Universitatea „Petrol – Gaze” din Ploiești

UPG modules Alarm Clock

Realizat de: Coșereanu Emanuel, INFO an 3, grupa 40322

Cuprins

Lista de figuri

Introducere

În cadrul acestei teme am realizat un ceas cu alarmă ce arată dacă în cadrul Universității Petrol-Gaze Ploiești este pauză, sau dacă se află în desfășurare un modul. Pentru aceasta am avut nevoie de următoarele lucruri:

- Placă ESP32S.
- Ecran LCD I2C 1602.
- Sonerie pasivă 12 mm 12085.
- Senzor de umiditate și temperatură DHT11.
- Bandă LED WS2812 5050 cu 8 led-uri.
- Fire conductoare.

Obiectele enumerate mai sus s-au folosit în următorul fel:

- Pe ecranul LCD s-a afișat pe primul rând ziua (nume și data), luna, ora și minutul curente, iar pe rândul 2 s-a afișat un mesaj pentru a informa dacă se află în desfășurare vreun modul, este pauză sau nu se mai fac cursuri.
- Banda LED s-a folosit pentru a semnaliza când e în desfășurare un modul, culoarea ei fiind roșie, respectiv verde atunci când este timp liber.
- Soneria a fost folosită, bineînțeles, pentru a suna atunci când începe și se termina pauza (sau când începe și se termină un modul).
- Senzorul de temperatură și umiditate a fost folosit pentru a obține temperatura și umiditatea din sală și a afișa-o în site-ul web.

Ca protocoale de comunicație s-a folosit MQTT pentru a trimite datele de la senzori, mesajul afișat pe ecranul LCD și luminozitatea bandei LED către serverul informatica2. Pentru că pe moment serverul informatica2 nu era funcțional pentru testarea am folosit HiveMQ, așa cum se va putea vedea mai târziu în document. Totodată am folosit și NTP pentru a putea obține ora curentă.

Mai jos voi lăsa și o poză (**Fig. 1**) cu proiectul asamblat, deși arată puțin dezordonat și nu se pot observa prea bine conexiunile dintre fire.

