():
    template = """ ... HTML + JS ... """
    return render_template_string(template,
threshold=DISCONNECT_THRESHOLD)

```

از **Chart.js** برای نمودارها استفاده می‌کنیم:

1. Temperature / Heat Index
2. Humidity
3. (Send Frequency (ms
4. (Latency (ms

با **AJAX fetch** هر ۳ ثانیه داده‌ها از مسیر /data گرفته می‌شود و نمودارها به‌روزرسانی می‌شوند.

تشخیص قطع اتصال:

بررسی می‌کنیم که آخرین پیام دریافتی بیش از DISCONNECT_THRESHOLD ثانیه قبل نبوده باشد.
وضعیت اتصال در بالای داشبورد نمایش داده می‌شود:

```
const disconnected = (now - lastMsgTime) > {{ threshold }};
statusDiv.innerHTML = disconnected
    ? "<span style='color:red'>Disconnected!</span>"
    : "<span style='color:green'>Connected</span>";

```

مسیر Flask برای داده‌ها /data:

```
@app.route("/data")
def data():
    return jsonify(data_points)

```

- این مسیر آخرین ۵۰ پیام را به صورت JSON برمی‌گرداند تا نمودارها در مرورگر به‌روزرسانی شوند.

اجرای Flask

```
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)

```

Flask روی تمام آی‌پی‌های دستگاه (0.0.0.0) و پورت ۵۰۰۰ اجرا می‌شود.

Debug فعال است برای مشاهده خطاها و توسعه راحت‌تر.

Flask روی تمام آی‌پی‌های دستگاه (0.0.0.0) و پورت ۵۰۰۰ اجرا می‌شود.

Debug فعال است برای مشاهده خطاها و توسعه راحت‌تر.

بخش سخت افزاری:

1. کتابخانه‌ها و تعاریف اولیه

```
#include "DHT.h"           // برای خواندن دما و رطوبت از DHT11 / DHT22
#include <WiFi.h>          // اتصال به شبکه WiFi
#include <PubSubClient.h>  // پروتکل MQTT
#include <ArduinoJson.h>   // تولید JSON برای ارسال پیام
```

پین و نوع سنسور

```
#define DHTPIN 13
#define DHTTYPE DHT22

```

DHTPIN → پین متصل به سنسور DHT22

DHTTYPE → نوع سنسور (DHT22 در این پروژه)

2. اطلاعات شبکه و MQTT

```
const char* ssid = "TestNetwork";                // نام شبکه WiFi
const char* password = "TestNetwork1234";        // رمز شبکه
const char* mqttServer = "broker.emqx.io";       // آدرس سرور MQTT

```

3. تعریف اشیا و متغیرها

```
DHT dht(DHTPIN, DHTTYPE);          // شی سنسور DHT
WiFiClient espClient;              // اتصال شبکه ESP32
PubSubClient client(espClient);    // MQTT client
StaticJsonDocument<200> doc;       // ساخت JSON برای ارسال داده

```

4. اتصال به WiFi

```
void setupWifi(){
  delay(10);
  WiFi.begin(ssid, password);
  Serial.print(F("Connecting to Wifi"));
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
}

```

تابع setupWifi تلاش می‌کند ESP32 به شبکه WiFi متصل شود.

اگر اتصال برقرار نباشد، در حلقه باقی می‌ماند تا اتصال برقرار شود.

Serial.print وضعیت اتصال را روی Serial Monitor نشان می‌دهد.

5. اتصال به MQTT

```
void setupWifi(){
  delay(10);
  WiFi.begin(ssid, password);
  Serial.print(F("Connecting to Wifi"));
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
}

```

اگر MQTT قطع شود، ESP32 تلاش می‌کند دوباره وصل شود.
client.connect با Client ID "ESP32ClientARMM" اتصال برقرار می‌کند.
خطای اتصال روی Serial Monitor چاپ می‌شود.

6. تابع setup

```
void setup() {
  Serial.begin(9600);
  Serial.println(F("Show DHT Temp And Humidity On Monitor:"));
  setupWifi();                    // اتصال به WiFi
  client.setServer(mqttServer, 1883); // تنظیم سرور و پورت MQTT
  dht.begin();                    // شروع کار سنسور DHT
}

```

- مقداردهی اولیه سریال، WiFi، MQTT و سنسور DHT.

7. حلقه اصلی loop

```
client.loop();
delay(2000);

```

client.loop() برای مدیریت اتصال و دریافت/ارسال پیام MQTT.

delay(2000) → هر ۲ ثانیه یک بار داده خوانده و ارسال می‌شود.

7.2 خواندن داده‌ها از سنسور

```
float humidity = dht.readHumidity();
float temperature = dht.readTemperature();
float temperatureFarenhait = dht.readTemperature(true);

if (isnan(humidity) || isnan(temperature) ||
isnan(temperatureFarenhait)) {
  Serial.println(F("Failed To Read From Sensor"));
  return;
}

float heatIndexCelsius = dht.computeHeatIndex(temperature, humidity,
false);

```

دما و رطوبت به **°C** و **°F** خوانده می‌شوند.

اگر خواندن موفق نبود، پیغام خطا چاپ می‌شود و حلقه ادامه نمی‌یابد.
شاخص حرارتی (Heat Index) محاسبه می‌شود.

7.3 ساخت JSON

```
doc["humidity"] = humidity;
doc["temperature"] = temperature;
doc["temperatureFarenhait"] = temperatureFarenhait;
doc["heatIndexCelsius"] = heatIndexCelsius;
doc["timeStamp"] = millis();  // timestamp ارسال

```

- اطلاعات دما، رطوبت، Heat Index و زمان فعلی ESP32 در JSON ذخیره می‌شود.

7.4 اتصال MQTT و ارسال داده

```
float humidity = dht.readHumidity();
float temperature = dht.readTemperature();
float temperatureFarenhait = dht.readTemperature(true);

if (isnan(humidity) || isnan(temperature) ||
isnan(temperatureFarenhait)) {
  Serial.println(F("Failed To Read From Sensor"));
  return;
}

float heatIndexCelsius = dht.computeHeatIndex(temperature, humidity,
false);

```

اگر MQTT قطع باشد، دوباره وصل می‌شود.
JSON سریالیزه شده و به Topic IOTProject4032ARMM ارسال می‌شود.
وضعیت ارسال روی Serial Monitor چاپ می‌شود.