Philadelphia University
Faculty of Engineering

(DC Motor speed control Using PIC16f877A)

Full Report

By:
(المنذر محمدمهدي الدهني)
201610825

Supervisor
(د.قاسم العبيدي)

December/2018

List of contents

Table of figures

Introduction

Direct current (DC) motors are very important integral parts of drive mechanisms both for domestic, entertainment, and industrial uses.
The wide applications of dc motors are largely dependent on the variability of their capacities of power and running speeds.
The possibility of variation and the ease of control of the drive speed and direction of dc motors have helped to increase the range of areas in which they are being applied, which include such areas as rolling mills, electric vehicle tractions, electric trains, electric bicycles, guided vehicles, home electric appliances, toys and robotic manipulators.

Characteristics of the DC motor that are required to be controlled or changed at any time usually include speed, torque and direction of rotation.
Important speed–related requirements for efficient and intelligent dc motor speed control include maintaining constant speed with variation of load, varying the running speed to suit particular needs, for energy saving, soft starting and stopping to increase motor reliability and availability.

Normally, in the conventional DC motors which is known to have high starting torque and a high no-load speed, the speed is poorly regulated in the sense that the speed changes when the amount of load it drives changes and would require additional supply potentials to maintain a steady running speed. But in the permanent magnet DC motors, the torque-speed characteristic is usually linear because the flux is generated by the permanent magnet rather than the wire wound poles in conventional dc motors. II.

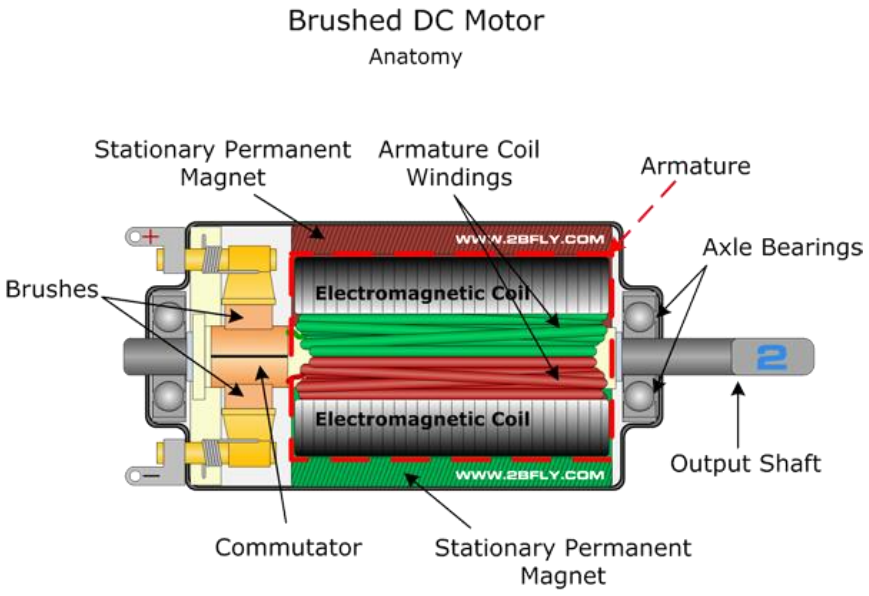


Figure 1

DC Motor speed control with PIC16F877A Microcontroller

The best way to control speed of a DC Motor is by using PWM.
(**Pulse Width Modulation**, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off.)

