CHAPTER 1

INTRODUCTION

1.1 Background

Direct current (DC) motors have variable characteristics and are used extensively in variable-speed drives. DC motor can provide a high starting torque and it is also possible to obtain speed control over wide range. Why do we need a speed motor controller? For example, if we have a DC motor in a robot, if we just apply a constant power to each motor on a robot, then the poor robot will never be able to maintain a steady speed. It will go slower over carpet, faster over smooth flooring, slower up hill, faster downhill, etc. So, it is important to make a controller to control the speed of DC motor in desired speed.

DC motor plays a significant role in modern industrial. These are several types of applications where the load on the DC motor varies over a speed range. These applications may demand high-speed control accuracy and good dynamic responses.

In home appliances, washers, dryers and compressors are good examples. In automotive, fuel pump control, electronic steering control, engine control and electric vehicle control are good examples of these. In aerospace, there are a number of applications, like centrifuges, pumps, robotic arm controls, gyroscope controls and so on.

1.2 Objective

The main core of this project is to design a speed control system of DC Motor by using microcontroller. This system will be able to control the DC motor speed at desired speed regardless the changes of load.

1.3 Organization of thesis

This thesis consists five chapters. In first chapter, it discuss about the objective and scope of this project as long as summary of works. While

Chapter 2 will discuss more on theory and literature reviews that have been done. It will discuss about how to measure speed of DC motor and speed controlling technique that can be used to control the speed of the motor.

In Chapter 3, the discussion will be on Arduino platform and AVR microcontroller Atmega2560.
In Chapter 4, the discussion will be on the methodology hardware and software implementation of this project. The result and discussion will be presented in Chapter 5.

At the end of this thesis the conclusion of this project is discussed and future work that can be done based on this work.

CHAPTER 2

THEORY

This chapter will discuss about theory portion. It will discuss about various speed measurement techniques of DC motor and speed controlling technique that can be used to control the speed of the motor.

2.1 Speed Measurement of DC Motor

To start with this project, we need a device that will measure the speed of the motor shaft. There are several methods which can use to measure the speed of motor. Here, we will only discuss about speed measurement by using tachometer and optical encoder.

2.1.1 Speed Measurement of DC Motor using Tachometer

Tachometer is an instrument that measure speed motor based on concept of back EMF induced in motor when it is running. The EMF is voltages appear on the commutator segments caused by rotated in the magnetic field by some external force.

The magnitude of the EMF is given by [1],

$$\boxed{\mathbf{EMF = K_E\, \varphi\, N}} \qquad [2.1]$$

Where, K_E = a constant based on motor construction
φ = magnetic flux
N = speed of motor (in rpm)

The actual relationship between motor speed and EMF follows and is derived from Equation 2.1,

$$\boxed{\mathbf{N = EMF/\, K_E\, \varphi}} \qquad [2.2]$$

Thus, the motor speed is directly proportional to the EMF voltage ad inversely proportional to the field flux. For permanent magnet DC motor, when the EMF measured is increases, the speed of the motor is also increases with the gain. So, the speed of motor can be measured by measuring the back EMF using tachometer.

2.1.2 Speed Measurement of DC Motor using Optical Encoder

The best way to measure speed is to fit an optical encoder. This shines a beam of light from a transmitter across a small space and detects it with a receiver the other end. If a disc is placed in the space, which has slots cut into it, then the signal will only be picked up when a slot is between the transmitter and receiver. As shown in below figure a a disk with number of slot is placed between a photo transmitter like Infrared LED and photo detector like photodiode or phototransistor.

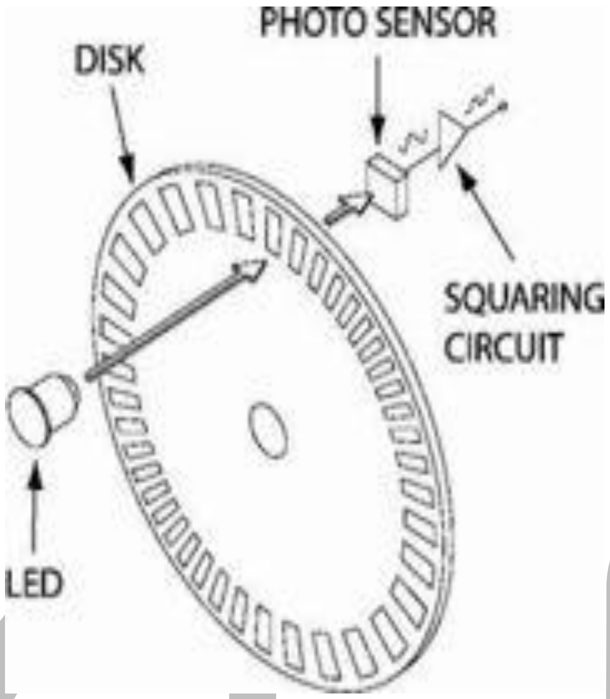


Figure 2.1 Encoder Wheel

The output of photo detector is then given to squaring circuit. This will have an output which swings to +5v when the light is blocked, and about 0.5 volts when light is allowed to pass through the slots in the disc. The frequency of the output waveform is given by,

$$\boxed{f_{out} = (N * Rpm)/ 60} \quad [2.3]$$

Where, f_{out} = frequency of output waveform

Rpm = speed in revolutions per minutes

N = number of slots at disc

So, from Equation 2.3, the speed of DC motor in rpm is given by,

$$\boxed{Rpm = f_{out} * 60 / N} \quad [2.4]$$

2.2 Speed Controlling of DC Motor

For precise speed control of servo system, closed-loop control is normally used. Basically, the block diagram and the flow chart of the speed control are shown in Figure 2.1 and Figure 2.1 respectively. The speed, which is sensed by analog sensing devices (e.g., tachometer), is compared with the reference speed to generate the error signal and to vary the armature voltage of the motor.

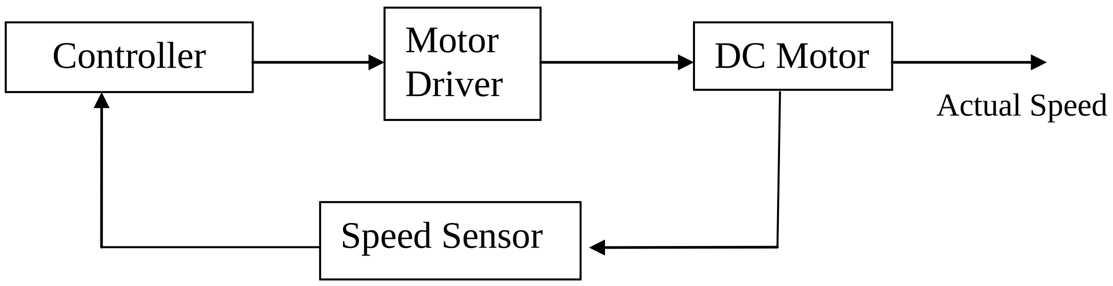


Figure 2.2 Basic block diagram for DC Motor speed control

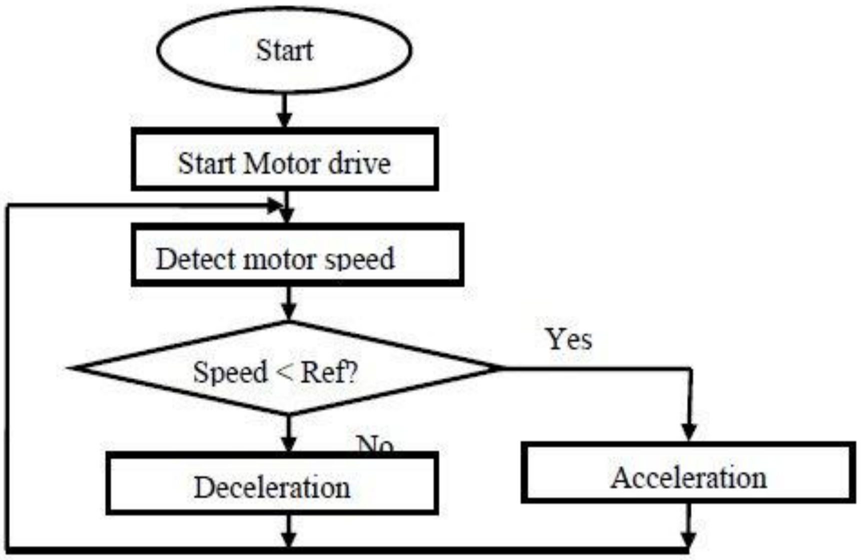


Figure 2.3 Basic flow chart of DC motor speed control

There are several controllers that can used to control the speed of the motor such as by using thyristor, phase-locked-loop control, chopper circuit, Fuzzy Logic Controller, PWM technique and etc. Here Speed Control through PWM technique is discussed.

