Ministerul Educaţiei, Culturii și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Departamentul Ingineria Software și Automatică

RAPORT

Lucrare de laborator Nr.4.2
Disciplina: IoT
Tema: Actuatori - DC Motor

A efectuat: st.gr.TI-212,
Vlasitchi Stefan

A verificat : asist. univ.
Lupan Cristian

Chișinău 2024

Definirea problemei:

Să 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.

Dispozitivele de acționare vor fi următoarele:

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

Obiective:

1. Implementarea comunicării seriale pentru recepționarea comenzilor;
2. Controlul motorului prin comenzi;
3. Afișarea stării pe LCD.

INTRODUCERE

Actuatorii reprezintă componente esențiale în sisteme automatizate, având rolul de a transforma un semnal de control (electronic sau mecanic) într-o acțiune fizică. În esență, un actuator este un dispozitiv care primește o comandă de la un sistem de control (precum un microcontroler) și o convertește într-o mișcare sau într-o altă formă de energie utilă, cum ar fi activarea unui releu pentru a aprinde un bec sau pentru a mișca un motor electric.

Într-un sistem automatizat, actuatoarele permit interacțiunea cu mediul fizic, permițând funcții precum mișcarea mecanică, controlul poziției, aprinderea sau stingerea unui dispozitiv și multe altele. Controlul precis al actuatorilor este important în aplicații variate, de la automatizarea industrială și robotică până la dispozitive inteligente de uz casnic.

Exemple de actuatori:

- Releurile – permit controlul dispozitivelor de putere (ex. becuri, motoare) printr-un semnal de control de joasă tensiune. Un releu poate fi acționat pentru a închide sau deschide un circuit electric.
- Motoarele electrice – utilizate pentru mișcarea componentelor, fiind acționate în funcție de intensitatea și direcția curentului.

EFECTUAREA LUCRĂRII

Materiale necesare:

- **Microcontroler** (Arduino Mega) folosit ca microcontroler principal pentru controlul releului și afișarea stării pe LCD. Acesta va primi comenzi prin interfața serială și va controla becul;
- **Surse de alimentare și fire de conectare** asigură alimentarea circuitului și conectarea între componente;
- **Modul de alimentare (VCC)** – în acest caz, simbolul de alimentare „VCC” este un nod virtual ce asigură o conexiune constantă de 5V.
- **Motor** Este controlat de motor driver
- **L298 motor driver :** Actioneaza asupra motorului

