Grigore Vlad-Gabriel

Grupa 341B2

Interfață grafică și control automatizat Arduino - simulare sistem SCADA

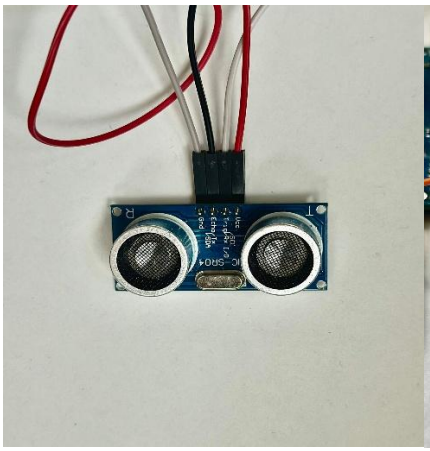
1.Introducere

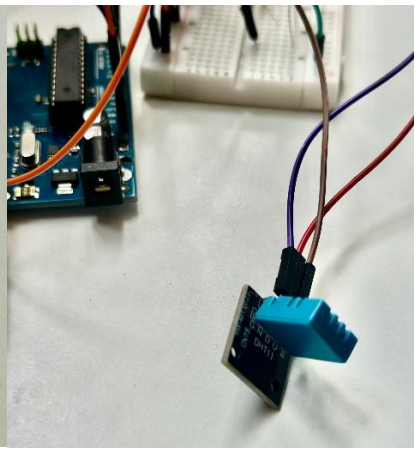
Proiectul propus urmărește realizarea unei aplicații demonstrative ce simulează funcționalitatea unui sistem de conducere integrat (SIC) utilizând platforma Arduino. Sunt integrate concepte fundamentale din SCADA (Supervisory Control and Data Acquisition) prin monitorizarea în timp real a unor parametri de mediu, controlul unui actuator (servo) și interacțiune cu operatorul prin intermediul unei interfețe fizice (LCD + butoane).

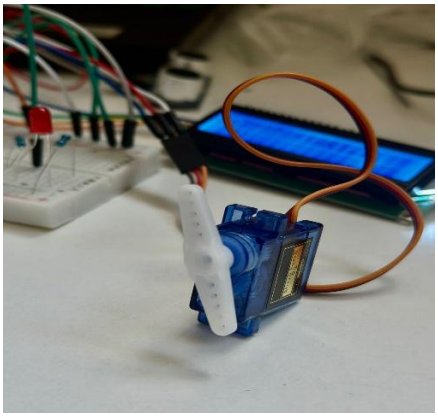
2. Justificarea alegerii componentelor

Nivelul 1 – Achiziție și execuție:

Acest nivel reprezintă punctul de contact direct cu procesul fizic, asigurând colectarea fidelă a datelor și transpunerea comenzilor de control în acțiuni concrete. În cadrul aplicației, achiziția de date se realizează prin intermediul a doi senzori: DHT11, care furnizează în timp real valorile temperaturii și umidității, și senzorul ultrasonic, utilizat pentru detectarea proximității inițiale a operatorului. Controlul procesului este concretizat prin comanda unui servo motor, care răspunde la semnalele primite, modificându-și poziția în funcție de rezultatele prelucrării algoritmice. Acest nivel îmbină măsurarea cu execuția, constituind fundamentul oricărui sistem de control automatizat.







Nivelul 2 – Control local:

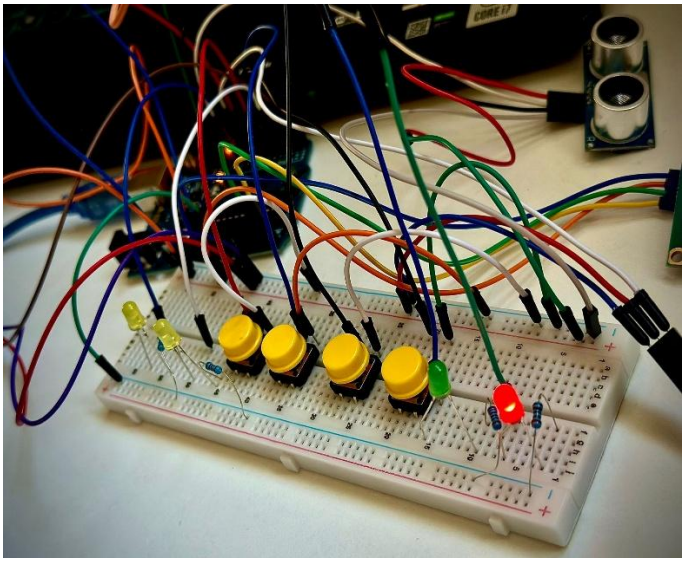
La acest nivel are loc prelucrarea decizională a informațiilor provenite din teren. Algoritmul implementat reproduce principiile de bază ale unui regulator bi- sau tri-pozițional, adaptat unei aplicații simplificate, dar ilustrative. Decizia de control este luată pe baza unei mărimi compuse – media aritmetică dintre temperatură și umiditate – care este apoi mapată într-un unghi de poziționare al servo motorului. Această abordare transformă un semnal complex în comenzi clare și cuantificabile, accentuând funcția de conversie inteligentă și reacție rapidă caracteristică nivelului de control local.

