**Ministerul Educaţiei, Culturii și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Departamentul Informatică și Ingineria Sistemelor**

**PROIECT de AN**

**Interfețe de Comunicare**

**Tema proiectului:** ***Sinteza unui nod senzorial în baza ESP32 + Rețea Mesh (2 Dev.)***

A elaborat:

st. gr. CR-221 FR **Cristina Serba**

A verificat:

Conf.univ.,dr.  **Ababii Victor**

Chişinău – 2024

**“APROB”**

Şef DIIS

“\_10\_”\_septembrie\_2022

conf.univ.,dr. V. Sudacevschi

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# SARCINĂ TEHNICĂ

**Pentru proiectul de an**

**La disciplina\_\_Interfete de comunicare**

Studentul: Cristina Serba Grupa: CR-221 FR

**Tema : Sinteza unui nod senzorial în baza ESP32 + Retea Mesh (2 Dev.)**

**Condiţii tehnice:** **WiFi (IoT) nod senzorial**

**Setul de elemente: Microcontroloare ESP32, senzor de umiditate DHT22, senzor de gaz MQ-2**

**Conţinutul proiectului de an:**

1. Introducere,
2. Analiza situaţiei în domeniul de proiectare,
3. Sinteza algoritmilor de funcţionare a sistemului proiectat,
4. Descrierea componentelor/dispozitivelor utilizate în procesul de proiectare,
5. Sinteza şi descrierea schemelor funcţionale/electrice de principiu (E-Drive, Proteus, Fritzing),
6. Dezvoltarea produsului program (Arduino IDE),
7. Descrierea modului de utilizare a sistemului proiectat,
8. Testarea funcționala a proiectului (ESP, NodeMCU, Genuino)
9. Concluzii,
10. Date bibliografice,
11. Anexe

**Conţinutul părţii grafice/program:**

1. Scheme bloc ale algoritmilor,
2. Scheme funcţionale/de structură,
3. Scheme electrice de principiu,
4. Codul sursa a produsului program,
5. Poze cu sistemul in proces de testare funcționala.

**Concluzii:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Conducătorul** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_conf.univ.,dr. V. Ababii\_)

**Eliberată “\_10.09.2022**\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura studentului)

**CUPRINS**

[SARCINĂ TEHNICĂ 2](#_Toc166703790)

[Introducere : 4](#_Toc166703791)

[1. Analiza situaţiei în domeniul de proiectare 5](#_Toc166703792)

[2. Sinteza algoritmilor de funcţionare a sistemului proiectat 6](#_Toc166703793)

[3. Descrierea componentelor/dispozitivelor utilizate în procesul de proiectare 6](#_Toc166703794)

[4. Sinteza şi descrierea schemelor funcţionale/electrice de principiu (E-Drive, Proteus, Fritzing) 11](#_Toc166703795)

[5. Dezvoltarea produsului program elaborat in Arduino IDE 13](#_Toc166703796)

[6. Descrierea modului de utilizare a sistemului proiectat 20](#_Toc166703797)

[7. Testarea funcționala a proiectului (ESP, NodeMCU, Genuino) 21](#_Toc166703798)

[Concluzii 23](#_Toc166703799)

[Date bibliografice: 24](#_Toc166703800)

# Introducere :

Umiditatea potrivita este esentiala pentru confortul locuintei noastre, dar mai ales pentru sanatate. Problemele apar mai ales in timpul lunilor cu temperaturi extreme, cand in aer se elibereaza prea putini vapori de apa, ceea ce duce la scaderea umiditatii,

Un aer prea uscat poate cauza probleme de sanatate serioase, mai ales pe timpul iernii, cand sistemul imunitar este mai slabit, astfel incat racesti mai usor si esti mai expus virusurilor si bacteriilor. O umiditate optima in casa, unde ne petrecem aproximativ 90% din timp in zilele libere, potrivit unui studiu, este extrem de importanta pentru a ne mentine sanatosi, dar si pentru ca locuinta noastra sa nu aiba de suferit din pricina aerului prea umed sau prea uscat. [[1]](#_Date_bibliografice:)

Astmul bronsic reprezinta o afectiune medicala cronica ce genereaza pacientului mari stari de disconfort. In cazul astmului, fluxul de aer de pe caile respiratorii este restrictionat, cauza fiind inflamatia lor cronica. Inflamatia cronica poate fi data de alergii, diferite substante sau schimbari de temperatura. [[2]](#_Date_bibliografice:)

Factorii genetici joaca un rol foarte important in declansarea acestei afectiuni, foarte multi pacienti dezvoltand astm in copilarie. Aceasta insa nu este o regula, astmul putand sa apara la orice varsta. Fie ca vorbim de alergii alimentare, alergie la praf si polen sau par de animale, toate acestea reprezinta teren propice de instalare al afectiunii manifestate prin imposibilitatea de a respira corespunzator. Fumul de tigara si substantele cu efect iritant pot fi o alta cauza a astmului bronsic. Si vremea rece sau fluctuatiile de temperatura se numara pe lista cauzelor care declanseaza astmul bronsic la majoritatea pacientilor.

