Ministerul Educației și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Ingineria Software și Automatică

Lucrare de laborator Nr. 4

Disciplina: Internetul lucrurilor (IOT)
Tema: Actuatori cu interfaţa binară şi analogică

A efectuat: Popa Cătălin, gr. TI-211

A verificat: Cristian Lupan, asist. univ.

Chișinău 2024

4.1 Problema propusă:

- Sa se realizeze o aplicație in baza de MCU care va controla dispozitivele de acționare cu comenzi recepționate de la interfața serială și raportare către LCD.Sa se achiziționeze semnalul de la senzor;

Dispozitivele de actionare vor fi urmatoarele:

- un bec electric prin intermediul releului cu comenzi de ON si OFF

4.2 Problema propusă:

- Sa se realizeze o aplicatie in baza de MCU care va controla dispozitivele de actionare cu comenzi receptionate de la interfata seriala si raportare catre LCD.

Dispozitivele de actionare vor fi urmatoarele:

- un motor in curent continuii cu comenzi de setare a puterii motorului intre (-100% .. 100%) adica inainte si inapoi, si viteza prin intermediul driverului L298.

Obiective

1. Dezvoltarea unei aplicații pe un microcontroler (MCU) care să controleze dispozitive de acționare, primind comenzi prin interfața serială și raportând starea către un ecran LCD.
2. Achiziționarea unui semnal de la un senzor.
3. Dispozitivul de acționare este un bec electric, care va fi controlat prin intermediul unui releu cu comenzi de ON și OFF.
4. Dispozitivul de acționare este un motor în curent continuu (DC), care va avea comenzi de setare a puterii între -100% și 100%, ceea ce înseamnă controlul direcției (înainte și înapoi).
5. Controlul vitezei și direcției motorului va fi realizat prin intermediul driverului L298**.**

Introducere

Microcontrolerele permit controlul și monitorizarea dispozitivelor fizice, aducând tehnologia mai aproape de utilizator prin intermediul unei interfețe intuitive și simplificate. Un exemplu clasic de interacțiune între utilizator și dispozitive fizice este controlul unui LED prin apăsarea unui buton sau trimiterea de comenzi printr-o interfață serială. Această tehnologie este utilizată într-o gamă largă de aplicații, de la sisteme de iluminat inteligente până la dispozitive electronice complexe utilizate în automatizări industriale. Unul dintre avantajele microcontrolerelor este costul redus și flexibilitatea lor, permițând implementarea rapidă de prototipuri și teste pentru diverse aplicații IoT. Microcontrolerele precum cele din familia Arduino sau ESP sunt folosite frecvent pentru prototiparea sistemelor IoT datorită ecosistemului extins de biblioteci și comunități de dezvoltatori care susțin aceste platforme .

Materiale și metode

Arduino UNO - platformă de microcontroler open-source bazată pe microcontrolerul ATmega328P. Este ușor de utilizat și permite conectarea ușoară a componentelor externe prin pini digitali și analogici. Acesta este ideal pentru prototiparea de sisteme IoT și proiecte electronice interactive. Afișarea este în figura 1.1.

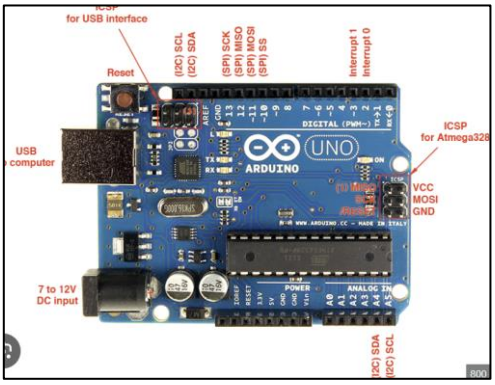


Fig. 1.1. Arduino UNO

- 14 pini digitali I/O (din care 6 sunt PWM);
- 6 intrări analogice;
- memorie flash de 32 KB;
- tensiune de operare: 5V;
- interfață de programare: Arduino IDE.

Un releu **SPDT** (Single Pole Double Throw) este un releu cu un singur pol și două poziții de comutare, utilizat în proiecte cu Arduino pentru a controla circuite externe, în special atunci când dorim să comutăm dispozitive

de putere mare sau tensiuni diferite de cele furnizate de Arduino (de exemplu, lămpi, motoare sau alte echipamente). Afișarea este în figura 1.2.

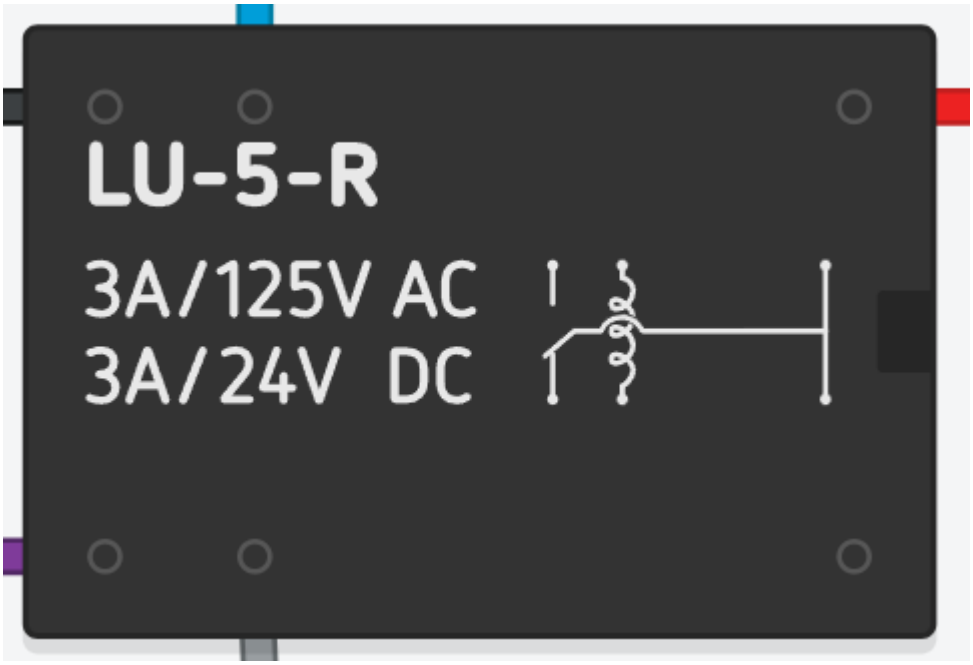


Fig. 1.2. Releu SPDT

- tensiune de operare: 2V-3V;
- curent: ~20mA;
- LED verde pentru semnalizarea codurilor valide;
- LED roșu pentru semnalizarea codurilor invalide.