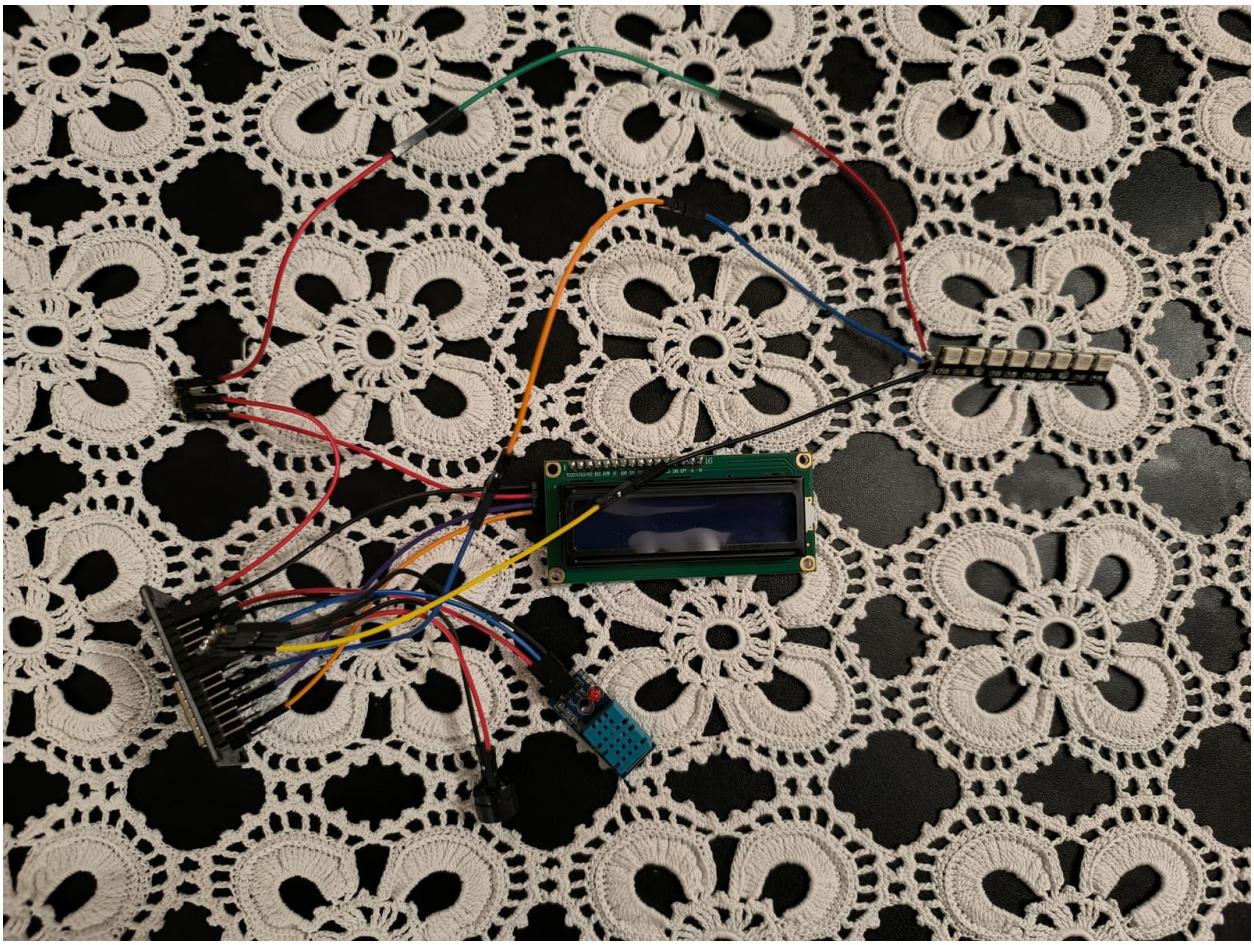


***Fig. 1** - UPG modules Alarm Clock*

Înainte de a începe să codez, bineînțeles că a trebui să conectez toți senzorii și actuatorii la plăcuța ESP, așa că voi lăsa mai jos detaliile importante despre conectare:

- Ecranul LCD l-am conecta la 5V (pinul VN al plăcuței), SDA la pinul D21 și SCL la pinul D22.
- Banda LED am conectat-o tot la 5V, și DIN la pinul 18 al plăcuței.
- Senzorul de temperatură și umiditate l-am conectat la 3.3V, iar DIN pe pinul 26 al plăcuței.
- Plusul soneriei l-am conectat la D25.

Acestea fiind spuse, în continuare voi prezenta cât mai scurt și la obiect proiectul, respectiv modul meu de gândire în realizarea acestuia.

Realizarea proiectului

Realizarea proiectului este puțin mai încâlcită, așa că voi sări peste împărțirea frumoasă a capitolelor și voi trece mai repede prin anumite lucruri deja cunoscute.

Cum am spus mai sus, am folosit protocolul NTP pentru a obține ora curentă și am setat **gmtOffset_sec = 7200**, pentru că aceasta este ora de iarnă a României (+ 2 ore). O dată obținută data și ora m-am folosit de ele pentru a le afișa pe ecranul LCD și în site-ul web.

Funcția startAlarmClock()

Nucleul central al proiectului este funcția *startAlarmClock()* în care se întâmplă în principiu toată magia afișării și a alarmei. După ce obținem data și ora le împărțim pe bucăți și le modificăm puțin pentru a afișa ziua și luna în română, așa cum se poate vedea mai jos.

```
int week_day = timeinfo.tm_wday;          //obtinem ziua (0 - Duminica, 1 - Luni....)
  int month = timeinfo.tm_mon;              //obtinem luna
  int month_day_number = timeinfo.tm_mday;  //obtinem ziua din luna
  int hour = timeinfo.tm_hour;              //obtinem ora
  int minute = timeinfo.tm_min;             //obtinem minutul


  char days[7][4] = { "DUM", "LUN", "MAR", "MIE", "JOI", "VIN", "SAM" };                                        //matrice de caractere pentru a putea afisa ziua
  char months[12][4] = { "IAN", "FEB", "MAR", "APR", "MAI", "IUN", "IUL", "AUG", "SEP", "OCT", "NOI", "DEC" };  //aceeasi chestie ca mai sus dar afisam luna:))


  char buffer[32];
  sprintf(buffer, "%s %d %s %02d:%02d", days[week_day], month_day_number, months[month], hour, minute);  //formatam tot si punem intr un buffer, apoi afisam
  lcd.print(buffer);
```

