TECHNICAL UNIVERSITY OF MOLDOVA

FACULTY OF COMPUTERS INFORMATICS AND MICROELECTRONICS

DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATION

Laboratory work 4.2

Subject: Actuators – DC Motors

Done by: **Konjevic Alexandra,**
st. gr. FAF-213

Verified by: **Moraru Dumitru,**
university lecturer

CHIŞINĂU, 2024

THE TASKS OF THE LABORATORY WORK

An application based on MCU will be developed to control actuation devices with commands received from the serial interface and report to the LCD. The actuation devices will be as follows:

A DC motor with commands to set the motor power between (-100% .. 100%), i.e., forward and backward, and speed through the L298 driver, also, an electric bulb using a relay. Peripheral control drivers will be implemented at abstraction levels.

- Transmit the commands through serial monitor
- Display the actuator state on the serial interface
- Implement on layers the control of the DC Motor

PROGRESS OF THE WORK

1.1 Description

Before proceeding to the laboratory work, we must first clarify some of the theoretical concepts needed for this work. First of all, let's define what is an actuator – in the strict meaning, an actuator is a device that can convert energy in movement. But if we chose not to be strict, we can call an actuator any device that converts electrical energy in an output, for example, a led, a display, a loudspeaker etc.

The first task of the laboratory work was to implement a circuit with a light bulb, that we would control via serial terminal, using commands "led on" and "led off". Considering that the electric bulb needs more voltage to be turned on, we need to use a relay. The Arduino board cannot handle directly high power, and this is where a relay can help. A relay is composed by an electromagnet, that moves a tiny metallic plank (the COM terminal), between two different positions (NC terminal and NO terminal). We can decide in which position the COM terminal will be connected to through activating/deactivating the electromagnet, by connecting a low power signal in the electromagnet control terminal. For NC (normally closed) configuration, we need to write a LOW signal to ACTIVATE the relay; for the NO (normally open) configuration, the logic will be inverted – we will use a HIGH signal to ACTIVATE the relay.

So, here are some reasons to use a relay when trying to connect a light bulb in the circuit:

- Isolation: Relays provide electrical isolation between the high-voltage circuit (the electric bulb) and the low-voltage control circuit (the Arduino). This is crucial for safety, as it prevents high-voltage fluctuations or surges from damaging the sensitive electronic components of the Arduino.
- Current Limitation: Relays are designed to handle higher currents than most microcontroller output pins. This means they can safely control devices with higher power requirements, such as electric bulbs, without risking damage to the Arduino.
- Voltage Compatibility: Some bulbs may operate at voltages that are not directly compatible with the Arduino's output pins. Relays provide a way to control devices that require higher voltages than the Arduino can directly provide.
- Switching High-Power Loads: Relays are well-suited for switching high-power loads like electric bulbs. They can handle the electrical load directly, while the Arduino only needs to provide a small control signal to activate the relay.

Next task of the laboratory work was related to the DC motor. A DC (Direct Current) motor is considered the simplest motor, which works based on the principle of electromagnetic induction. It means that the rotation of the motor depends on the force generated by the electromagnetic field. It converts electrical energy into mechanical energy. Such motors can be powered from the direct current.

The DC motor consists of a stator, rotor, armature, and a commutator. The commutator comes with brushes. There are two stationary magnets in the stator that are responsible for

producing the magnetic field. The armature present in the DC motor carries the alternating current. Electrical energy is converted into mechanical energy in the form of torque by the armature. It further transfers this mechanical energy via shaft. The commutator is defined as the electrical switch. It can also reverse the direction of the current between the external circuit and the motor. The brushes act as an intermediate between the external power supply and the rotating coils. The iron core at the center is wrapped with insulated wires concentrating on the magnetic field when current passes through the wires. The windings of insulated wire have many turns around the core of the motor. The wire ends are connected to the commutator. The commutator further energizes the armature coils and connects the power supply and the rotating coils through brushes.