Rezistor - Rezistorul este utilizat pentru a limita curentul ce trece prin LED, prevenind arderea acestuia. Pentru a proteja LED-urile, se vor folosi rezistoare de aproximativ 220Ω. Afișarea este în figura 1.3.

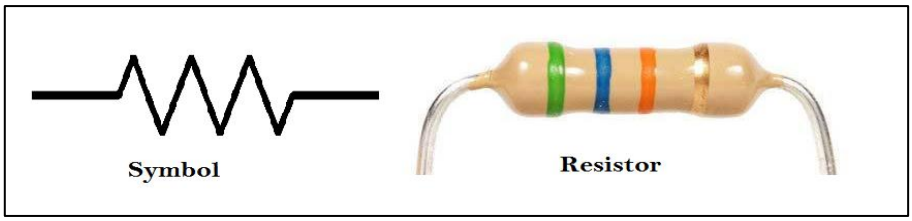


Fig. 1.3. Rezistor

Fire Jumper - Firele jumper sunt utilizate pentru a face legături electrice între diverse componente și plăcuța Arduino. Acestea permit conectarea componentelor fără a fi necesară lipirea acestora. Afișarea este în figura 1.6.

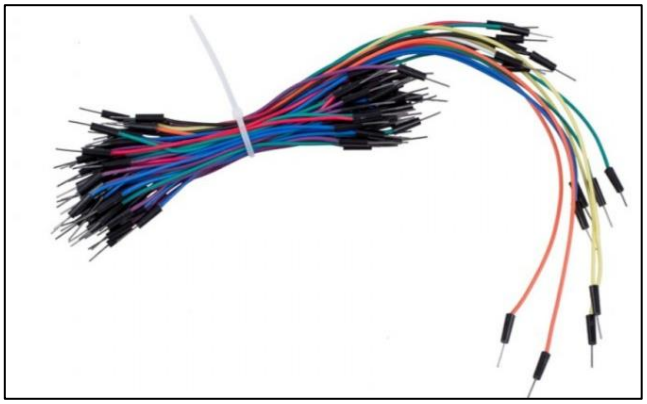


Fig. 1.6. Fire Jumper

Driverul L298 este un circuit integrat utilizat pentru a controla motoare de curent continuu (DC) și motoare pas cu pas. Este un driver de punte H care permite controlul atât al direcției, cât și al vitezei motoarelor. L298 este capabil să controleze două motoare DC simultan sau un singur motor pas cu pas, funcționând pe o tensiune de alimentare de până la 46V și curent maxim de 2A pe fiecare canal. Afișarea este în figura 1.7.

Fig. 1.7. Driver L298

L293D conține două punți H, ceea ce îi permite să controleze fie două motoare DC simultan, fie un motor pas cu pas. O punte H este un circuit electric care permite controlul motorului în două direcții (înainte și înapoi). De asemenea, suportă controlul vitezei motorului prin semnale PWM (Pulse Width Modulation).

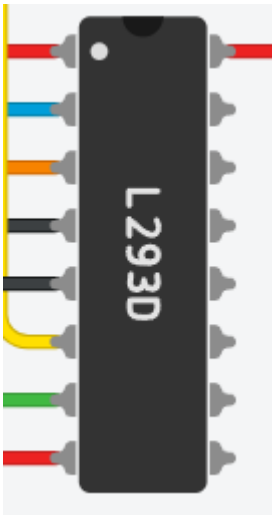


Fig. 1.8 Driver L293D

Un motor DC (motor de curent continuu) este un tip de motor electric care funcționează prin utilizarea curentului continuu (DC) pentru a converti energia electrică în energie mecanică (mișcare). Aceste motoare sunt foarte populare datorită simplității lor, fiind utilizate într-o gamă largă de aplicații, de la jucării la automobile și dispozitive industriale. Funcționarea unui motor DC se bazează pe fenomenul forței magnetice care apare atunci când un curent electric trece printr-un conductor aflat într-un câmp magnetic. Aceasta forță creează o mișcare de rotație în axul motorului.

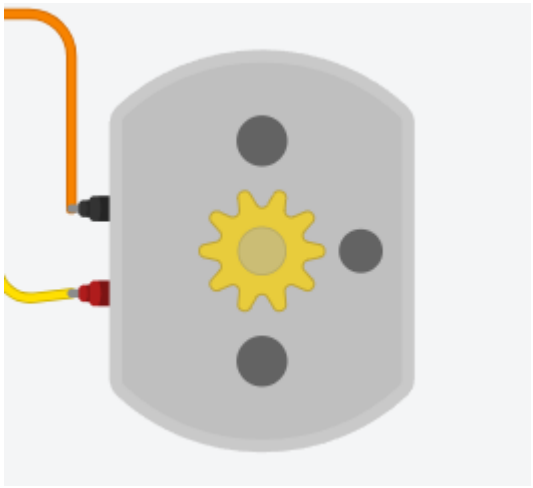


Fig. 1.9 DC Motor

Metode

Configurarea Arduino UNO: Arduino va fi configurat folosind Arduino IDE. Se va implementa codul pentru a citi inputurile de la tastatura 4x4 și pentru a controla LED-urile și afișarea pe LCD.

Interacțiunea cu utilizatorul: Sistemul va primi comenzi fie prin terminalul serial și va răspunde prin intermediul LCD-ului şi serial monitor.

Validare în simulator: Codul și conexiunile vor fi testate inițial într-un simulator de circuite, cum ar fi Tinkercad, pentru a verifica funcționalitatea corectă. După simulare, se va testa și pe un circuit real.

4.1

Pentru prima sarcină am avut nevoie de o lampă, o baterie externă de 9V, un DC motor, arduino UNO, un driver L293D şi un driver L298. Simularea realizată este în figura 1.10.

