## 

## Structuri software pentru aplicatii de timp real

## Logger de temperatura si umiditate folosind MQTT

## Student: Cimpan Dan

## Grupa: 31461/1

## Introducere

Acest proiect doreste replicarea functionalitatii unui echipamentului de logging si plotare a datelor de temperatura si umiditate folosite pentru verificarea camerelor climatice folosite in mediul industrial.

Componentele electronice utilizate in industia auto trebuie sa respecte standarul AEC-Q200. Standardul AEC-Q200, stabilit de Automotive Electronics Council (AEC), definește testele de stres pentru calificarea componentelor pasive utilizate în aplicații auto. Acesta specifică diferite grade de temperatură pentru a asigura fiabilitatea componentelor în diverse condiții de operare:

|  |  |  |
| --- | --- | --- |
| Grad | Interval de temperatura | Aplicatii tipice |
| 0 | De la -40°C până la +40°C | Componente generale Auto |
| 1 | De la -40°C până la +80°C | Componentele din compartimentul motor |
| 2 | De la -40°C până la +65°C | Componentele din componenta sistemului de climatizare |
| 3 | De la -40°C până la +40°C | Componentele din habitaclu |
| 4 | De la 0°C până la +30°C | Aplicatii non-auto (stand de test, computere de diagnoza) |

Umiditate și teste asociate

În ceea ce privește umiditatea, standardul AEC-Q200 include teste pentru a evalua performanța componentelor în condiții de umiditate ridicată. De exemplu:

Testul de umiditate cu tensiune aplicată expune componentele la 85% umiditate relativă la 40°C cu o tensiune de 12V aplicată timp de 1.000 de ore.

Testul de rezistență la umiditate implică expunerea componentelor la temperaturi variabile (25°C - 40°C) și umiditate ridicată (80% - 98%) pe 10 cicluri, fiecare cu durata de 24 de ore.

Aceste teste se realizeaza cu ajutorul unor camere climatice ce contin in ansamblul lor pompe de calduca capabile atat sa controleze temperatura cat si umiditatea aerului care este apoi ventilat in camerele climatice.

Pentru a respecta standardul AEC-Q200, performantele camerelor climatice trebuie verificat cu un echipament extern dupa fiecare ciclu de mentenanta. Pentru aceasta se folosesc echipamente speciale, precum cel din Figura 1, de data logging ce au in componenta partea de transmitter (care se introduce in camera de climatizare si contine senzorii si componentele electronice necesare) si partea de receiver care stocheaza datele primite de la senzori si le ploteaza.



Figure 1 Sistem achizitionare date

Proiectul "Logger de temperatura si umiditate folosind MQTT" documentează procesul de colectare, stocare și vizualizare a datelor de temperatură și umiditate utilizând un ESP32, un senzor DHT22 și tehnologia MQTT.

**1. Componentele hardware**

**ESP32**

ESP32 este un microcontroller puternic, cu conectivitate WiFi și Bluetooth, ideal pentru proiecte IoT. Acesta permite colectarea și transmiterea datelor de la senzori către un server MQTT.

**Senzor de temperatură și umiditate DHT22**

DHT22 este un senzor digital capabil să măsoare temperatura și umiditatea cu o precizie ridicată. Acesta este conectat la ESP32 și transmite datele citite prin protocol digital.

**Servomotorul DS3225**

Acest servomotor este special conceput pentru medii neprielnice de lucru, fiind rezistent la o plaja larga de temperaturi de operare iar mecanismul intern este lubrifiat cu grafit ce nu isi schimba proprietatile la temperaturi negative.

**Simulare în Wokwiki**

Pentru a simula aceste compoente s-a folosit Wokwiki. Acesta este un simulator de circuite electronice care permite testarea și dezvoltarea codului fără a avea nevoie de hardware fizic. În acest proiect, Wokwiki a fost utilizat pentru a emula ESP32 și senzorul DHT22, servomotorul DS3225 cat si elemente luminoase de semnalizare, vizibile in figura 2.