2.2.1 Speed Control of DC motor using PWM technique

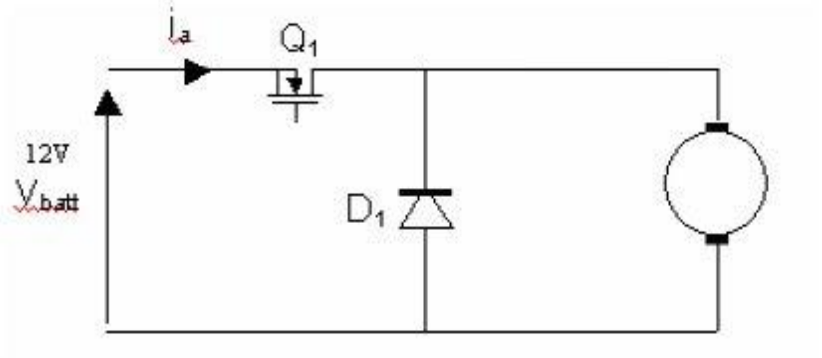


Figure 2.4 simple motor circuit

Let us consider a simple circuit that connects a battery as power supply through a switch MOSFET (Metal-Oxide-Semiconductor Field Effect Transistor) as shown in Figure 2.3. When the switch is closed, the motor sees 12 Volts, and when it is open it sees 0 Volts. If the switch is open for the same amount of time as it is closed, the motor will see an average of 6 Volts, and will run more slowly accordingly.

This *on-off* switching is performed by power MOSFETs. A MOSFET (Metal-Oxide-Semiconductor Field Effect Transistor) is a device that can turn very large currents on and off under the control of a low signal level voltage.

The average of voltage that supply to DC motor is given by,

$$\boxed{\mathbf{V_{ave} = (\ t_{on}/T\ ) * V_{in}}} \qquad [2.5]$$

Where, V_{ave} = average voltage supply to DC motor

t_{on} = time ON of switches

T = period of PWM

$(T_{on/T})$ = DC, duty cycle

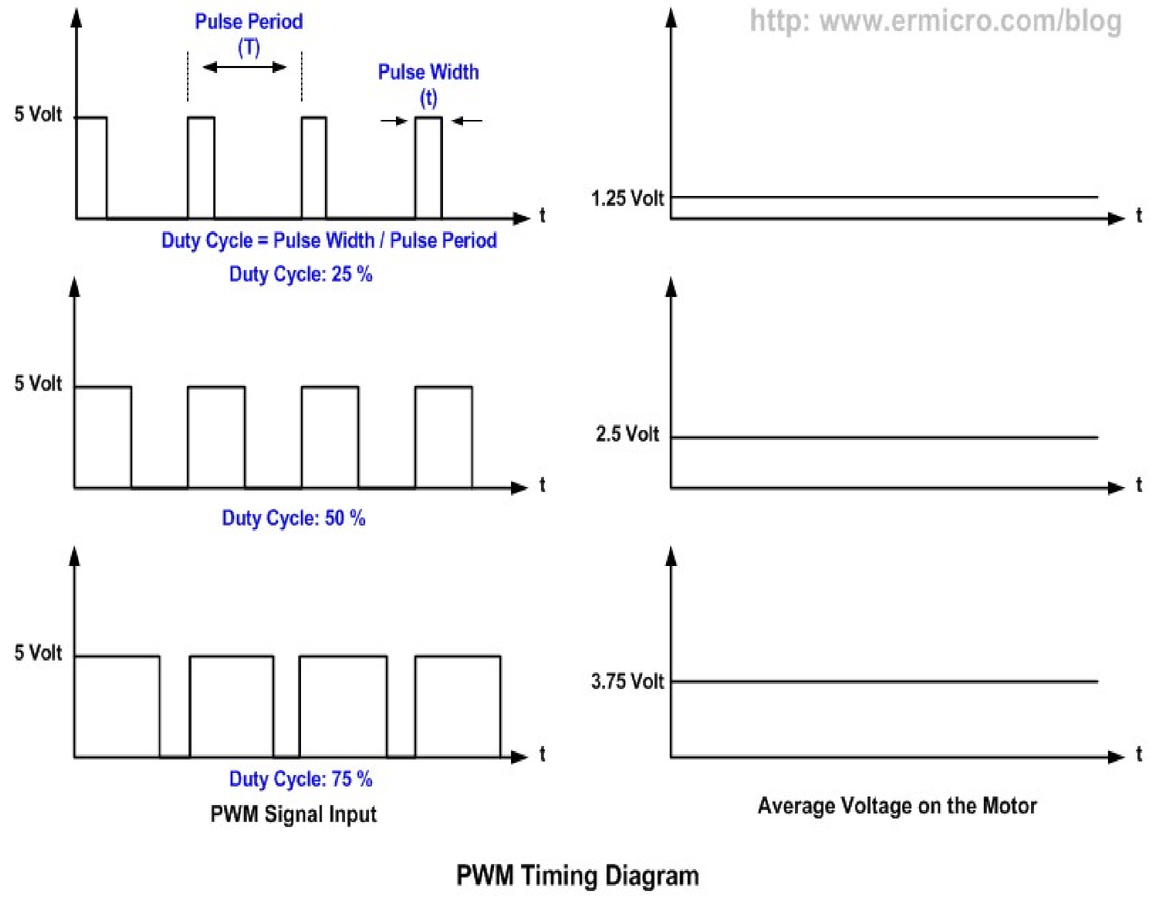


Figure 2.5 PWM signal and average voltage

As the amount of time that the voltage is *on* increases compared with the amount of time that it is *off*, the average speed of the motor increases and vice versa.

The time that it takes a motor to speed up and slow down under switching conditions is depends on the inertia of the rotor (basically how heavy it is), and how much friction and load torque there is. Figure 2.5 shows the speed of a motor that is being turned on and off fairly slowly:

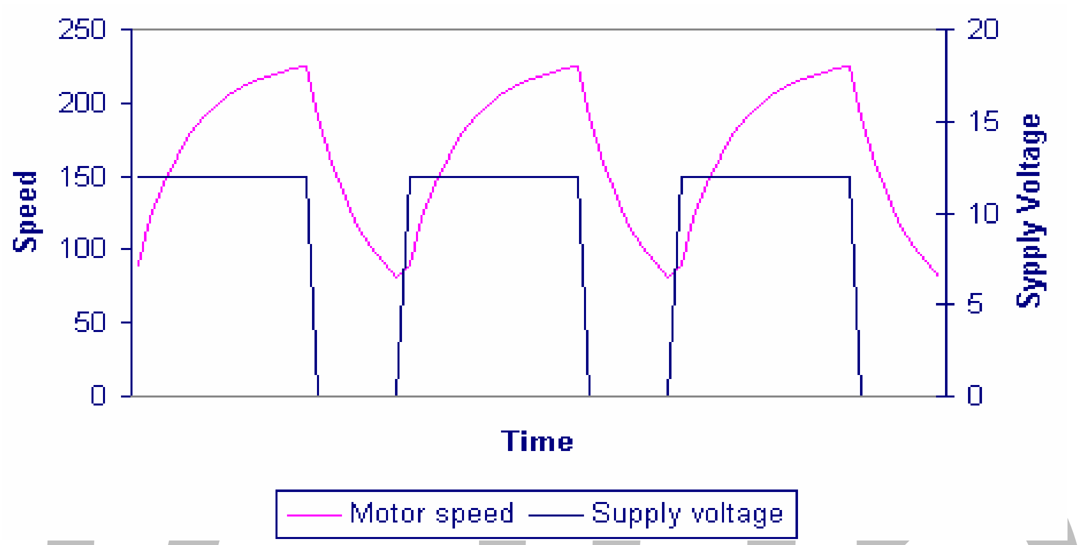


Figure 2.6 Relation of supply voltage with motor speed

We can see that the average speed is around 150 rpm, although it varies quite a bit. If the supply voltage is switched fast enough, it won't have time to change speed much, and the speed will be quite steady. This is the principle of switch mode speed control. Thus the speed is set by PWM – Pulse Width Modulation.

A full bridge circuit is shown in the diagram below. Each side of the motor can be connected either to battery positive, or to battery negative. Only one MOSFET on each side of the motor must be turned on at any one time otherwise they will short out the battery and burn out.

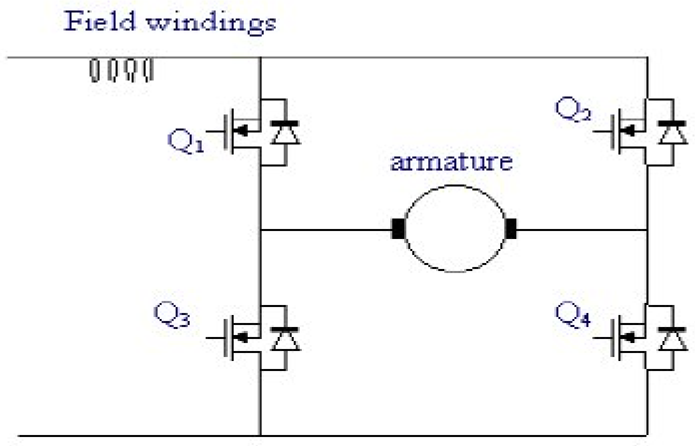


Figure 2.7
Full H-bridge
Motor drive

To make the motor go forwards, Q4 is turned on, and Q1 has the PWM signal applied to it. Meanwhile, to make the motor go backwards, Q3 is turned on, and Q2 has the PWM signal applied to it.

CHAPTER 3

ARDUINO PLATFORM

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing).