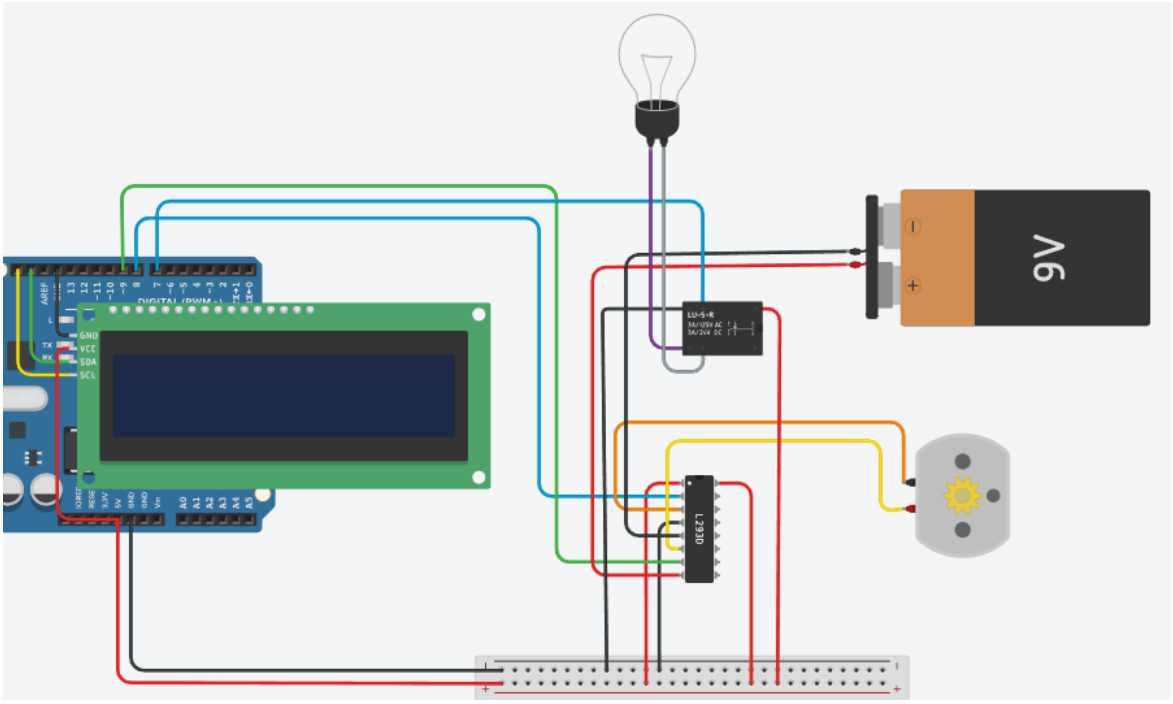


Figura 1.10. – Simulare microcontroler

Mai întâi definim pinii de intrare pentru releu şi motor, care sunt 7,8 şi 9.

```
const int pinMotorA1 = 8;
const int pinMotorA2 = 9;
const int pinRelay = 7;
```

În continuare, iniţializăm interfaţa serială, setăm ecran LCD la 16x2 caractere şi activăm backlight. Setăm pinii pentru control motor şi releu şi setăm ca iniţial să fie motorul şi becul oprit.

```
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.setBacklight(1);

  pinMode(pinMotorA1, OUTPUT);
  pinMode(pinMotorA2, OUTPUT);
  pinMode(pinRelay, OUTPUT);

  digitalWrite(pinMotorA1, LOW);
  digitalWrite(pinMotorA2, LOW);
  digitalWrite(pinRelay, HIGH);
```

În loop verificăm dacă există comenzi noi primite pe interfaţa serială. Citim comanda până la newline şi eliminăm spaţiile albe suplimentare, iar în final procesăm comenzile pentru bec şi motor.

```
void loop() {
  if (Serial.available() > 0) {
    String comanda = Serial.readStringUntil('\n');
    comanda.trim();
    procesareComenzi(comanda);
  }
}
```

Funcţia pentru procesarea comenzilor:

```
void procesareComenzi(String comanda) {
  if (comanda == "MOTOR_FWD") {
    controlMotor(1);
  }
  else if (comanda == "MOTOR_REV") {
    controlMotor(-1);
  }
  else if (comanda == "MOTOR_STOP") {
    controlMotor(0);
  }
  else if (comanda == "LIGHT_ON") {
    controlBec(true);
  }
```

```
  else if (comanda == "LIGHT_OFF") {
    controlBec(false);
  }
  else {
    Serial.println("Comanda necunoscuta.");
  }
}
```

Layer pentru controlul motorului:

```
void controlMotor(int directie) {
  if (directie == 1) {
    digitalWrite(pinMotorA1, HIGH);
    digitalWrite(pinMotorA2, LOW);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Motor: Inainte");
    Serial.println("Motor inainte.");
  }
  else if (directie == -1) {
    digitalWrite(pinMotorA1, LOW);
    digitalWrite(pinMotorA2, HIGH);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Motor: Invers");
    Serial.println("Motor invers.");
  }
  else {
    digitalWrite(pinMotorA1, LOW);
    digitalWrite(pinMotorA2, LOW);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Motor: Oprit");
    Serial.println("Motor oprit.");
  }
}
```

Layer pentru controlul becului:

```
void controlBec(bool stare) {
  if (stare) {
    digitalWrite(pinRelay, LOW);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Bec: ON");
    Serial.println("Bec aprins.");
  }
  else {
    digitalWrite(pinRelay, HIGH);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Bec: OFF");
```

```
    Serial.println("Bec stins.");
  }
}
```

Pentru realizarea punctului 2, am fost nevoit să modificăm codul pentru a putea controla motorul, căruia îi atribuim prin Serial monitor viteza între -100 şi +100. Am avut nevoie de un motor DC, un driver L293D şi o baterie externă de 9V. Simularea este repreyentată în figura 1.11.