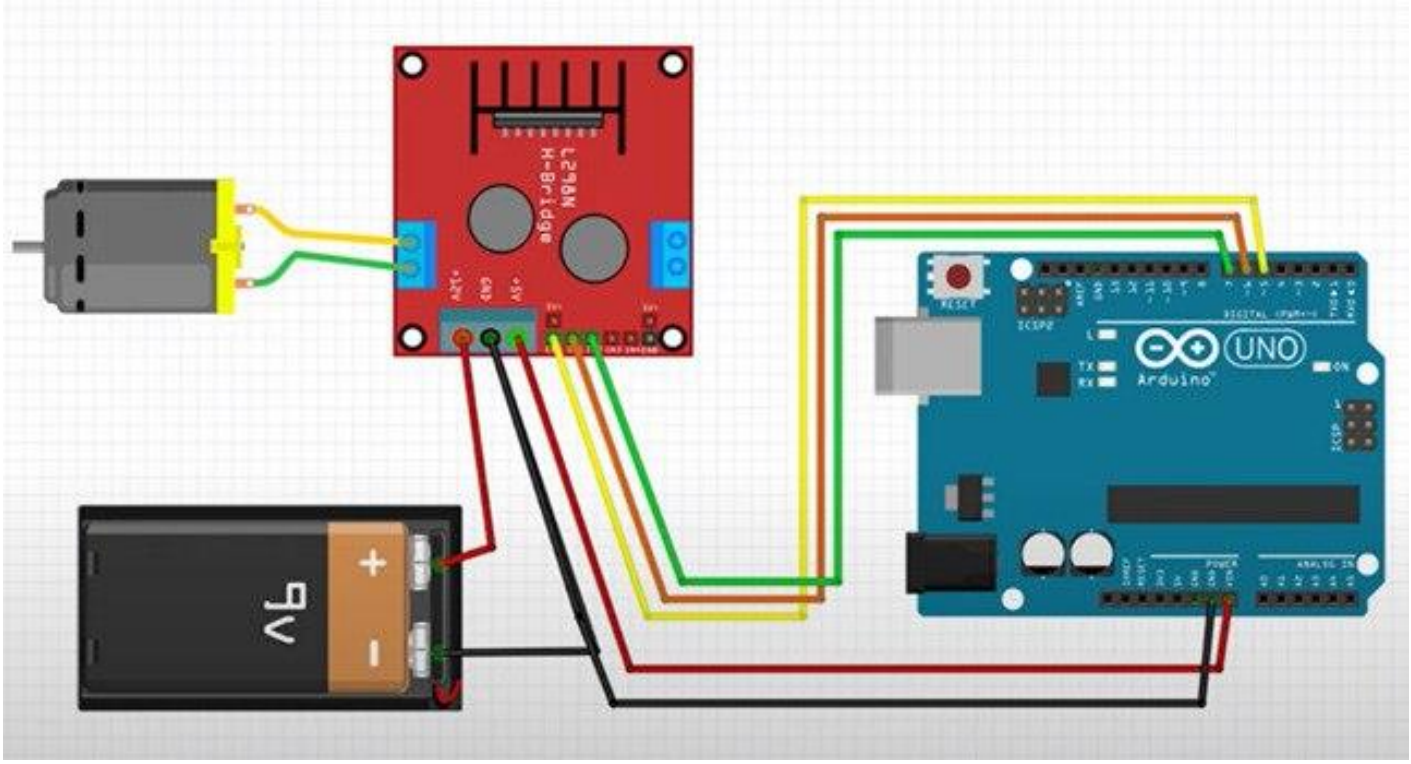

Figura 1 – Ansamblarea circuitului virtual

Modul de lucru

Utilizând această aplicație în bază de MCU, putem gestiona starea motorului cu ajutorul motor driver prin comenzile in Serial

```
void loop() {
  int16_t speed = 0;
```

```
  char buffer[20];

  printf("Enter SPEED: ");
  speed = getSpeed();

  dtostrf(speed, 5, 2, buffer);
  printf("%s\n", buffer);

  setSpeed(speed);
}

int16_t getSpeed() {
  bool valid = false;
  int16_t speed;

  while (!valid) {
    speed = getNumericValue();
    if (speed < -255 || speed > 255) {
      valid = false;
      printf("Value out of range. Please enter a value between -255 and 255.\n");
    } else {
      valid = true;
    }
  }
  return speed;
}

void setSpeed(int16_t speed) {
  if (speed < 0) {
    spinLeft(abs(speed));
  }
  else if (speed > 0) {
    spinRight(abs(speed));
  }
  else {
    stopSpinning();
  }
}
```

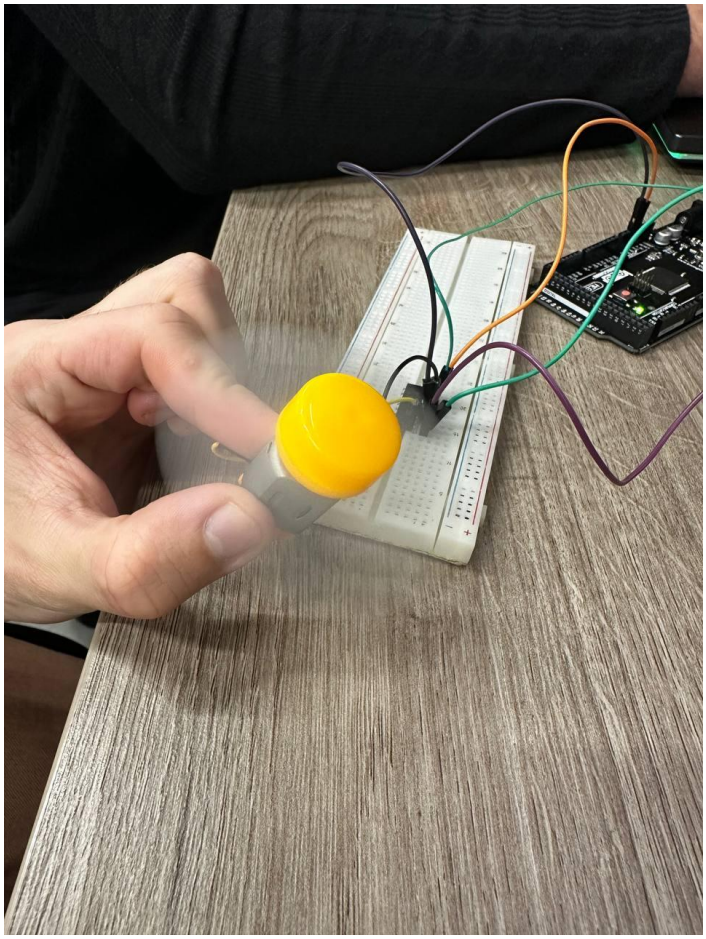

Figura 2 – Simularea aplicației în circuit fizic

CONCLUZIE

Lucrarea de laborator a demonstrat aplicarea practică a controlului unui actuator simplu prin intermediul unui microcontroler Arduino Mega. Prin utilizarea interfeței seriale, sistemul primește comenzi binare (-255 - 255) pentru a controla un motor si directia de miscare a acestuia , simulând funcționalitatea unui sistem IoT de automatizare.