O dată obținute aceste detalii și afișate pe ecran, trecem la rândul 2 al ecranului unde vom afișa un anumit mesaj în funcție de oră (adică dacă e oră de modul, dacă e pauză sau dacă nu mai sunt deloc cursuri). Pentru că este un cod lung, voi lăsa doar o mică parte din el mai jos.

```
else if ((hour == 18 && minute >= 30) || (hour == 19 && minute <= 59)) {
    lcd.print("Modulul 7!!!");
    colorWipe(strip.Color(255, 0, 0));
    strip.show();  //aprindem led urile
  } else if ((hour >= 20 && minute >= 1 && hour <= 24) || (hour <= 8 && minute <= 29)) {
    lcd.print("Fara cursuri!!!");
```

```
    colorWipe(strip.Color(0, 255, 0));
    strip.show();  //aprindem led urile
  } else {
    lcd.print("PAUZAAA!!!");
    colorWipe(strip.Color(0, 255, 0));
    strip.show();  //aprindem led urile
  }
```

Totodată, atunci când afișăm mesajul pe rândul 2 al ecranului schimbăm culoarea **bandei LED** în verde dacă nu este oră de modul, și roșu dacă este oră de modul.

Funcția AlarmClockSound()

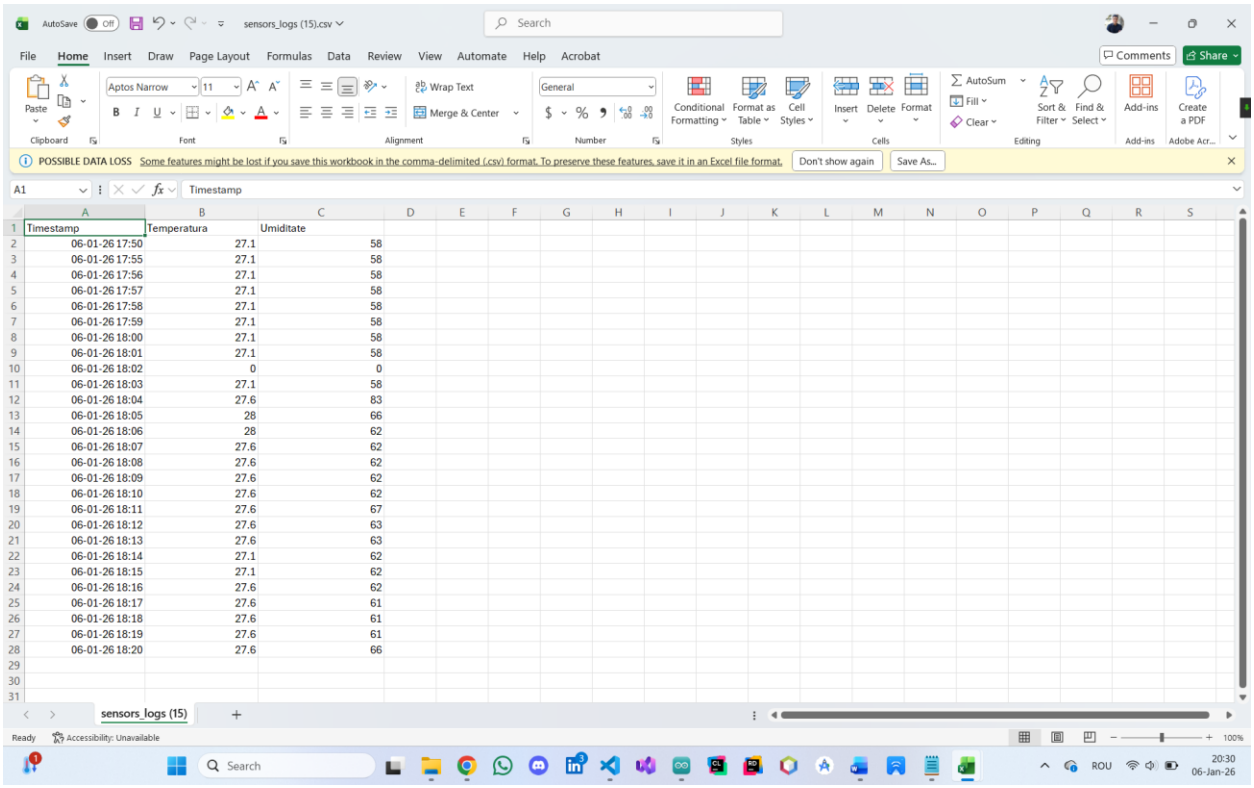
Pentru a porni buzzer-ul (soneria) trebuie să verificăm momentele în care începe o oră de modul, sau o poză, fapt ce se face foarte simplu, motiv pentru care nu voi mai lăsa codul aici. În cazul în care ne aflăm pe un caz pozitiv se v-a apela funcția *AlarmClockSound()*, al cărei cod va fii lăsat mai jos.

```
//functie pentru a porni buzzerul atunci cand e pauza sau curs
void AlarmClockSound() {
  //frecvente pentru notele muzicale (Do Re Mi Fa Sol La Si Do - Octava 4
:)) )
  int notes[] = {
    262, 294, 330, 349, 392, 440, 494, 523
  };

  unsigned long start = millis();  //obtinem timpul in secunde de cand placa
a fost pornita

  while (millis() - start < 10000) {  // 10 secunde
    for (int i = 0; i < 8; i++) {
      tone(buzzerPin, notes[i]);
      delay(150);
      noTone(buzzerPin);
      delay(50);
    }
  }
}
```