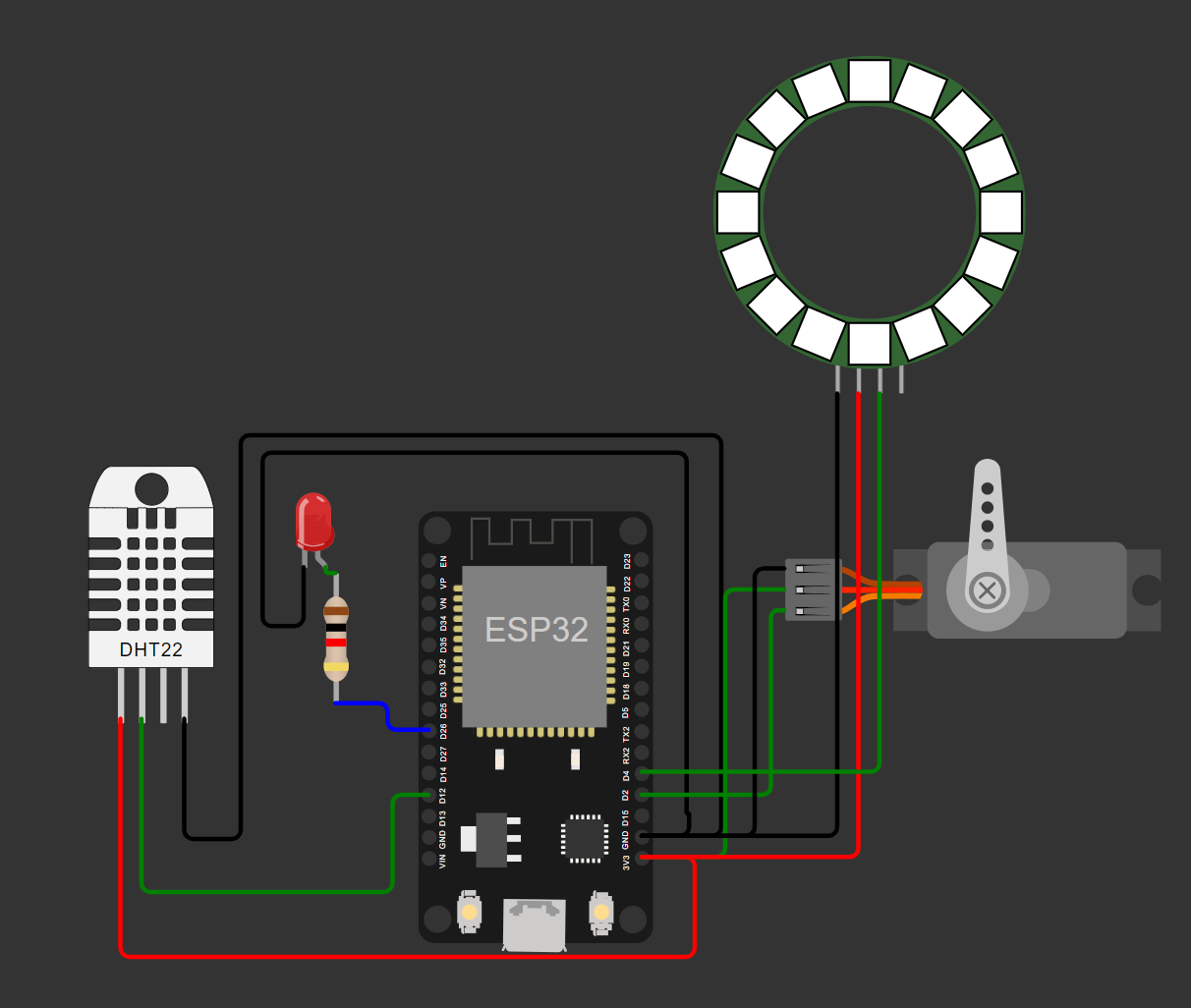


Figure 2

**2. Codul utilizat pentru simulare**

Codul implementat pe ESP32 permite colectarea și publicarea datelor prin MQTT. Acesta include:

* Configurarea conexiunii WiFi

WiFiClient espClient;

PubSubClient client(espClient);

void setup\_wifi() {

  delay(10);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

**Serial**.print(".");

  }

**Serial**.println("");

**Serial**.println("WiFi connected");

**Serial**.println("IP address: ");

**Serial**.println(WiFi.localIP());

}

void reconnect() {

  while (!client.connected()) {

    if (client.connect(clientID)) {

**Serial**.println("MQTT connected");

      client.subscribe("lights");

      client.subscribe("servo"); // Subscribe la servo topic

      client.subscribe("lights/neopixel"); // Subscribe la neopixel topic

**Serial**.println("Topic Subscribed");

    }

    else {

**Serial**.print("failed, rc=");

**Serial**.print(client.state());

**Serial**.println(" try again in 5 seconds");

      delay(5000);  // wait 5sec and retry

    }

  }

}

* Citirea datelor de la senzorul DHT22

    // citeste temperatura si umiditatea

    sensors\_event\_t event;

    dht.temperature().getEvent(&event);

    if (isnan(event.temperature)) {

**Serial**.println(F("Error reading temperature!"));

    }

    else {

**Serial**.print(F("Temperature: "));

      temp = event.temperature;

**Serial**.print(temp);

**Serial**.println(F("°C"));

    }

    // ia umiditatea si printeaza valoarea

    dht.humidity().getEvent(&event);

    if (isnan(event.relative\_humidity)) {

**Serial**.println(F("Error reading humidity!"));

    }

    else {

**Serial**.print(F("Humidity: "));

      hum = event.relative\_humidity;

**Serial**.print(hum);

**Serial**.println(F("%"));

    }

* Publicarea datelor pe un topic MQTT

    msgStr = String(temp) + "," + String(hum) + ","; // formateaza datele citite de la senzor intr-un mesaj de tip string

    byte arrSize = msgStr.length() + 1; // converteste acest mesaj intr-un aray de caractere pentru a putea fi scris prin MQTT

    char msg[arrSize];

**Serial**.print("PUBLISH DATA: ");

**Serial**.println(msgStr);

    msgStr.toCharArray(msg, arrSize);

    client.publish(topic, msg); // publica mesajul pe topicul MQTT definit

    msgStr = "";

    delay(1);

* Controlul unui LED, al unui servo motor și al unui inel LED WS2812

Cand se primește un mesaj pe topicul "lights", LED-ul conectat la pinul 26 este activat sau dezactivat, în funcție de valoarea mesajului „ON” sau „OFF”.

  if (String(topic) == "lights") {

    if (data == "ON") {

**Serial**.println("LED");

      digitalWrite(LED, HIGH);// aprinde led-ul daca acesta este setat ca ON din dashboard

    }

    else {

      digitalWrite(LED, LOW);

    }

Când se primește un mesaj pe topicul "servo", se citește valoarea în grade și se mișcă servo-motorul la acea poziție.

  else if (String(topic) == "servo") {

    int degree = data.toInt(); // converteste inputul in numar intreg

**Serial**.print("Moving servo to degree: ");

**Serial**.println(degree);

    servo.write(degree); // misca servomotorul in pozitia dorita in grade

  }

In functia callback, când se primește un mesaj pe topicul "lights/neopixel", se citesc valorile RGB și se setează culoarea pentru întreaga bandă de LED-uri WS2812 conectată la pinul 4:

  else if (String(topic) == "lights/neopixel") {

    int red, green, blue;

    sscanf(data.c\_str(), "%d,%d,%d", &red, &green, &blue); // scrierea datelor in valori RGB

**Serial**.print("Setting NeoPixel color to (R,G,B): ");

**Serial**.print(red);

**Serial**.print(",");

**Serial**.print(green);

**Serial**.print(",");

**Serial**.println(blue);

    fill\_solid(leds, NUM\_LEDS, CRGB(red, green, blue)); // seteaza toate ledurile din banda sa aiba aceeasi culoare

