Faculty of Computers, Informatics and Microelectronics
Technical University of Moldova

REPORT

Laboratory work #4
Embedded Systems

Performed by:
st. gr. FAF – 141, Postica Denis

Verified by:
Senior Lecturer, Andrei Bragarenco

Chișinău 2017

Topic

Pulse Width Modulation. Controlling motor with H-Bridge.

Objectives

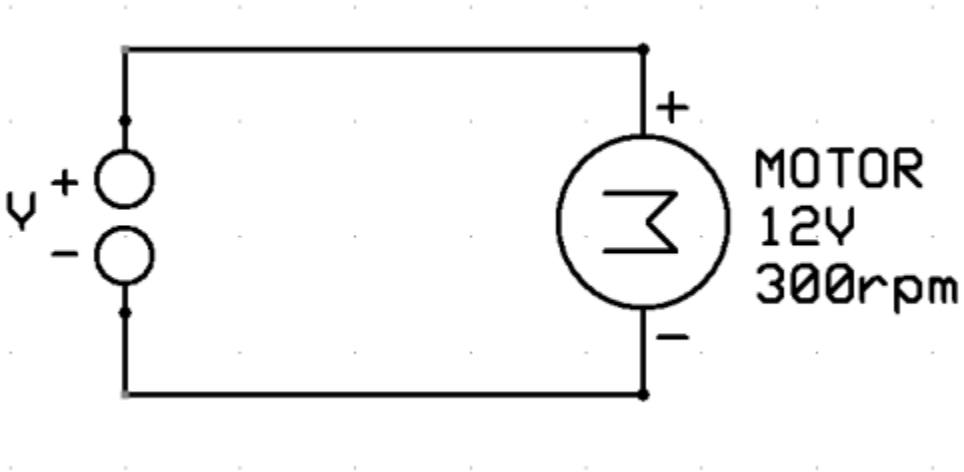
- Implement keyboard control with USART and Virtual Terminal
- Implement PWM
- Implement HBridge
- Create basic 2wd car using elements from previous point.

Task

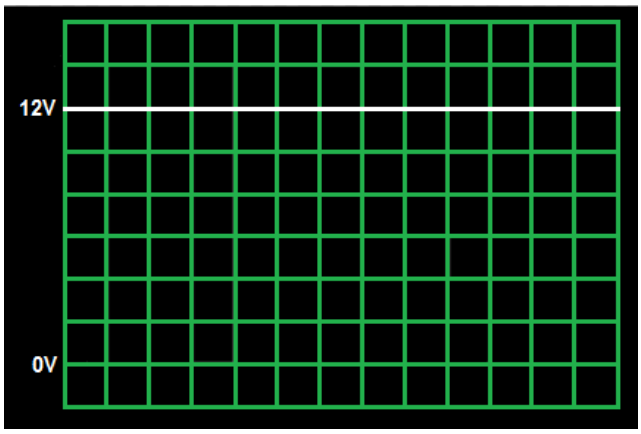
Write a C program and schematics for 2WD car using **Universal asynchronous receiver/transmitter**, **h-bridge, pulse width modulation**. Use keyboard as control for wheels. Car should be able to steer,increase velocity, decrease velocity,stop or free wheeling.