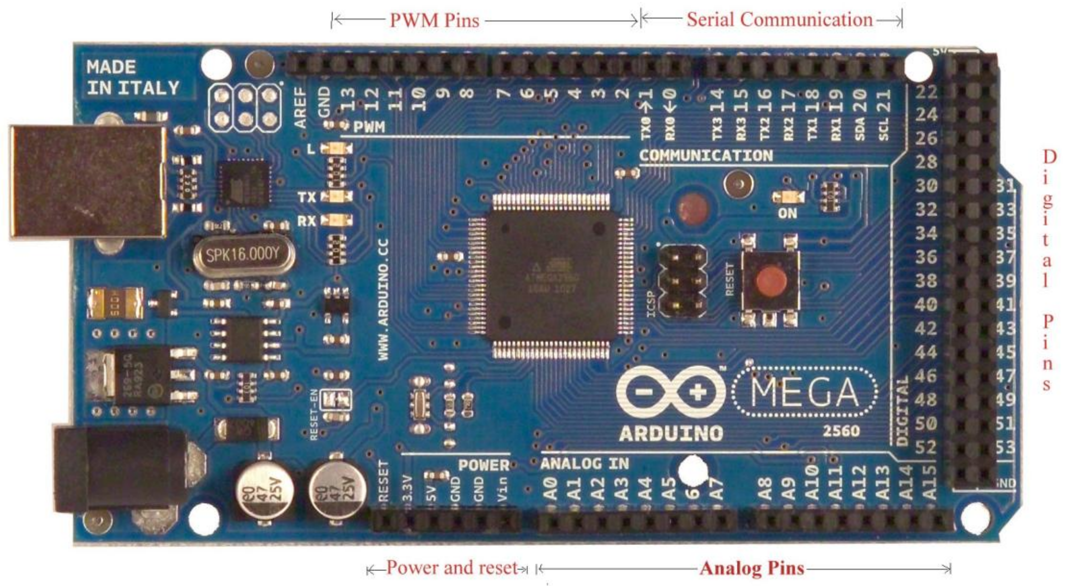


Figure 3.1 Arduino Mega 2560

In this project Arduino Board is used as controller, we have used Arduino Mega 2560, which is shown in above figure. Arduino Mega contains 100 Pin Microcontroller AVR Atmega 250. The IC pin diagram is shown in figure 4.10.

The summary of Board is given below:

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA

DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by boot loader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

The power supply to the Arduino Board is provided through USB port from Computer. It is having easy coding with good libraries. It can also be load by using various other developing environments like BASCOM Avr, AVR Studio, and Atmel studio.

Other features like interrupts, serial communication pins, timer, SPI of board are described below:

Serial:

Pin no: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.

External Interrupts:

Pin no: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. To use interrupt the **attachInterrupt ()** function is used.

PWM:

Pin no: 0 to 13. Provide 8-bit PWM output with the **analogWrite ()** function.

SPI:

Pin no: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.

LED Indication:

Pin no: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

USB Overcurrent Protection

The Arduino Mega2560 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Programming

The ATmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol.

Arduino has vast libraries which can be complied on Arduino software or alpha complier. Arduino is case sensitive language. These libraries make programming language easy. Some common function are given below

pinMode (pin no, INPUT/OUTPUT) : This instruction sets pin of Microcontroller as input in or output pin. The pin no is the number of the pin, which is indicated on Arduino board. For example to set pin 31 as input pin one should write “pinMode(31,INPUT) ”.

digitalWrite(Pin no, HIGH/LOW) : This instruction sets the defined pin as high or low level.

analogWrite (Pin no , PWM): This instruction gives PWM at the decided pin the value of PWM must be between 0 to 255.

Serial. Begin (baud rate): This instruction starts the serial communication circuitry. According to the selected baud rate the timer 2 is settled for serial communication.

attachInterrupt (interrupt pin no, function, mode): This instruction enables the hardware interrupt, interrupt pin no indicates the pin of at which interrupt must set enable. Function indicates high to low transition or low to high transition , while mode indicates the function in the program which should be executed while interrupt occurred.

And many more instructions help to make programming easy. The program can easily upload using USB from computer to board. The Arduino Board is having Atmega 8U2 Microcontroller which is having preloaded program of USB to Serial converter.

CHAPTER 4

METHODOLOGY

4.1 Block Diagram of Optical Encoder speed controller

In this project, microcontroller will be used as the controller to control DC motor speed at desired speed. The block diagram of the system is shown in Figure 4.1. It is a closed-loop with real time control system.

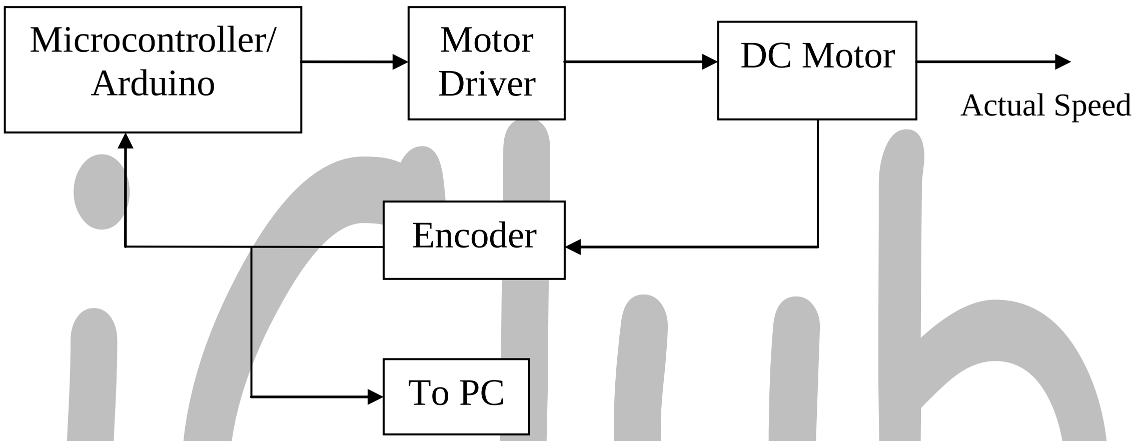


Figure 4.1 Block diagram of DC motor speed control system

The actual speed of DC motor will be measured by encoder and feedback to microcontroller. In microcontroller, it will calculate the error between the desired speeds with the actual speed. The error will determine duty cycle of pulse-width-modulation (PWM) in microcontroller. Then, the duty cycle will send to DC motor driver either accelerate or decelerate DC motor to maintain it at desired speed.

In Our Project we used Arduino mega 2560 board which having on board program facility. L293D is used as motor driver and 300 rpm, 12 V DC Motor is used. Encoder is made of disk having slots on the periphery of it. Infrared Led is used as light source and photodiode is used photo detector. The output of photo detector is given to the squaring circuit. In Our project Schmitt trigger is used as squaring circuit using OP-AMP 741 and positive feedback.

Figure 4.2 shows the picture of the project. The project is divides into two parts that are software and hardware implementation. Each part of the project will discuss in the following section.

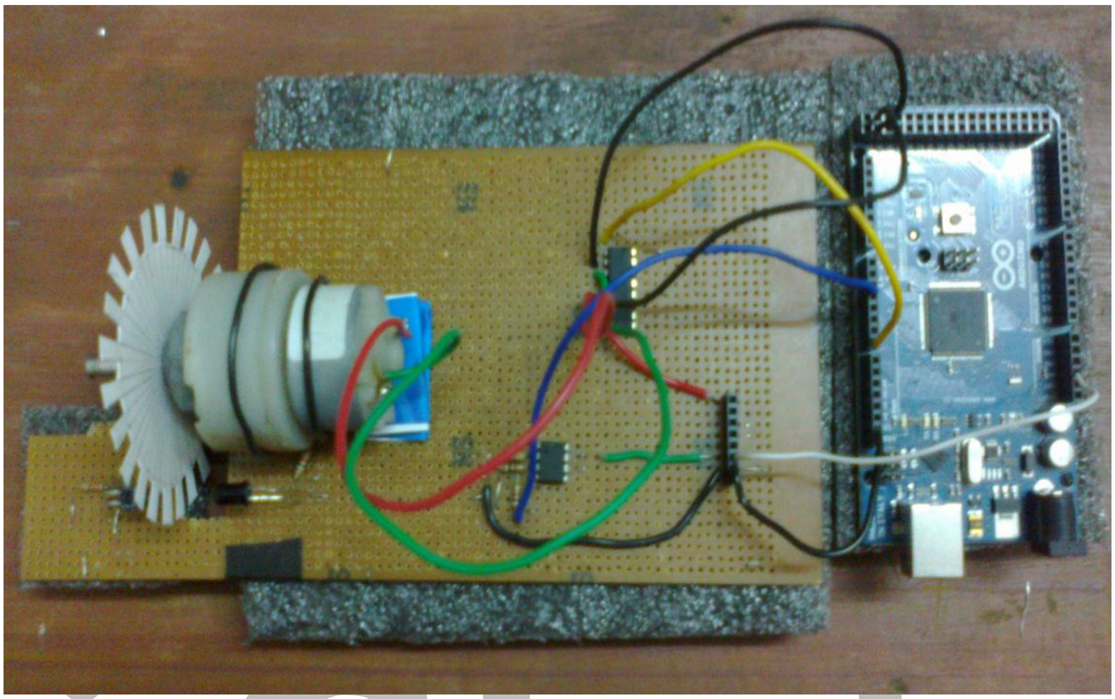


Figure 4.2 Picture of the project

4.2 Hardware implementation

This section will discuss about components that had been used included DC motor, optical encoder, power supply 5V, Arduino microcontroller board, pulse-width-modulation (PWM) and DC Motor drive.

4.2.1 Power Supply

Two different power supplies are used in this project.