Așa cum se poate observa din cod, am ales ca soneria să sune folosind notele Do, Re, Mi, Fa, Sol, La, Si, Do, deoarece alt sunet mai drăguț nu am găsit pe moment.

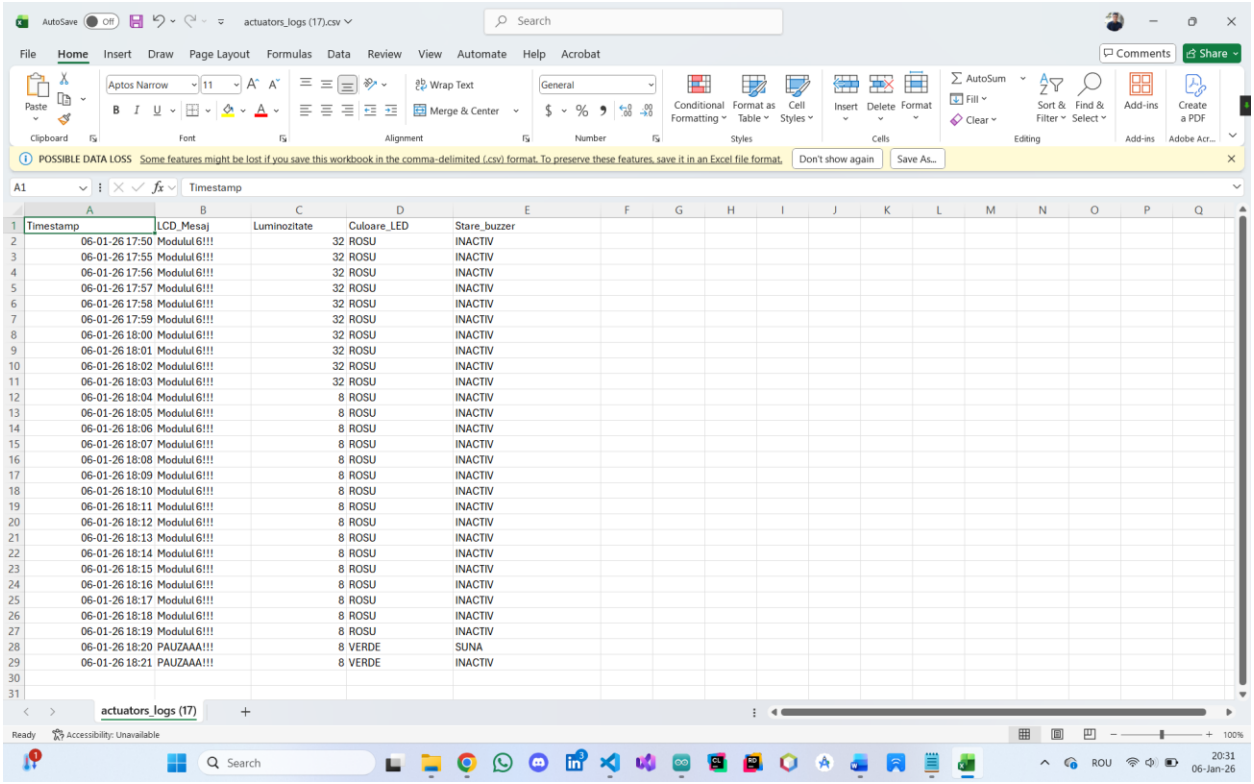
Scrierea în fișierele logging a datelor

Toate datele afișate pe ecran sau pe server au fost memorate în două fișiere denumite **sensors_logs** și **acutoators_logs**, în primul dintre ele memorez date despre temperatură și umiditate, respectiv data la care au fost încărcate (**Fig. 2**), iar în cel de al doilea memorez mesajul aflat pe rândul 2 al ecranului, luminozitatea bandei led, culoarea led-urilor, starea soneriei și data la care au fost înregistrate aceste date (**Fig. 3**), aceste două fișiere se pot descărca de pe serverul web (**Fig. 4**).