    FastLED.show(); // actualizare banda led cu o noua culoare

     fill\_solid(leds, NUM\_LEDS, CRGB(red, green, blue));

     FastLED.show();

  }

**3. Implementarea MQTT**

**Cum este utilizat MQTT pentru comunicarea între ESP32 și Node-Red**

MQTT este un protocol de mesagerie ușor utilizat pentru comunicarea între dispozitive IoT. ESP32 publică datele de temperatură și umiditate pe un broker MQTT (HiveMQ), iar Node-Red le preia pentru prelucrare și stocare. ESP32 se conectează la o rețea WiFi și la un broker MQTT (HiveMQ) utilizând biblioteca PubSubClient. Dispozitivul se reconectează automat dacă pierde conexiunea.

**4. Node-Red**

**Ce este Node-Red**

Node-Red este un instrument de dezvoltare vizuală pentru automatizare și IoT. Acesta permite colectarea, procesarea și afișarea datelor într-un mod intuitiv.

Functiile utilizate in Node-Red sunt utilizate pentru a prelua datele primite de la senzori si de a le afisa intr-un mod grafic si intuitiv de citit pe dasboard.

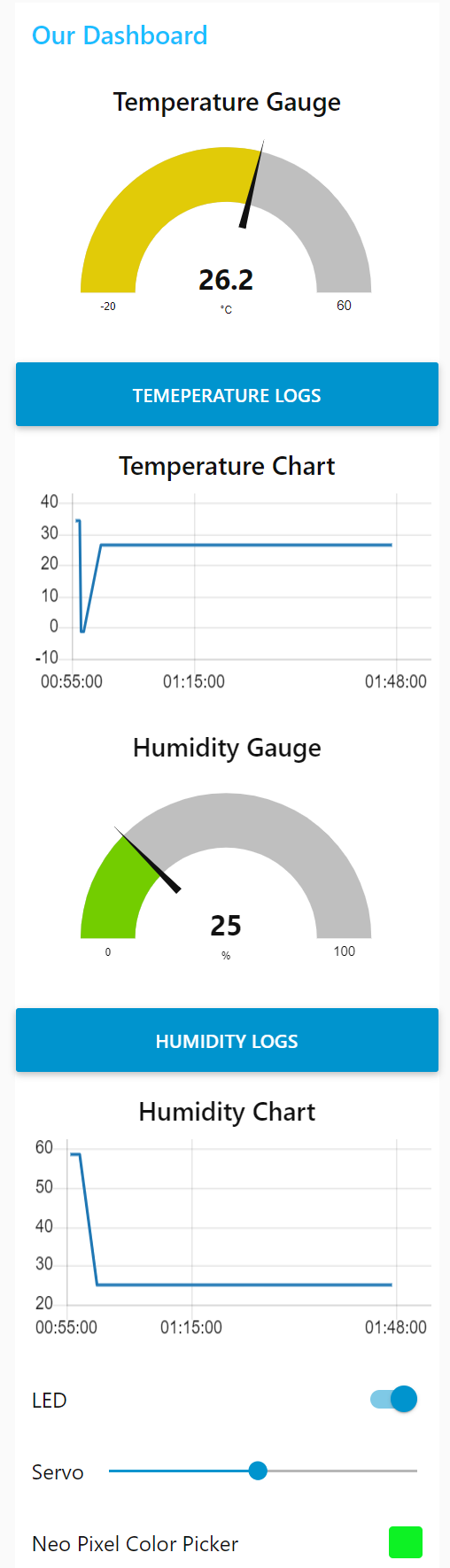
De asemenea, se adauga un timestamp datelor primite pentru a putea avea o referinta in timp a datelor stocate in baza de date. Astfel se poate monitoriza evolutia in timp a valorilor de temperatura si umiditate. Astfel sunt prezente toate datele necesare realizarii unor grafice pentru o citire mai usoara a datelor.