1) 5V from PC through USB port to run Arduino Board.
2) 12V, which is converted from 230 V ac supply to run DC Motor.

The circuit diagram of AC to DC converter circuit from AC supply is shown below.

The circuit consists of 230-15 V, 500 mA transformer, with bridge rectifier circuit to convert AC voltage in to DC. The output of rectifier is driven to the voltage regulator circuit. Here two voltage regulator ICs are used 7805 and LM317. 7805 provides constant 5 V at the output node, while LM317 provides variable output between 2.5 V to 14 V.1000μF and 10μF capacitors are used for filtering purpose, which remove high peaks from the output signal.

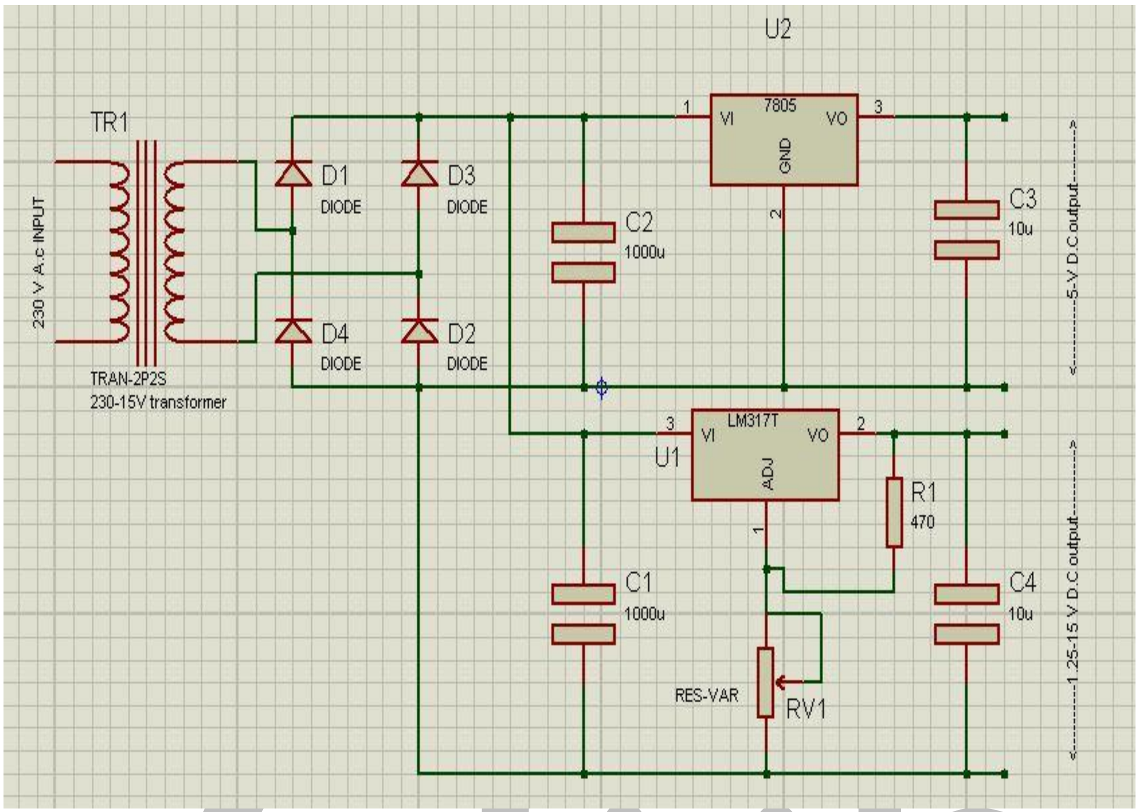


Figure 4.3 230V AC to 2.5 – 14 V DC Converter

Above circuit can able to produce voltage between 2.5 – 14 V. in our project the 12 V dc motor is used ,therfore to run DC motor 12 V is used. Which is apllied to the Motor Driver IC

4.2.2 DC Motor

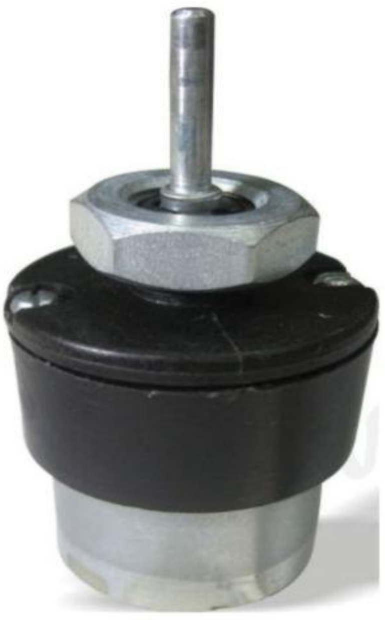


Figure 4.4
12 V, 300 Rpm,
Geared DC Motor

In this project 12V DC Motor is used as shown in figure. It is having speed of 300 Rpm with gears to control the speed. The encoder plate will be mounted on the shaft the motor as shown in figure 4.2

4.2.3 Optical Encoder

This project, an optical encoder will be used to measure the DC motor speed. The fundamental reason for the superiority of this system is that the optical encoder used as the velocity sensor, is capable of much better performance than the generator type of tachometer.

When the optical disc is properly mounted on the motor shaft, it generates a frequency directly proportional to motor speed. Changes in gap, temperature, and magnet strength simply have no effect on the output of the optical tachometer. By contrast, an analog tachometer is directly affected by all the problems listed above.

The encoder in the market is very expensive. In order to reduce the cost of the project, an optical encoder is built. Figure 4.5 show the basic configuration and the schematic of the optical encoder in action respectively.

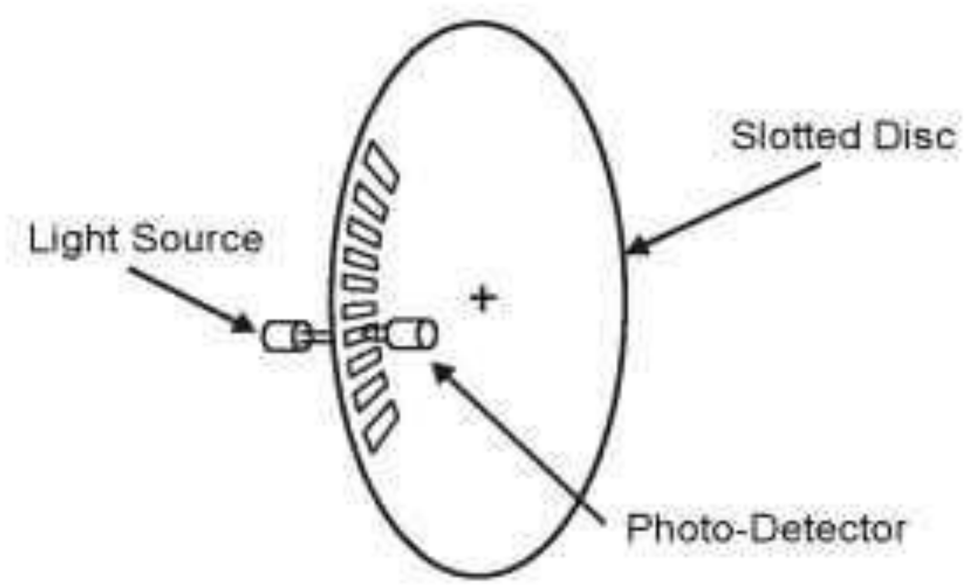


Figure 4.5 Schmeatic of Optical Encoder

The below, Figure 4.6 shows the circuit diagram of Sensors (Photodiode, Infrared LED). The output of Photodiode is based on the input of infrared rays. During the running when slot come across the infrared led and photodiode the infrared rays incidents on photo diode which generate 4 V across resistance R2 as shown in figure. When there is no short between diode and led the output will be 0.5V.

For better response the output is fed to the Schmitt trigger circuit, which is shown in below figure 4.6. OP-AMP 741 is used with positive feedback to build Schmitt trigger. The Schmitt trigger is generally used to generate square pulses and maintain the output state. The operation of Schmitt trigger is such that when 0.5 v is applied to the input of circuit the output will be low or logic state ‘0’, which is maintained low.

When the input reaches above the output state will be high, we can also consider it as logic state '1'. The output is given to the next stage Arduino Board to count the pulses generated by Schmitt trigger in finite time.