# Analiza situaţiei în domeniul de proiectare

**Problema: Calitatea aerului și relația acesteia cu bolile respiratorii**

**Sarcina: Crearea unui sistem autonom de monitorizare a calității aerului**

Problema aerului prea uscat poate afecta în mod semnificativ persoanele cu probleme respiratorii, cum ar fi astmul, bronșita cronică sau alte afecțiuni pulmonare. Potrivit rapoartelor OMS, astmul afectează aproximativ 334 de milioane de oameni la nivel global. Umiditatea scăzută în aer poate agrava simptomele acestor afecțiuni și poate provoca disconfort respirator și alte probleme de sănătate.

Potrivit OMS, poluarea aerului este responsabilă de aproximativ 7 milioane de decese premature în fiecare an la nivel global. Aceste decese sunt asociate cu expunerea la particule fine și gaze poluante, inclusiv dioxid de azot (NO2), monoxid de carbon (CO), ozon (O3) și particule în suspensie (PM2.5 și PM10).

Umiditatea scăzută poate contribui la apariția și agravarea problemelor respiratorii, cum ar fi tusea, iritațiile nazale, gâtul uscat și disconfortul respirator general.Conform unui studiu publicat în revista "Respiratory Care", umidificatoarele pot reduce simptomele de tuse și congestie la persoanele cu bronșită cronică și alte afecțiuni respiratorii. Aerul prea uscat poate usca mucoasele căilor respiratorii, făcându-le mai vulnerabile la infecții. Umiditatea scăzută poate, de asemenea, să reducă eficiența sistemului imunitar al căilor respiratorii în combaterea virusurilor și bacteriilor. Pe de altă parte, aerul prea umed poate favoriza creșterea mucegaiului și a altor alergeni care pot agrava afecțiunile respiratorii și pot crește riscul de infecții respiratorii.

Temperaturile extreme, fie prea ridicate, fie prea scăzute, pot afecta sistemul imunitar și pot face organismul mai susceptibil la infecții respiratorii. De asemenea, schimbările bruște de temperatură pot irita căile respiratorii și pot agrava simptomele unor afecțiuni respiratorii preexistente.

Cum poate ajuta un sistem de monitorizare a calității aerului în aceste situații:

1. **Umiditate optimă:** Un sistem de monitorizare a calității aerului poate măsura nivelul de umiditate din aer și poate oferi informații despre cât de umed sau uscat este aerul într-o anumită încăpere sau mediu. Pentru persoanele cu probleme respiratorii, menținerea unei umidități optime în aer poate fi crucială pentru confortul și sănătatea lor respiratorie. Un aer prea uscat poate irita căile respiratorii și poate agrava simptomele, în timp ce un nivel optim de umiditate poate ajuta la menținerea căilor respiratorii umede și la reducerea disconfortului.
2. **Temperatura adecvată:** Un sistem de monitorizare a calității aerului poate oferi și informații despre temperatura aerului. Temperaturile extreme, fie prea ridicate, fie prea scăzute, pot afecta respirația și pot agrava simptomele persoanelor cu probleme respiratorii. Menținerea unei temperaturi confortabile și adecvate poate contribui la gestionarea simptomelor și la îmbunătățirea calității respirației.
3. **Monitorizarea nivelurilor de gaze:** În plus, un sistem de monitorizare a calității aerului poate detecta și măsura nivelurile de gaze nocive sau iritante din aer, cum ar fi dioxidul de carbon (CO2) sau compușii organici volatili (VOC). Acești compuși pot proveni de la diferite surse, cum ar fi fumul de țigară, produsele de curățenie, mobilierul sau materialele de construcție, și pot afecta negativ sănătatea respiratorie. Identificarea și monitorizarea acestor compuși pot ajuta la reducerea expunerii la substanțe nocive și la protejarea sănătății respiratorii.

Prin monitorizarea și gestionarea acestor factori, un sistem de monitorizare a calității aerului poate oferi informații importante și instrumente utile pentru persoanele cu probleme respiratorii, ajutându-le să creeze un mediu mai sigur și mai sănătos pentru respirație. Aceste date pot fi utilizate pentru luarea deciziilor informate privind gestionarea mediului interior și pentru adaptarea mediului la nevoile specifice ale persoanelor cu afecțiuni respiratorii.

# Sinteza algoritmilor de funcţionare a sistemului proiectat

Algoritmul are la bază conectarea prin rețea mesh a mai multor microcontroloare ESP32 care vor colecta date de la senzori de umiditate, temperatură și gaz.

Colectarea datelor va fi realizată separat de la fiecare microcontroler responsabil de propriul set de date. Acestea vor fi transmise între ele, dar microcontrolerul principal le va selecționa și afișa pe un web server în timp real.

De asemenea, pentru fiecare microcontroler a fost adăugat un ecran LCD pentru afișarea stărilor intermediare și informațiilor utile despre statusul acestuia.