	A	B	C
1	Timestamp	Temperatura	Umiditate
2	06-01-26 17:50	27.1	58
3	06-01-26 17:55	27.1	58
4	06-01-26 17:56	27.1	58
5	06-01-26 17:57	27.1	58
6	06-01-26 17:58	27.1	58
7	06-01-26 17:59	27.1	58
8	06-01-26 18:00	27.1	58
9	06-01-26 18:01	27.1	58
10	06-01-26 18:02	0	0
11	06-01-26 18:03	27.1	58
12	06-01-26 18:04	27.6	83
13	06-01-26 18:05	28	66
14	06-01-26 18:06	28	62
15	06-01-26 18:07	27.6	62
16	06-01-26 18:08	27.6	62
17	06-01-26 18:09	27.6	62
18	06-01-26 18:10	27.6	62
19	06-01-26 18:11	27.6	67
20	06-01-26 18:12	27.6	63
21	06-01-26 18:13	27.6	63
22	06-01-26 18:14	27.1	62
23	06-01-26 18:15	27.1	62
24	06-01-26 18:16	27.6	62
25	06-01-26 18:17	27.6	61
26	06-01-26 18:18	27.6	61
27	06-01-26 18:19	27.6	61
28	06-01-26 18:20	27.6	66

***Fig. 2** - Fișierul sensors_log*



Timestamp	LCD_Mesaj	Luminozitate	Culoare_LED	Stare_buzzer
06-01-26 17:50	Modulul 6!!!	32	ROSU	INACTIV
06-01-26 17:55	Modulul 6!!!	32	ROSU	INACTIV
06-01-26 17:56	Modulul 6!!!	32	ROSU	INACTIV
06-01-26 17:57	Modulul 6!!!	32	ROSU	INACTIV
06-01-26 17:58	Modulul 6!!!	32	ROSU	INACTIV
06-01-26 17:59	Modulul 6!!!	32	ROSU	INACTIV
06-01-26 18:00	Modulul 6!!!	32	ROSU	INACTIV
06-01-26 18:01	Modulul 6!!!	32	ROSU	INACTIV
06-01-26 18:02	Modulul 6!!!	32	ROSU	INACTIV
06-01-26 18:03	Modulul 6!!!	32	ROSU	INACTIV
06-01-26 18:04	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:05	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:06	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:07	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:08	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:09	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:10	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:11	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:12	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:13	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:14	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:15	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:16	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:17	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:18	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:19	Modulul 6!!!	8	ROSU	INACTIV
06-01-26 18:20	PAUZAAA!!!	8	VERDE	SUNA
06-01-26 18:21	PAUZAAA!!!	8	VERDE	INACTIV

***Fig. 3** - Fișierul actuators_log*

***Fig. 4** - Interfața web*

Pentru a memora aceste date în fișiere s-au folosit 2 funcții principale și anume: *logData()* și *logActuatorsPeriodic()*. În prima dintre ele s-a obținut data și ora, respectiv temperatura și umiditatea, folosindu-mă de *getHumidity()* și *getTemperature()*, ce îmi returnau ca string aceste două valori. Apoi s-au pus toate datele într-un string și au fost scrise în fișier. În cea de a doua funcție a trebui să obțin mesajul scris pe ecranul LCD folosind funcția *getMessage()*, să aflu culoarea led-urilor și starea soneriei, am formatat toate datele și le-am încărcat în fișier. Voi lăsa mai jos doar codul de la prima funcție.

```
void logData() {
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.print("No time available (yet)");
    return;
  }

  //pregatim datele in format CSV: data, ora, temperatura, umiditate
  char timestamp[20];
  strftime(timestamp, sizeof(timestamp), "%Y-%m-%d %H:%M:%S", &timeinfo);

  String dataString = String(timestamp) + "," + getTemperature() + "," +
getHumidity() + "\n";

  //deschidem fisierul in modul append pentru a adauga la ginal
  File file = LittleFS.open("/logs.csv", FILE_APPEND);
  if (file) {
    file.print(dataString);
    file.close();
    Serial.println("Log Senzori: " + dataString);
  };
}
```

Ambele funcții sunt apelate în loop() o dată la un minut!!!