Tools and techniques

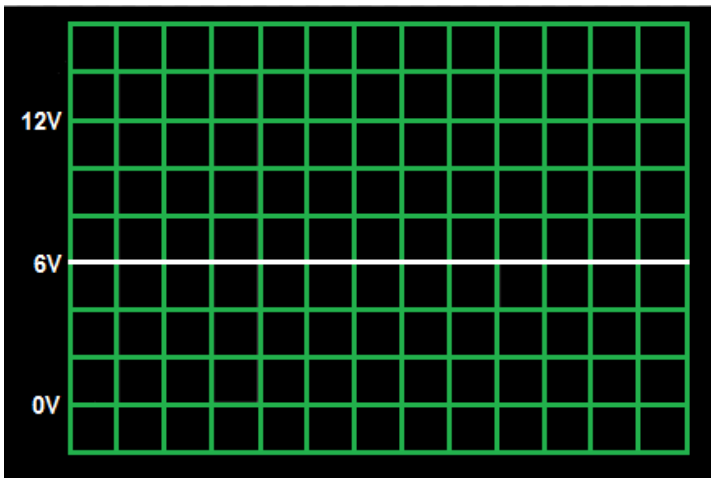
Motor speed control



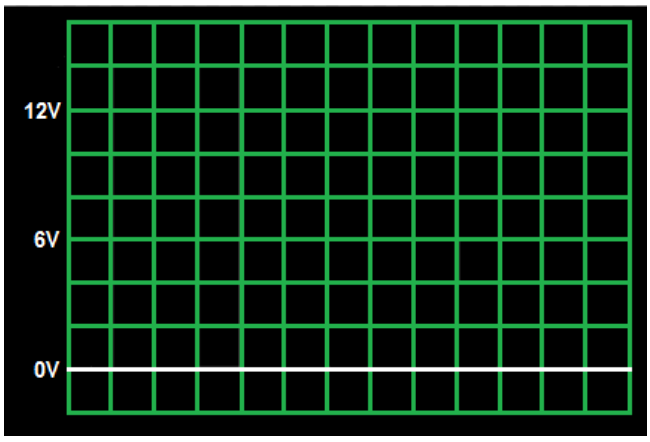
The motor is rated 12V/300rpm. This means that (assuming ideal conditions) the motor will run at 300 rpm only when 12V DC is supplied to it. If we apply 6V, the motor will run at only 150 rpm.



The motor will rotate at 300 rpm. Now let us change the voltage level as follows (6V DC).



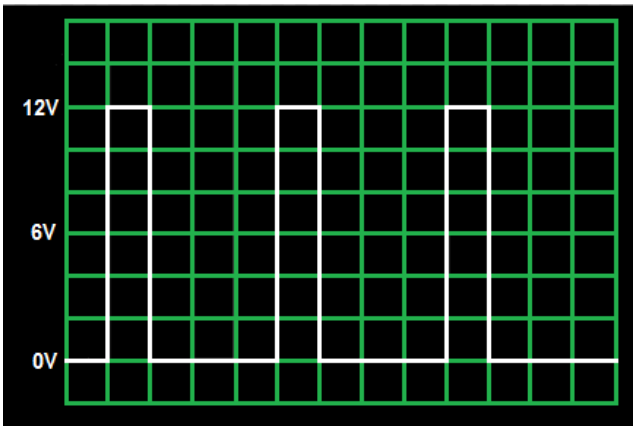
We find that the motor rotates at 150 rpm. Now let us change the voltage level once again as follows (0V DC).



This time, unsurprisingly, the motor doesn't run at all. Okay, so let's make it more interesting. What if we provide the following supply to the motor.

Solution

Each and every body in this world has some inertia. Say the motor above rotates whenever it is powered on. As soon as it is powered off, it will tend to stop. But it doesn't stop immediately, it takes some time. But before it stops completely, it is powered on again! Thus it starts to move. But even now, it takes some time to reach its full speed. But before it happens, it is powered off, and so on. Thus, the overall effect of this action is that the motor rotates continuously, but at a lower speed. In the above case, the motor will behave exactly as if a 6V DC is supplied to it, i.e. rotate at 150 rpm!



Now what happens? Yes! You guessed it right! (I hope so ;)) Since the on-time is less than the off-time, the effective speed of the motor reduce. In this case, the speed becomes 75 rpm (since off-time = 3 times on-time, i.e. speed = 300/4 = 75 rpm).
This is what we call **Pulse Width Modulation**, commonly known as **PWM**.

PWM – Pulse Width Modulation

PWM stands for Pulse Width Modulation. It is basically a modulation technique, in which the width of the carrier pulse is varied in accordance with the analog message signal. As described above, it is commonly used to control the power fed to an electrical device, whether it is a motor, an LED, speakers, etc.

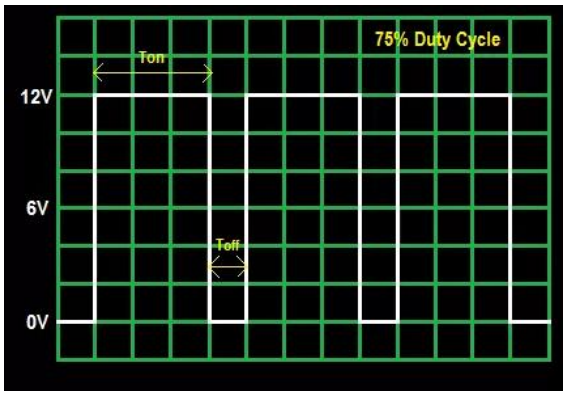
PWM Generation

The simplest way to generate a PWM signal is by comparing the a predetermined waveform with a fixed voltage level.
It has three **compare output modes** of operation:

- **Inverted Mode –** In this mode, if the waveform value is greater than the compare level, then the output is set high, or else the output is low. This is represented in figure A above.
- **Non-Inverted Mode –** In this mode, the output is high whenever the compare level is greater than the waveform level and low otherwise. This is represented in figure B above.
- **Toggle Mode –** In this mode, the output toggles whenever there is a compare match. If the output is high, it becomes low, and vice-versa.