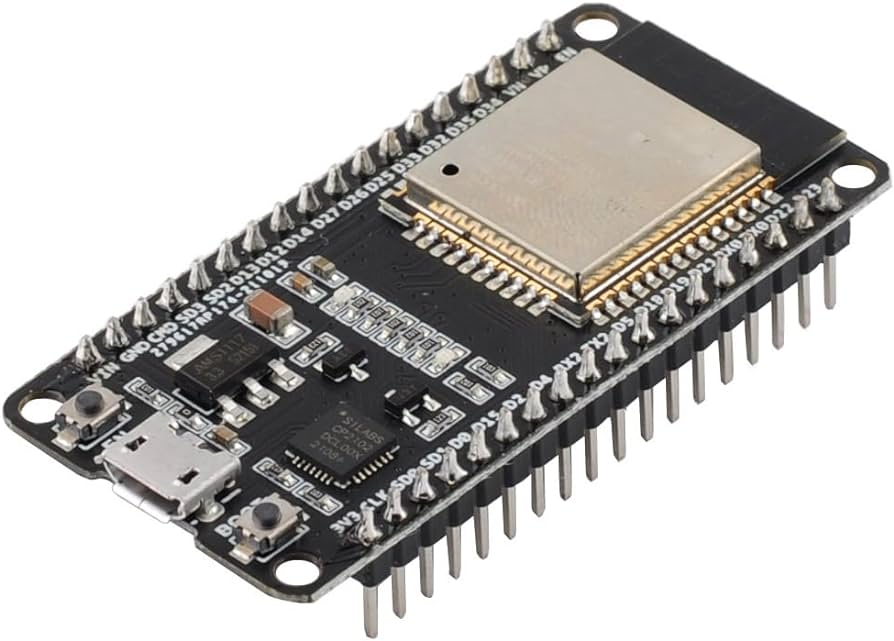
Pentru determinarea limitelor datelor achiziționate, a fost conectat câte un LED printr-un releu pentru fiecare microcontroler colector, care va fi aprins atunci când valorile date nu intră în intervalul de siguranță.

Astfel, algoritmul constă din conectarea a:

* 3 microcontroloare ESP32
* 2 LED-uri prin releuri
* 1 senzor de umiditate și temperatură
* 1 senzor de gaz

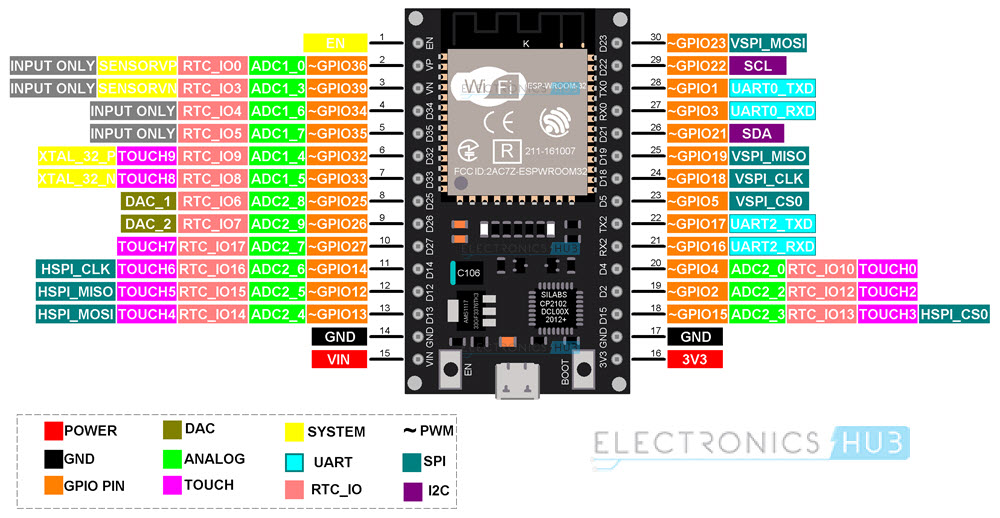
# Descrierea componentelor/dispozitivelor utilizate în procesul de proiectare

La baza proiectului stă microcontrolerul ESP-WROOM-32 (Figura 3.1) de tip sistem pe cip (SoC).Din funcționalitățile sale principale, în proiect au fost folosite dotarea cu WiFi și periferiile UART, GPIO, SPI. Este o platformă hardware extrem de versatilă și puternică, ideală pentru dezvoltarea unei game variate de aplicații IoT (Internet of Things) și embedded. [[3]](#_Date_bibliografice:)



**Figura 3.1**. Modulul ESP32.

Pinajul NodeMCU este prezentat în Figura 3.2.



**Figura 3.2.** Pinajul NodeMCU.

DHT22 (Figura 3.3), cunoscut și sub denumirea de AM2302, este un senzor de temperatură și umiditate digitală, proiectat pentru a furniza date precise și fiabile despre mediul înconjurător. Folosește un senzor capacitiv pentru umiditate și un termistor pentru a măsura temperatura aerului. Este foarte ușor de folosit; primul pin de pe partea stângă se conectează la tensiuni de 3-5V, cel de-al doilea pin la pinul de date și pinul cel mai din dreapta la masă. [[4]](#_Date_bibliografice:)



**Figura 3.3.** Senzorul DHT-22.

Senzorul de gaz MQ2 (Figura 3.4) este util pentru detectarea scurgerilor de gaz (casă și industrie). Este potrivit pentru detectarea H2, Gaz petrolier lichefiat, CH4, CO, alcool, fum sau propan. Datorită sensibilității sale ridicate și timpului de răspuns rapid, măsurarea poate fi efectuată cât mai curând posibil. Sensibilitatea senzorului poate fi reglată prin potențiometru. [[5]](#_Date_bibliografice:)



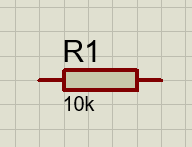
**Figura 3.4.** Sensor de gaz MQ-2.

