**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-191 Boico Alexandr**

**Controlled: univ. lecturer Moraru Dumitru**

**Chișinău 2022**

**The task of the laboratory work:**

Develop an MCU-based application that will control the drive devices with commands received from the serial interface and reporting them in some format.

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 bulb. If we get the command “lamp ON”, the board sends a signal to the transistor that lodges electricity on the relay, but this closes the circuit. The “lamp OFF” executes the opposite algorithm.

**Appendix 1**

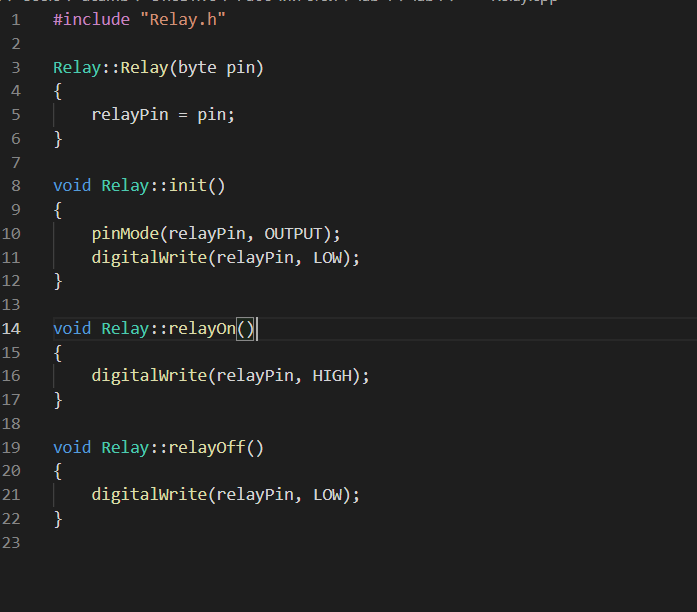


Fig 1. Writing To Relay

This part of code describes functions which will be used to turn on/off relay

**Appendix 2**

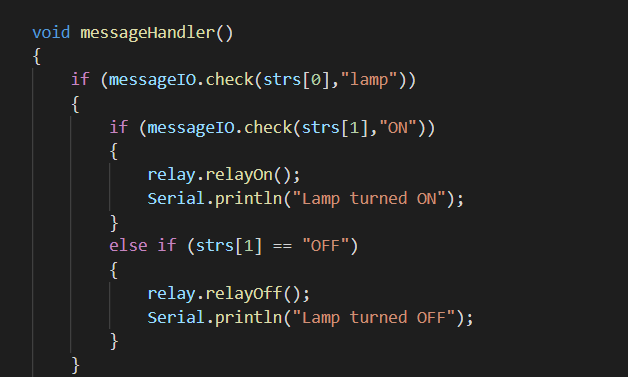


Fig 2. Handling Coming Messages (Relay)

This is the implementation of the relay handler depending on the command in the serial view.

**Task 2:**

In this laboratory work is used a DC motor with nominal voltage 12V. One can control it using a hardware driver between the Arduino board and the motor component, for example, in the given electrical scheme the L298 driver is used.

**Appendix 3**



Fig 3. Motor Logic (Part 1)

This part of code makes an abstraction on the motor movement states.

