**MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**RAPORT**

Lucrare de laborator nr. 7

la cursul ***„Interfețe de comunicare”***

**Tema: Programarea aplicațiilor cu acces Wireless. Regim Client și Client-Server.**

**A efectuat :**  **St. gr. CR-221FR Serba Cristina**

**A verificat:**  **Ababii Victor**

**Chișinău 2024**

# Tema lucrării

Programarea aplicațiilor cu acces Wireless. Regim Client și Client-Server.

# Scopul lucrării:

# Cercetarea, proiectarea și programarea dispozitivelor cu comunicare Wireless (ESP 8266, ESP32 sau Genuino 101).

# Resurse hardware şi software necesare pentru efectuarea lucrării:

1. Calculator PC;

2. Acces la Internet;

3. Mediul de proiectare Proteus sau Fritzing;

4. Mediul de programare Arduino IDE.

5. Îndrumare electronice pentru utilizarea, proiectarea și programarea sistemelor în baza Kit-ului ESP8266 / ESP32 / Arduino UNO / Genuino 101.

# Sarcina tehnică pentru efectuarea lucrării de laborator:

1. Să se instaleze mediul de dezvoltare Arduino IDE și bibliotecile respective pentru Kit-ul ESP8266, ESP32 sau Genuino 101;

2. Să se instaleze mediul de dezvoltare Proteus și bibliotecile respective pentru Kit-ul ESP8266 si ESP32;

3. Să se asambleze schema electrică de principiu (Proteus/Fritzing) a sistemului casa inteligenta în baza Kit-ului de dezvoltare ESP8266 / ESP32 / Genuino 101. In calitate de set de senzori sunt selectați toate dispozitivele care sunt in dispoziția sistemului.

4. Să se elaboreze programul care efectuează operații de afisare si control al sistemului casa inteligenta.

5. Starile intermediare ale sistemului sa se afiseze in fereastra Serial Terminal a mediului de dezvoltare Arduino IDE;

6. Sa se asambleze schema electrica de principiu a sistemului Casa Inteligenta in baza Kit-ului ESP8266 / ESP32 / Genuino 101 cu conectarea setului de senzori care sunt la dispozitie;

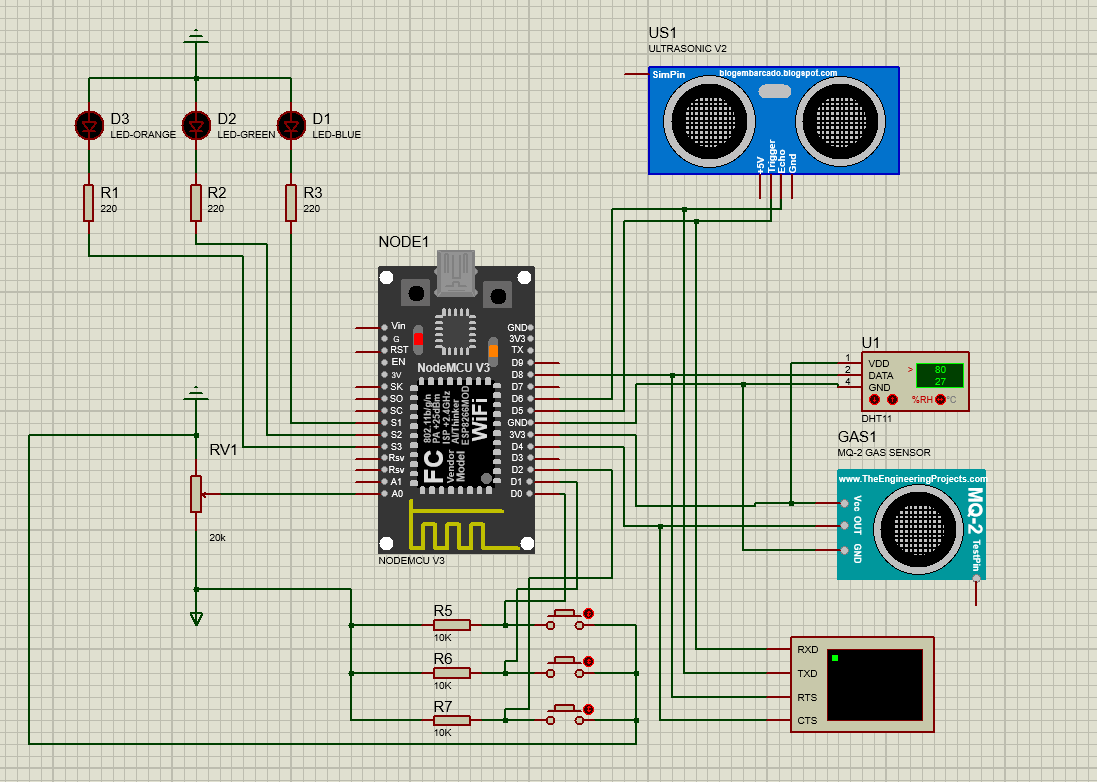
7. Să se incarce codul HEX al programului elaborat în mediul Proteus pentru testare funcțională (Daca sistemul permite);