Nivelul 3 – Interfață om-mașină (HMI)

Interfața om-mașină este realizată cu ajutorul unui ecran LCD 16x2, care permite vizualizarea în timp real a valorilor achiziționate și a stării sistemului. Această componentă are rol dublu: informativ și decizional. Operatorul poate interpreta cu ușurință datele aflate pe prima linie a ecranului – temperatura și umiditatea mediului ambiant – în timp ce linia a doua reflectă starea sistemului: modul de operare (automat sau manual) și unghiul actual al actuatorului. Asemănător unui panou sinoptic din sistemele industriale, acest HMI oferă o imagine sintetică și clară a procesului, consolidând relația om-mașină într-un mod accesibil și intuitiv.

Nivelul de siguranță operațională (Safety):

Pentru asigurarea protecției operatorului și a sistemului în ansamblu, a fost implementat un mecanism de oprire de urgență (STOP), care întrerupe instantaneu orice funcționare activă. Acest mecanism nu este doar o simplă oprire logică, ci o revenire controlată la o stare de siguranță: servo motorul revine la poziția inițială, toate LED-urile indică oprirea, iar afișajul semnalează clar statusul STOP. Astfel, se respectă principiile esențiale ale sistemelor de control sigure – prevenirea deteriorării echipamentului, protejarea utilizatorului și menținerea unei conduite predictibile în caz de eroare sau intervenție externă. Acest nivel reflectă preocuparea pentru responsabilitate și fiabilitate, esențială în orice sistem modern automatizat.



3. Structura sistemului

Canale de intrare (analogice și digitale):

Sistemul integrează două tipuri de canale de intrare, utilizate pentru captarea informațiilor esențiale din mediul fizic și pentru interacțiunea cu operatorul:

Achiziție analogică (AI): Se realizează prin intermediul senzorului DHT11, care furnizează simultan valorile temperaturii și umidității aerului, și al senzorului ultrasonic, utilizat pentru detectarea prezenței în proximitate. Aceste date sunt convertite și prelucrate pentru a fundamenta deciziile de control.

Intrări digitale (DI): Patru butoane – START, STOP, MODE și STEP – permit declanșarea, întreruperea, comutarea modului de funcționare și navigarea manuală prin secvențele definite ale sistemului.

Canale de ieșire (digitale și analogice simulate):

Comanda elementelor de execuție este realizată prin ieșiri digitale și analogice simulate, care transmit în mod direct semnalele de control către actuator și elementele de semnalizare:

Ieșiri digitale (DO): Patru LED-uri indică în mod vizual stările fundamentale ale sistemului: activarea comenzii START, intrarea în regim STOP, funcționarea în mod AUTO, respectiv MANUAL.

Ieșire analogică simulată (AO): Servo motorul este controlat prin semnal PWM, iar poziția acestuia reflectă decizia algoritmului de control în raport cu datele senzorilor sau cu interacțiunea manuală.

Interfață om-mașină (HMI):

Interacțiunea utilizatorului cu sistemul este mediată printr-o interfață intuitivă, compusă din: Display LCD cu interfață I2C: Acesta asigură afișarea în timp real a parametrilor monitorizați (temperatură, umiditate), precum și a stării curente a sistemului (mod de funcționare și unghiul servo motorului).

Butoane fizice de control: Acestea oferă utilizatorului posibilitatea de a selecta modul de operare dorit și de a acționa manual asupra secvenței de funcționare, contribuind astfel la testarea și supravegherea activă a procesului.

4. Algoritm de funcționare

1. Inițial, se cere apropierea de senzorul de distanță pentru inițializarea interacțiunii.
2. Se apasă pe start pentru a putea începe procesul.
3. Se selectează modul (AUTO sau MANUAL) prin apăsarea butonului MODE o dată sau de două ori.
4. Modul AUTO:
    a. Se calculează media valorilor T + H.
    b. Se mapează la un unghi între 0°–180°.
    c. Servo-ul se mișcă treptat în pași de +30° cu întârzieri (profil de viteză).
5. Modul MANUAL:
    a. Se parcurg poziții discrete: 0°, 60°, 90°, 120° la fiecare apăsare STEP, urmând ca după ce se atinge valoarea de 120° să se revină la 0°.
6. STOP general – întrerupe funcționarea atat a modului automat cât și a celui manual, revenind în starea de la 1.