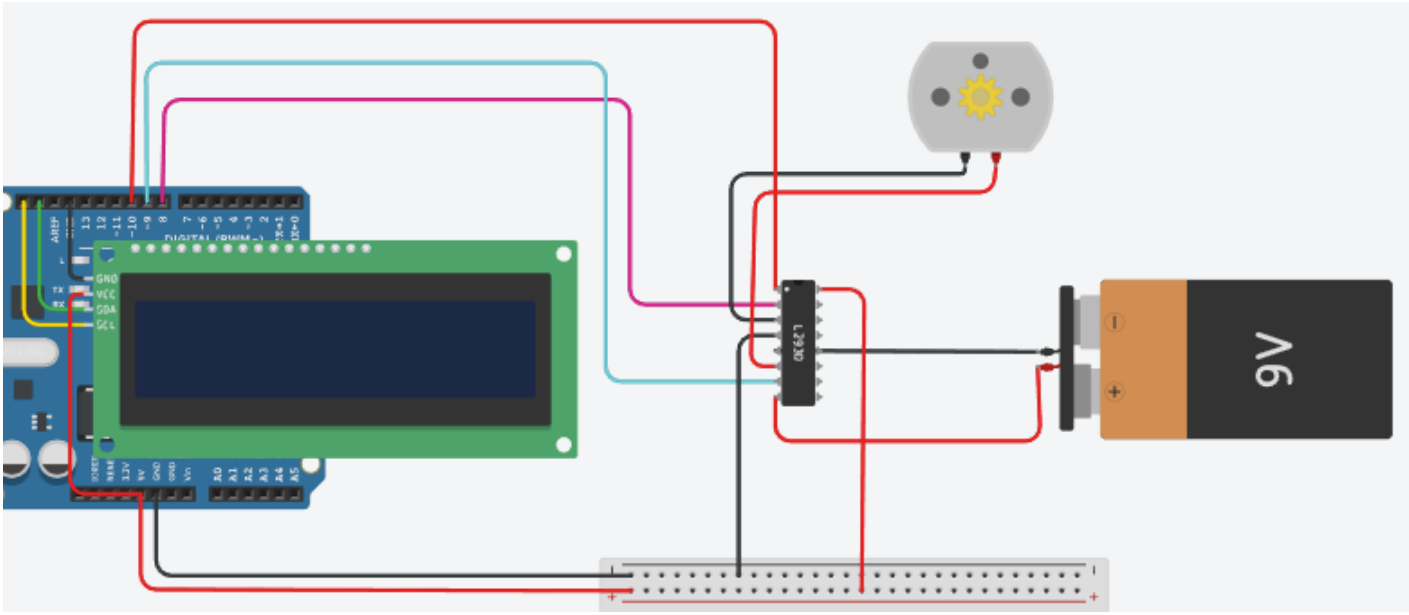


Figura 1.11 – Simulare microcontroler

Mai întai setăm pinii pentru controlul motorului.

```
const int pinIN1 = 8;
const int pinIN2 = 9;
const int pinENA = 10;
```

În setup configurăm pinii şi activăm ăinul pentru motor, pentru a ne asigura că motorul e pornit.

```
pinMode(pinIN1, OUTPUT);
 pinMode(pinIN2, OUTPUT);
 pinMode(pinENA, OUTPUT);
 digitalWrite(pinENA, HIGH);
```

Apoi în loop citim comanda din serial monitor, eliminăm spaţiile şi obţinem valoarea puterii.

```
void loop() {
    if (Serial.available() > 0) {
        String command = Serial.readStringUntil('\n');
        command.trim();

        if (command.startsWith("SET_POWER")) {
            int power = command.substring(10).toInt();
            setMotorPower(power);
        }
    }
}
```

Ultima funcţie creată, este pentru a seta puterea motorului. Mai întâi limităm puterea între +100 şi +100. Apoi verificăm dacă puterea este pozitivă sau negativă, şi în dependenţă de răspuns realizăm prin funcţia map mişcarea motorului înainte sau înapoi.

```
void setMotorPower(int power) {
    if (power > 100) power = 100;
    if (power < -100) power = -100;

    if (power >= 0) {
        digitalWrite(pinIN1, HIGH);
        digitalWrite(pinIN2, LOW);
        analogWrite(pinENA, map(power, 0, 100, 0, 255));
        lcd.clear();
        lcd.print("Motor: Forward");
        lcd.setCursor(0, 1);
        lcd.print("Power: ");
        lcd.print(power);
    } else {
        digitalWrite(pinIN1, LOW);
        digitalWrite(pinIN2, HIGH);
        analogWrite(pinENA, map(-power, 0, 100, 0, 255));
        lcd.clear();
        lcd.print("Motor: Backward");
```

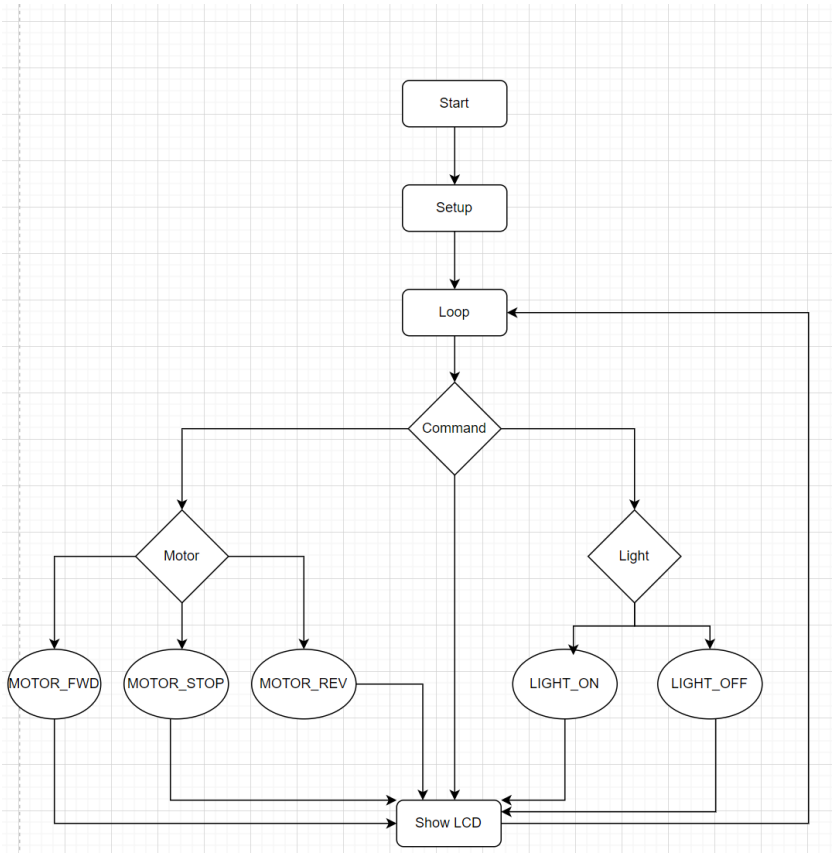
```
        lcd.setCursor(0, 1);
        lcd.print("Power: ");
        lcd.print(-power);
    }
}
```

