Ministerul Educatiei al Republicii Moldova
Universitatea Tehnica a Moldovei
Filiera Anglofona

Report

Embedded Systems

Laboratory Work #4

Performed by: Vlad Croitoru

Verified by: Andrei Bragarenco

Chisinau 2017

Topic:

Pulse Width Modulation. Controlling motor with H-Bridge.

Scope:

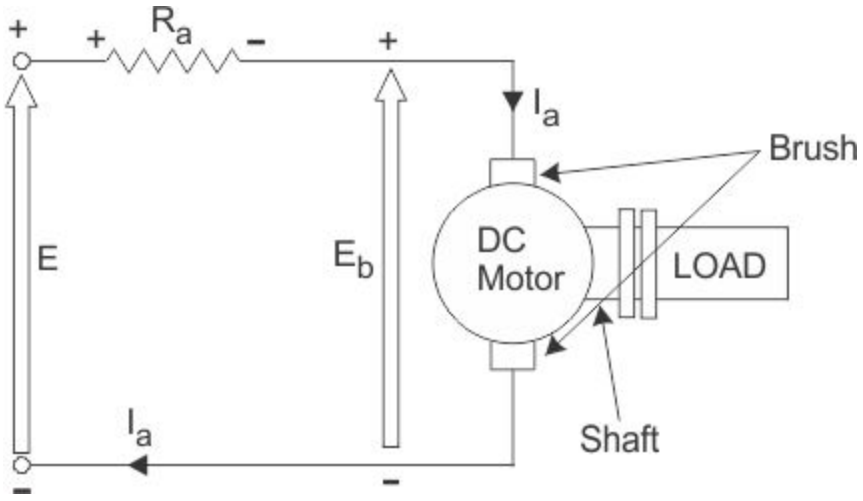
- Implement keyboard control with USART and Virtual Terminal
- Implement PWM
- Implement HBridge

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.

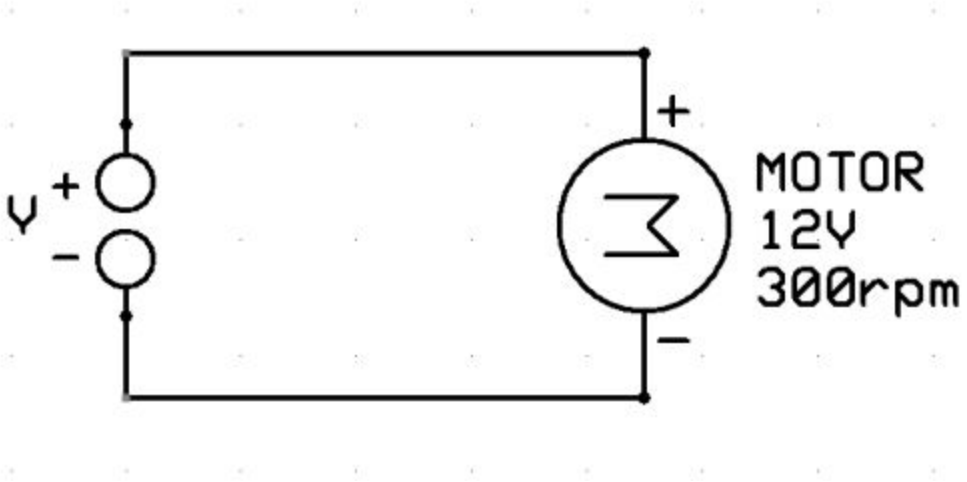
Overview:

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.

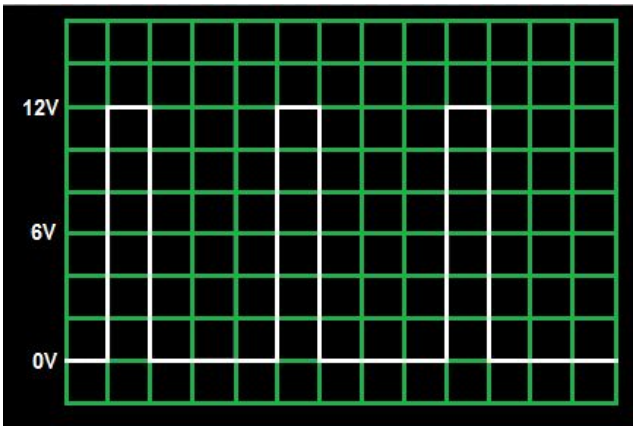
We find that the motor rotates at 150 rpm. Now let us change the voltage level once again as follows (0V DC).