Cod de funcționare:

```
#include <Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

#define DHTPIN 2
#define DHTTYPE DHT11
#define BUTTON_START 3
#define BUTTON_STOP 7
#define BUTTON_MODE 5
#define BUTTON_STEP 6
#define SERVO_PIN 9

#define TRIG_PIN 11
#define ECHO_PIN 12

#define LED_STOP 13
#define LED_START 8
#define LED_AUTO 4
#define LED_MANUAL 10

DHT dht(DHTPIN, DHTTYPE);
Servo myservo;
LiquidCrystal_I2C lcd(0x27, 16, 2);

bool senzorAtins = false;
bool programPornit = false;
bool secventaActiva = true;
bool modManual = false;
int pasCurent = 0;
```

```
int prevStepState = HIGH;

void setup() {
  Serial.begin(9600);
  dht.begin();
  myservo.attach(SERVO_PIN);

  pinMode(BUTTON_START, INPUT_PULLUP);
  pinMode(BUTTON_STOP, INPUT_PULLUP);
  pinMode(BUTTON_MODE, INPUT_PULLUP);
  pinMode(BUTTON_STEP, INPUT_PULLUP);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  pinMode(LED_STOP, OUTPUT);
  pinMode(LED_START, OUTPUT);
  pinMode(LED_AUTO, OUTPUT);
  pinMode(LED_MANUAL, OUTPUT);

  digitalWrite(LED_STOP, HIGH);
  digitalWrite(LED_START, LOW);
  digitalWrite(LED_AUTO, LOW);
  analogWrite(LED_MANUAL, 0);

  lcd.init();
  lcd.backlight();
}
```

```
void loop() {
  if (digitalRead(BUTTON_STOP) == LOW) {
    senzorAtins = false;
    programPornit = false;
    secventaActiva = true;
    modManual = false;
    pasCurent = 0;
    myservo.write(0);

    digitalWrite(LED_STOP, HIGH);
    digitalWrite(LED_START, LOW);
    digitalWrite(LED_AUTO, LOW);
    analogWrite(LED_MANUAL, 0);

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("STOP activat!");
    delay(1000);
    return;
  }

  if (!senzorAtins) {
    long duration;
    float distance;

    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
```

```
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = duration * 0.034 / 2;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Atinge senzorul");

  if (distance > 0 && distance < 20) {
    senzorAtins = true;
    lcd.setCursor(0, 1);
    lcd.print("OK! Apasa START");
    delay(1000);
  }
  delay(300);
  return;
}

if (!programPornit && senzorAtins) {
  digitalWrite(LED_STOP, LOW);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Apasa START...");

  if (digitalRead(BUTTON_START) == LOW) {
    programPornit = true;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Mod: Auto/Man");
    lcd.setCursor(0, 1);
    lcd.print("Apasa MODE 1/2x");
```

```
    int apasari = 0;
    bool aFostApasat = false;
    unsigned long startTime = millis();
    while (millis() - startTime < 2500) {
      if (digitalRead(BUTTON_MODE) == LOW) {
        apasari++;
        aFostApasat = true;
        while (digitalRead(BUTTON_MODE) == LOW);
        delay(200);
      }
    }

    if (!aFostApasat) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Nu ai ales mod!");
      lcd.setCursor(0, 1);
      lcd.print("Reincearca!");
      programPornit = false;
      senzorAtins = false;
      modManual = false;
      delay(2000);
      return;
    }
```

```
    lcd.clear();
    lcd.setCursor(0, 0);
    if (apasari == 2) {
      modManual = true;
      lcd.print("Mod: MANUAL");
    } else {
      modManual = false;
      lcd.print("Mod: AUTO");
    }

    lcd.setCursor(0, 1);
    lcd.print("Apasa START...");
    secventaActiva = true;

    while (digitalRead(BUTTON_START) == HIGH);
    while (digitalRead(BUTTON_START) == LOW);
    delay(300);
  } else {
    delay(300);
    return;
  }
}

float temperatura = dht.readTemperature();
float umiditate = dht.readHumidity();

if (isnan(temperatura) || isnan(umiditate)) {
  lcd.setCursor(0, 0);
  lcd.print("Eroare senzor");
  delay(1000);
  return;
}
```

5. Timp real și execuție în buclă

Structura codului implementat reproduce arhitectura specifică unui sistem de conducere în timp real, în care fluxul de execuție este guvernat de o buclă infinită. Această buclă asigură un ciclu continuu de achiziție a datelor de la senzori, prelucrare logică a informațiilor și aplicare a comenzilor către elementele de execuție. Prin această strategie de actualizare constantă, sistemul răspunde dinamic la variațiile parametrilor monitorizați, oferind o reacție imediată la modificările din mediul înconjurător sau la interacțiunea utilizatorului. Funcționarea în timp real presupune respectarea unui ritm predictibil de execuție, ceea ce permite menținerea coerenței între evoluția fizică a procesului și deciziile software aplicate în timp util.