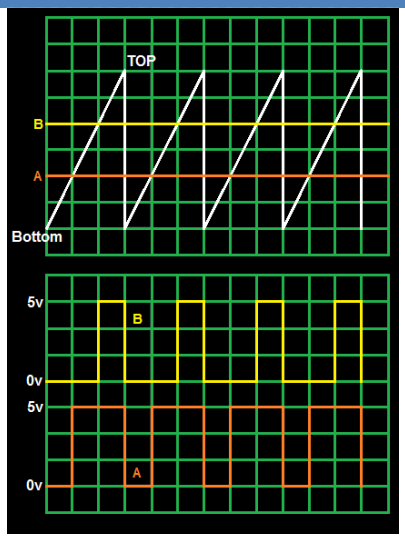
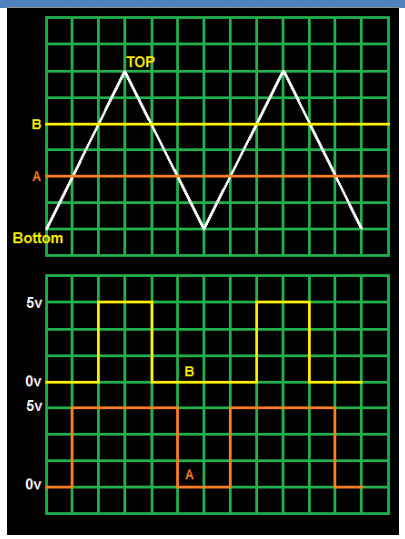
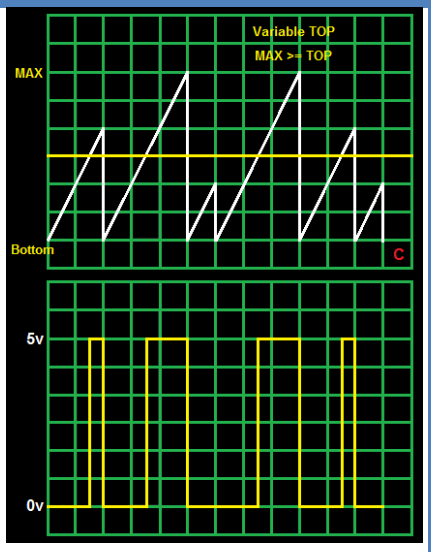
The Duty Cycle of a PWM

Waveform is given by



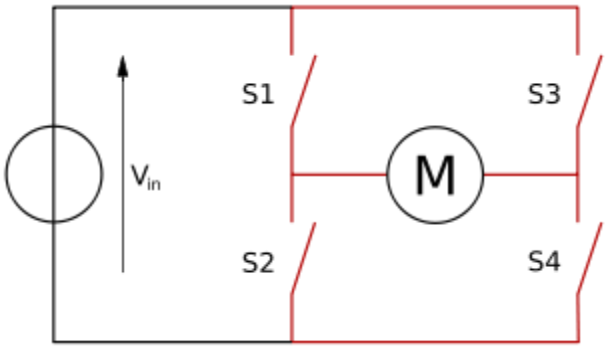
$$Duty\ Cycle = \frac{T_{on}}{T_{on} + T_{off}} \times 100\ \%$$

PWM Modes of Operation

Fast PWM	Phase Correct PWM	Frequency and Phase Correct PWM
		

Used Resources

HBridge



H bridge is an electronic circuit that enables a voltage to be applied across a load in either direction. These circuits are often used in robotics and other applications to allow DC motors to run forwards or backwards.
Most DC-to-AC converters (power inverters), most AC/AC converters, the DC-to-DC push–pull converter, most motor controllers, and many other kinds of power electronics use H bridges. In particular, a bipolar stepper motor is almost invariably driven by a motor controller containing two H bridges.

The H-bridge arrangement is generally used to reverse the polarity/direction of the motor, but can also be used to 'brake' the motor, where the motor comes to a sudden stop, as the motor's terminals are shorted, or to let the motor 'free run' to a stop, as the motor is effectively disconnected from the circuit. The following table summarises operation, with S1-S4 corresponding to the diagram above.

S1	S2	S3	S4	Result
1	0	0	1	Motor moves right
0	1	1	0	Motor moves left
0	0	0	0	Motor coasts
0	1	0	1	Motor brakes
1	0	1	0	Motor brakes
1	1	0	0	Short circuit
0	0	1	1	Short circuit
1	1	1	1	Short circuit