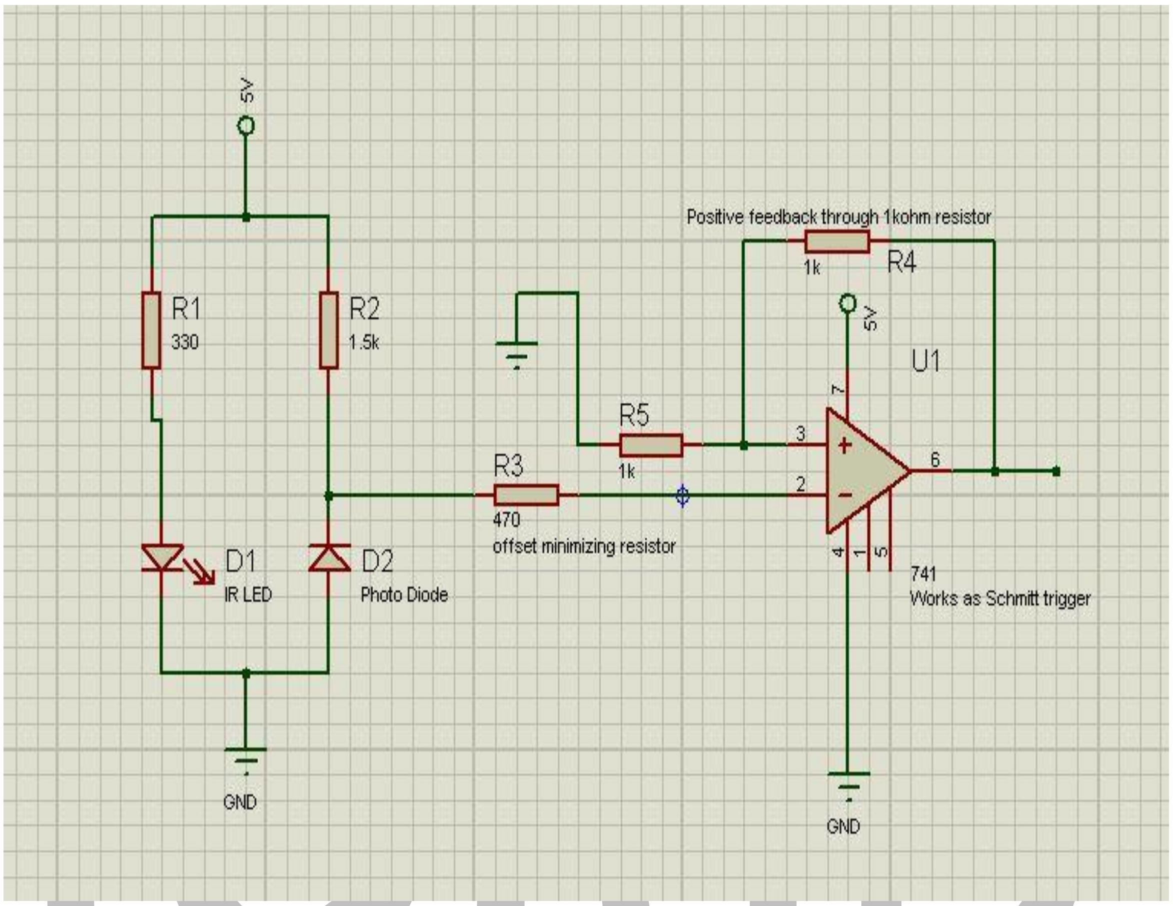


Figure 4.6 Sensor circuit of Optical Encoder and Schmitt trigger

4.2.4 Motor Driver IC

If a DC motor is connected directly to the voltage supply, the constant power will be supplied to the DC motor all the time. Due to the constant power to motor, the speed of motor will slow down when the load is heavier and speed up when the load is lighter. So, DC motor drive is needed where we can control the magnitude of supply voltage in order to control the speed of DC motor.

The DC motor drive that will be used in this project is a dual full bridge driver, chip L293D. The Pin configuration of L293D is shown in Figure 4.7. The operating supply voltage of chip L293D is up to 36V and the total DC current up to 2A.

The time to enable the chip L298 will be determined by the duty cycle pulse that sent from PWM in microcontroller. The average of voltage that supply to DC motor is given by below Equation,

$$\boxed{V_{ave} = (t_{on}/T) * V_{in}}$$

The pin diagram of L293D is shown in below figure

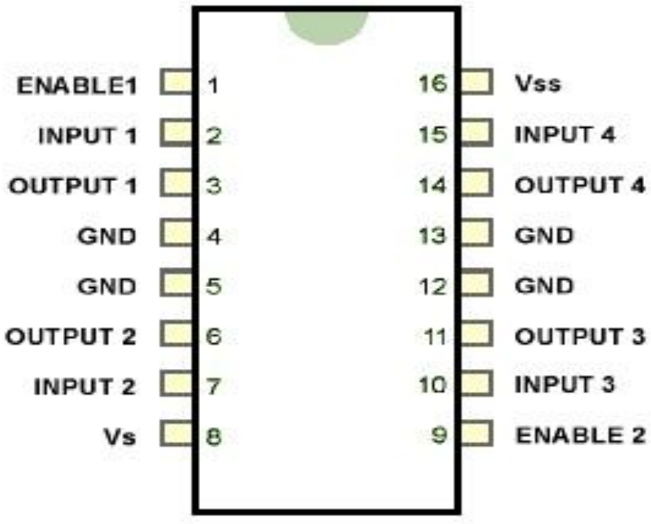


Figure 4.7 Pin Diagram of L293D

Figure 3.10 shows the schematic circuit of L293D with a 12V DC motor. To supply the voltage to motor through L293D, it should be enabled by +5V at pin 1 (Enable 1) or pin 9 (Enable 2). If the chip is disabled, V_{ave} is zero, there is no voltage supply to motor, and the motor will stop running. In this project, the duty cycle of PWM will sent to V_{en} to drive the motor. Therefore the PWM Pin of Arduino board is connected to the pin 1 of L293D.

To run motor at 12V, 12V supply is given to the $V_{s,}$ V_{ss} Pin of the L293D. pin 4,5,13 and 12 should be grounded. Motor Driver provides facility to run DC motor in clock wise and anticlockwise direction without interchanging the cables of DC motor. According to the direction the pin 2, 7 and 15, 10 are settled. This pins are connected with Arduino, which are simple digital pins that runs on digital logic levels '1' and '0'. Two pins are used to control one motor; L293D is dual motor driver so it has 4 control pins. For any direction one pin of the two pins must be high, while other pin must be low. This will change the direction of rotation. According to control pins output pins 3 and 6 are adjusted automatically. These pins are connected with 12V DC motor.

To stop the motor L293D has quick breaking facility. Both control pins 2 and 7 should be high for instant breaking.

Below Figure 4.8 shows L293D with 12V DC motor.

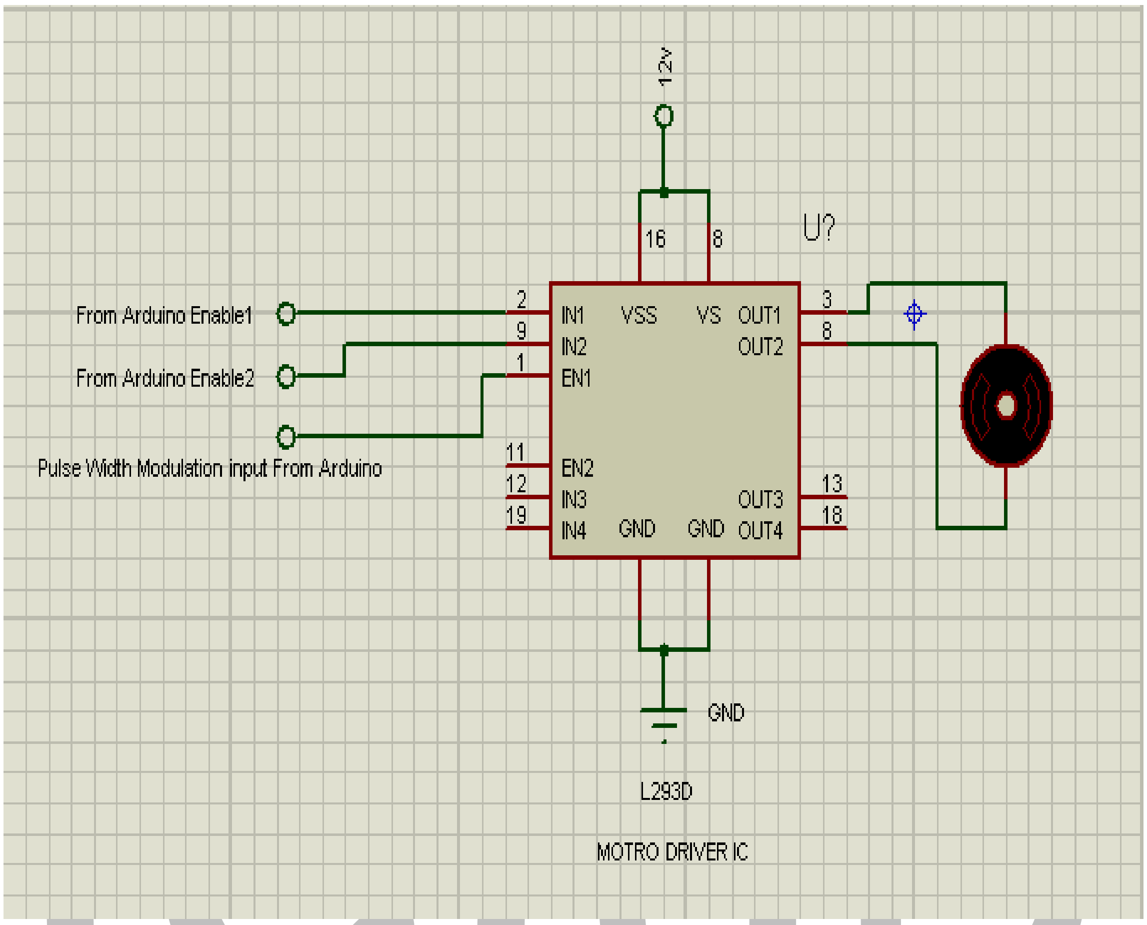


Figure 4.8 L293D with 12V DC motor

4.2.5 Arduino Board Interfacing

As shown in figure 4.9 Arduino Board is having 74 pins. We are using 1 hardware interrupt, 1 timer interrupt, 1 PWM signal and 2 digital pins.