LM016L (Figura 3.5) este un afișaj cu cristale lichide (LCD) de tip alphanumeric cu 16 caractere pe 2 linii, produs de diferiți producători. Este un dispozitiv popular și larg utilizat în aplicații electronice și informatice. LM016L utilizează tehnologia LCD (cristale lichide) pentru afișarea informațiilor, are o dimensiune de 16 caractere pe 2 linii, ceea ce înseamnă că poate afișa un total de 32 de caractere simultan și folosește caractere alfanumerice, adică poate afișa litere, cifre și simboluri speciale. [[6]](#_Date_bibliografice:)



**Figura 3.5**. Prezentarea LM016L LCD

Un rezistor (Figura 3.6) este un component electronic pasiv utilizat în circuitul electric pentru a limita curentul electric sau pentru a stabili o anumită tensiune într-un anumit punct al circuitului. Rezistența electrică a unui rezistor determină cât de mult va limita curentul electric care trece prin el. [[7]](#_Date_bibliografice:)



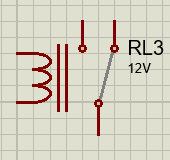
**Figura 3.6.** Prezentarea rezistorului.

Masa sau pământul (Figura 3.7) în contextul electronic este un termen utilizat pentru a se referi la un punct de referință electric comun într-un circuit electric sau electronic. Acest punct este adesea conectat la carcasa metalică a echipamentului sau la țărâm (solul) fizic și este folosit pentru a crea un punct de referință pentru tensiunea electrică într-un circuit. [[8]](#_Date_bibliografice:)



**Figura 3.7.** Prezentarea masei.

Un releu (Figura 3.8) este un dispozitiv electronic sau electromecanic utilizat pentru a întrerupe sau a conecta circuite electrice sau electronice în funcție de prezența sau absența unui semnal de control. Releul acționează ca un comutator controlat de un semnal electric și este utilizat într-o varietate de aplicații pentru a izola și a proteja circuitele, pentru a controla dispozitive de putere și pentru a realiza interconectarea între diferite circuite sau dispozitive. [[9]](#_Date_bibliografice:).



**Figura 3.8.** Prezentarea unui releu.

Tranzistorul 2N2369 (Figura 3.9) este un tranzistor bipolar de joncțiune de siliciu (BJT) utilizat în aplicații generale de amplificare și comutare în domeniul electronic. [[10]](#_Date_bibliografice:)



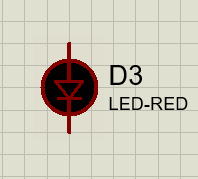
**Figura 3.9.** Prezentarea unui transistor.

Puterea în contextul unui component electronic se referă la cantitatea de energie pe care o consumă sau o livrează componentul respectiv. Aceasta poate fi exprimată în termeni de putere consumată sau putere livrată, în funcție de rolul și funcționarea specifică a componentului în circuitul electronic. [[11]](#_Date_bibliografice:)



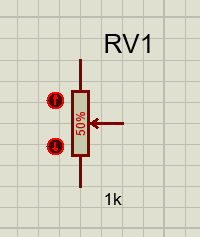
**Figura 3.10.** Prezentarea generatorului de curent continuu.

LED-ul (Figura 3.11) , sau dioda emițătoare de lumină, este un dispozitiv semiconductor care convertește energie electrică în lumină coerentă (monocromatică). LED-ul funcționează pe baza efectului de electroluminiscență în materialele semiconductoare. Atunci când un curent electric trece prin LED, electronii și găurile sunt recombinați într-o zonă activă din interiorul dispozitivului, emițând fotoni în proces. [[12]](#_Date_bibliografice:)



**Figura 3.11**. Prezentarea unui LED

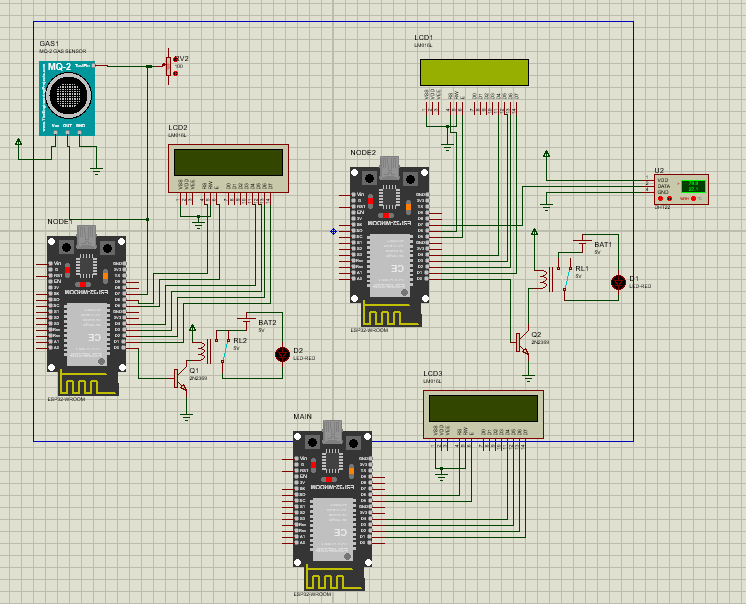
Un potențiometru (Figura 3.12) este un dispozitiv electronic utilizat pentru a ajusta nivelul unei tensiuni sau al unui curent într-un circuit electric. Este un component pasiv, adică nu conține componente active precum tranzistoare sau amplificatoare, și funcționează pe principiul schimbării rezistenței electrice în funcție de poziția unui cursor. [13]