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!



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.

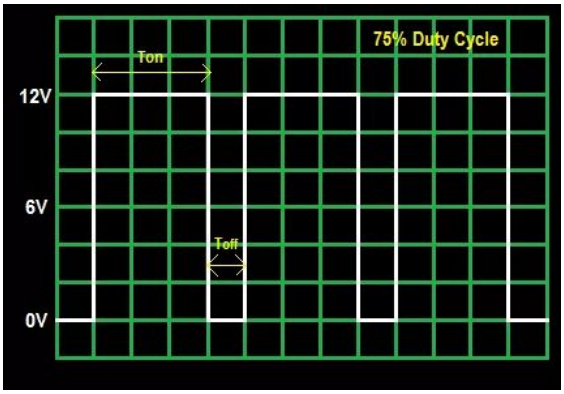
It has three **compare output modes** of operation:

- **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.

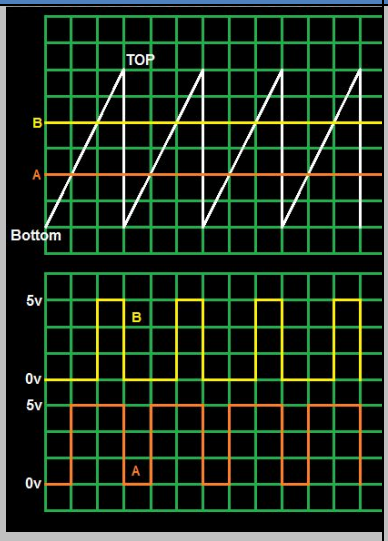
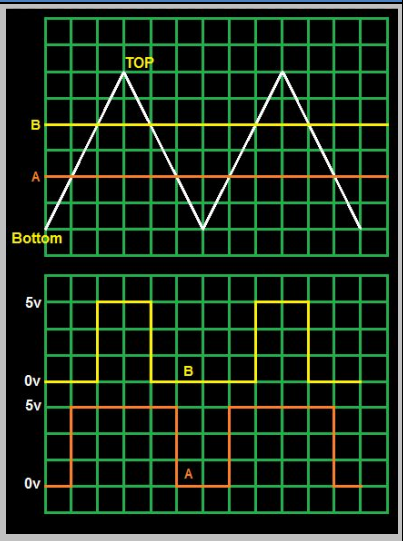
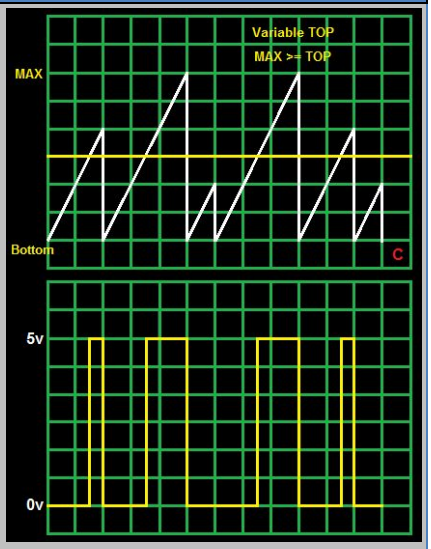
The Duty Cycle of a PWM

Waveform is given by

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

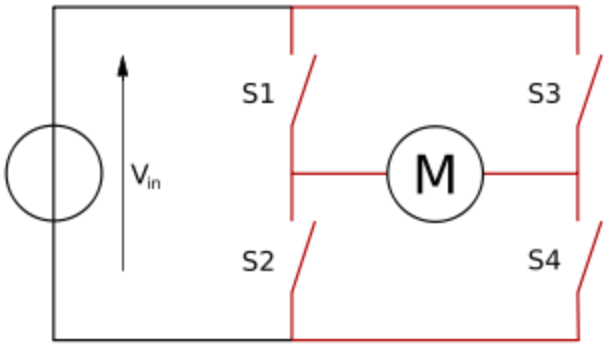


PWM Modes of Operation

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:

Project Structure:

UART Driver

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

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);
```

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 operation);
```

