Education Ministry of Moldova

Technical University

Computer Science and Microelectronics

Report

Embedded Systems

laboratory work #4

Author:
Bîrcu Maxim

Supervisor:
Bragarenco Andrei

January 16, 2017

Introduction

Topic

Pulse width modulation. Controlling motor with H-Bridge.

Objectives

1. Implement keyboard with USART and Virtual Terminal.
2. Implement PWM
3. Implement H-Bridge.
4. Create a basic car using elements described higher.

Objectives

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

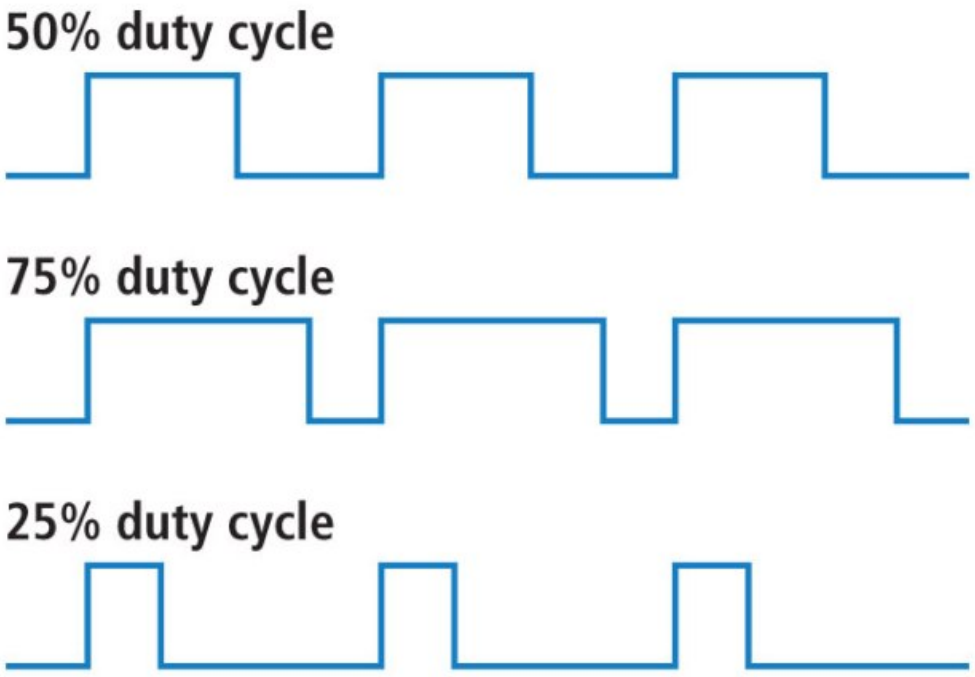
1 PWM

1.1 Definition

Pulse width modulation (PWM) is a fancy term for describing a type of digital signal. Pulse width modulation is used in a variety of applications including sophisticated control circuitry. A common way we use them here at SparkFun is to control dimming of RGB LEDs or to control the direction of a servo motor. We can accomplish a range of results in both applications because pulse width modulation allows us to vary how much time the signal is high in an analog fashion. While the signal can only be high (usually 5V) or low (ground) at any time, we can change the proportion of time the signal is high compared to when it is low over a consistent time interval.

