Ministerul Educației al Republicii Moldova

Universitatea Tehnică a Moldovei

RAPORT

la Programarea aplicațiilor incorporate și independente de platformă

Lucrare de laborator Nr. 4

Tema: : Actuators. Generate PWN Signal

A efectuat st. gr. FAF-141: Oxana Dunav

A verificat: Andrei Bragarenco

Chişinău 2016

Topic

Actuators. Generate PWN Signal

Purpose

- Controlling motor with a button
- PWN Signal

Task

Develop an application that will read data from a button press and rotate the motor either clockwise or counterclockwise.

Domain

➔ PWN signal

Pulse-width modulation (PWM), or pulse-duration modulation (PDM), is a modulation technique used to encode a message into a pulsing signal. Although this modulation technique can be used to encode information for transmission, its main use is to allow the control of the power supplied to electrical devices, especially to inertial loads such as motors. In addition, PWM is one of the two principal algorithms used in photovoltaic solar battery chargers, the other being maximum power point tracking.

➔ L293 Driver

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

➔ DC motors

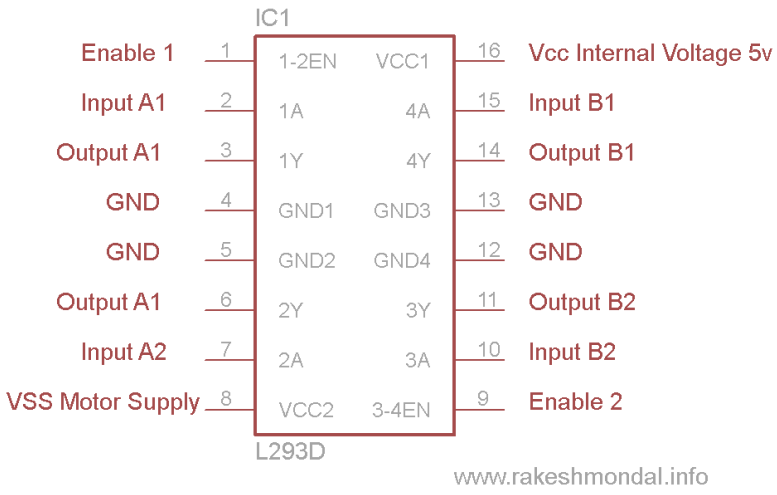
The DC motor uses a combination of schematic and programmatic modelling techniques. The schematic model is shown below, and demonstrates rather nicely how electrical circuits may be used to simulate mechanical phenomena.

Used Resources

L293 – Motor Driver

L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction.

L293D Pin Diagram



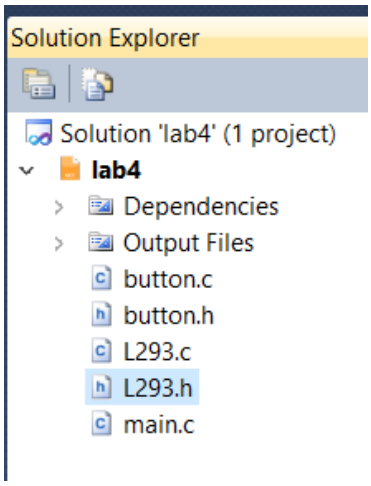
L293D Logic Table.

Lets consider a Motor connected on left side output pins (pin 3,6). For rotating the motor in clockwise direction the input pins has to be provided with Logic 1 and Logic 0.

- **Pin 2** = **Logic 1** and **Pin 7** = **Logic 0** | Clockwise Direction
- **Pin 2** = **Logic 0** and **Pin 7** = **Logic 1** | Anticlockwise Direction
- **Pin 2** = **Logic 0** and **Pin 7** = **Logic 0** | Idle [No rotation] [Hi-Impedance state]
- **Pin 2** = **Logic 1** and **Pin 7** = **Logic 1** | Idle [No rotation]

Solution

The *Project Structure* looks in this way



L293.h / L293.c

Contains declaration and implementation of L293 Driver

button.h /button.c

Contains declaration and implementation of the buttons that control the motor.

Main Program

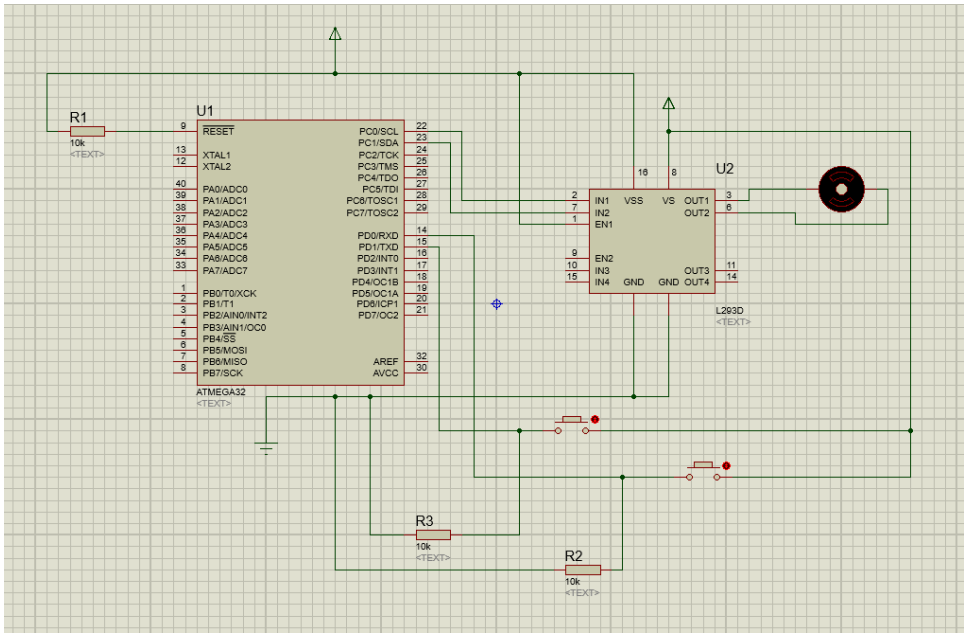
Main Program is responsible for :