Let's discuss the need to use the L293D H-Bridge motor driver with the DC motor. L293 is defined as the motor driver IC that permits the DC motor to drive in any direction. It can also simultaneously control two DC motors. It is a 16-pin Integrated Circuit (IC). It receives signals from the microprocessor present on the Arduino board and transmits this signal to the motor. It has two VCC or voltage pins, where one pin draws current for its working and another is used to provide voltage to the DC motor. The motor usually requires high current for its operation. We can use the microcontroller present on the Arduino, but high current might damage the microcontroller. To overcome this, the motor driver is used. L293D is one of the most popular motor drivers used to drive the DC motors. It can run DC motors up to 1 Ampere current load.

The program that I wrote is designed to control a hobby gearmotor and an electric bulb via the serial monitor. It begins with defining pin assignments for the gearmotor control and the relay for the bulb. It also includes constants for motor speed control and initializes the LCD display. The program employs the "LiquidCrystal_I2C" library to facilitate communication with the LCD.

There are three main classes: LCDController, L293DController, and MotorController. The LCDController manages the initialization and display functions for the LCD. It shows an initial message prompting the user to enter the speed and updates the display with the current speed set for the motor. The L293DController initializes and controls the L293D motor driver. It sets the speed and direction of the motor rotation. The MotorController initializes the L293D motor driver controller and provides a method to change the speed of the motor. In the setup function, the program initializes the pins, motor controller, serial communication, and LCD controller. It also sets up the standard output stream to redirect printf() to the serial monitor.

In the loop function, the program continuously reads input from the serial monitor. If the input is "led on" or "led off", it turns the electric bulb on or off, respectively, via the relay. Otherwise, it interprets the input as a speed command for the motor. If the speed is within the valid range, it adjusts the motor speed and updates the LCD display accordingly. If the speed is out of range, it displays an error message on the LCD.

Overall, this program provides a user-friendly interface for controlling a gearmotor and an electric bulb using the Arduino board and serial communication.

1.2 Block Diagram

First of all, below are presented three diagrams for three methods that are contained in the LCDController class: the first one is for displaying the initial message; the second one, for displaying the speed introduced by the user in the serial monitor, and the last one – for displaying an error message when the user introduces a value that is out of range of the -100 to 100.

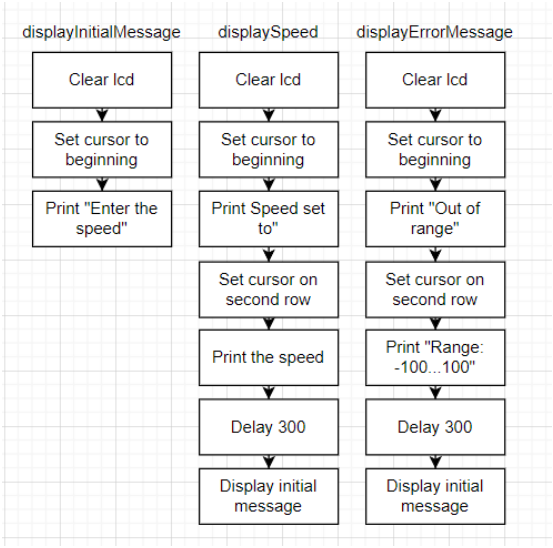

Figure 1. LCDController class

Next, we have the diagram of the methods located in the second class: L293DController. Here we can see two methods how calculating the speed of the motor and setting it to rotate either clockwise or counter clockwise. The rotation depends on the input from the user.

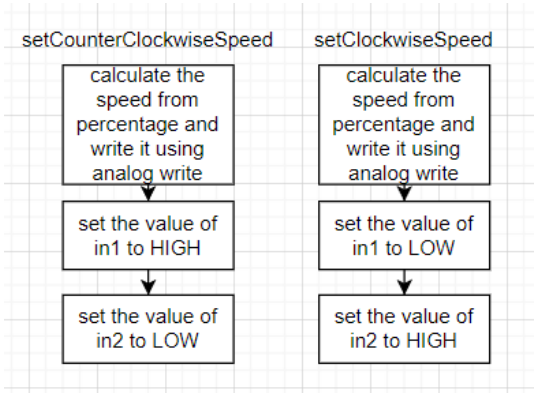

Figure 2. L293DController class

In the figure below, we can see the flow chart for the next class – MotorController, which has a method for invoking the method to rotate the motor clockwise or counter clockwise based on the input of user – if it is either a positive value or a negative one.