Exemplul functiei ce converteste timestamp-ul din format ISO 8601 in formatul (an-luna-zi ora:minut:secunda) acceptat de SQL.

var timestamp = new Date().toISOString();

// Convereste formatul ISO 8601 la formatul MySQL DATETIME (YYYY-MM-DD HH:MM:SS)

var mysqlTimestamp = timestamp.replace("T", " ").substring(0, 19);



var value = msg.payload;

// Adaugarea datelor in tabelul SQL

msg.topic = "INSERT INTO temperature\_logs (timestamp, temperature) VALUES (?, ?)";

msg.payload = [mysqlTimestamp, value];

return msg;

**Dashboard-ul aplicației**

Aplicația include un dashboard interactiv, figura 3, care permite vizualizarea datelor în timp real. Acesta conține:

Figure 3

* **Indicator pentru temperatură**: Afișează temperatura actuală și își schimbă culoarea în funcție de valoare.
* **Indicator pentru umiditate**: Afișează umiditatea actuală și își schimbă culoarea în funcție de valoare.
* **Grafice de evoluție**: Monitorizează variațiile temperaturii și umidității în timp.
* **Controller Servomotor**: actioneaza pozitia servomotorului
* **Buton LED**: aprinde sau stinge LED-ul
* **Picker RGB**: controleaza culoarea benzii de LED-uri

**5. Stocarea și vizualizarea datelor**

**Stocarea datelor**

Datele primite de la senzori sunt stocate atât local, sub formă de fișiere CSV (compatibile cu Excel), cât și într-o bază de date MySQL.

Conexiunea dintre baza de date SQL si Node-Red se face pin intermediul unei librari numite „node-red-node-mysql-dynamic”. Pentru o conexiune, este necesara configurarea conexiunii prin setarea detelor precum: Host, Port, User, Password, Database.

**6. Utilizare**

1. Pentru utilizarea acestei aplicatii se incepe prin pornirea serverului prin intermediul comenzii „node-red” in Powershell.
2. Odata ce serverul este activ se conecteaza microcontroller-ul la HiveMQ. Dupa conectare prin dublu-click pe senzorul DHT22 se pot ajusta valorile de temperatura si umiditate ce vor fi apoi vizibile in dashboard.
3. Pentru a utiliza dashboard-ul este encesara parasirea vederii de flow de la adresa <http://127.0.0.1:1880/#flow/1464fd4c2ecd8845> si accesarea vederii de dashboard de la adresa: <http://127.0.0.1:1880/ui/#!/0?socketid=MERimw3aw3k0d_DqAAAF>
4. Datele salvate in logs se pot vedea in MySQL workbench prin comenzi SQL de citire a tabelelor, precum SELECT \* FROM sensor\_logs; De asemenea datele sunt stocate si local sub forma de fisiere CSV, figura 4.

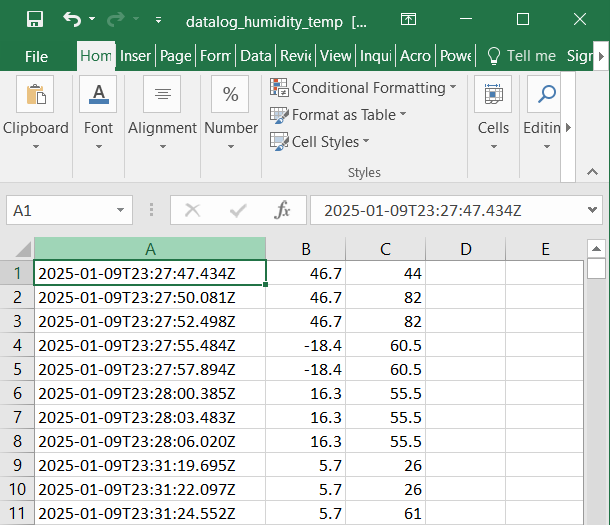


Figure 4