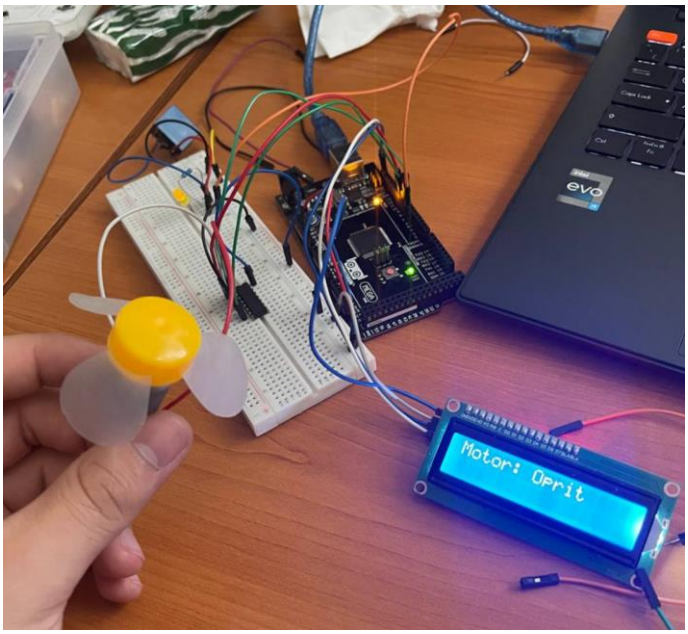
Concluzii

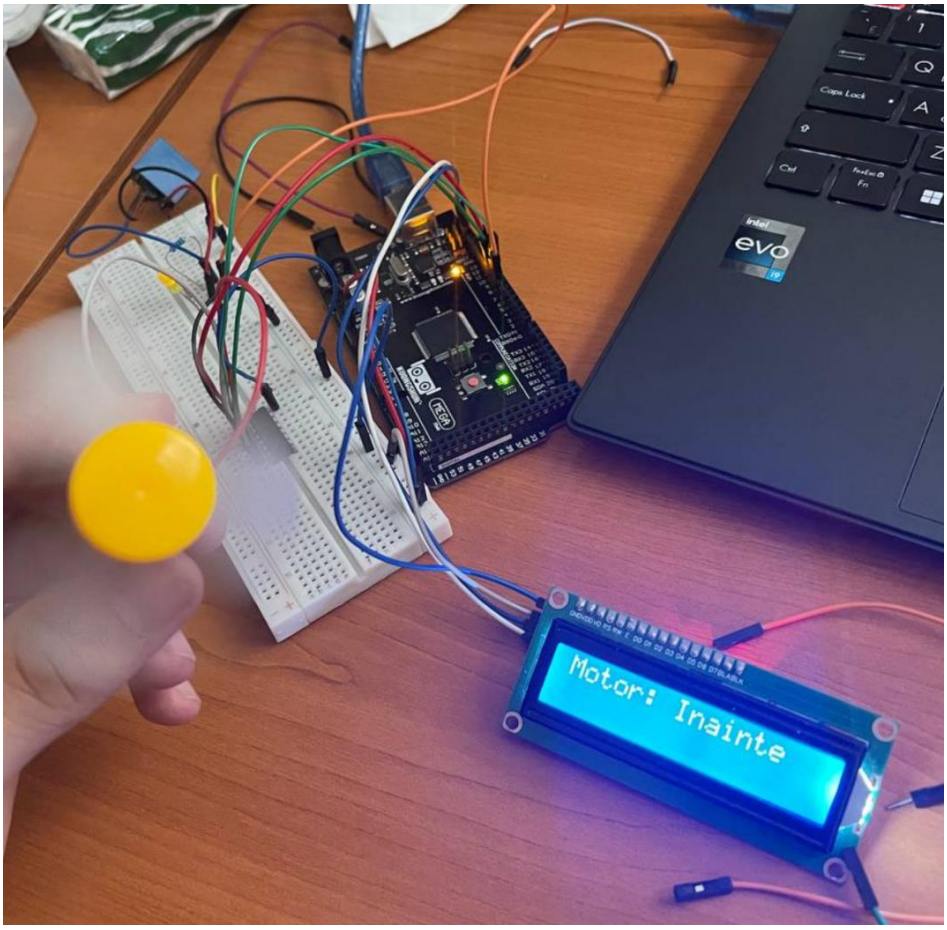
Acest laborator a demonstrat controlul eficient al dispozitivelor de acționare, un bec electric și un motor DC, prin intermediul unui microcontroler (MCU), utilizând comenzi seriale și raportând starea către un ecran LCD. Prin utilizarea unui releu pentru bec și a unui driver L298 pentru motor, s-au ilustrat concepte cheie precum controlul vitezei și direcției motorului, precum și funcționarea on/off a unui bec. Această abordare pune bazele unor aplicații de control automatizat, cu potențial de extindere în diverse sisteme IoT și automatizări industriale.

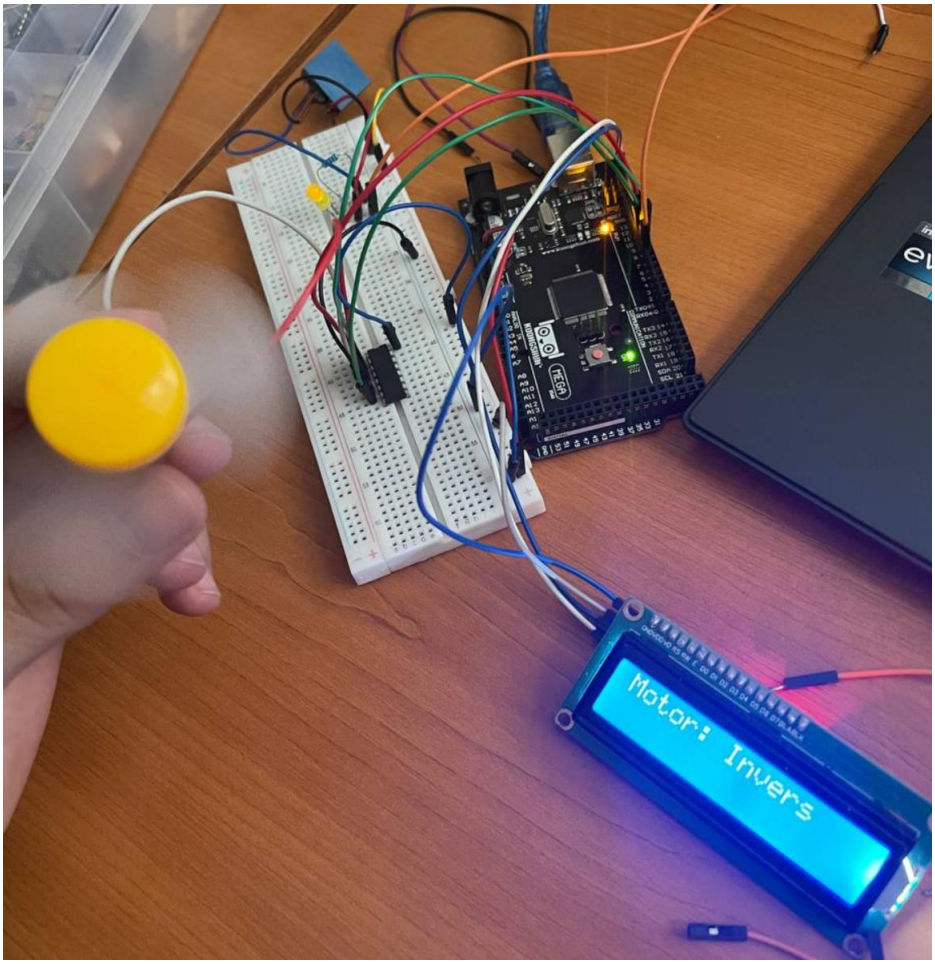
Anexă

4.1


Start
Setup
Loop
Command
Motor
Light
MOTOR_FWD
MOTOR_STOP
MOTOR_REV
LIGHT_ON
LIGHT_OFF
Show LCD