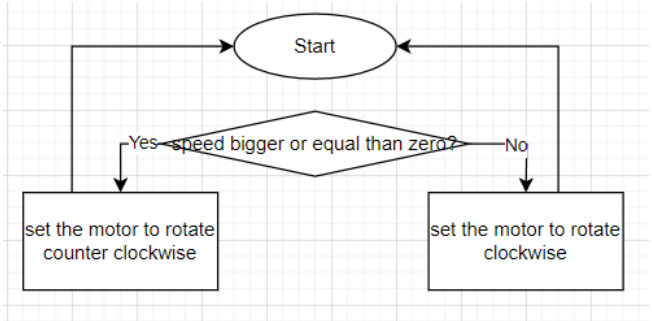

Figure 3. MotorController class

1.3 Electrical Schematic

To simulate the electrical schematic for this project, I began by placing an Arduino Uno board at the center. I then connected the needed devices for the first task: an electric bulb, an relay and a power supply. Also, we need to have there an NPN transistor. For the second task, I used a motor, a battery and a motor driver.

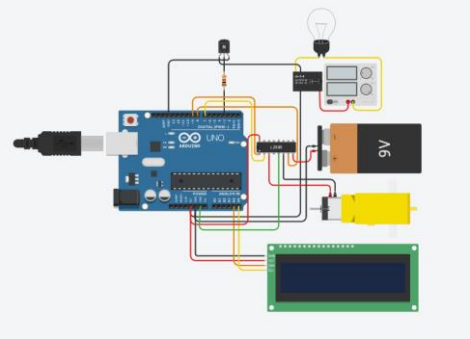

Figure 5. Electric circuit

1.4 Running Simulation

In the image below we can see all the devices and IC that are used in the circuit. For the first task, we can see an electric bulb, a power supply for the bulb, a SPDT relay and a NPN transistor (the transistor acts like a switch, controlling the power to the motor). For the second task, we can observe a hobby gearmotor, a 9V battery, for supplying the motor with power, and the L293 motor driver.

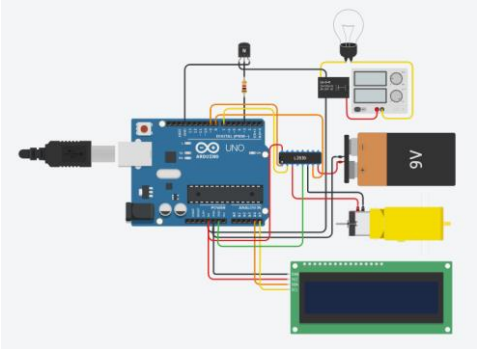

Figure 6. Electric circuit

In the following image there is presented the implementation of the first task: we can see that the electric light turned on, and the output of this action is written in the serial monitor.

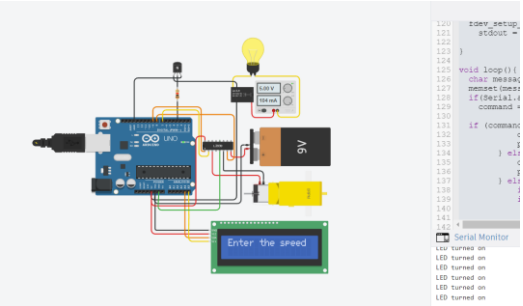

Figure 6. Simulation with the bulb turned on

Next, there is the second task: controlling the motor by giving it a specific speed. We can see that after entering the desired speed in the serial monitor, the motor started rotating with the corresponding power. The speed is also displayed on the LCD screen. First of all, there is a demonstration of giving a positive speed value.

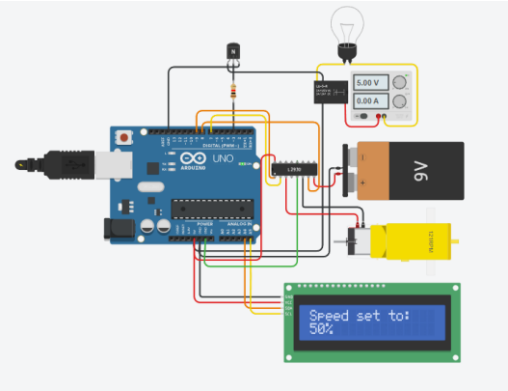

Figure 7. Simulation with the introduced positive speed 50%

After that, we can also observe what happens when the input is a negative value (in the range): the negative speed is displayed on the LCD screen, and also, the motor start rotating in the opposite direction.