**Figura 3.12.** Prezentarea unui potențiometru

# Sinteza şi descrierea schemelor funcţionale/electrice de principiu (E-Drive, Proteus, Fritzing)

Rezultatul proiectării schemei electrice de principiu dezvoltată în mediul Proteus este prezentat în Figura 4.1.

****

**Figura 4.1.** Schema electrică de principiu.

**Modul de funcționare a sistemului.**

Senzorul de temperatură și umiditate DHT22măsoară temperatura și umiditatea aerului prin generarea semnalelor proporționale cu aceste măsurători, care sunt citite de modulul ESP32 pe pinul D7. Datele sunt afișate pe un ecran LCD și sunt transmise prin rețeaua mesh. Dacă valorile măsurate depășesc limitele definite, releul conectat la pinul D0 al modulului este activat (setat la HIGH) și dezactivat (setat la LOW) în caz contrar.

Același principiu stă la al doilea modul ESP32, care măsoară concentrația de gaz prin senzorul MQ-2, o afișează la ecranul LCD, transmite datele prin rețeaua mesh și activează releul dacă valorile nu intră în intervalul prestabilit, ceea ce activează LED-ul roșu.

Nodul principal colectează datele transmise prin rețeaua mesh de la celaltele module ESP32, este conectat la rețeaua WiFi și trimite datele unui web server. Nodul este conectat prin portul 80 al routerului rețelei accesate. De asemenea, este conectat un modul LCD la nodul dat pentru afișarea datelor intermediare.

Nodurile ESP32 colectează date independent la un interval de 1s și sunt conectate la rețeaua mesh utilizând biblioteca painlessMesh [[13]](#_Date_bibliografice:). Colectarea datelor se face în funcția de trimtere a acestora spre restul nodurilor prin rețea. Fiecare nod primește notificare atunci când sunt date noi în rețea, adică primirea acestora se face în mod asincron.

# Dezvoltarea produsului program elaborat in Arduino IDE

**Descrierea aplicației Arduino IDE**

Arduino IDE este un software open-source, care este folosit pentru a scrie și a încărca cod pe plăcile Arduino. Aplicația IDE este potrivită pentru diferite sisteme de operare, cum ar fi Windows, Mac OS X și Linux. Suportă limbajele de programare C și C++. Aici, IDE înseamnă Integrated Development Environment. [[14]](#_Date_bibliografice:)