Motor: Oprit


Motor: Inainte




```
#include <Adafruit_LiquidCrystal.h>

Adafruit_LiquidCrystal lcd(0);

const int pinMotorA1 = 8;
const int pinMotorA2 = 9;
const int pinRelay = 7;

void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.setBacklight(1);
```

```
  pinMode(pinMotorA1, OUTPUT);
  pinMode(pinMotorA2, OUTPUT);
  pinMode(pinRelay, OUTPUT);

  digitalWrite(pinMotorA1, LOW);
  digitalWrite(pinMotorA2, LOW);
  digitalWrite(pinRelay, HIGH);

  lcd.setCursor(0, 0);
  lcd.print("Sistem pornit");
}

void loop() {
  if (Serial.available() > 0) {
    String comanda = Serial.readStringUntil('\n');
    comanda.trim();
    procesareComenzi(comanda);
  }
}

void procesareComenzi(String comanda) {
  if (comanda == "MOTOR_FWD") {
    controlMotor(1);
  }
  else if (comanda == "MOTOR_REV") {
    controlMotor(-1);
```

```
  }
  else if (comanda == "MOTOR_STOP") {
    controlMotor(0);
  }
  else if (comanda == "LIGHT_ON") {
    controlBec(true);
  }
  else if (comanda == "LIGHT_OFF") {
    controlBec(false);
  }
  else {
    Serial.println("Comanda necunoscuta.");
  }
}



void controlMotor(int directie) {
  if (directie == 1) {
    digitalWrite(pinMotorA1, HIGH);
    digitalWrite(pinMotorA2, LOW);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Motor: Inainte");
    Serial.println("Motor inainte.");
  }
  else if (directie == -1) {
    digitalWrite(pinMotorA1, LOW);
```

```
    digitalWrite(pinMotorA2, HIGH);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Motor: Invers");
    Serial.println("Motor invers.");
  }
  else {
    digitalWrite(pinMotorA1, LOW);
    digitalWrite(pinMotorA2, LOW);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Motor: Oprit");
    Serial.println("Motor oprit.");
  }
}


void controlBec(bool stare) {
  if (stare) {
    digitalWrite(pinRelay, LOW);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Bec: ON");
    Serial.println("Bec aprins.");
  }
  else {
    digitalWrite(pinRelay, HIGH);
```

```
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Bec: OFF");
    Serial.println("Bec stins.");
  }
}
```

Abstractizare 4.1

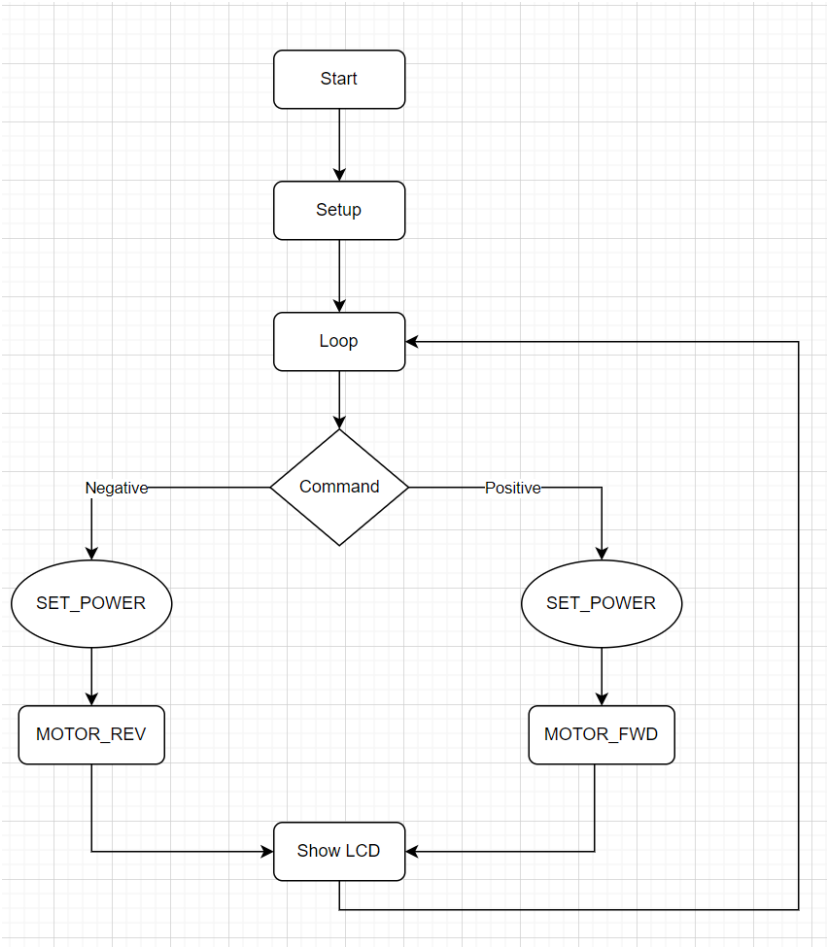
```
#include <Adafruit_LiquidCrystal.h>

Adafruit_LiquidCrystal lcd(0);

const int pinMotorA1 = 8;
const int pinMotorA2 = 9;
const int pinRelay = 7;
const int pinENA = 10;

void setup() {
    Serial.begin(9600);
    lcd.begin(16, 2);
    lcd.setBacklight(1);

    pinMode(pinMotorA1, OUTPUT);
    pinMode(pinMotorA2, OUTPUT);
    pinMode(pinRelay, OUTPUT);
    pinMode(pinENA, OUTPUT);
```