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 direction);

void MOTOR_set_speed(Motor *descriptor,uint8_t speed);

void MOTOR_reset_speed(Motor *descriptor);


void MOTOR_brake(Motor *descriptor);
```

PWM Driver

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

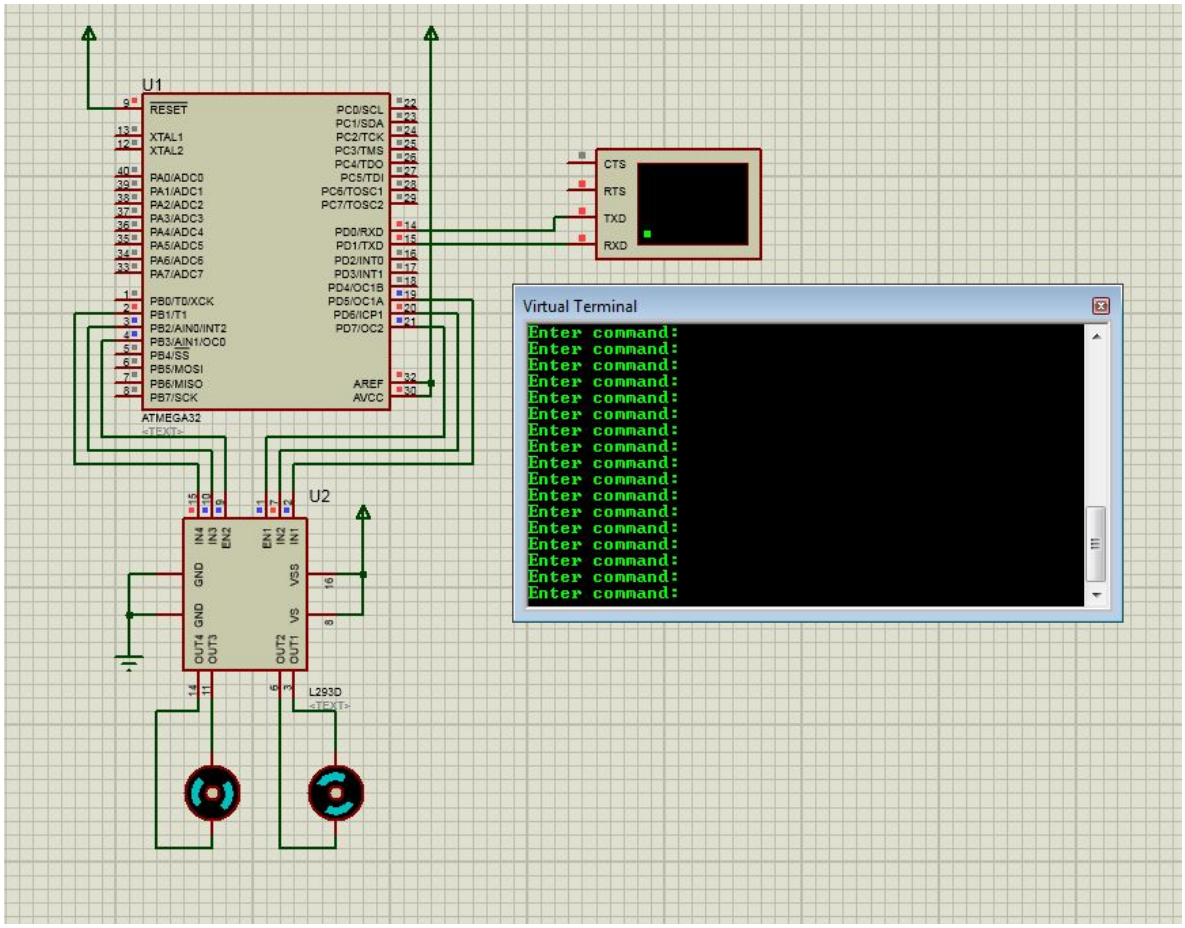
```
/**
 * Sets first 8bit timer for PWM with phase
 */
void pwm_0_set(uint8_t time_on);

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

Schematics

Used elements:

1. Virtual Terminal
2. L932D Hbridge
3. 2 DC Motors



Conclusion

This laboratory work gave us a basic concepts about Timers, PWM and how to control a motor. PWM allows us easily to control the amount of voltage we provide to the motor. So 8bit timers were used to control PWM.

I had set both timer to work on 256 PRESCALER, because it allowed me a CYCLE to be (For 1mhz frequency)

$$\frac{1}{10^6} * 256 * 256 = 0.065536$$

That's good enough. This means we have ~ 65ms a cycle. It's enough.

I used L932D as **dual bridge** for controlling wheels and pwm connected to HBridge **Enable** to control it's speed.