Ministerul Educatiei al Republicii Moldova
Universitatea Tehnica a Moldovei
Filiera Anglofona

Report

Laboratory Nr.4

Embeded Systems

Performed By:	Daniel Ciobanu
Verified By:	Andrei Bragarenco

Chisinau 2017

Topic : Acutators. General PMW signal

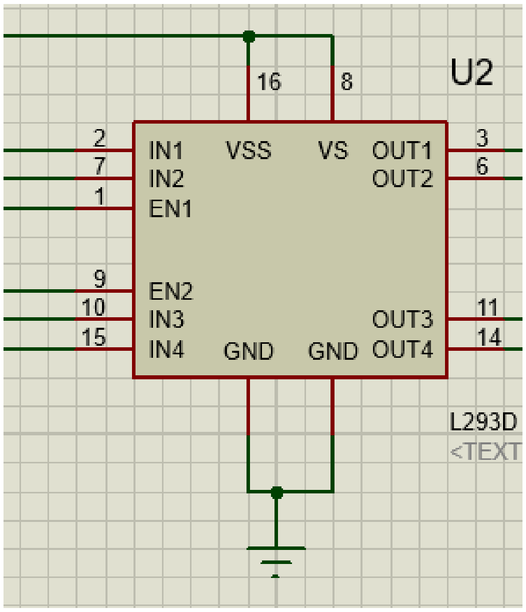
Task: We have to make to motors to work in order to simulate a car movement using an H-bridge and control the speed using PMW signal.

Theory:

A **microcontroller** is a self-contained system with peripherals, memory and a processor that can be used as an embedded system. Most programmable microcontrollers that are used today are embedded in other consumer products or machinery including phones, peripherals, automobiles and household appliances for computer systems. Due to that, another name for a microcontroller is "embedded controller." Some embedded systems are more sophisticated, while others have minimal requirements for memory and programming length and a low software complexity. Input and output devices include solenoids, LCD displays, relays, switches and sensors for data like humidity, temperature or light level, amongst others.

L293D driver:

The L293 and L293D devices 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.



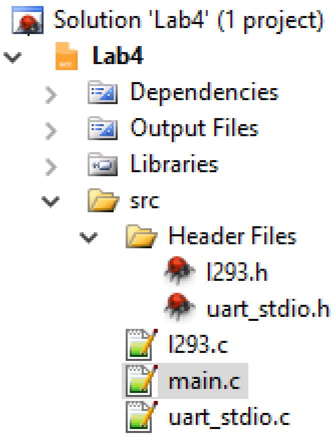
PMV –signal:

Pulse width modulation (PWM) is a powerful technique for controlling analog circuits with a microprocessor's digital outputs. PWM is employed in a wide variety of applications, ranging from measurement and communications to power control and conversion.

One of the advantages of PWM is that the signal remains digital all the way from the processor to the controlled system; no digital-to-analog conversion is necessary. By keeping the signal digital, noise effects are minimized. Noise can only affect a digital signal if it is strong enough to change a logical-1 to a logical-0, or vice versa.

Implementation:

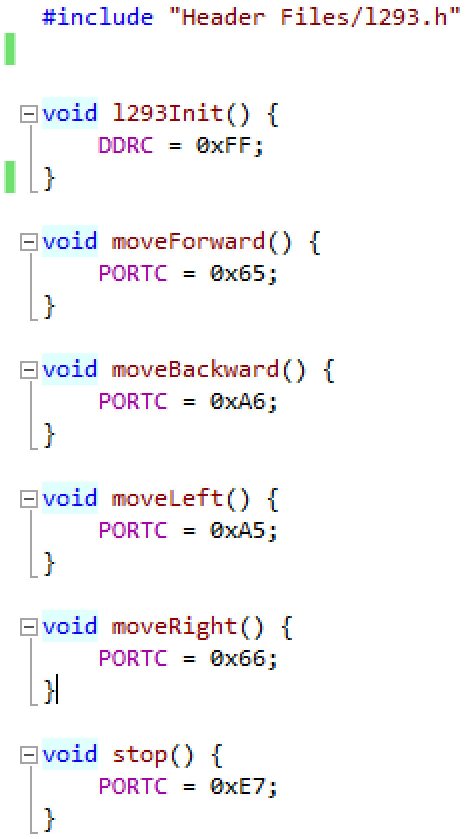
The structure of the laboratory 4 looks like this:



```
Solution 'Lab4' (1 project)
  Lab4
    Dependencies
    Output Files
    Libraries
    src
      Header Files
        l293.h
        uart_stdio.h
      l293.c
      main.c
      uart_stdio.c
```