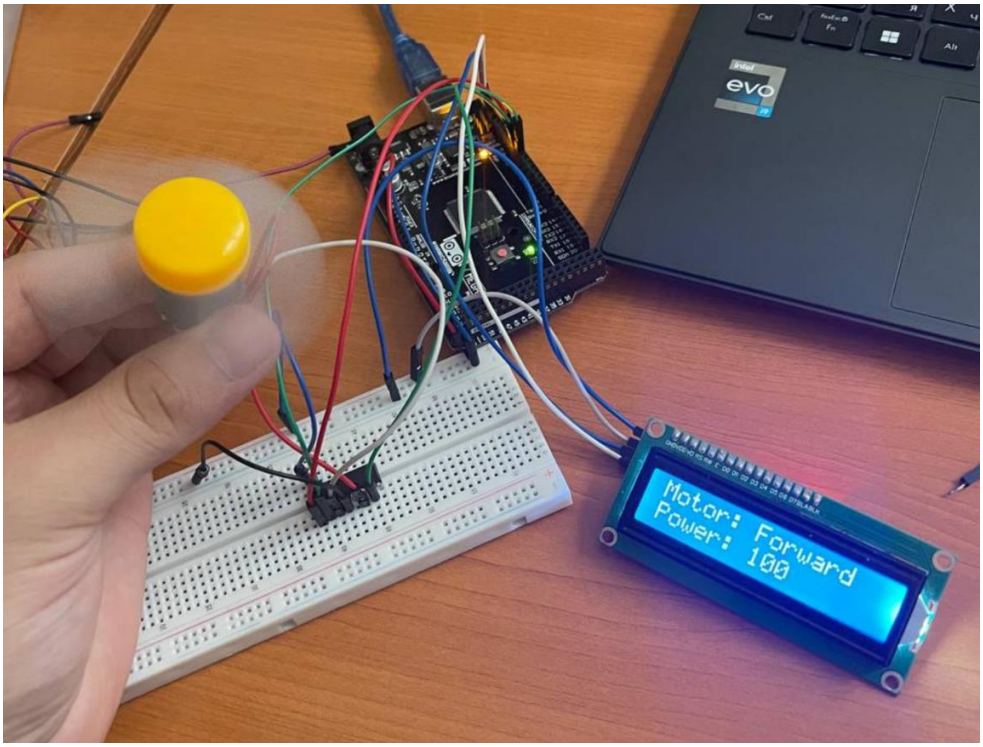
```
    digitalWrite(pinMotorA1, LOW);
    digitalWrite(pinMotorA2, LOW);
    digitalWrite(pinRelay, HIGH);
    analogWrite(pinENA, 0);

    lcd.setCursor(0, 0);
    lcd.print("Sistem pornit");
}

void loop() {
    if (Serial.available() > 0) {
        String comanda = Serial.readStringUntil('\n');
        comanda.trim();
        procesareComenzi(comanda);
    }
}

void procesareComenzi(String comanda) {
    if (comanda == "MOTOR_FWD") {
        controlMotor("Inainte", 1, 255);
    }
    else if (comanda == "MOTOR_REV") {
        controlMotor("Invers", -1, 255);
    }
    else if (comanda == "MOTOR_STOP") {
        controlMotor("Oprit", 0, 0);
```

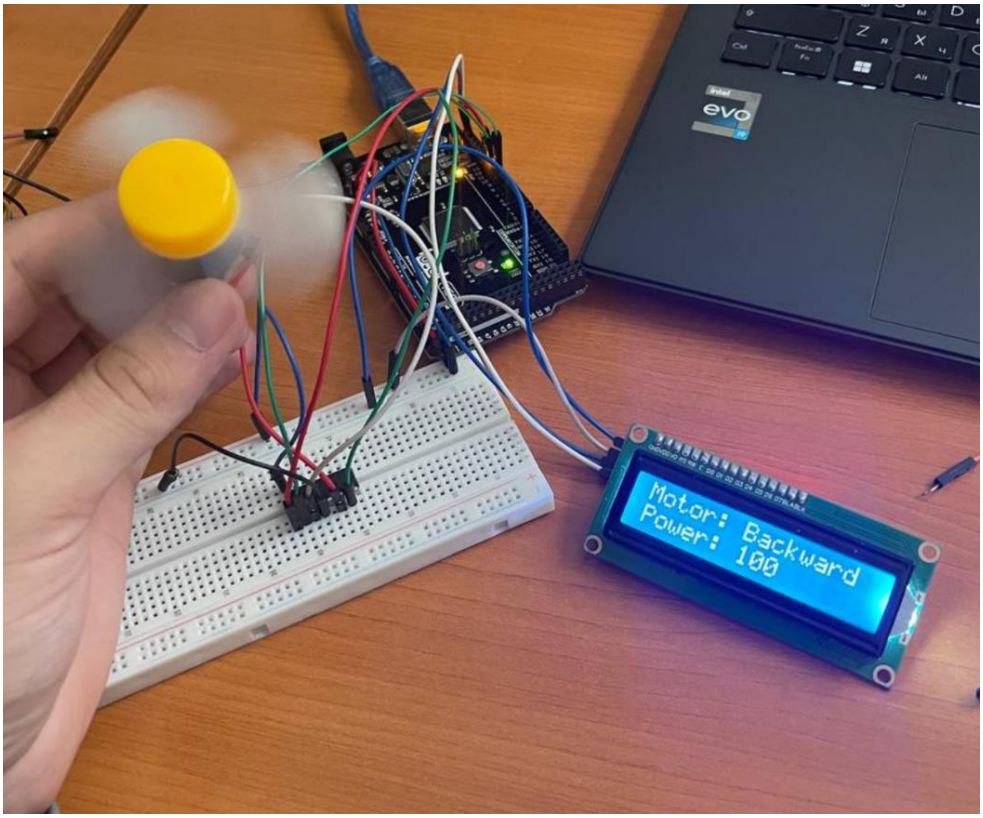
```
    }
    else if (comanda == "LIGHT_ON") {
        controlBec(true, "Bec: ON");
    }
    else if (comanda == "LIGHT_OFF") {
        controlBec(false, "Bec: OFF");
    }
    else {
        Serial.println("Comanda necunoscuta.");
    }
}

void controlMotor(String direction, int directie, int putere) {
    if (directie == 1) {
        digitalWrite(pinMotorA1, HIGH);
        digitalWrite(pinMotorA2, LOW);
        analogWrite(pinENA, putere);
    }
    else if (directie == -1) {
        digitalWrite(pinMotorA1, LOW);
        digitalWrite(pinMotorA2, HIGH);
        analogWrite(pinENA, putere);
    }
    else {
        digitalWrite(pinMotorA1, LOW);
        digitalWrite(pinMotorA2, LOW);
        analogWrite(pinENA, 0);
```