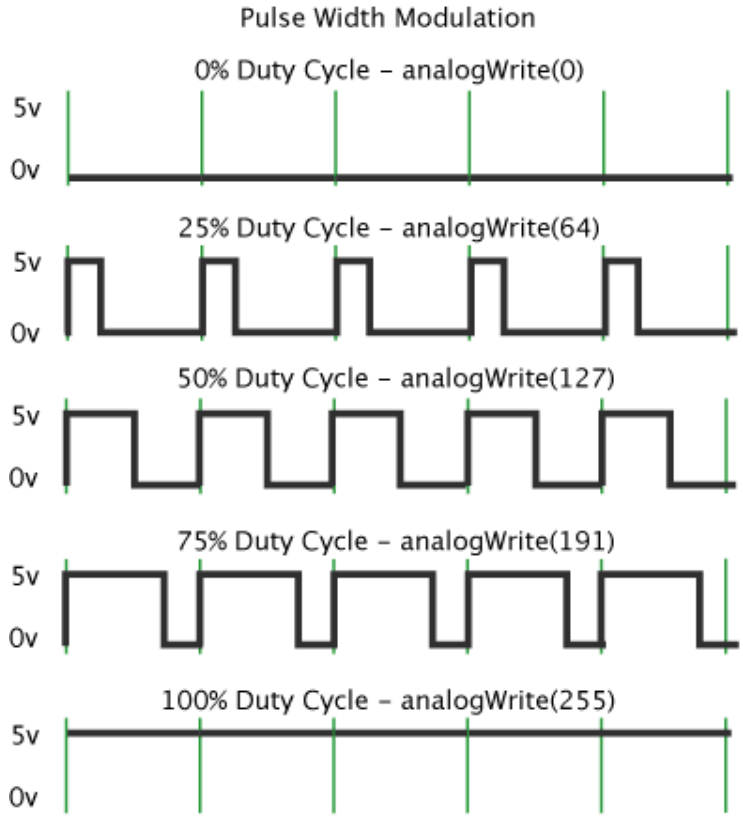


Figure 2

From the figure above when the PWM signal duty cycle is varied and if the duty cycle is changed the power delivered to the motor will be changed also.

The idea of speed control is to switch the motor ON and OFF at varying speeds, let us say we have a 12V DC motor and we applied a constant 12V signal to that motor, the motor would run in its full power (full speed).
Now assume we apply a 50% duty cycle signal to the motor (at several KHz frequency), the motor will turn ON and OFF continuously and the effective voltage applied to the motor is 6V, this would decrease the motor speed by the half, and the motor would be running at 50% of its full power.
Varying the duty cycle of the applied signal would cause the speed to vary, a 0% duty cycle signal would turn OFF the motor, and a 100% one would run the motor at its full speed.

The PIC16F84A does not have PWM generator so we use PIC16F877A which has one.

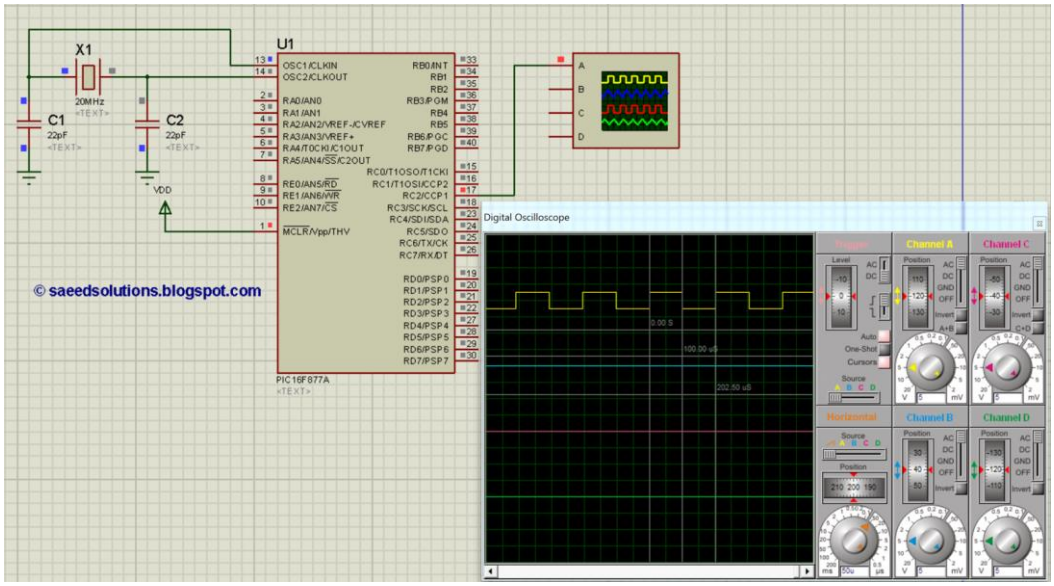


Figure 3

16F877a includes two PWM modules included in the CCP1, CCP2 modules (CCP stands for Capture-Compare-PWM), and there are 3 steps for using those modules:

- Set the desired CCP module to PWM mode.
- Setup the TIMER2 module, which controls the frequency and the base of your PWM signal.
- Set your duty cycle, rely on the equations below to select it properly.

Both PWM modules use Timer2 to generate signals which means the two modules will have the same frequency.

We useTimer2 to setup the PWM frequency.
Using the following equation:
PWM period = [(PR2) + 1] • 4 • Tosc • (TMR2 Prescale Value)
Where PWM frequency is defined as 1/[PWM period].
PR2 is Timer2 preload value,
Tosc = 1/(Fosc)
TMR2 Prescale Value can be 1, 4 or 16.

1-Hardware design:

Hardware design should look like this:

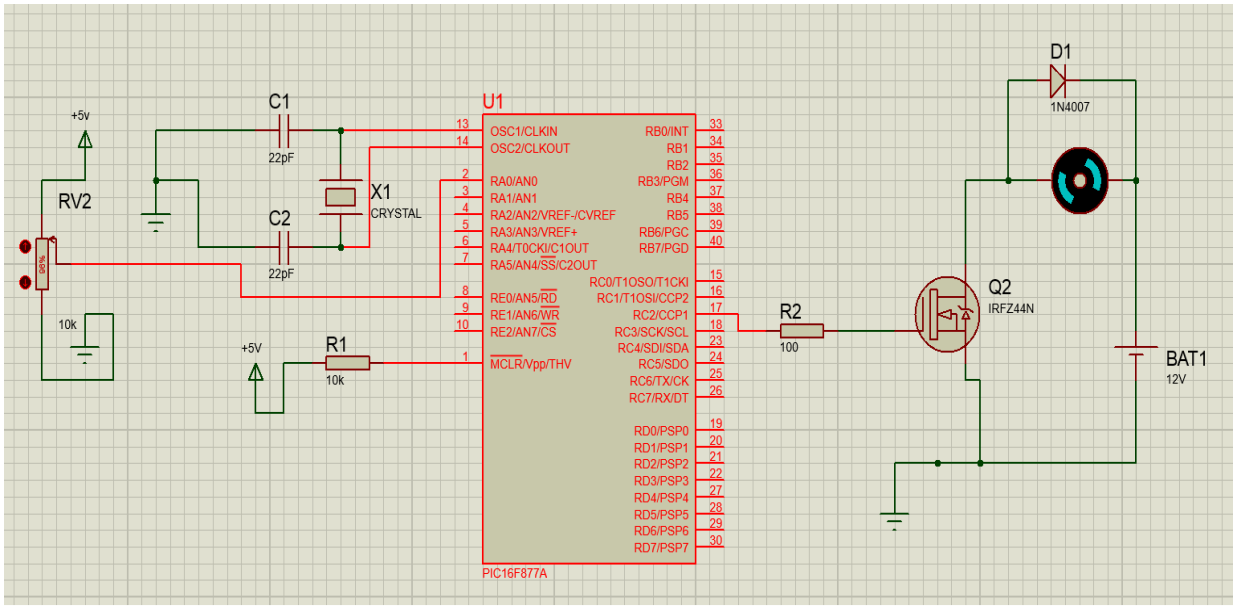


Figure 4

Components:

- pic16f877A
- 10k, 100Ω
- 12v DC motor
- 10k variable resistor
- Two 22pf capacitors
- Diode
- 8 kHz crystal
- N-type Mostfit

We connected pin 2 which is RA0 to a 10k potentiometer as an input and this will be the tool to control the speed by turning its knob.

The output is pin 17 which is CCP1 that generates PWM connected to a N-type mostfit then connected to a 12v DC Motor.