Anexa 1

Fișierul stdinout.h

```
#ifndef _STDINOUT_H
#define _STDINOUT_H

// no need to make an instance of this yourself
class initializeSTDINOUT
{
        static size_t initnum;
public:
        // Constructor
        initializeSTDINOUT();
};

// Call the constructor in each compiled file this header is included in
// static means the names won't collide
static initializeSTDINOUT initializeSTDINOUT_obj;

#endif
```

Fișierul stdinout.cpp

```
#if ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif
#include <stdio.h>
#include "stdinout.h"

// Function that printf and related will use to print
static int serial_putchar(char c, FILE *f)
{
  if (c == '\n') {
    serial_putchar('\r', f);
  }

  return Serial.write(c) == 1 ? 0 : 1;
}
// Function that scanf and related will use to read
static int serial_getchar(FILE *)
{
  // Wait until character is avilable
  while (Serial.available() <= 0) { ; }

  return Serial.read();
}

static FILE serial_stdinout;

static void setup_stdin_stdout()
```

```
{
  // Set up stdout and stdin
  fdev_setup_stream(&serial_stdinout, serial_putchar, serial_getchar, _FDEV_SETUP_RW);
  stdout = &serial_stdinout;
  stdin  = &serial_stdinout;
  stderr = &serial_stdinout;
}

// Initialize the static variable to 0
size_t initializeSTDINOUT::initnum = 0;

// Constructor that calls the function to set up stdin and stdout
initializeSTDINOUT::initializeSTDINOUT()
{
  if (initnum++ == 0) {
    setup_stdin_stdout();
  }
}
```

Anexa 2

main.cpp

```
#include "DisplayController.h"
#include "serial_utils.h"
#include "L298.h"

int16_t getSpeed();
void setSpeed(int16_t speed);


void setup() {
  initializeSerial(115200);
  initializeLcd();
  initializeDcMotor();
}

void loop() {
  int16_t speed = 0;
  char buffer[20];

  printf("Enter SPEED: ");
  speed = getSpeed();

  dtostrf(speed, 5, 2, buffer);
  printf("%s\n", buffer);

  setSpeed(speed);
}

int16_t getSpeed() {
  bool valid = false;
  int16_t speed;
```

```
  while (!valid) {
    speed = getNumericValue();
    if (speed < -255 || speed > 255) {
      valid = false;
      printf("Value out of range. Please enter a value between -255 and 255.\n");
    } else {
      valid = true;
    }
  }
  return speed;
}

void setSpeed(int16_t speed) {
  if (speed < 0) {
    spinLeft(abs(speed));
  }
  else if (speed > 0) {
    spinRight(abs(speed));
  }
  else {
    stopSpinning();
  }
}
```

SerialUtilis.cpp

```
#include "serial_utils.h"

void initializeSerial(int64_t baud){
  Serial.begin(baud);
  printf("Serial is rady!\n");
}

int16_t getNumericValue() {
  char input[10];  // Șir pentru a stoca inputul utilizatorului
  int16_t speed = 0;
  bool valid = false;

  while (!valid) {
    // Citim inputul ca un șir de caractere
    scanf("%s", input);

    // Verificăm dacă inputul este numeric și poate începe cu '-' sau '+'
    valid = true;  // Presupunem că este valid până la dovada contrară
    int i = 0;
    if (input[0] == '-' || input[0] == '+') {
      i = 1;  // Dacă începe cu '-' sau '+', începem verificarea de la următorul caracter
    }
```

```
    // Verificăm restul caracterelor să fie doar cifre
    for (; input[i] != '\0'; i++) {
      if (input[i] < '0' || input[i] > '9') {
        valid = false;
        break;
      }
    }

    // Dacă inputul este valid, convertim la număr și verificăm intervalul
    if (valid) {
      speed = atoi(input);  // Convertim șirul la număr întreg

    } else {
      printf("Invalid input. Please enter a numeric value.\n");
    }
  }
  return speed;  // Returnăm valoarea introdusă
}
```

SerialUtilis.h

```
#ifndef SERIAL_UTILS_H
#define SERIAL_UTILS_H

#include <Arduino.h>

void initializeSerial(int64_t baud);
int16_t getNumericValue();

#endif // SERIAL_UTILS_H
```

L2980.cpp

```
#include "L298.h"
#include "lcd_utils.h"

void initializeDcMotor(){
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENA, OUTPUT);

  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
```

```
}

void stopSpinning(){
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, 0);
  resetLcd(0, "--");
}

void spinRight(uint8_t speed){
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, speed);
  resetLcd(speed, "->");
}

void spinLeft(uint8_t speed){
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  analogWrite(ENA, speed);
  resetLcd(speed, "<-");
}
```

L2980.h

```
#ifndef DC_MOTOR_UTILS_H
#define DC_MOTOR_UTILS_H

#include <Arduino.h>

//MotorPins
#define IN1 5
#define IN2 6
#define ENA 7 // for speed control

void initializeDcMotor();
void stopSpinning();
void spinRight(uint8_t);
void spinLeft(uint8_t);
```

```
#endif // LCD_UTILS_H
```