**Appendix 4**

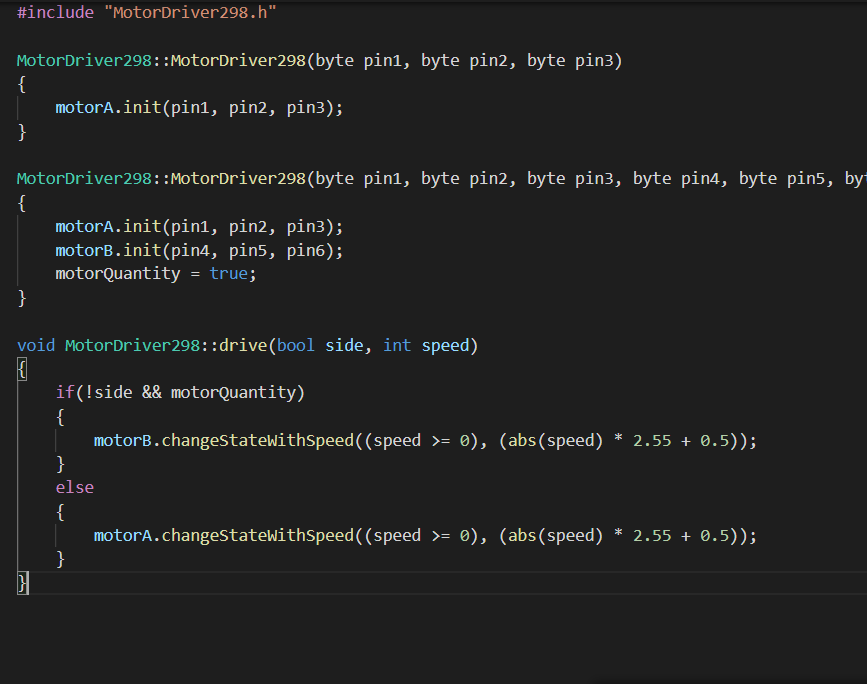


Fig 4. Motor Logic (Part Two)

This piece of code describes which motor should change the speed depending on the side parameter and how fast should it work depending on the speed parameter.

**Appendix 5**

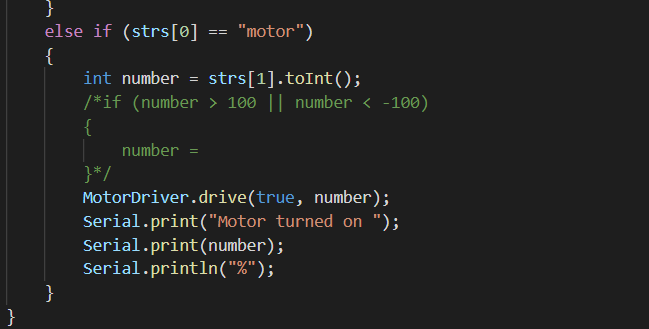


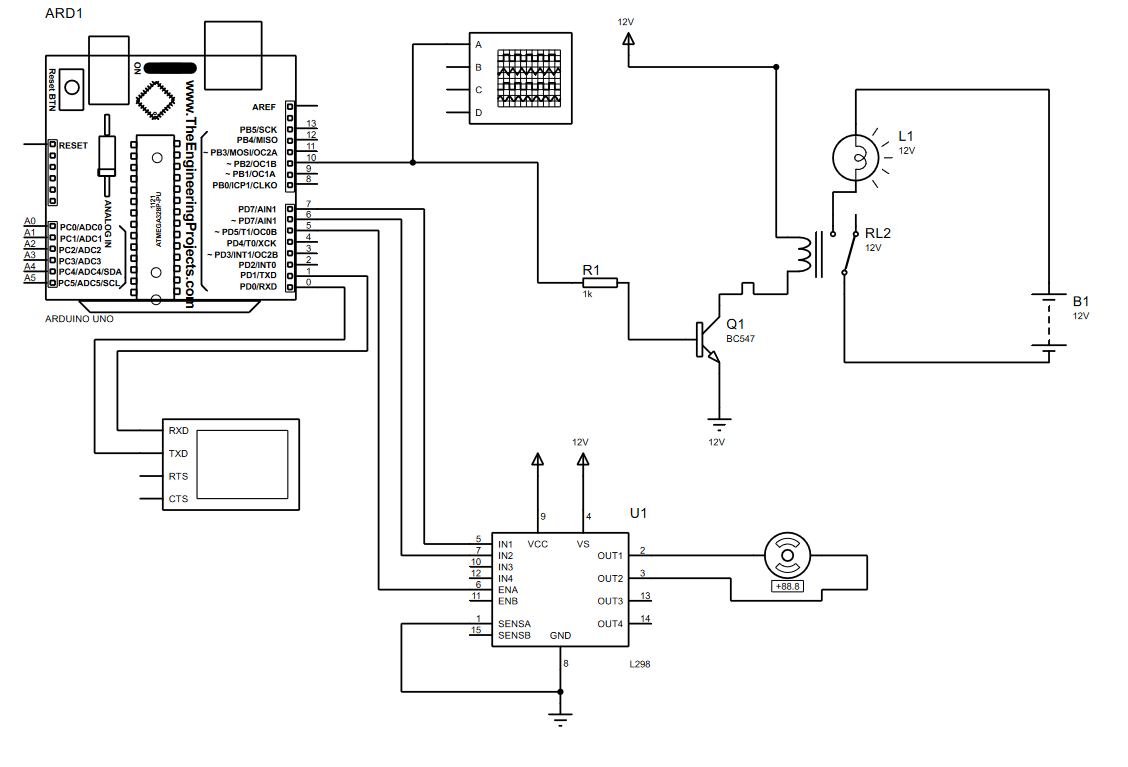
Fig 5. Handling Coming Messages (Motor)