6. Asemănare cu sistemele SCADA reale

- **Afișajul LCD reproduce funcționalitatea sinopticelor industriale, caracteristice interfețelor de tip HMI (Human-Machine Interface).** Acesta are rolul de a furniza în timp real operatorului informații esențiale privind starea sistemului și valorile parametrilor măsurați, precum temperatura, umiditatea sau unghiul actuatorului. Structura sa dual (două linii de text) permite afișarea simultană a datelor tehnice și a mesajelor de avertizare, replicând comportamentul panourilor sinoptice din industrie, unde lizibilitatea rapidă și claritatea mesajului sunt critice pentru operare eficientă și sigură.LED-urile funcționează ca semnalizare vizuală de stare, analog semnalizărilor luminoase din industrie.

- **LED-urile de stare îndeplinesc funcția unor elemente de semnalizare vizuală, analogă sistemelor de iluminare utilizate în instalațiile industriale.** Fiecare LED reflectă o stare specifică a sistemului: activare, oprire, funcționare în regim automat sau manual. Aceste indicatoare permit o evaluare vizuală imediată, fără a necesita o interpretare textuală sau navigarea prin meniuri. Astfel, se realizează un feedback instant pentru utilizator, conform principiului ergonomiei industriale, unde informația vizuală codificată (prin culoare sau prezență/absență) este preferată pentru reacție rapidă.Modularitatea codului și selecția modurilor de operare reflectă tendințele moderne de portabilitate, control local și integrare a operatorului.

- **Butonul STOP implementează un mecanism de oprire de urgență („fail-safe”), analog unui interlock de siguranță regăsit în sistemele de automatizare industriale.**
  Acest mecanism asigură trecerea imediată a sistemului într-o stare neutră și sigură, întrerupând toate funcțiile active și restabilind condițiile de start. Prin eliminarea oricărui risc asociat continuării necontrolate a procesului, se respectă principiile de bază ale arhitecturilor Safety Instrumented Systems (SIS), în care protecția operatorului și a echipamentelor are prioritate absolută.

- **Modularitatea codului și flexibilitatea modurilor de operare reflectă orientările contemporane în dezvoltarea sistemelor de control distribuit, care favorizează portabilitatea, adaptabilitatea la cerințe locale și implicarea operatorului în lanțul decizional.**

  Prin posibilitatea de a comuta între regimurile AUTO și MANUAL, sistemul oferă atât autonomie funcțională, cât și control direct. Această abordare dual este specifică sistemelor moderne SCADA, în care operatorul poate interveni în proces fie pentru a ajusta manual parametrii, fie pentru a permite rularea automată pe baza unui set de reguli prestabilite. Structura modulară a codului permite, totodată, extinderea și personalizarea ulterioară, favorizând mentenanța, testarea și scalarea sistemului.

7.Concluzii

Proiectul realizat ilustrează cu succes integrarea funcțiilor esențiale ale unui Sistem Integrat de Conducere (SIC) într-o aplicație educațională bazată pe platforma Arduino. Prin intermediul componentelor utilizate – senzori, actuator, interfață de afișare și elemente de comandă – sunt acoperite toate etapele fundamentale ale unui sistem SCADA: achiziția de date, prelucrarea în timp real, decizia locală și comunicarea către operator. Deși natura implementării este una demonstrativă, arhitectura sa modulară și logică replică fidel principiile sistemelor industriale, permițând o scalare ușoară către aplicații reale, cu multiple variabile monitorizate, integrare de protocoale de comunicație (ex: Modbus, OPC) și extinderea către stocarea și analiza datelor istorice. Prin acest demers, proiectul nu doar validează conceptele teoretice studiate, ci oferă și un cadru experimental concret pentru înțelegerea și aprofundarea paradigmei automatizării moderne.

Bibliografie

1. Cursuri SIC - prof. dr. ing. Ciprian Lupu (Cap. 1–10)
2. Documentație Arduino: www.arduino.cc
3. Datasheet-uri componente: DHT11, HC-SR04, SG90
4. Exemple de interfețe SCADA: WinCC, LabView, RSView
5. Cod propriu și testare practică pe placă Arduino UNO