1.2 Duty Cycle

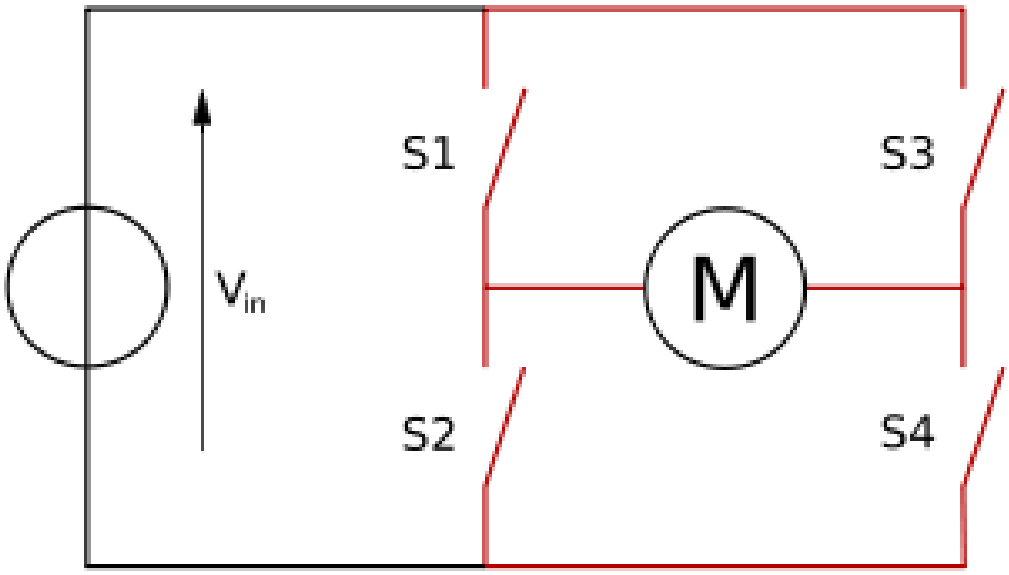
When the signal is high, we call this "on time". To describe the amount of "on time" , we use the concept of duty cycle. Duty cycle is measured in percentage. The percentage duty cycle specifically describes the percentage of time a digital signal is on over an interval or period of time. This period is the inverse of the frequency of the waveform. If a digital signal spends half of the time on and the other half off, we would say the digital signal has a duty cycle of 50% and resembles an ideal square wave. If the percentage is higher than 50%, the digital signal spends more time in the high state than the low state and vice-versa if the duty cycle is less than 50%. Here is a graph that illustrates these three scenarios



2 H-Bridge

2.1 Definition

An 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.



2.2 Operation

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

3 Resources

3.1 Atmel Studio

Atmel Studio 7 is the integrated development platform (IDP) for developing and debugging Atmel® SMART ARM®-based and Atmel AVR® microcontroller (MCU) applications. Studio 7 supports all AVR and Atmel SMART MCUs. The Atmel Studio 7 IDP gives you a seamless and easy-to-use environment to write, build and debug your applications written in C/C++ or assembly code. It also connects seamlessly to Atmel debuggers and development kits. Additionally, Atmel Studio includes Atmel Gallery, an online apps store that allows you to extend your development environment with plug-ins developed by Atmel as well as by third-party tool and embedded software vendors. Atmel Studio 7 can also able seamlessly import your Arduino sketches as C++ projects, providing a simple transition path from Makerspace to Marketplace.

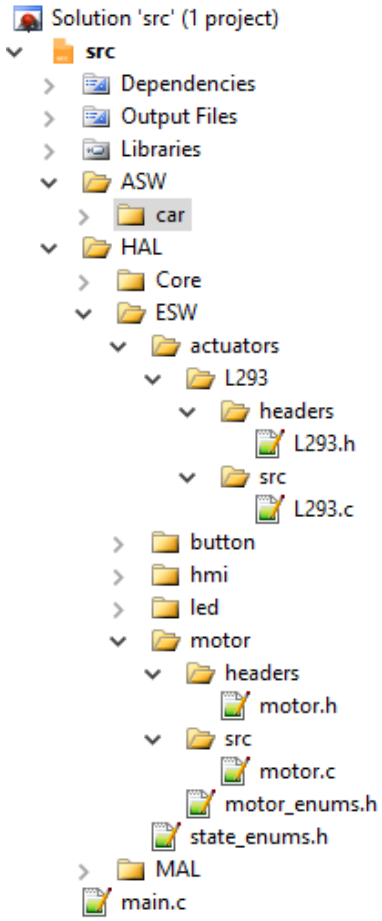
3.2 Proteus

The Proteus Design Suite is an Electronic Design Automation (EDA) tool including schematic capture, simulation and PCB Layout modules. It is developed in Yorkshire, England by Lab center Electronics Ltd with offices in North America and several overseas sales channels. The software runs on the Windows operating system and is available in English, French, Spanish and Chinese languages.

