TECHNICAL UNIVERSITY OF MOLDOVA
FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS
DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS

Report of laboratory work №4

Theme: Actuators

Fulfilled: st. gr. FAF-202 **Popa Eugeniu**

Controlled: univ. lecturer **Moraru Dumitru**

Chișinău 2023

THE TASKS OF THE LABORATORY WORK

Main task: To design an MCU-based application that will drive the actuation devices with commands received from a serial terminal with feedback messages of the order fulfillment.

The actuators will be the following:

- an electric light bulb via the relay with ON and OFF controls

- a continuous current motor with motor power setting commands between (-100% .. 100%) ie forward and reverse, and speed via the L298 driver

THE PROGRESS OF THE WORK

Task 1:

In this laboratory work, a relay is used to control the lamp. If we execute the “on” command, the board sends a signal to the transistor that lodges electricity on the relay, but this closes the circuit. The “off” command executes the opposite algorithm (Fig 1.1).

```
#include "Relay.h"

Relay::Relay() {}

void Relay::setPin(int pin)
{
  relayPin = pin;
}

void Relay::setup()
{
  pinMode(relayPin, OUTPUT);
}

void Relay::switchLight(bool relayState)
{
    bool state = relayState;
    if (state == true)
    {
        Serial.println("Turning on the lamp");
        digitalWrite(relayPin, HIGH);
    } else
    {
        Serial.println("Turning off the lamp");
        digitalWrite(relayPin, LOW);
    }
}
```

Fig 1.1 Functions from Relay.cpp

The loop function receives the messages from the serial interface and handles the relay (Fig 1.2).

```
if (receivedInput == turnOnLamp)
{
  relay.switchLight(true);
}
else if (receivedInput == turnOffLamp)
{
  relay.switchLight(false);
}
```

Fig 1.2 Relay handling from main.cpp

Task 2:

In this laboratory work, a DC motor with nominal voltage 5V is used. One can control it using a hardware driver between the Arduino board and the motor component, for example, in the given electrical scheme the L293D electrical circuit is used.

The “setRotation()” function sets the rotation direction of the motor at the beginning of the loop (Fig 2.1).

```
void Motor::setRotation()
{
    if (rotationDirection == 0)
    {
        analogWrite(motorIn1, motorSpeed);
        analogWrite(motorIn2, 0);
    }
    else if (rotationDirection == 1)
    {
        analogWrite(motorIn1, 0);
        analogWrite(motorIn2, motorSpeed);
    }
}
```

Fig 2.1 “setRotation()” function from Motor.cpp

The “increaseSpeed()” and “decreaseSpeed()” functions set the rotation speed of the motor, depending on the received message (Fig 2.2).

```
void Motor::increaseSpeed()
{
    motorSpeed = motorSpeed + 25;
    if (motorSpeed > maxSpeed)
    {
      Serial.println("Maximum speed reached");
      motorSpeed = maxSpeed;
    }
    else Serial.println("Increasing the speed");
}

void Motor::decreaseSpeed()
{
    motorSpeed = motorSpeed - 25;
    if (motorSpeed < minSpeed)
    {
      Serial.println("Minimum speed reached");
      motorSpeed = minSpeed;
    }
    else
      Serial.println("Decreasing the speed");
}
```

Fig 2.2 “increaseSpeed()” and “decreaseSpeed()” functions from Motor.cpp

The “changeRotation()” function changes the rotation direction of the motor, depending on the received message (Fig 2.3).

```
void Motor::changeRotation()
{
    if(rotationDirection == 0)
    {
      rotationDirection = 1;
    }
    else rotationDirection = 0;
    Serial.println("Reversing the rotation");
}
```

Fig 2.3 “changeRotation()” function from Motor.cpp

The loop function receives the messages from the serial interface and handles the motor (Fig 2.4).

```
else if (receivedInput == increaseSpeed)
{
  motor.increaseSpeed();
}
else if (receivedInput == decreaseSpeed)
{
  motor.decreaseSpeed();
}
else if (receivedInput == changeRotation)
{
  motor.changeRotation();
}
else Serial.println("Unknown command");
```

Fig 2.4 Motor handling from main.cpp

Electrical scheme made in Proteus

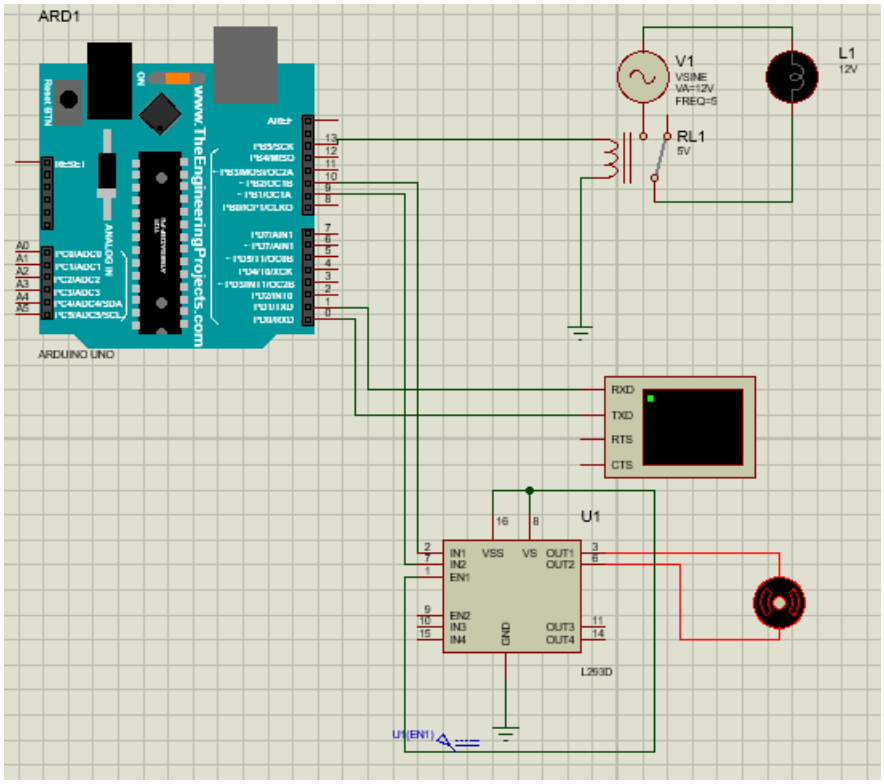

Fig 3 Electrical scheme

CONCLUSIONS

To summarize, the laboratory project I undertook has imparted me with valuable knowledge and experience on working with actuators in embedded systems, which is an essential component of the Internet of Things (IoT). I have discovered that adopting a modular approach to code development is an efficient way to create sophisticated yet intuitive systems while allowing for easy integration of new features down the line. Moreover, I have realized that programming actuators necessitates a comprehensive understanding of numerous aspects, making it a critical skill for budding engineers.