Faculty of Computers, Informatics and Microelectronics
Technical University of Moldova

Embeded Systems
Laboratory work #4

Authors:
Mereuta Alex

Supervisor:
Bragarenco Andrei

2016

1 Topic

Pulse width Modulation. Controlling motor with H-Brigde.

2 Objectives

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

3 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.

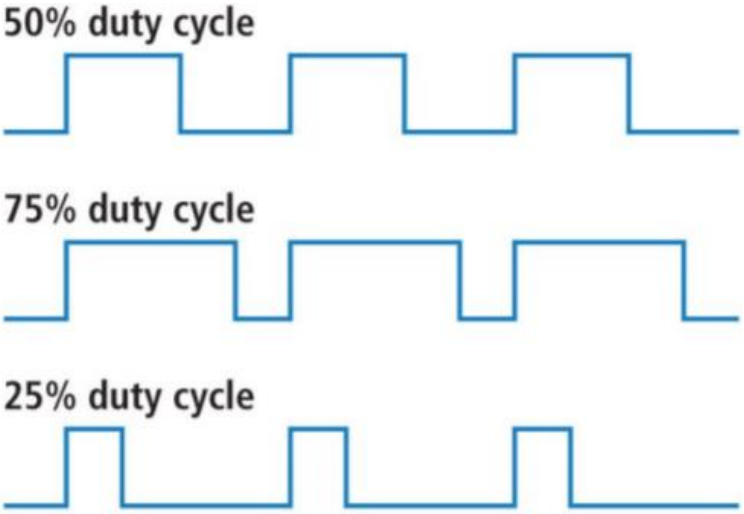
4 Domain

4.1 Pulse width modulation

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.

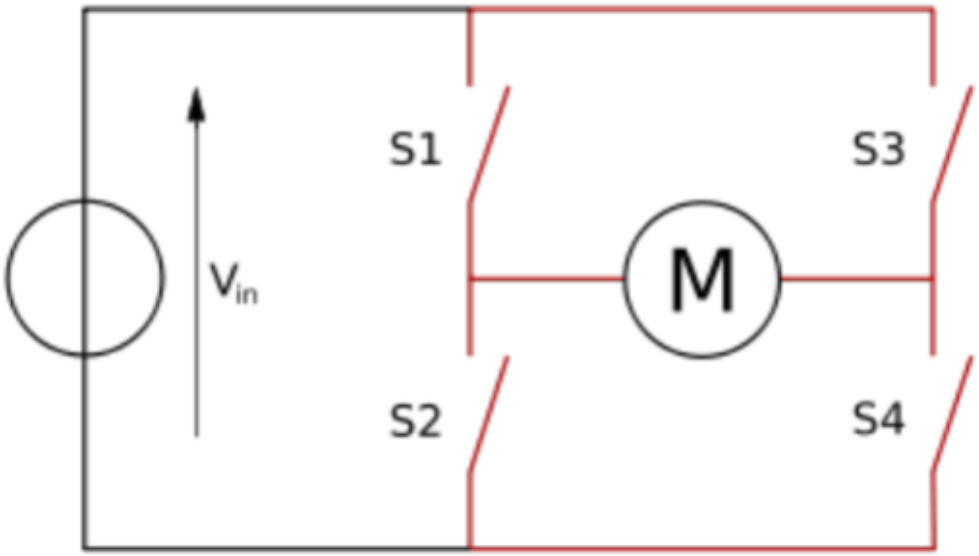
4.2 Duty Cycle

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%. Her is a graph that illustrates these three scenarios.



4.3 H-Bridge

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 DCto-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.



4.4 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.

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

5 Resources

5.1 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.

5.2 Proteus Design Suite

Proteus lets you create and deliver professional PCB designs like never before. With over 785 microcontroller variants ready for simulation straight from the schematic, built in STEP export and a world class shape based autorouter as standard, Proteus Design Suite delivers the complete software package for today and tomorrow's engineers. Proteus let’s use simulate our hardware before creating it. It’s very useful tool especially for beginners. It makes virtual “hardware” which will work like real one.