First we initialize the l293d driver and then we write the functions for moving the motors. We are doing that by giving to PORTC a value that sets the pins in order to pass the voltage according to our needs. For that we need to have l293d logic table.

Input1	Input2	Input3	Input4	Motor State
1	0	0	1	Clockwise rotation
0	1	1	0	Anticlockwise Rotation
0	0	0	0	Idle [High Impedence State]
1	1	1	1	Idle

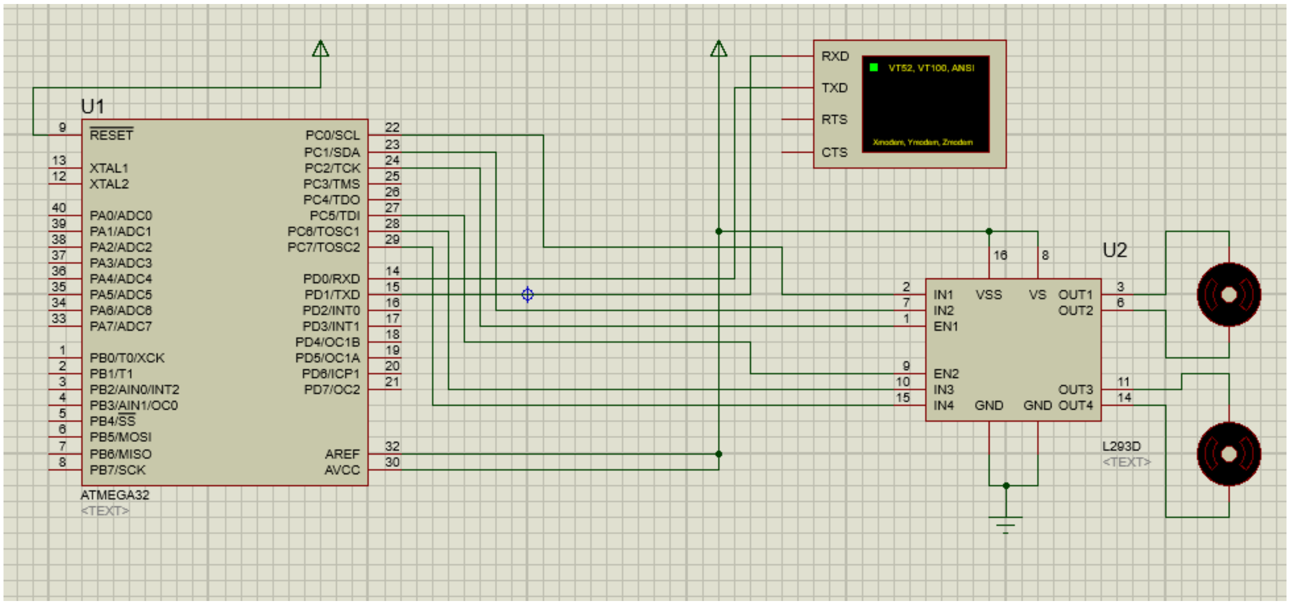


```
#include "Header Files/l293.h"

void l293Init() {
    DDRC = 0xFF;
}

void moveForward() {
    PORTC = 0x65;
}

void moveBackward() {
    PORTC = 0xA6;
}

void moveLeft() {
    PORTC = 0xA5;
}

void moveRight() {
    PORTC = 0x66;
}

void stop() {
    PORTC = 0xE7;
}
```