Function	**Arduino Pin No**	**Microcontroller Pin no**
Hardware Interrupt	2	PE4 (OC3B/INT4)
PWM Pin	7	PH4 (OC4B)
Digital pin 1	22	PA0 (AD0)
Digital pin 2	23	PA1 (AD1)

.

Table 4.1 Pin allocation of Arduino Board

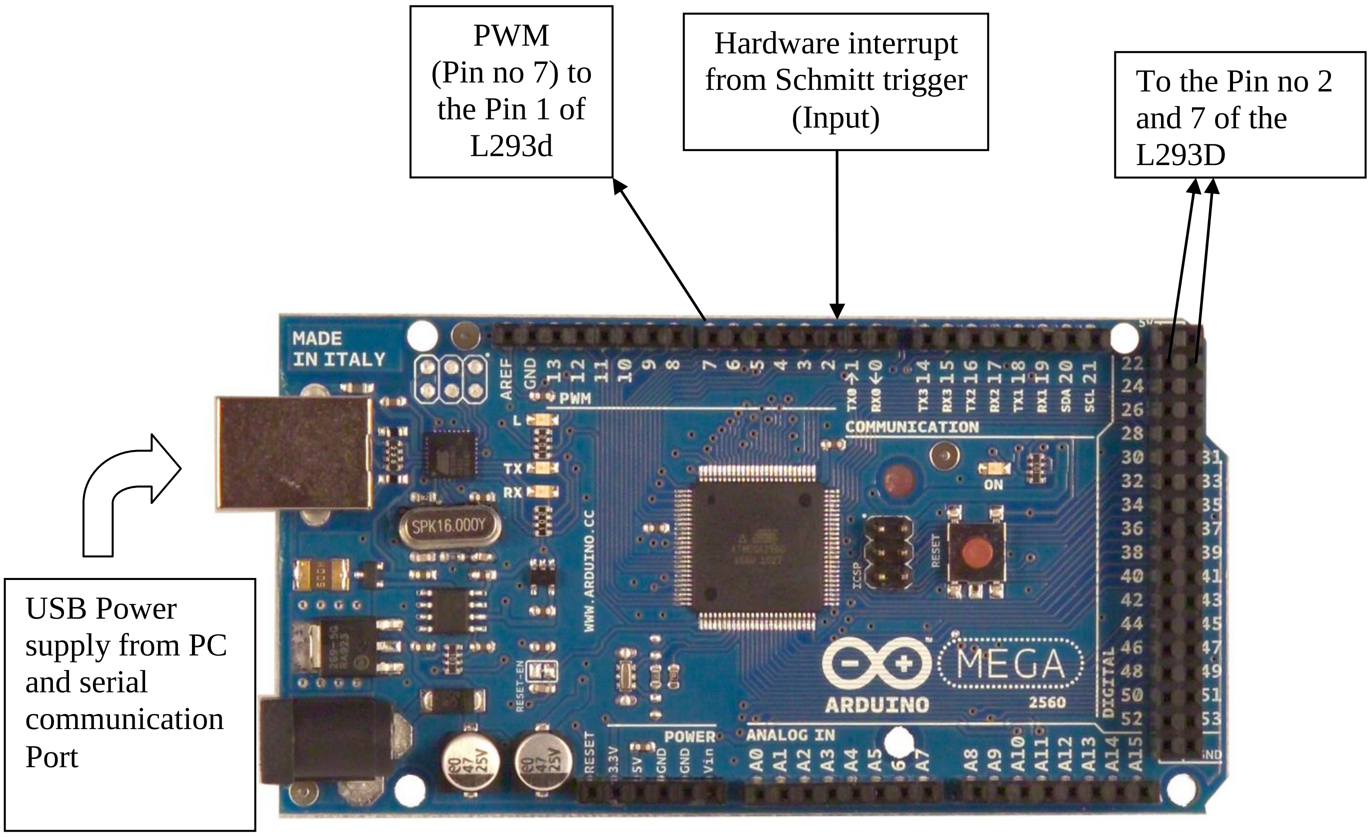

PWM (Pin no 7) to the Pin 1 of L293d
Hardware interrupt from Schmitt trigger (Input)
To the Pin no 2 and 7 of the L293D
USB Power supply from PC and serial communication Port

Figure 4.9 Arduino Mega 2560 interfacing

As shown in above figure Arduino is connected to the PC and motor driver L293D. 5V supply is allocated to the board through USB port from PC. Here 5V supply cannot be given from the power supply circuit of the project because the 12 V from power supply is given to the DC motor, when DC motor is under the load the speed decreases and it generates distortion in the supply that will be result in unwanted interrupts in Arduino Board. Therefore a separate 5V must have to be provided to the Arduino Board.

4.3 Software Implementation

For software implementation, Arduino Alpha compiler is used to program microcontroller in C and C++ language. Besides, Visual Basic 10 is used for user interface purpose and for monitoring the speed response of the system. This section discuss about coding of Arduino Board and Visual basic 2010 and flowcharts of programs.

4.3.1 Microcontroller Programming

In this project Arduino Mega 2560 board is used. It is having Atmega 2560 microcontroller of Atmel Corporation. Programming in project is done in c and c++ language by using Arduino libraries.

4.3.1.1 Flowchart

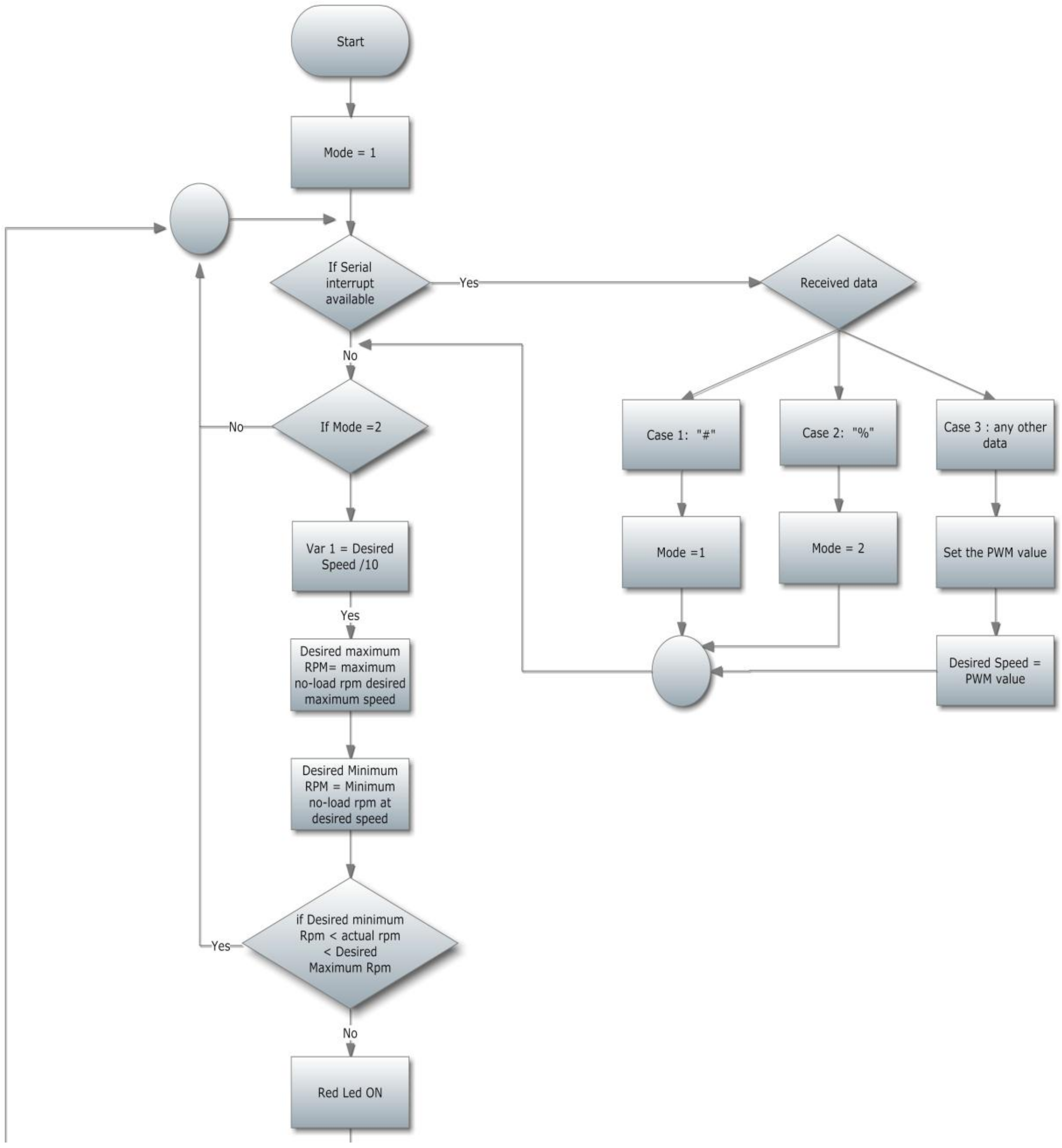
The flowchart contains facility of two modes of project

1) Free running No-load Mode (Mode 1)
2) On-load Capturing Mode (Mode 2)

❖ *Free running No-load Mode:*

In this mode the whole platform of project will run on free running. The input from Visual basic software can able to vary the speed of the motor. The software sends PWM value according to the change in software. The Mode 1 can be activated by selecting button of No-load in software

.