4 Solution

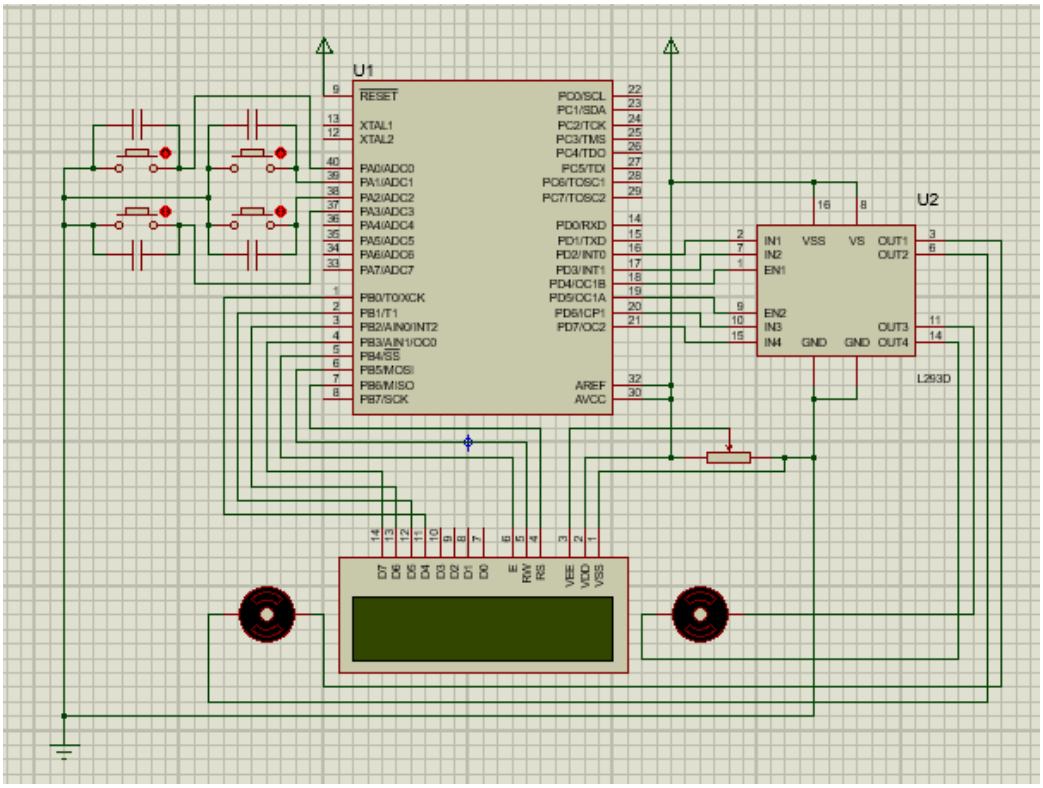
4.1 Project Structure

Project contains MAL(Micro-Controller abstraction layer) that internal rivers interfacing, also it has a ESW(Electronic Software) that holds internal drivers abstractions and external drivers interfacing as for example L293, and of course solution contains ASW(Application Software) which is the most abstract part of the project and which is the car implementation.