Folosirea protocolului de comunicație MQTT

Pentru a trimite datele necesare m-am conectat ca și client la un server public, deoarece informatica2 nu funcționa pentru moment (**Fig. 5**), dar pentru a mă conecta la informatica2 pur și simplu voi lăsa linia scrisă așa:

```
//setari MQTT
const char* mqtt_server = "informatica2.upg-ploiesti.ro"; // Sau IP-ul
serverului daca e local
const int mqtt_port = 1883;
```

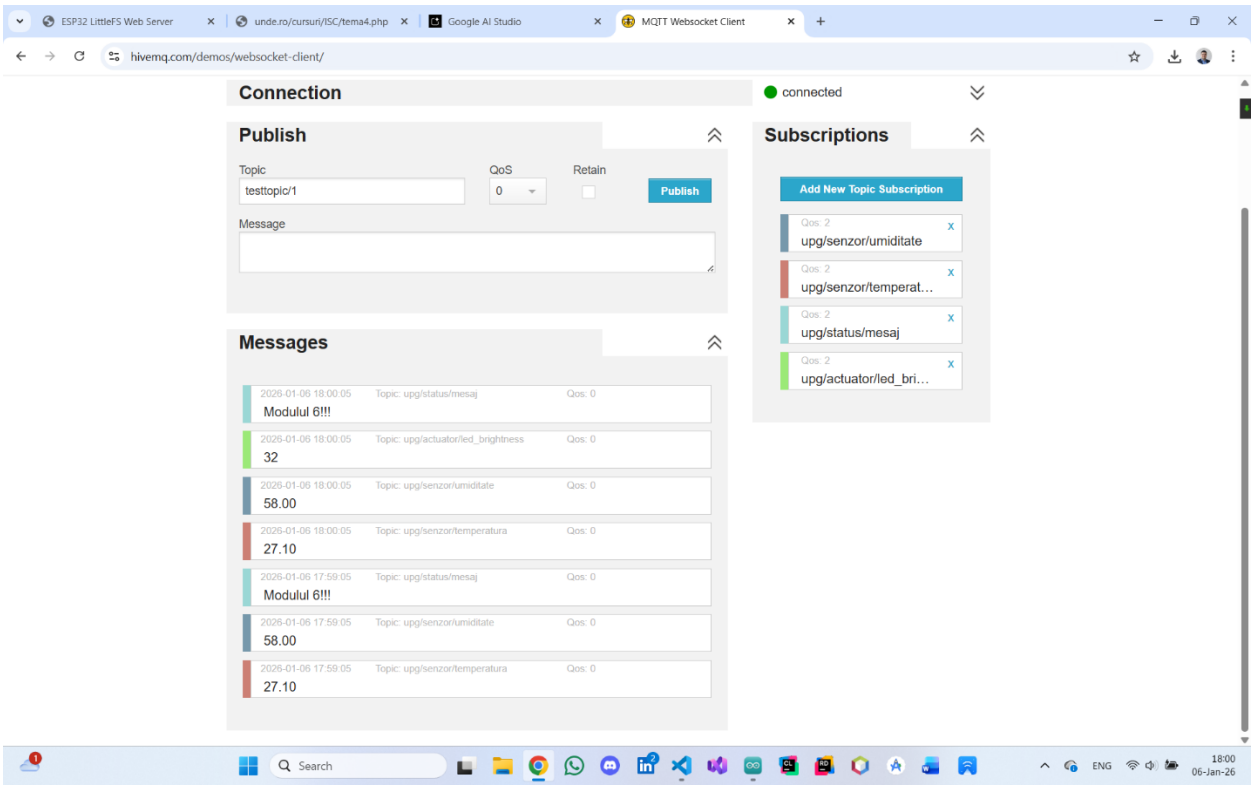


***Fig. 5** - Transmiterea datelor la severul MQTT*

Deoarece există posibilitatea deconectării de la server am creat o funcție *reconnectMQTT()* ce este apelată la un anumit interval de timp în loop(). Pentru a trimite (publica) datele am creat funcția *sendMQTTData()*, în care m-am folosit din nou de funcțiile necesare pentru a obține temperatura, umiditatea, mesajul de pe ecran și luminozitatea bandei led ca string-uri, setând topicurile necesare fiecărui parametru.

```
//functie pentru a trimite date la server (publish)
void sendMQTTData() {
  if (!mqttClient.connected()) {
    reconnectMQTT();
  }
  mqttClient.loop();

  //trimitem temperatura
  mqttClient.publish("C_Emanuel/upg/senzor/temperatura",
getTemperature().c_str());

  //trimitem tmiditatea
  mqttClient.publish("C_Emanuel/upg/senzor/umiditate",
getHumidity().c_str());
```

```
  //trimitem tesajul de pe LCD (Modul/Pauza)
  mqttClient.publish("C_Emanuel/upg/status/mesaj", getMessage().c_str());

  //trimitem tuminozitatea LED
  mqttClient.publish("C_Emanuel/upg/actuator/led_brightness",
SliderValue.c_str());

  Serial.println("Date trimise prin MQTT!");
}
```

Interfața Web

Interfața Web (**Fig. 4**), a cărei poză a fost afișată mai sus, este folosită pentru a face următoarele lucruri:

- Pentru a afișa data și ora, aproximativ exact ca pe banda LED (acum afișăm ziua și luna total, fără prescurtări).
- Afișarea temperaturii în grade Celsius.
- Afișarea umidității.
- Afișarea luminozității bandei LED și permiterea modificării acesteia.

Toate aceste date sunt actualizate o dată la 5 secunde, datorită următoarea bucăți de cod din javascript:

```
const updateSensorValue = (id, endpoint) => {
    fetch(endpoint)