Start
Mode = 1
If Serial interrupt available
Yes
Received data
No
If Mode =2
No
Case 1: "#"
Case 2: "%"
Case 3 : any other data
Mode =1
Mode = 2
Set the PWM value
Var 1 = Desired Speed /10
Yes
Desired Speed = PWM value
Desired maximum RPM= maximum no-load rpm desired maximum speed
Desired Minimum RPM = Minimum no-load rpm at desired speed
if Desired minimum Rpm < actual rpm < Desired Maximum Rpm
Yes
No
Red Led ON

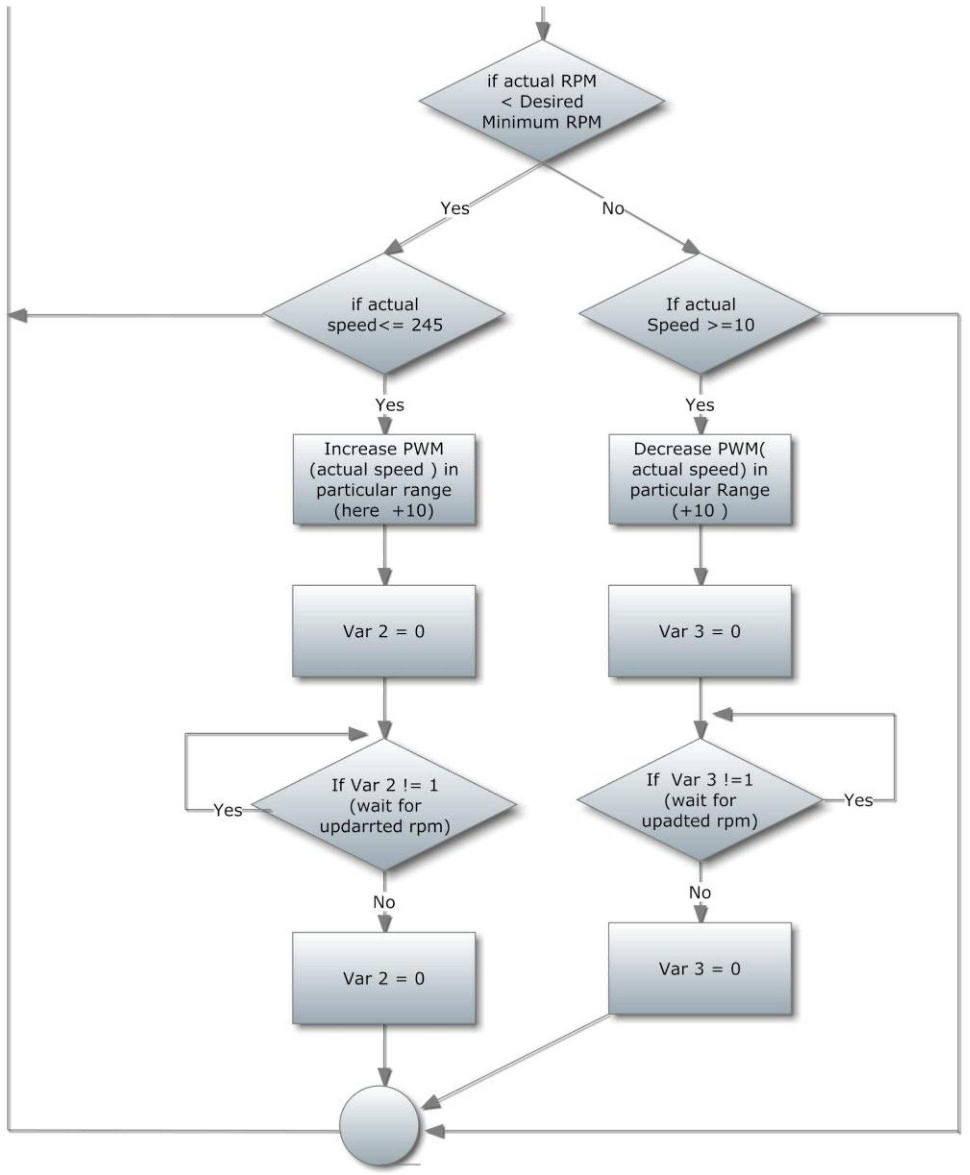

if actual RPM < Desired Minimum RPM
Yes
No
if actual speed<= 245
If actual Speed >=10
Yes
Yes
Increase PWM (actual speed) in particular range (here +10)
Decrease PWM(actual speed) in particular Range (+10)
Var 2 = 0
Var 3 = 0
If Var 2 != 1 (wait for updarrted rpm)
Yes
If Var 3 !=1 (wait for upadted rpm)
Yes
No
No
Var 2 = 0
Var 3 = 0

Figure 4.10 Flowchart of main program having free running and on-load modes

❖ *On –load Capturing Mode*:

This mode compares the actual RPM of motor the Desired RPM of motor. If actual speed is low than desired speed then the Arduino will automatically increase the PWM and hence RPM will be increased. Same for when actual speed is high than desired speed then the Arduino will automatically decrease the PWM and hence RPM will be decreased. This mode can be selected by on-load mode button in the software. The basic difference between this two differences is that mode 1 will not check that actual speed of rpm is same as desired speed or not.

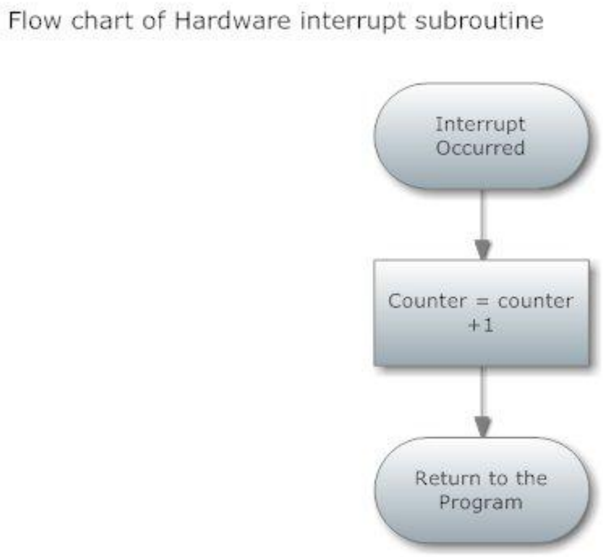


Figure 4.11 Flowchart of hardware interrupts subroutine

Above flow chart is for hardware interrupt subroutine, when interrupt generated the counter increase the count and return to the program.

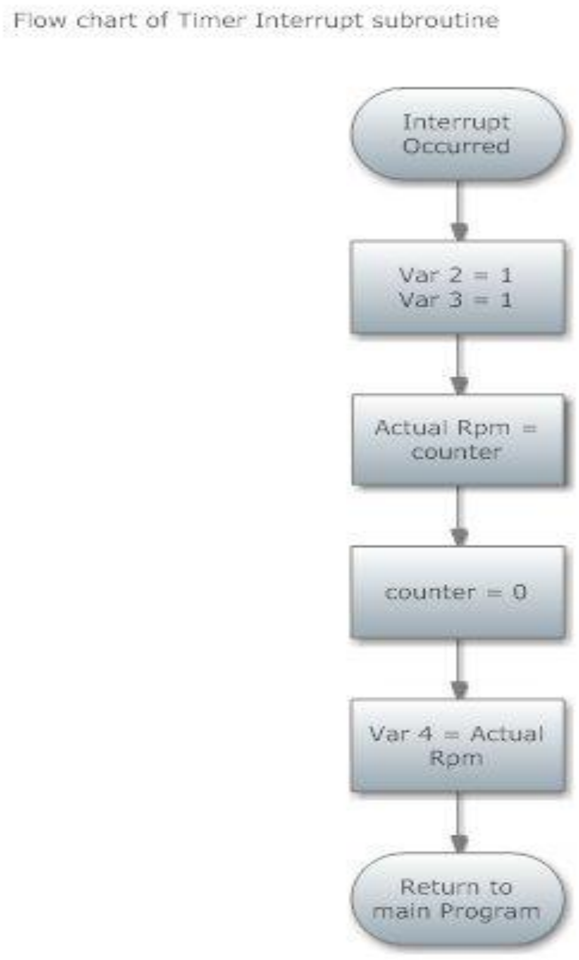


Figure 4.12 Flowchart of timer interrupt subroutine

The flow chart for Timer interrupt subroutine is shown as above, the timer sets the flag after each time loop. The program counter enters to the timer interrupt subroutine. Var 2 and Var 3 are set to one for future use, while counter value is copied to Actual rpm. Var 4 is used for backup of counter value.