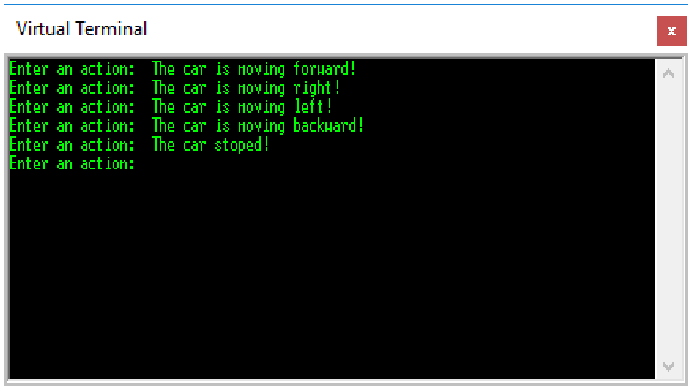
Car controls:

W – move forword
S – move backward
A - move left
D - move right
Q - stop

The scheme in Proteus:



The Result:



```
Enter an action:  The car is moving forward!
Enter an action:  The car is moving right!
Enter an action:  The car is moving left!
Enter an action:  The car is moving backward!
Enter an action:  The car stoped!
Enter an action:
```

Conclusion :

In this laboratory work we gained knowledge how l293d driver works and how to implement it in order to obtain an H-bridge for rotating the motors.

Appendix:

uart_stdio.h

```
#ifndef UART_STDIO_H_
#define UART_STDIO_H_

#define F_CPU 1000000UL
#include <stdio.h>

void uart_stdio_Init(void);
int uart_PutChar(char c, FILE *stream);
char uart_ReadChar();

#endif /* UART_STDIO_H_ */
```

uart_stdio.c

```
#include "Header Files/uart_stdio.h"

#define UART_BAUD  9600

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

FILE uart_output = FDEV_SETUP_STREAM(uart_PutChar, NULL,
_FDEV_SETUP_WRITE);
FILE uart_input = FDEV_SETUP_STREAM(NULL, uart_ReadChar,
_FDEV_SETUP_READ);

void uart_stdio_Init(void) {
     stdout = &uart_output;
     stdin = &uart_input;

     #if F_CPU < 2000000UL && defined(U2X)
     UCSRA = _BV(U2X);             /* improve baud rate error by using
2x clk */
     UBRRL = (F_CPU / (8UL * UART_BAUD)) - 1;
     #else
     UBRRL = (F_CPU / (16UL * UART_BAUD)) - 1;
     #endif
     UCSRB = _BV(TXEN) | _BV(RXEN); /* tx/rx enable */
}
```

```
int uart_PutChar(char c, FILE *stream) {
    if (c == '\n')
    uart_PutChar('\r', stream);

    while (~UCSRA & (1 << UDRE));
    UDR = c;


    return 0;
}

char uart_ReadChar() {
    //Wait untill a data is available
    while(!(UCSRA & (1<<RXC)))
    {
        //Do nothing
    }
    //Now USART has got data from host
    //and is available is buffer

    return UDR;
}
```

main.c

```
int main(void) {

    uart_stdio_Init();
    l293Init();
    char command;

    while(1){
        printf("Enter an action:  ");

        command = getchar();

        switch (command) {

            case 'w':
                    moveForward();
                    printf("The car is moving forward!\r");
                    break;
            case 's':
                    moveBackward();
                    printf("The car is moving backward!\r");
```

```
                              break;
                  case 'a':
                              moveLeft();
                              printf("The car is moving left!\r");
                              break;
                  case 'd':
                              moveRight();
                              printf("The car is moving right!\r");
                              break;
                  case 'q':
                              stop();
                              printf("The car stoped!\r");
                              break;
                  default:
                              printf("command not found");
                              break;
            }

      }
```

L293.h

```
#ifndef L293_H_
#define L293_H_

#include <avr/io.h>

void l293Init();
void moveForward();
void moveBackward();
void moveLeft();
void moveRight();
void stop();

#endif /* LN293_H_ */
```

L293.c

```
#include "Header Files/l293.h"


void l293Init() {
      DDRC = 0xFF;
}

void moveForward() {
      PORTC = 0x65;
}
```

```
void moveBackward() {
      PORTC = 0xA6;
}

void moveLeft() {
      PORTC = 0xA5;
}

void moveRight() {
      PORTC = 0x66;
}

void stop() {
      PORTC = 0xE7;
}
```