The mostfit is used as a switch here to flow the PWM signal through it

The Source is connected to the PWM output pin of microcontroller.
The Drain is connected to the negative of the motor.
The Gate is connected to the ground.

Notice that we are using two power supplies here:
5v to operate the microcontroller.
12v to operate the DC Motor.

We can use a voltage 7805 voltage regulator to give 5 volt to variable resistor and supply the microcontroller and just use the 12 volt power supply.

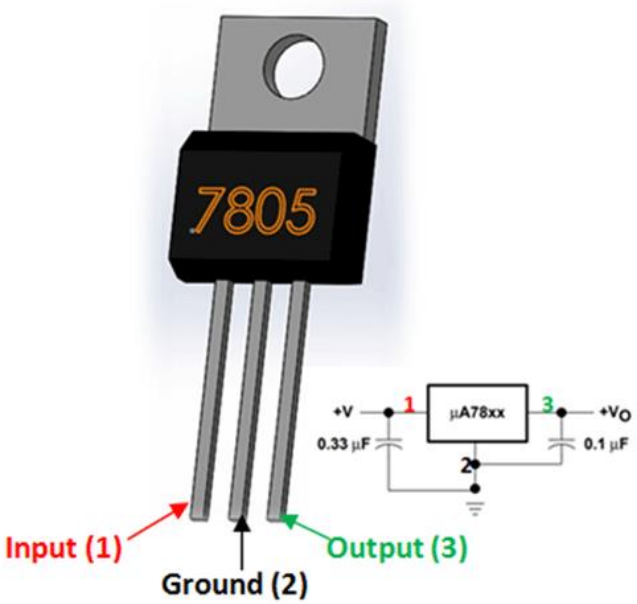


Figure 5

Diode 1N4007 is used as a freewheeling fast recovery diode. Because motor is a inductive load and its back emf (Electromotive force) may damage circuit. Free wheel diode will provide flow path to back emf current and avoid sparking across motor terminals.