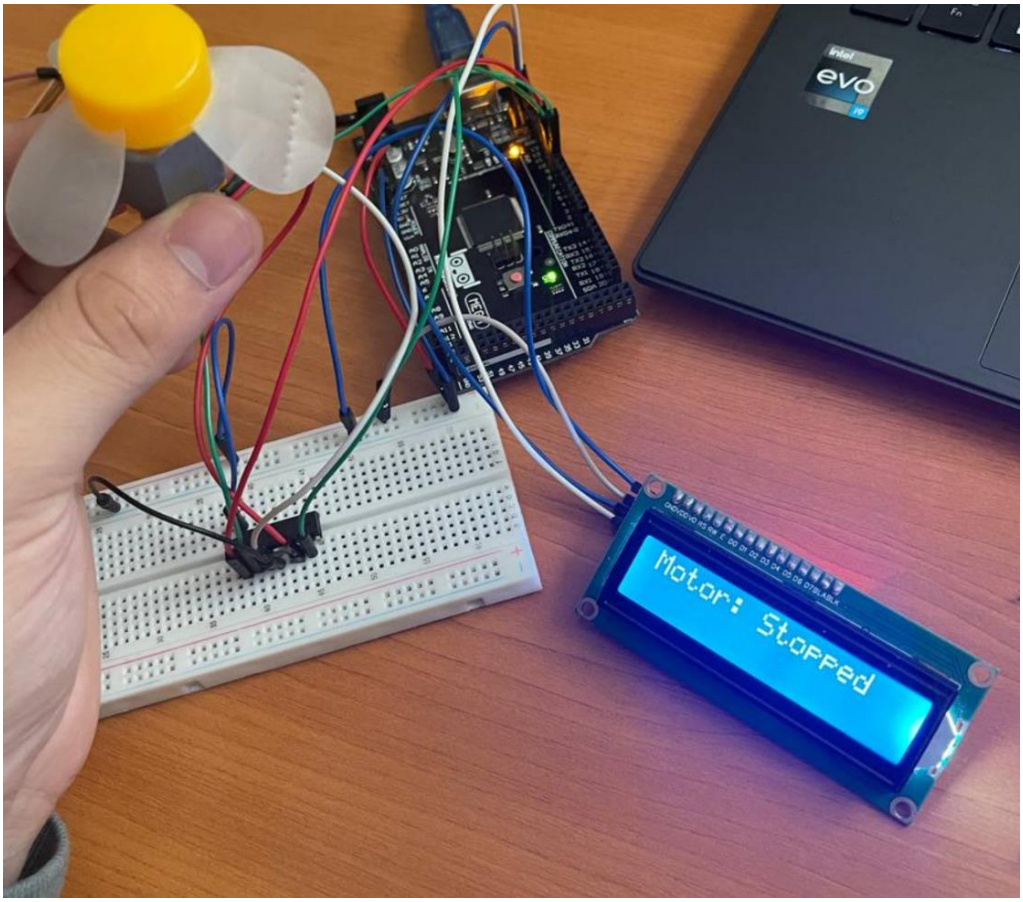
```
    }

    updateLcdAndSerial("Motor: " + direction, "Motor " + direction + ".");
}

void controlBec(bool stare, String lcdText) {
    digitalWrite(pinRelay, stare ? LOW : HIGH);
    updateLcdAndSerial(lcdText, stare ? "Bec aprins." : "Bec stins.");
}

void updateLcdAndSerial(String lcdMessage, String serialMessage) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(lcdMessage);
    Serial.println(serialMessage);
}
```

4.2


Start
Setup
Loop
Command
Negative
Positive
SET_POWER
SET_POWER
MOTOR_REV
MOTOR_FWD
Show LCD


Motor: Forward
Power: 100






```
#include <Adafruit_LiquidCrystal.h>
```

```
Adafruit_LiquidCrystal lcd(0);
const int pinIN1 = 8;
const int pinIN2 = 9;
const int pinENA = 10;

void setup() {
    pinMode(pinIN1, OUTPUT);
    pinMode(pinIN2, OUTPUT);
    pinMode(pinENA, OUTPUT);
    digitalWrite(pinENA, HIGH);
    lcd.begin(16, 2);
    lcd.print("Motor: Stopped");
    Serial.begin(9600);
}

void loop() {
    if (Serial.available() > 0) {
        String command = Serial.readStringUntil('\n');
        command.trim();

        if (command.startsWith("SET_POWER")) {
            int power = command.substring(10).toInt();
            setMotorPower(power);
        }
    }
}

void setMotorPower(int power) {
    if (power > 100) power = 100;
    if (power < -100) power = -100;
```

```
    if (power >= 0) {
        digitalWrite(pinIN1, HIGH);
        digitalWrite(pinIN2, LOW);
        analogWrite(pinENA, map(power, 0, 100, 0, 255));
        lcd.clear();
        lcd.print("Motor: Forward");
        lcd.setCursor(0, 1);
        lcd.print("Power: ");
        lcd.print(power);
    } else {
        digitalWrite(pinIN1, LOW);
        digitalWrite(pinIN2, HIGH);
        analogWrite(pinENA, map(-power, 0, 100, 0, 255));
        lcd.clear();
        lcd.print("Motor: Backward");
        lcd.setCursor(0, 1);
        lcd.print("Power: ");
        lcd.print(-power);
    }}
```

Abstractizare 4.2

```
#include <Adafruit_LiquidCrystal.h>

Adafruit_LiquidCrystal lcd(0);

const int pinIN1 = 8;

const int pinIN2 = 9;

const int pinENA = 10;
```

```
void setup() {
    pinMode(pinIN1, OUTPUT);
    pinMode(pinIN2, OUTPUT);
    pinMode(pinENA, OUTPUT);
    digitalWrite(pinENA, HIGH);
    lcd.begin(16, 2);
    lcd.print("Motor: Stopped");
    Serial.begin(9600);
}

void loop() {
    if (Serial.available() > 0) {
        String command = Serial.readStringUntil('\n');
        command.trim();

        if (command.startsWith("SET_POWER")) {
            int power = command.substring(10).toInt();
            String direction = (power >= 0) ? "Forward" : "Backward";
            String lcdText = "Motor: " + direction;
            setMotorState(direction, abs(power), lcdText);
        }
    }
}
```

```
void setMotorState(String direction, int power, String lcdText) {
    if (power > 100) power = 100;
    if (direction == "Forward") {
        digitalWrite(pinIN1, HIGH);
        digitalWrite(pinIN2, LOW);
        analogWrite(pinENA, map(power, 0, 100, 0, 255));
    } else if (direction == "Backward") {
        digitalWrite(pinIN1, LOW);
        digitalWrite(pinIN2, HIGH);
        analogWrite(pinENA, map(power, 0, 100, 0, 255));
    }

    lcd.clear();
    lcd.print(lcdText);
    lcd.setCursor(0, 1);
    lcd.print("Power: ");
    lcd.print(power);
}
```