8. Procesul de testare functionala / simularea sa se înregistreze prin Fotografii sau PrintScrean-uri.

# Algoritmul de funcționare al sistemului

La placa de baza NodeMCU au fost conectate 3 LED-uri pentru controlul datelor, un sensor ultrasonic, sensor de umiditate si temperatura DHT11 si un sensor de gaz MQ-2. Pentru controlul celor 3 senzori au fost conectate 3 butoane. Astfel, la apasarea unui buton si primirea de date valide de la sensor, va fi iluminat LED-ul corespunzator.

# Schema model pentru proiectare si cercetare



# Codul sursa C și HEX ale programelor elaborate:

|  |
| --- |
| #include <ESP8266WiFi.h>  #include <DHT.h>  #define DHTPIN      D8  #define DHTTYPE     DHT11  #define LED1        1  #define LED2        2  #define LED3        3  #define GAS\_SENSOR  D4  #define TRIGGER\_PIN D5  #define ECHO\_PIN    D6  #define BUTTON1     D0  #define BUTTON2     D1  #define BUTTON3     D3  DHT dht(DHTPIN, DHTTYPE);  void setup()  {    Serial.begin(9600);    /\* Pinii pentru afisarea datelor \*/    pinMode(LED1, OUTPUT);    pinMode(LED2, OUTPUT);    pinMode(LED3, OUTPUT);    /\* Pinii pentru primirea datelor de la senzori \*/    pinMode(GAS\_SENSOR, INPUT);    pinMode(TRIGGER\_PIN, OUTPUT);    pinMode(ECHO\_PIN, INPUT);    /\* Pinii pentru butoane \*/    pinMode(BUTTON1, INPUT);    pinMode(BUTTON2, INPUT);    pinMode(BUTTON3, INPUT);  }  void loop()  {    int buttonState1 = digitalRead(BUTTON1);    int buttonState2 = digitalRead(BUTTON2);    int buttonState3 = digitalRead(BUTTON3);    /\* Primeste date despre umiditate si temperatura de la DHT 11 \*/    float humidity = dht.readHumidity();    float temperature = dht.readTemperature();    /\* Primeste date despre nivelul gazului de la MQ-2 \*/    int gasLevel = analogRead(GAS\_SENSOR);    /\* Primeste date despre distanta de la ultrasonic \*/    long duration, distance;    digitalWrite(TRIGGER\_PIN, LOW);    delayMicroseconds(2);    digitalWrite(TRIGGER\_PIN, HIGH);    delayMicroseconds(10);    digitalWrite(TRIGGER\_PIN, LOW);    duration = pulseIn(ECHO\_PIN, HIGH);    distance = (duration / 2) / 29.1;    /\* Scrie pe LED1 daca datele intra in interval \*/    if ((temperature > 25 || humidity > 70) && (buttonState1 == HIGH)) {      digitalWrite(LED1, HIGH);    } else {      digitalWrite(LED1, LOW);    }    /\* Scrie pe LED2 daca datele intra in interval \*/    if ((gasLevel > 500) && (buttonState2 == HIGH)) { // Adjust threshold as needed      digitalWrite(LED2, HIGH);    } else {      digitalWrite(LED2, LOW);    }    /\* Scrie pe LED3 daca datele intra in interval \*/    if ((distance < 20) && (buttonState2 == HIGH)) {      digitalWrite(LED3, HIGH);    } else {      digitalWrite(LED3, LOW);    }    Serial.print("Temperatura: ");    Serial.print(temperature);    Serial.print(" °C\tUmiditatea: ");    Serial.print(humidity);    Serial.print(" %\tNivel Gaz: ");    Serial.print(gasLevel);    Serial.print("\tDistanta: ");    Serial.print(distance);    Serial.println(" cm");    delay(1000);  } |

# Concluzii:

În concluzie, in urma efectuarii lucrarii cu scopul de a cerceta, proiecta și programa dispozitivelor cu comunicare, sistemul a încorporat diverși senzori, inclusiv butoane, senzor de gaz, senzor de umiditate (DHT11) și senzor cu ultrasunete pentru a monitoriza condițiile de mediu și intrarea utilizatorului. În plus, indicatorii LED au fost folosiți pentru a oferi feedback vizual bazat pe citirile senzorilor și interacțiunile utilizatorului. Au fost implementate:

* Proiectarea schemei electrice în Proteus pentru a reprezenta arhitectura sistemului de casă inteligentă.
* Dezvoltarea programului Arduino pentru a gestiona achiziția datelor senzorilor, procesarea intrărilor utilizatorului și controlul LED-urilor.
* Integrarea funcționalității butoanelor pentru a permite interacțiunea utilizatorului cu sistemul.