      .then(response => {
        if (!response.ok) {
          throw new Error(`Failed to fetch ${endpoint}`);
        }
        return response.text();
      })
      .then(data => {
        document.getElementById(id).textContent = data;
      })
      .catch(error => {
        console.error(error);
        document.getElementById(id).textContent = 'Error';
      });
  };

  function updateSlider(element) {
```

```
    var SliderValue = document.getElementById("Slider").value;
    document.getElementById("textSliderValue").innerHTML = SliderValue;
    console.log(SliderValue);
    var httpRequest = new XMLHttpRequest();
    httpRequest.open("GET", "/slider?value=" + SliderValue, true);
    httpRequest.send();
  }

  // Update all sensors every 5 seconds
  setInterval(() => {
    //Senzori
    updateSensorValue('temperature', '/temperature');
    updateSensorValue('humidity', '/humidity');

    // Timp și Dată
    updateSensorValue('day_name', '/dayname');
    updateSensorValue('day_number', '/daynumber');
    updateSensorValue('month_name', '/monthname');
    updateSensorValue('hour', '/hour');
    updateSensorValue('minute', '/minute');

    // Mesaj (Modul/Pauză)
    updateSensorValue('message', '/message');
  }, 5000);
```

Pentru a afișa parametrii s-au folosit PLACEHOLDER ce au fost înlocuiți cu valorile necesare, după caz, așa cum se poate vedea din codul C de mai jos.

```
//Replaces placeholder with temperature and humidity value
String processor(const String &var) {
  //Serial.println(var); debug
  if (var == "SLIDERVALUE") {
    return SliderValue;
  } else if (var == "DAY_NAME") {
    return getDayName();
  } else if (var == "DAY_NUMBER") {
    return getDayNumber();
  } else if (var == "MONTH") {
    return getMonthName();
  } else if (var == "HOUR") {
    return getHour();
  } else if (var == "MINUTE") {
    return getMinutes();
```

```
  } else if (var == "MESSAGE") {
    return getMessage();
  } else if (var == "TEMPERATURE") {
    return getTemperature();
  } else if (var == "HUMIDITY") {
    return getHumidity();
  }

  return String();
}
```

Pentru a seta valoarea luminozității bandei LED s-a folosit funcția *updateSlider()*.