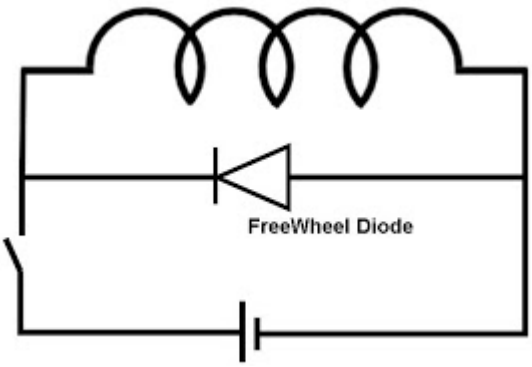
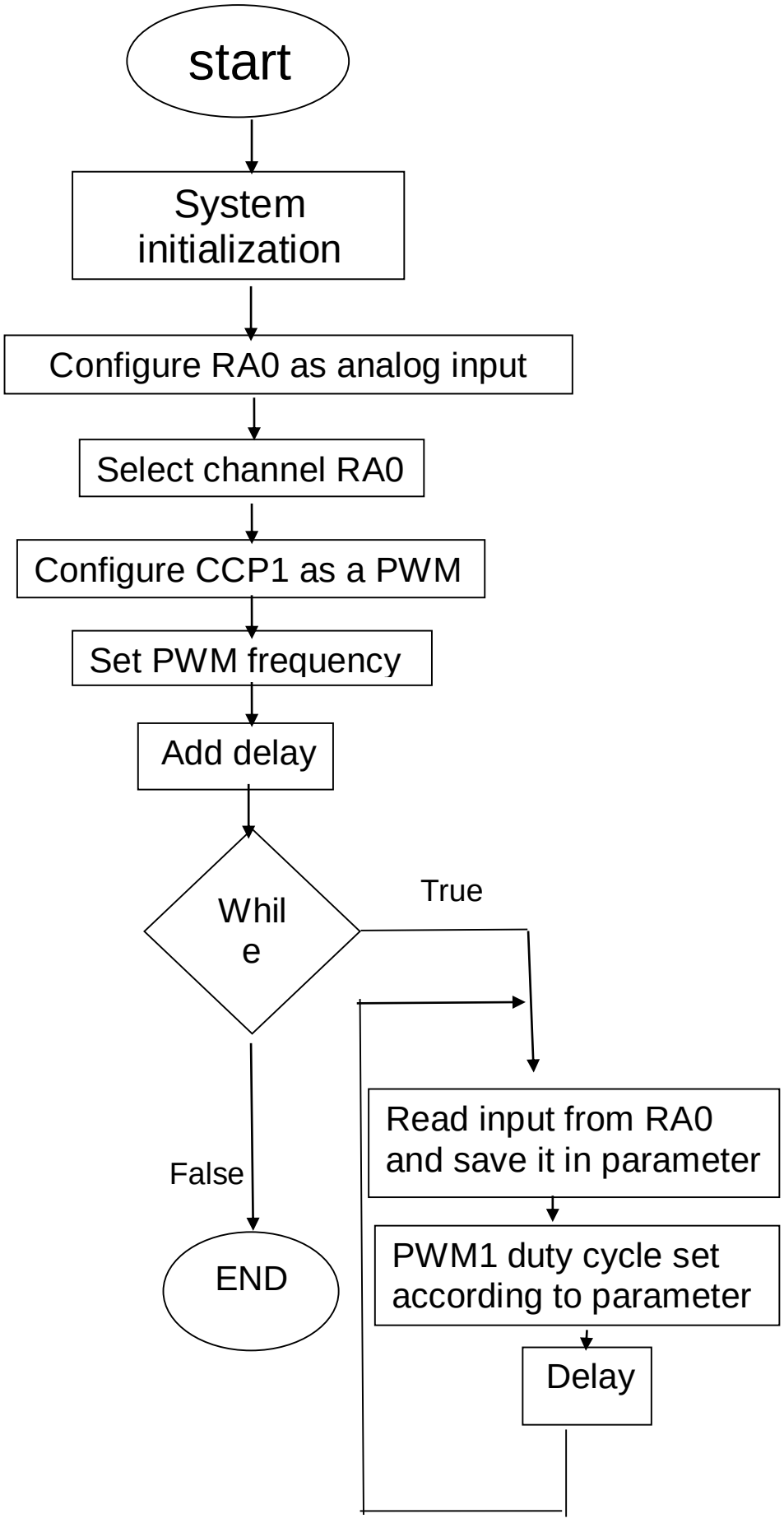


Figure 6

2- Software Flowchart:



Final code:

Notes:
This code is written using C language.
This code is compiled using MikroC PRO for PIC.

```
unsigned int ADR;

void main()
 {

PORTC = 0;
 TRISC = 0; //all PORTC pins output
 PORTA = 0;
 TRISA = 0x01; //RA0 input for pot
CMCON = 7; //Disable analog comparators
T2CON = 0; //Prescaler 1:1, TMR2 off
 ADCON1 = 14; //AN0 only analog
 ADC_Init(); //mikroC initializes ADC
CCP1CON = 12; //PWM mode
PR2 = 249; //16kHz PWM frequency
 while (1){
ADR = ADC_Get_Sample(0) >> 2; //Get ADC reading for channel 0 //and divide reading by
4
if (ADR < 10){ //Setting lower bound
TMR2ON_bit = 0; //Stop Timer and so stop PWM
TMR2 = 0; //Clear Timer
 RC2_bit = 0; //Clear PWM output
 }
else{
 if (ADR > 240){ //Setting upper bound
TMR2ON_bit = 0; //Stop Timer and so stop PWM
 TMR2 = 0; //Clear Timer RC2_bit = 1; //Keep PWM output high }
else{ //Within upper and lower bounds
 CCPR1L = ADR; //Set PWM duty cycle
 TMR2ON_bit = 1; //If not on, turn on TMR2. If on, keep on
              }
          }
      }
   }
```