**Conclusion**

After performing the 4th laboratory work, I have learned how to add the functionality for the DC motor, connected to the Arduino board through the L298 hardware driver and a bulb through an relay. There were launched 2 sequential tasks that detect the commands introduced from the Serial and, depending on the input, execute different functions.

**Appendix**

**Electrical schema in proteus**

****

**Code**

* main.cpp

#include <Arduino.h>

#include "Relay.h"

#include "MotorDriver298.h"

#include "MessageIO.h"

#define relayPin 10

#define motorPin1 7

#define motorPin2 6

#define motorPin3 5

const char delimiter = ' ';

Relay relay(relayPin);

MessageIO messageIO;

MotorDriver298 MotorDriver(motorPin1, motorPin2, motorPin3);

String strs[2];

String messages[10] = {

//"motor -10"

"lamp ON", "motor -10", "motor 100", "lamp OFF", "motor 0", "motor 100", "lamp fuf", "motor ajdf", "lamp ON", "lamp OFF"

};

int i = 0;

void setup()

{

Serial.begin(9600);

relay.init();

Serial.println("write lamp ON or lamp OFF to turn on(or off) the lamp");

Serial.println("write motor \*number from -100 to 100\* to turn on the engine");

}

void loop()

{

String message = messages[i]; //messageIO.readIn();

messageSpliter(message);

messageHandler();

i++;

delay(3000);

}

void messageSpliter(String str)

{

byte StringCount = 0;

while (str.length() > 0)

{

int index = str.indexOf(delimiter);

if (index == -1) // No space found

{

strs[StringCount++] = str;

break;

}

else

{

strs[StringCount++] = str.substring(0, index);

str = str.substring(index + 1);

}

}

//return strs;

}

void messageHandler()

{

if (messageIO.check(strs[0],"lamp"))

{

if (messageIO.check(strs[1],"ON"))

{

relay.relayOn();

Serial.println("Lamp turned ON");

}

else if (strs[1] == "OFF")

{

relay.relayOff();

Serial.println("Lamp turned OFF");

}

}

else if (strs[0] == "motor")

{

int number = strs[1].toInt();

/\*if (number > 100 || number < -100)

{

number =

}\*/

MotorDriver.drive(true, number);

Serial.print("Motor turned on ");

Serial.print(number);

Serial.println("%");

}

}

* MessageIO.h

#ifndef MessageIO\_h

#define MessageIO\_h

#include <stdio.h>

//#include <cstring>

#include <Arduino.h>

class MessageIO

{

private:

char charMotor[5];

public:

MessageIO();

String readIn();

bool check(String input, String pattern);

bool motor(String motor);

};

#endif

* MessageIO.cpp

#include "MessageIO.h"

MessageIO::MessageIO()