6 Solution

6.1 Schematics

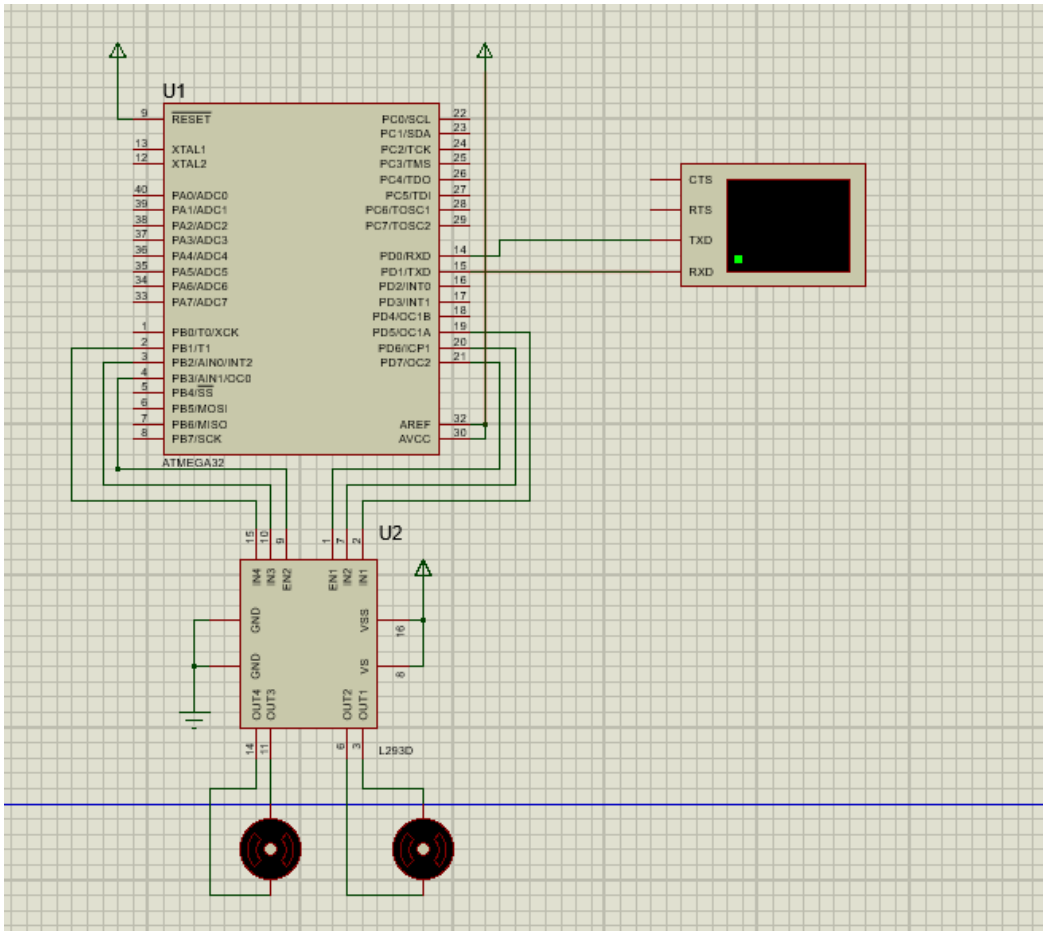


Figure 1 Here we have a virtual terminal, L932D H-Bridge and 2 DC Motors

7 Conclusion

During this laboratory work I've learned the basic concepts about Timers, PMW and how to control a motor. For example, the PMW allows us to control how much voltage is given to the motor. We used 8 bit timers to control the PWM.

8 Annex

8.1 UART Driver

Initializes serial IO for UART. It's used as keyboard controller.

8.2 CAR_2WD Driver

CAR_2WD has descriptor which consists of 2 motors. It has enought methods to control car from keyboard.

```
typedef struct Car {
    Motor *leftMotor;
    Motor *rightMotor;
} Car;

Car* CAR_create(Motor *leftMotor,Motor *rightMotor);

void CAR_start(Car *descriptor);

void CAR_stop(Car *descriptor);

void CAR_left(Car *descriptor);

void CAR_right(Car *descriptor);

void CAR_forward(Car *descriptor);

void CAR_backward(Car *descriptor);

void CAR_brake(Car *descriptor);

void CAR_calibrate_speed(Car *descriptor,uint8_t increment);
```

8.3 HBRidge Driver

HBridge driver uses descriptor for bridge which has 3 pins, ENABLE, Input 1 and Input 2. Also there is a enum definition of HBridge Operation. This driver has set of methods which

```
typedef struct HBridge {
    GPIO *en;
    GPIO *in1;
    GPIO *in2;
```

```
  } HBridge;

  typedef enum HBridge_Operation {
       HBRIDGE_OPERATION_LEFT,
       HBRIDGE_OPERATION_RIGHT,
       HBRIDGE_OPERATION_BREAK
  } HBridge_Operation;

  HBridge* HBRIDGE_create(GPIO *en,GPIO *in1,GPIO *in2);
  void HBRIDGE_init(HBridge *descriptor);
  void HBRIDGE_enable(HBridge *descriptor);
  void HBRIDGE_disable(HBridge *descriptor);
  void HBRIDGE_set_operation(HBridge *descriptor,HBridge_Operation op-
eration);
```

8.4 Motor Driver

```
  typedef enum MOTOR_Direction {
       MOTOR_DIRECTION_LEFT,
       MOTOR_DIRECTION_RIGHT,
  } Motor_Direction;

  typedef struct Motor {
       HBridge *hbridge;
       uint8_t speed;
       void (*pwm)(uint8_t);
  } Motor;

  Motor* MOTOR_create(HBridge *descriptor,void (*pwm)(uint8_t));

  void MOTOR_start(Motor *descriptor);

  void MOTOR_stop(Motor *descriptor);

  void MOTOR_set_direction(Motor *descriptor,Motor_Direction direc-
tion);

  void MOTOR_set_speed(Motor *descriptor,uint8_t speed);

  void MOTOR_reset_speed(Motor *descriptor);


void MOTOR_brake(Motor *descriptor);
```

8.5 PWM Driver

PWM constists of 2 methods, which sets TIMER0 and TIMER1 for PWM(Fast PWM Mode) with 256 prescaller.

```
  /**
   * Sets first 8bit timer for PWM with phase
   */
```

```
void pwm_0_set(uint8_t time_on);

/**
 * Sets second 8bit timer for PWM with phase
 */
void pwm_2_set(uint8_t time_on);
```