- Initialization of 2 buttons , which control the motor

- Initialization of L293 Driver
- program rotates the motor in dependency of which button is pressed.

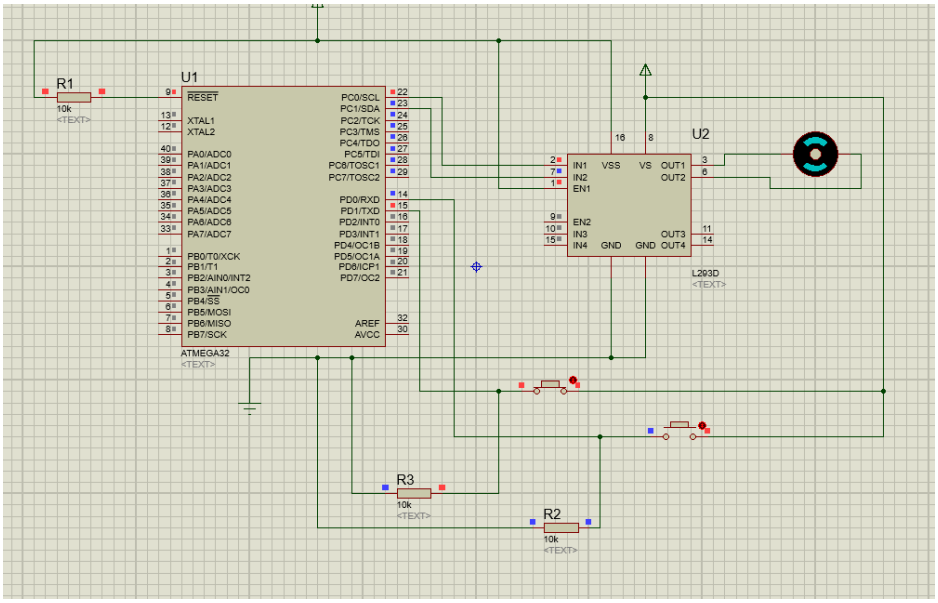
Schematics

For our laboratory work we need ATMega32 MCU, 2 Push Buttons, L293 and DC motor.



Simulation Result

The motor is rotating anticlockwise.



Conclusion

During this laboratory work I learned how to control a motor using PWN signal, how to make it move in 2 directions using L293 driver.

Being able to control motors is an important concept in mobile robotics and it gives possibilities to control the robot.

Appendix

main.c

```
#define F_CPU 8000000ul
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#include "L293.h"
#include "button.h"

int main(void)
{
	L293_init();
	initButtonOne();
	initButtonTwo();


	  while(1) {

		if(isButtonOnePressed()) {
			L293_clockwise();
		} else if(isButtonTwoPressed()) {
			L293_antiClockwise();
		} else {
			L293_stop();
		}

	  }
}
```

L293.h

```
#ifndef LAB4_SRC_L293_H_
#define LAB4_SRC_L293_H_

void L293_init();
void L293_antiClockwise();
void L293_clockwise();
void L293_stop();
void L293_free();

#endif /* LAB4_SRC_L293_H_ */
```

L293.c

```
#include <avr/io.h>
#include "L293.h"

void L293_init() {
	DDRC = 0xFF; //PORTB as Output
}

void L293_clockwise()
{
	PORTC = 0x02; //00000010
}

void L293_antiClockwise()
{
	 //Rotates Motor in Antilockwise
	 PORTC = 0x01; //00000001
}

void L293_stop()
{
	PORTC = 0x00; //00000000
}
```

button.h

```
#ifndef BUTTON_H_
#define BUTTON_H_
#include <avr/io.h>

void initButtonOne();
void initButtonTwo();
int isButtonOnePressed();
int isButtonTwoPressed();

#endif /* BUTTON_H_ */
```

button.c

```
#include "button.h"

void initButtonOne() {
	DDRD &= ~(1 << PORTD0) ;
}

void initButtonTwo() {
	DDRD &= ~(1 << PORTD1) ;
}

int isButtonOnePressed() {
```

```
	return PIND & (1<<PORTD0);
}

int isButtonTwoPressed() {
	return PIND & (1<<PORTD1);
}
```

FlowChart