**Rezultatul dezvoltării produsului program**

***Codul nodului 1 (colectarea datelor de umiditate și temperatură):***

|  |
| --- |
| #include <painlessMesh.h>  #include <DHT.h>  #include <LiquidCrystal.h>  #include <Arduino\_JSON.h>  #define DHTPIN 7u  #define DHTTYPE DHT22  #define RELEPIN 0u  #define MESH\_PREFIX "MYMESH"  #define MESH\_PASSWORD "password"  #define MESH\_PORT 5555  #define NODE\_NUMBER 2u  #define TEMP\_LOWER\_THRESHOLD 0u  #define TEMP\_UPPER\_THRESHOLD 30u  #define HUM\_LOWER\_THRESHOLD 20u  #define HUM\_UPPER\_THRESHOLD 80u  LiquidCrystal lcd(6, 5, 4, 3, 2, 1);  Scheduler userScheduler;  painlessMesh mesh;  DHT dht(DHTPIN, DHTTYPE);    void sendMessage();  Task taskSendMessage( TASK\_SECOND \* 1 , TASK\_FOREVER, &sendMessage );  void validateData(double temperature, double humidity)  {  if ((temperature < TEMP\_LOWER\_THRESHOLD) || (temperature > TEMP\_UPPER\_THRESHOLD))  {  analogWrite(RELEPIN, HIGH);  }  else  {  analogWrite(RELEPIN, LOW);  }  if ((humidity < HUM\_LOWER\_THRESHOLD) || (humidity > HUM\_UPPER\_THRESHOLD))  {  analogWrite(RELEPIN, HIGH);  }  else  {  analogWrite(RELEPIN, LOW);  }  }  String getData ()  {  JSONVar jsonReadings;  String data;  jsonReadings["node"] = NODE\_NUMBER;  jsonReadings["temp"] = dht.readTemperature();  jsonReadings["hum"] = dht.readHumidity();  validateData(jsonReadings["temp"], jsonReadings["hum"]);  data = JSON.stringify(jsonReadings);  return data;  }  void sendMessage ()  {  String msg = getData();  mesh.sendBroadcast(msg);  }  void receivedCallback(uint32\_t source, String &msg)  {  JSONVar data = JSON.parse(msg.c\_str());  String nodeType = data["node"];    if (nodeType == "temperature\_humidity")  {  double temp = data["temp"];  double hum = data["hum"];    lcd.clear();  lcd.setCursor(0, 0);  lcd.print("Temp: ");  lcd.print(temp);  lcd.setCursor(0, 1);  lcd.print("Humidity: ");  lcd.print(hum);  }  else if (nodeType == "gas\_level")  {  double gasLevel = data["gas\_level"];    lcd.clear();  lcd.print("Gas Level:");  lcd.print(gasLevel);  }  else  {  lcd.clear();  lcd.print("Invalid data received");  }  }  void newConnectionCallback(uint32\_t nodeId)  {  lcd.clear();  lcd.printf("New Connection, nodeId = %u\n", nodeId);  }  void changedConnectionCallback()  {  lcd.clear();  lcd.printf("Changed connections\n");  }  void nodeTimeAdjustedCallback(int32\_t offset)  {  lcd.clear();  lcd.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(),offset);  }  void setup()  {  lcd.begin(16, 2);  dht.begin();  mesh.setDebugMsgTypes( ERROR | STARTUP );  mesh.init( MESH\_PREFIX, MESH\_PASSWORD, &userScheduler, MESH\_PORT );  mesh.onReceive(&receivedCallback);  mesh.onNewConnection(&newConnectionCallback);  mesh.onChangedConnections(&changedConnectionCallback);  mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);  userScheduler.addTask( taskSendMessage );  taskSendMessage.enable();  lcd.clear();  }  void loop()  {  mesh.update();  } |

**Codul nodului 2 (Colectarea datelor de concentrație a gazului):**

|  |
| --- |
| #include <painlessMesh.h>  #include <DHT.h>  #include <LiquidCrystal.h>  #include <Arduino\_JSON.h>  #define MQ2\_PIN 7u  #define RELEPIN 0u  #define MESH\_PREFIX "MYMESH"  #define MESH\_PASSWORD "password"  #define MESH\_PORT 5555  #define NODE\_NUMBER 2u  #define GAS\_LOWER\_THRESHOLD 200u  #define GAS\_UPPER\_THRESHOLD 800u  LiquidCrystal lcd(6, 5, 4, 3, 2, 1);  Scheduler userScheduler;  painlessMesh mesh;    void sendMessage();  Task taskSendMessage( TASK\_SECOND \* 1 , TASK\_FOREVER, &sendMessage );  void validateData(int gasLevel)  {  if ((gasLevel < GAS\_LOWER\_THRESHOLD) || (gasLevel > GAS\_UPPER\_THRESHOLD))  {  analogWrite(RELEPIN, HIGH);  }  else  {  analogWrite(RELEPIN, LOW);  }  }  String getData()  {  JSONVar jsonReadings;  String data;  jsonReadings["node"] = NODE\_NUMBER;  jsonReadings["gas"] = analogRead(MQ2\_PIN);  validateData(jsonReadings["gas"]);  data = JSON.stringify(jsonReadings);  return data;  }  void sendMessage ()  {  String msg = getData();  mesh.sendBroadcast(msg);  }  void receivedCallback(uint32\_t source, String &msg)  {  JSONVar data = JSON.parse(msg.c\_str());  int nodeType = data["node"];    if (1u == nodeType)  {  double temp = data["temp"];  double hum = data["hum"];    lcd.clear();  lcd.setCursor(0, 0);  lcd.print("Temp: ");  lcd.print(temp);  lcd.setCursor(0, 1);  lcd.print("Humidity: ");  lcd.print(hum);  }  else if (2u == nodeType)  {  double gasLevel = data["gas"];    lcd.clear();  lcd.print("Gas Level:");  lcd.print(gasLevel);  }  else  {  lcd.clear();  lcd.print("Invalid data received");  }  }  void newConnectionCallback(uint32\_t nodeId)  {  lcd.clear();  lcd.print("New nodeId = ");  lcd.print(nodeId);  }  void changedConnectionCallback()  {  lcd.clear();  lcd.print("Changed connections");  }  void nodeTimeAdjustedCallback(int32\_t offset)  {  lcd.clear();  lcd.print("Adjusted time, offset = ");  lcd.print(offset);  }  void setup()  {  pinMode(MQ2\_PIN, INPUT);  lcd.begin(16, 2);  mesh.setDebugMsgTypes( ERROR | STARTUP );  mesh.init( MESH\_PREFIX, MESH\_PASSWORD, &userScheduler, MESH\_PORT );  mesh.onReceive(&receivedCallback);  mesh.onNewConnection(&newConnectionCallback);  mesh.onChangedConnections(&changedConnectionCallback);  mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);  userScheduler.addTask( taskSendMessage );  taskSendMessage.enable();  lcd.clear();  }  void loop()  {  mesh.update();  } |