Solution

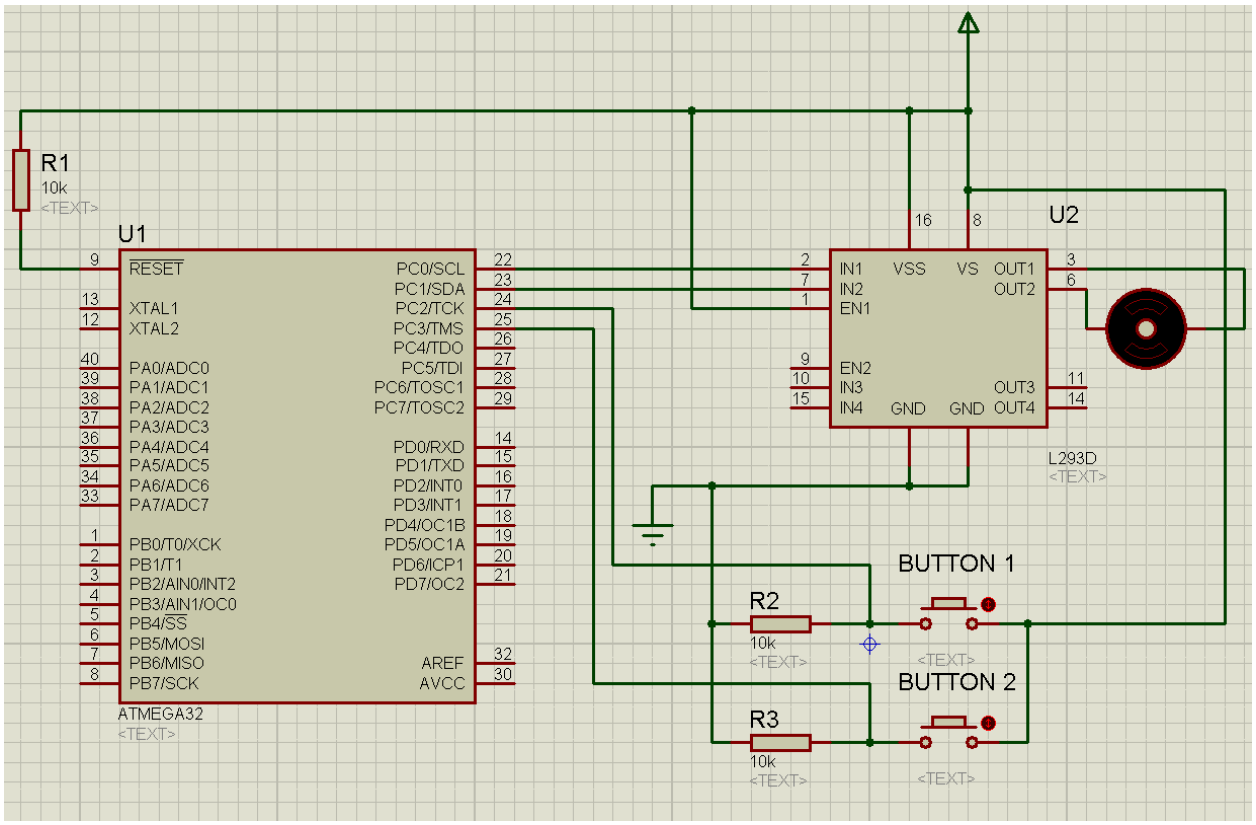
button.h

```
void initButton1();
void initButton2();
int button1Pressed();
int button2Pressed();
```

L293.h

```
void L293_init();
void motorLeft();
void motorRight();
void stopMotor();
void freeMotor();
```

Schematics



Conclusion

This laboratory work gave us a basic concepts about Timers, PWM and how to control a motor. PWM allows us easily to control “how much voltage to provide to motor”. In this laboratory work I used 8bit timers to control PWM.

Appendix:

L293.h

```
#ifndef L293_H_
#define L293_H_


void L293_init();
void motorLeft();
void motorRight();
void stopMotor();
void freeMotor();


#endif /* L293_H_ */
```

L293.c

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

#include "L293.h"


void L293_init() {
    DDRC = 0b11111111;

}


void motorRight()
{
    PORTC = 0b00000010;
}

void motorLeft()
{
     PORTC = 0b00000001;
}

void stopMotor()
{
    PORTC = 0b00000011;
}

void freeMotor()
{
    PORTC = 0b00000000;
}
```

button.h

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

void initButton1();
void initButton2();
int button1Pressed();
int button2Pressed();


#endif /* BUTTON_H_ */
```

button.c

```
#include "button.h"


void initButton1() {
    DDRC &= ~(1 << PORTC2) ;
}

void initButton2() {
    DDRC &= ~(1 << PORTC3) ;
}

int button1Pressed() {
    return PINC & (1<<PORTC2);
}

int button2Pressed() {
    return PINC & (1<<PORTC3);
}
```

lab4.c

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

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



int main(void)
{
    L293_init();
    initButton1();
    initButton2();


      while(1) {

          if(button1Pressed() && button2Pressed()) {
              stopMotor();
          }

          else if(button1Pressed()) {
              motorLeft();
          }

          else if(button2Pressed()) {
              motorRight();
          }

          else {
              freeMotor();
          }

      }
}
```