Pentru a trimite valorile necesare și pentru a le actualiza constant s-au definit rute HTPP GET ce trimit către server textul dori (umiditatea, temperatura, ziua, ora, ...). Voi lăsa mai jos 3 dintre ele.

```
//adaugam rute pentru temperatura, umiditatte, zi, ora, minut, pentru a
putea sa le actualizam constant
  server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request) {
    request->send(200, "text/plain", getTemperature().c_str());
  });

  server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request) {
    request->send(200, "text/plain", getHumidity().c_str());
  });

  server.on("/dayname", HTTP_GET, [](AsyncWebServerRequest *request) {
    request->send(200, "text/plain", getDayName().c_str());
  });
```

Și pentru a actualiza luminozitatea bandei LED:

```
server.on("/slider", HTTP_GET, [](AsyncWebServerRequest *request) {
    String inputMessage;
    // GET input1 value on <ESP_IP>/slider?value=<inputMessage>
    if (request->hasParam(INPUT_PARAMETER)) {
      inputMessage = request->getParam(INPUT_PARAMETER)->value();
      SliderValue = inputMessage;
      int brightness = SliderValue.toInt();  // 0-255

      strip.setBrightness(brightness);
      strip.show();
    } else {
      inputMessage = "No message sent";
```

```
    }
    Serial.println(inputMessage);
    request->send(200, "text/plain", "OK");
  });
```

Concluzii și poze

Acesta a fost unul dintre cele mai interesante și distractive proiecte de pe acest semestru. Din păcate mi-am pierdut totuși puțin răbdarea și nu am mai făcut o documentație atât de detaliată.

Videoclipul pe care îl voi atașa la temă va arăta funcționara proiectului cât și evoluția acestuia în timp, în mare parte se va axa pe demonstrarea faptului că banda LED poate fii controlată din interfața web și că alarma funcționează.

Mai jos vor fii lăsate 2 poze, una pentru a prezenta mesajele și culoarea led-urilor atunci când este pauză, respectiv atunci când este modul.

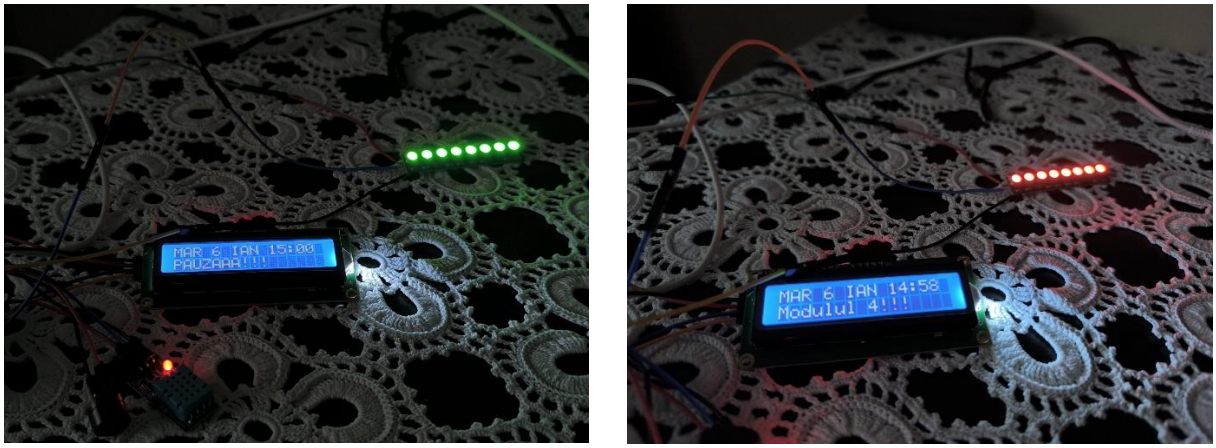


***Fig. 6** - Afișarea atunci când este pauză vs modul*

Bibliografie

1. https://www.youtube.com/watch?v=syQyrjJu0WE&t=65s - ESP32 Project - Digital Clock with LCD Display At Home
2. https://www.youtube.com/watch?v=K98h51XuqBE - ESP32 Tutorial - DHT11/DHT22 (Temperature and Humidity Sensor)
3. https://randomnerdtutorials.com/esp32-web-server-littlefs/ - ESP32 Web Server using LittleFS Filesystem (serve files from filesystem)
4. https://soldered.com/documentation/ws2812b/arduino/brightness-control/?srsltid=AfmBOops9cJR3x9ecRb8-yjA-DkPBTlKxZGcji2dN8W56mOZCE1u6Dl0 - WS2812B – Brightness Control and Effects
5. https://www.youtube.com/watch?v=s-NFdMXA0H4 - PWM Slider Bar Control on ESP32 ESP8266 WebServer | Control Brightness Of LEDs | Out of the Box IoT
6. https://www.w3schools.com/howto/howto_js_rangeslider.asp - How TO - Range Sliders
7. https://github.com/mishmashlabs/ESP32_PWM_SLIDER_WEBSERVER - ESP32_PWM_SLIDER_WEBSERVER