**Codul nodului principal (de trimitere a datelor la web server):**

|  |
| --- |
| #include <LiquidCrystal.h>  #include <painlessMesh.h>  #include <Arduino\_JSON.h>  #define SSID "Ailin"  #define PASSWORD "22042016"  #define MESH\_PREFIX "MYMESH"  #define MESH\_PASSWORD "password"  #define MESH\_PORT 5555  #define NODE\_NUMBER 0u  Scheduler userScheduler;  painlessMesh mesh;  WiFiServer server(80);  LiquidCrystal lcd(6, 5, 4, 3, 2, 1);  void sendMessage(void);  Task taskSendMessage( TASK\_SECOND \* 1 , TASK\_FOREVER, &sendMessage );  double temp;  double hum;  double gasLevel;  void sendMessage(void)  {  /\* avoid compiler warnings \*/  }  void receivedCallback(uint32\_t source, String &msg)  {  JSONVar data = JSON.parse(msg.c\_str());  int nodeType = data["node"];    if (1u == nodeType)  {  temp = data["temp"];  hum = data["hum"];    lcd.clear();  lcd.setCursor(0, 0);  lcd.print("Temp: ");  lcd.print(temp);  lcd.setCursor(0, 1);  lcd.print("Humidity: ");  lcd.print(hum);  }  else if (2u == nodeType)  {  gasLevel = data["gas"];    lcd.clear();  lcd.setCursor(0, 0);  lcd.print("Gas Level: ");  lcd.print(gasLevel);  }  else  {  lcd.clear();  lcd.setCursor(0, 0);  lcd.print("Invalid data received");  }  }  void newConnectionCallback(uint32\_t nodeId)  {  lcd.clear();  lcd.setCursor(0, 0);  lcd.print("New nodeId = ");  lcd.print(nodeId);  }  void changedConnectionCallback()  {  lcd.clear();  lcd.setCursor(0, 0);  lcd.print("Changed connections\n");  }  void nodeTimeAdjustedCallback(int32\_t offset)  {  lcd.clear();  lcd.setCursor(0, 0);  lcd.print("Adjusted time");  }  void setup()  {  lcd.begin(16, 2);    mesh.setDebugMsgTypes( ERROR | STARTUP );  mesh.init( MESH\_PREFIX, MESH\_PASSWORD, &userScheduler, MESH\_PORT );  mesh.onReceive(&receivedCallback);  mesh.onNewConnection(&newConnectionCallback);  mesh.onChangedConnections(&changedConnectionCallback);  mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);  userScheduler.addTask( taskSendMessage );  taskSendMessage.enable();    connectToWiFi();  server.begin();  lcd.clear();  lcd.setCursor(0, 0);  lcd.print("HTTP server started");  }  void loop()  {  WiFiClient client = server.available();  if (client)  {  client.println("HTTP/1.1 200 OK");  client.println("Content-Type: text/html");  client.println();  client.print("<!DOCTYPE html><html><head><title>Monitorizarea aerului</title></head><body>");  client.print("<h1>Temperatura: ");  client.print(temp);  client.print(" °C</h1>");  client.print("<h1>Umiditatea: ");  client.print(hum);  client.print(" %</h1>");  client.print("<h1>Nivelul de gaze: ");  client.print(gasLevel);  client.print("</h1");  client.print("</body></html>");  delay(100);  client.stop();  }  mesh.update();  }  void connectToWiFi()  {  lcd.clear();  lcd.print("Connecting to WiFi");  WiFi.begin(SSID, PASSWORD);  while (WiFi.status() != WL\_CONNECTED)  {  delay(1000);  lcd.clear();  lcd.print("Connecting...");  }  lcd.clear();  lcd.print("Connected to WiFi");  } |

# Descrierea modului de utilizare a sistemului proiectat

Sistemul elaborat este relativ simplu. Utilizatorul trebuie să amplaseze fiecare modul independent împreună cu senzorul respectiv în locul unde va dori să fie monitorizate caracteristicile date. Datele vor fi colectate în dependență de intervalul de actualizare a datelor din rețeaua mesh, adică 1s.

Ca sistemul să fie mai prietenos utilizatorului, pe câte un ecran LCD vor fi afișate datele colectate intermediare. Iar ca măsură de precauție, un LED roșu se va aprinde atunci când valorile depășesc norma prestabilită pentru condițiile normale.

Ca o posibilitate de adăugare în viitor poate fi folosit un umidificator care va fi conectat la modulul de monitorizare a umidității aerului, sau, alternativ, la un alt modul ESP32 conectat la rețeaua mesh dată.

Utilizatorul poate monitoriza, de asemenea, datele pe web serverul routerului la care a fost conectat modulul principal, adică va putea accesa website-ul folosind adresa principală a routerului, pe portul 80.