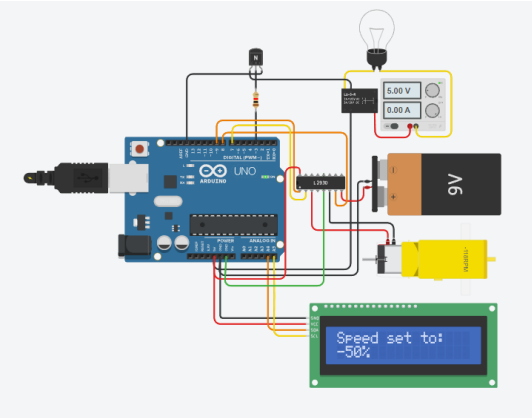

Figure 8. Simulation with the introduced negative speed -50%

Finally, below you can see what happens when we give as input a value of the speed that is not in the range (-100…100). We can see that the changes are not applied on the motor, and we can identify the error in introduced value thanks to the output on the LCD screen.

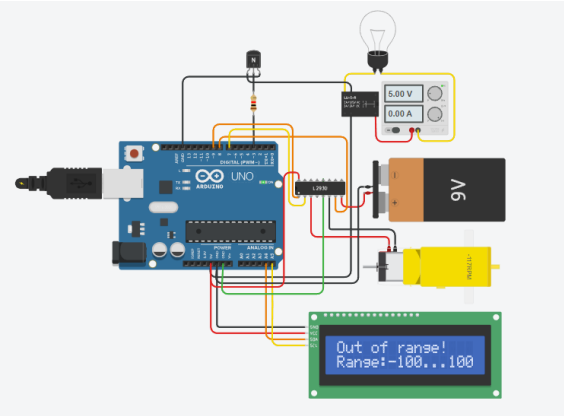

Figure 9. Simulation with the introduced out of range speed

CONCLUSION

In this laboratory work, I successfully developed an application based on an MCU to control actuation devices via commands received from a serial interface and reported the states on an LCD. The tasks involved controlling a DC motor's power and direction using the L298 driver and managing an electric bulb through a relay.

Through theoretical understanding and practical implementation, I addressed the complexities involved in interfacing with high-power devices such as electric bulbs and DC motors. Utilizing relays provided crucial isolation between high-voltage circuits and the MCU, ensuring safety and preventing damage to sensitive components.

The integration of the L293D H-Bridge motor driver facilitated bi-directional control of the DC motor, enabling smooth and efficient operation. I employed a structured programming approach, dividing functionalities into distinct classes, ensuring modularity and ease of maintenance.

Block diagrams and electrical schematics aided in visualizing the program's architecture and circuit connections, enhancing comprehension and troubleshooting. Simulations effectively demonstrated the system's functionality, showcasing seamless control of both the electric bulb and the DC motor through user inputs via the serial interface.

Overall, this laboratory work provided invaluable hands-on experience in MCU-based actuation control, reinforcing theoretical concepts and practical skills in embedded systems development. The developed application offers a user-friendly interface for controlling actuation devices, laying a solid foundation for further exploration and application in diverse engineering projects.