{

}

//String SerialIO::readIn()

String MessageIO::readIn()

{

String input = "";

if (Serial.available() > 0) {

input = Serial.readStringUntil('\n');

Serial.print("Received: ");

Serial.println(input);

}

return input;

}

bool MessageIO::check(String input, String pattern)

{

if(input == pattern) return true;

else return false;

}

* Relay.h

#ifndef Relay\_h

#define Relay\_h

#include <Arduino.h>

class Relay

{

private:

byte relayPin;

public:

Relay(byte pin);

void init();

void relayOn();

void relayOff();

};

#endif

* Relay.cpp

#include "Relay.h"

Relay::Relay(byte pin)

{

relayPin = pin;

}

void Relay::init()

{

pinMode(relayPin, OUTPUT);

digitalWrite(relayPin, LOW);

}

void Relay::relayOn()

{

digitalWrite(relayPin, HIGH);

}

void Relay::relayOff()

{

digitalWrite(relayPin, LOW);

}

* Motor.h

#ifndef Motor\_h

#define Motor\_h

#include <Arduino.h>

class Motor

{

private:

byte MotorPin1;

byte MotorPin2;

byte ControlPin;

public:

Motor();

void init(byte pin1, byte pin2, byte pin3);

void changeStateWithSpeed(bool direction, int speed);

// direction = true -> clockwise

// direction = false -> anti-clockwise

void changeState(bool direction);

};

#endif

* Motor.cpp

#include "Motor.h"

Motor::Motor()

{

}

void Motor::init(byte pin1, byte pin2, byte pin3)

{

MotorPin1 = pin1;

MotorPin2 = pin2;

ControlPin = pin3;

pinMode(MotorPin1, OUTPUT);

pinMode(MotorPin2, OUTPUT);

pinMode(ControlPin, OUTPUT);

}

void Motor::changeState(bool direction)

{

if (direction)

{

digitalWrite(MotorPin1, HIGH);

digitalWrite(MotorPin1, LOW);

}

else

{

digitalWrite(MotorPin1, LOW);

digitalWrite(MotorPin1, HIGH);

}

}

void Motor::changeStateWithSpeed(bool direction, int speed)

{

if (direction)

{

digitalWrite(MotorPin1, HIGH);

digitalWrite(MotorPin2, LOW);

analogWrite(ControlPin, speed);

}

else

{

digitalWrite(MotorPin1, LOW);

digitalWrite(MotorPin2, HIGH);

analogWrite(ControlPin, speed);

}

}

* MotorDriver.h

#ifndef MotorDriver298\_h

#define MotorDriver298\_h

#include <Arduino.h>

#include "Motor.h"

class MotorDriver298

{

private:

Motor motorA;

Motor motorB;

bool motorQuantity = false; //true = 2 motors, false - 1 motor

public:

MotorDriver298(byte pin1, byte pin2, byte pin3); //only motor A

MotorDriver298(byte pin1, byte pin2, byte pin3, byte pin4, byte pin5, byte pin6); //motorA and motorB

void drive(bool side, int speed);

// motor "A" -> side = true

// motor "B" -> side = false

};

#endif

* MotorDriver.cpp

#include "MotorDriver298.h"

MotorDriver298::MotorDriver298(byte pin1, byte pin2, byte pin3)

{

motorA.init(pin1, pin2, pin3);

}

MotorDriver298::MotorDriver298(byte pin1, byte pin2, byte pin3, byte pin4, byte pin5, byte pin6)

{

motorA.init(pin1, pin2, pin3);

motorB.init(pin4, pin5, pin6);

motorQuantity = true;

}

void MotorDriver298::drive(bool side, int speed)

{

if(!side && motorQuantity)

{

motorB.changeStateWithSpeed((speed >= 0), (abs(speed) \* 2.55 + 0.5));

}

else

{

motorA.changeStateWithSpeed((speed >= 0), (abs(speed) \* 2.55 + 0.5));

}

}