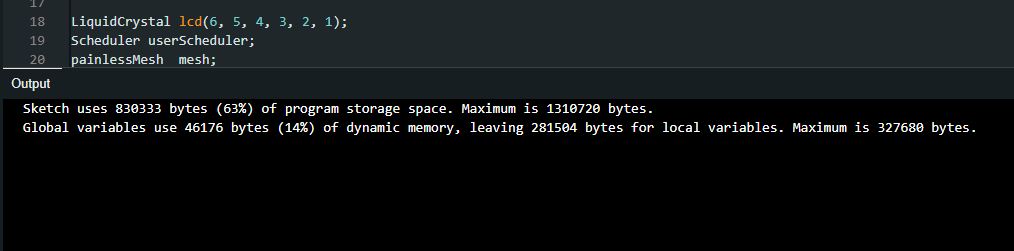
# Testarea funcționala a proiectului (ESP, NodeMCU, Genuino)

În Figura 7.1 este prezentată interfața GUI a sistemului.

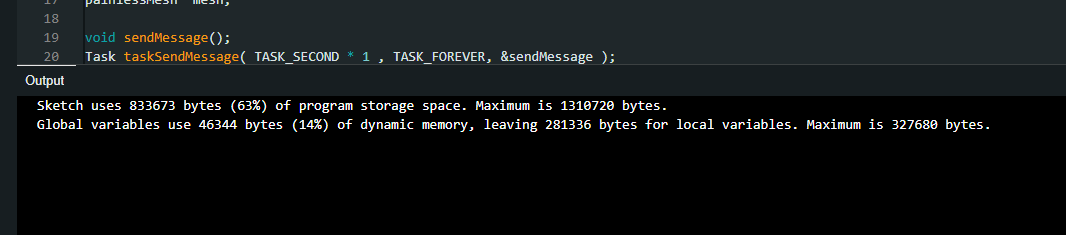
****

**Figura 7.1.** Interfața GUI a sistemului.

Rezultatul compilării produsului program elaborat în mediul de dezvoltare Arduino IDE este prezentat mai jos.

****

**Figura 7.2.** Rezultatul compilării programului pentru nodul 1.



**Figura 7.3.** Rezultatul compilării programului pentru nodul 2.

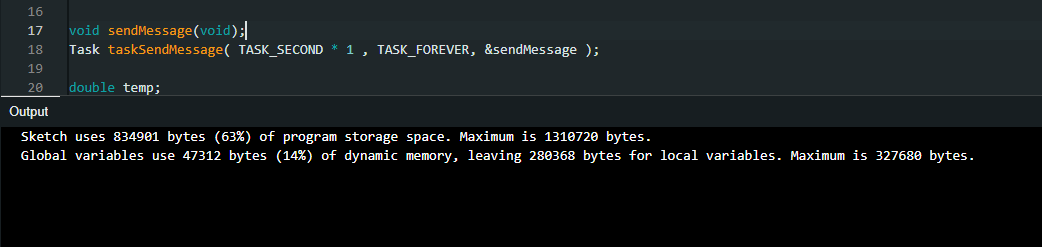


Figura 7.4. Rezultatul compilării programului pentru nodul principal.

# Concluzii

În urma cercetării și realizării acestui proiect cu tema : “Proiectarea unui sistem de monitorizare a calității aerului” am ajuns la următoarele concluzii:

1. Existența unui astfel de sistem în casele oamenilor ar ușura viața nu doar a celor care suferă de afecțiuni respiratorii, ci și a celor sănătoși, dat fiind că aerul prea fierbinte sau uscat provoacă senzații neplăcute sau efecte temporare precum sângerări nazale.
2. Ca protențial plan pentru proiectul dat ar putea fi un umidificator care poate înlocui LE  
   D-ul existent ce este activat cu ajutorul releului la depășirea limitelor prestabilite.
3. Mașina virtuală Proteus este un instrument folositor, însă nu are toate librăriile necesare. Astfel, am întâlnit probleme la determinarea pinajului plăcii ESP32, fiind diferit față de cel fizic, și nu a fost posibilă adăugarea unui umidificator deoarece nu există librăria pentru modelul ales.

# Date bibliografice:

[1] <https://alecoair.ro/blog/cum-te-ajuta-un-umidificator-sa-iti-pastrezi-sanatatea-efectele-nocive-ale-aerului-prea-uscat-asupra-organismului>

[2] <https://www.medicover.ro/despre-sanatate/astm-bronsic-cauze-simptome-si-tratament,355,n,295>

[3] <https://en.wikipedia.org/wiki/ESP32>

[4] <https://robotica.md/senzor-umiditate-temperatura-DHT22>

[5] <https://wiki.seeedstudio.com/Grove-Gas_Sensor-MQ2/>

[6] <https://embeddedcenter.wordpress.com/ece-study-centre/display-module/lcd-16x2-lm016l/>

[7] <https://eepower.com/resistor-guide/resistor-fundamentals/what-is-a-resistor/>

[8] <https://www.thespruce.com/what-is-grounding-1152859>

[9] <https://www.galco.com/comp/prod/relay.htm>

[10] <https://www.alldatasheet.com/datasheet-pdf/pdf/15068/PHILIPS/2N2369.html>

[11] <https://www.bbc.co.uk/bitesize/guides/zsgtw6f/revision/1>

[12] <https://ro.wikipedia.org/wiki/LED>

[13] <https://www.arduino.cc/reference/en/libraries/painless-mesh/>

[14] <https://www.arduino.cc/en/Guide/Introduction>