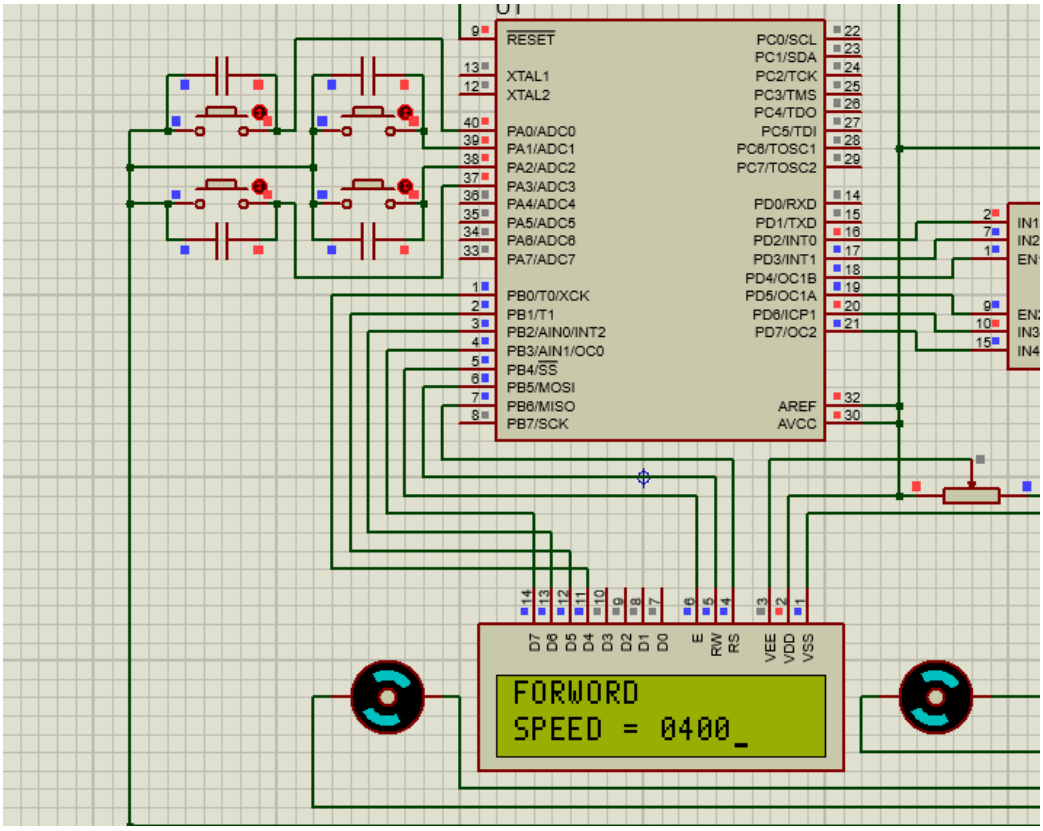


4.2 Circuit in Proteus

I've connected a lcd display to visualize the current state of the car also I connected a H-Bridge circuit L293 to micro-controller and connected 2 DC motors to this H-Bridge.



4.3 Simulation



Conclusion

In this laboratory work I've learned basic concepts of MCU programming in C language and building a simple printed circuit using Proteus. I've implemented a project that simulates the car behavior using a H-Bridge circuit and 2 DC motors.

Apendix

Main

```
/*
 * src.c
 *
 * Created: 11/20/2016 3:59:23 PM
 *  Author: max
 */


#include <avr/io.h>
#include <util/delay.h>
#include "HAL/ESW/motor/headers/motor.h"
#include "HAL/MAL/usart/headers/uart_stdio.h"
#include "HAL/ESW/led/headers/led.h"
#include "HAL/ESW/button/headers/button.h"
#include "ASW/car/headers/car.h"
#include "HAL/ESW/hmi/lcd/headers/lcd.h"

void displayMsg(char* direction, int speed){
        LCDClear();
        LCDWriteString(direction);
        LCDGotoXY(0,1);
        LCDWriteString("SPEED = ");
        LCDWriteInt(speed,4);
}

void displayInfo(Car *car){
        if(car->motor1.direction == forward){
                displayMsg("FORWORD", car->motor1.speed);
        } else {
                displayMsg("BACKWARD", car->motor1.speed);
        }
}
```

```
int main(void)
{
        Car car;
        init_car(&car);

        Button start_btn = create_button(0, &DDRA, &PINA, &PORTA);
        Button switch_direction_btn = create_button(3, &DDRA, &PINA, &PO
        Button speed_up_btn = create_button(1, &DDRA, &PINA, &PORTA);
        Button speed_down_btn = create_button(2, &DDRA, &PINA, &PORTA);

        LCDInit(LS_ULINE);
        LCDWriteString("OFF");

        while(1)
        {
                if(button_release(&start_btn)){
                        if(car_get_state(&car) == off){
                                start_car(&car);
                                displayInfo(&car);
                        } else{
                                stop_car(&car);
                                LCDClear();
                                LCDWriteString("OFF");
                        }
                }

                if(button_release(&switch_direction_btn)){
                        car_switch_direction(&car);
                        displayInfo(&car);
                }

                if(button_release(&speed_up_btn)){
                        car_speed_up(&car);
                        displayInfo(&car);
                }
```

```
                if(button_release(&speed_down_btn)){
                        car_speed_down(&car);
                        displayInfo(&car);
                }
        }
}
```