APPENDIX A: Source code

```
#include <LiquidCrystal_I2C.h>

int relayPin = 3;
String command;
#define enA 9
#define in1 7
#define in2 8
#define lcdAddress 32
const int defaultSpeed = 255;
const int minSpeed = -100;
const int linkSpeed = 0;
const int maxSpeed = 100;
const float percentTransform = 0.01;
const int timeDelay = 3000;
const int nullSpeed = 0;
const int lcdDimensions[2] = {16, 2};
const float normalizeFactor = 0.01;
int baudRate = 9600;

LiquidCrystal_I2C lcd(lcdAddress, 16, 2);

int serial_putchar(char c, FILE *stream)
{
    Serial.write(c);
    return 0;
}

FILE serial_stdout;

class LCDController
{
public:
    void init()
    {
        lcd.begin(lcdDimensions[0], lcdDimensions[1]);
        lcd.init();
        lcd.backlight();
        displayInitialMessage();
    }

    void displayInitialMessage()
    {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Enter the speed");
```

```
    }

    void displaySpeed(int speed)
    {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Speed set to: ");
        lcd.setCursor(0, 1);
        lcd.print(speed);
        lcd.print("%");
        delay(300);
        displayInitialMessage();
    }

    void displayErrorMessage()
    {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Out of range!");
        lcd.setCursor(0, 1);
        lcd.print("Range:-100...100");
        delay(300);
        displayInitialMessage();
    }
};

class L293DController
{
public:
    void init()
    {
        pinMode(enA, OUTPUT);
        pinMode(in1, OUTPUT);
        pinMode(in2, OUTPUT);
    }

    void setCounterClockwiseSpeed(int speed)
    {
        analogWrite(enA, (speed * normalizeFactor) * defaultSpeed);
        digitalWrite(in1, HIGH);
        digitalWrite(in2, LOW);
    }

    void setClockwiseSpeed(int speed)
    {
```

```
        analogWrite(enA, (abs(speed) * normalizeFactor) * defaultSpeed);
        digitalWrite(in1, LOW);
        digitalWrite(in2, HIGH);
    }
};

class MotorController
{
public:
    L293DController driverController;

    void init()
    {
        driverController.init();
    }

    void changeSpeed(int speed)
    {
        if (speed >= 0)
        {
            driverController.setCounterClockwiseSpeed(speed);
        }
        else if (speed < 0)
        {
            driverController.setClockwiseSpeed(speed);
        }
    }
};

MotorController motorController;
LCDController lcdController;

void setup()
{
    pinMode(relayPin, OUTPUT);
    motorController.init();
    Serial.begin(baudRate);
    lcdController.init();

    fdev_setup_stream(&serial_stdout, serial_putchar, NULL, _FDEV_SETUP_WRITE);
    stdout = &serial_stdout;
}

void loop()
{
```

```
    char message[100];
    memset(message, 0, 100);
    if (Serial.available() > 0)
    {
        command = Serial.readStringUntil('\n');
    }

    if (command.startsWith("led on"))
    {
        digitalWrite(relayPin, HIGH);
        printf("LED turned on\n");
    }
    else if (command.startsWith("led off"))
    {
        digitalWrite(relayPin, LOW);
        printf("LED turned off\n");
    }
    else
    {
        int speed = command.toInt();
        if (speed >= nullSpeed && speed <= maxSpeed)
        {
            motorController.changeSpeed(speed);
            lcdController.displaySpeed(speed);
        }
        else if (speed >= minSpeed && speed <= nullSpeed)
        {
            motorController.changeSpeed(speed);
            lcdController.displaySpeed(speed);
        }
        else
        {
            lcdController.displayErrorMessage();
        }
    }
}
```

BIBLIOGRAPHY

[1] Actuators, Arduino - Sensors and Actuators, [Quoted: 10.03.2024], access link: https://www.instructables.com/Arduino-Sensors-and-Actuators/

[2] Relays, Arduino 4 Relays Shield Basics, [Quoted: 02.02.2023], access link: https://docs.arduino.cc/tutorials/4-relays-shield/4-relay-shield-basics/

[3] DC Motors, Arduino DC motor, [Quoted: 2024], access link: https://www.javatpoint.com/arduino-dc-motor

[4] L293 motor driver, Arduino DC motor, [Quoted: 2024], access link: https://www.javatpoint.com/arduino-dc-motor

[5] Example of circuit with bulb, Arduino relay activated lamp [Quoted: 02.11.2020] access link: https://www.tinkercad.com/things/kCHoQ9WGGFY-arduino-relay-activated-lamp

[6] Example of circuit with hobby gearmotor, Obstacle Avoiding Robot Using Tinkercad [Quoted: 2024], access link: https://www.instructables.com/Obstacle-Avoiding-Robot-Using-Tinkercad-1/

[7] Formula for calculating percentage to analog output, [Quoted: 01.09.2014], access link: https://forum.arduino.cc/t/displaying-percentage-on-lcd/255276