4.3.1.2 Code

```
#include <TimerOne.h>
int maximumrpm[26]={0,0,11,42,74,103,128,148,164,177,
188,197,202,211,216,220,225,229,233,233,235,238,240,242,246,
255};
int minimumrpm[26]={0,0,0,11,43,75,104,129,149,165,178,
189,198,200,212,217,221,226,230,233,233,236,239,241,243,247};

int counter=0;
int counter1=0;
int counterflag=0;
int desiredrpm,actualrpm;
int desiredspeed,actualspeed=0;
int endflag=0;
int x[4]={0,0,0,0};
int a[4]={0,0,0,0};
int d,i,k,y;
int c=0;
int var5,var1=0;
int mindesiredrpm,maxdesiredrpm=0;
int endflag2=0;
int mode=1;

/* setup of input and output pins and interrupts */
void setup()
{
  pinMode(13,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(22,OUTPUT);
  pinMode(23,OUTPUT);
  pinMode(25,OUTPUT);
  digitalWrite(13,LOW);
  Serial.begin(9600);
  Timer1.initialize(240000);
  Timer1.attachInterrupt(alert);
  attachInterrupt(0,blockdetect,RISING);
  digitalWrite(13,HIGH);

}
```

```
/*main program starts here */

void loop()
{
  do
  {
      if(Serial.available())
      {
              Timer1.stop();
              d=Serial.read();
              if(d=='!')
              {
                  mode=1;
                  Timer1.initialize(2400000);
                  Timer1.attachInterrupt(alert);
                  digitalWrite(13,HIGH);
              }
              else if(d=='#')
              {
                  mode=2;
                  Timer1.initialize(2400000);
                  Timer1.attachInterrupt(alert);
                  digitalWrite(13,LOW);
              }
              else
              {
                  i=d-48;
                  k=0;
                  do
                  {
                    if(Serial.available())
                    {
                        x[i]=Serial.read();
                        if(x[i]=='%')
                        {
                          k=1;
                        }
                        a[i]=x[i]-48;
                        i--;
                     }
                  }while(k==0);

                   Timer1.initialize(2400000);
                  Timer1.attachInterrupt(alert);
                  y=d-48;
                  switch(y)
                  {
                    case 1: c=a[y];
                            break;
                    case 2: c=a[y]*10+a[y-1];
                            break;
                    case 3: c=a[y]*100+a[y-1]*10+a[y-2];
                            break;
                  }
                  digitalWrite(22,HIGH);
                  digitalWrite(23,LOW);
                  analogWrite(7,c);
                  desiredspeed=c;
                  var5=c;
           }
      }
      else
      {
            if(mode==2)
            {
                var1=var5/10;
                mindesiredrpm=minimumrpm[var1];
                maxdesiredrpm=maximumrpm[var1];
    if(actualrpm>=mindesiredrpm && actualrpm<=maxdesiredrpm)
                {
                    endflag2=1;
                    digitalWrite(52,HIGH);
                }
                else
                {
                   digitalWrite(52,LOW);
                   endflag2=0;
                   if(actualrpm<mindesiredrpm)
                   {
                         if(actualspeed<=245)
                         {
                           actualspeed=actualspeed+10;
                           analogWrite(7,actualspeed);
                         }
                         else
                         {
                           endflag2=1;
                         }
                   }
                   else
                   {
                         if(actualspeed>=10)
                         {
                           actualspeed=actualspeed-10;
                           analogWrite(7,actualspeed);
                         }
                         else
                         {
                           endflag2=1;
                         }
                   }
                   counterflag=0;
                   while(counterflag=0);
                   counterflag=0;
                }
            }
      }
  }while(1);
}

// hardware interrupt subroutine for slot detecting
void blockdetect()
{
  counter++;
}

/* timer interrupt subroutine for counting value of pulse for
predefined time */

void alert()
{
  counter1=counter;
  actualrpm=counter1;
  Serial.print(counter1);
  Serial.print('\n');
  counterflag=1;
  counter=0;
 }
```

4.3.2 Visualbasic 2010 code

This section describes coding of software PWM Controller. To develop software visual basic 2010 is used. The software has facility to send the PWM value at particular COM Port. The COM Port can also be able to select.

In starting of program the software detects all available COM Ports and display on the screen. User can select any COM Port, and baud rate from the list of baud rates. This software has facility to make hardware run in two different modes. In this project two modes are free-run mode and on-load condition.

Software has facility to display data which is sent by hardware to the computer. In this project hardware (Arduino) sends slots cuts by encoder wheel in 2.4 seconds. From that we can able to understand the mechanism of feedback to maintain the rpm of the motor.

Code:

```
Imports System
Imports System.ComponentModel
Imports System.Threading
Imports System.IO.Ports

Public Class mainwindow

    Dim cj As String
    Dim comi As String
    Dim comj As String
    Dim myport As Array
    Dim f As String
    Dim N As String
    Dim D As String
    Dim z As Integer
    Dim fp1, fp2 As String
    Delegate Sub settextcallback(ByVal [text] As String)

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        aboutwindow.Close()
        freerunbtn.Enabled = False
        serialrefresh()

End Sub

Private Sub TrackBar1_Scroll(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TrackBar1.Scroll

        cj = TrackBar1.Value()
        TextBox1.Text = cj
        D = cj.Length()
        If SerialPort1.IsOpen = True Then
            SerialPort1.Write(D)
            f = 0
            z = 0
            SerialPort1.Write(cj)
            SerialPort1.Write("%")
        End If
```

End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

```
        End
```

End Sub

Private Sub connectbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles connectbtn.Click

```
        connectbtn.Enabled = False
        SetTheSpeedToolStripMenuItem.Enabled = False
        disconnectbtn.Enabled = True
        refreshbtn.Enabled = False
        DisconnectToolStripMenuItem.Enabled = True
        modestatus.Text = "Free-runing Mode is activated"
        SerialPort1.PortName = newcomportlist.Text
        SerialPort1.BaudRate = baudratelist.Text
        SerialPort1.Parity = IO.Ports.Parity.None
        SerialPort1.StopBits = IO.Ports.StopBits.One
        SerialPort1.DataBits = 8
        SerialPort1.Open()
        serialstatus.Text = "Connected"
        cj = TrackBar1.Value()
        D = cj.Length()
        SerialPort1.Write(D)
        f = 0
        z = 0
        SerialPort1.Write(cj)
        SerialPort1.Write("%")
```

End Sub

Private Sub disconnectbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles disconnectbtn.Click

```
        SerialPort1.Close()
        connectbtn.Enabled = True
        SetTheSpeedToolStripMenuItem.Enabled = True
        disconnectbtn.Enabled = False
        refreshbtn.Enabled = True
        DisconnectToolStripMenuItem.Enabled = False
        serialstatus.Text = "Disconnected!"
        modestatus.Text = " "
```

End Sub

Private Sub serialport1_datareceived(ByVal sender As Object, ByVal e As System.IO.Ports.SerialDataReceivedEventArgs) Handles SerialPort1.DataReceived

```
        N = SerialPort1.ReadExisting()
        datareception(N)
```

End Sub

Private Sub COMPortToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles COMPortToolStripMenuItem.Click

```
        TextBox1.Select()
        TextBox1.SelectAll()
```

End Sub

Private Sub SetTheSpeedToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles SetTheSpeedToolStripMenuItem.Click

```
        connectbtn.Enabled = False
        SetTheSpeedToolStripMenuItem.Enabled = False
        disconnectbtn.Enabled = True
        DisconnectToolStripMenuItem.Enabled = True
        SerialPort1.PortName = newcomportlist.Text
        SerialPort1.BaudRate = baudratelist.Text
        SerialPort1.Parity = IO.Ports.Parity.None
        SerialPort1.StopBits = IO.Ports.StopBits.One
        SerialPort1.DataBits = 8
        SerialPort1.Open()
        serialstatus.Text = "Connected"
        cj = TrackBar1.Value()
        D = cj.Length()
        SerialPort1.Write(D)
        ' Do Until f = "$"
        ' Loop
        SerialPort1.Write(cj)
        SerialPort1.Write("%")
        refreshbtn.Enabled = False
```

End Sub

Private Sub DisconnectToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles DisconnectToolStripMenuItem.Click

```
        SerialPort1.Close()
        serialstatus.Text = "Disconnected!"
        BaudRateToolStripMenuItem.Enabled = True
        connectbtn.Enabled = True
        SetTheSpeedToolStripMenuItem.Enabled = True
        disconnectbtn.Enabled = False
        DisconnectToolStripMenuItem.Enabled = False
        refreshbtn.Enabled = True
```

End Sub

Private Sub ToolStripMenuItem2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ToolStripMenuItem2.Click

```
        If SerialPort1.IsOpen = False Then
            baudratelist.Text = 9600
        Else
             MessageBox.Show("Disconnect the hardware to
change Baudrate ")
        End If
```

End Sub

Private Sub ToolStripMenuItem3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ToolStripMenuItem3.Click

```
        If SerialPort1.IsOpen = False Then
            baudratelist.Text = 19200
        Else
            MessageBox.Show("Disconnect the hardware to change
Baudrate ")
        End If
```

End Sub

Private Sub ToolStripMenuItem4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ToolStripMenuItem4.Click

```
        If SerialPort1.IsOpen = False Then
            baudratelist.Text = 38400
        Else
            MessageBox.Show("Disconnect the hardware to change
Baudrate")
        End If
```

End Sub

Private Sub ToolStripMenuItem5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ToolStripMenuItem5.Click

```
       If SerialPort1.IsOpen = False Then
            baudratelist.Text = 57600
        Else
            MessageBox.Show("Disconnect the hardware to change
Baudrate")
        End If
```

End Sub

Private Sub ToolStripMenuItem6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ToolStripMenuItem6.Click

```
        If SerialPort1.IsOpen = False Then
            baudratelist.Text = 115200
        Else
            MessageBox.Show("Disconnect the hardware to change 
Baudrate")
        End If
```

End Sub

Private Sub ExitToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ExitToolStripMenuItem.Click

```
        SerialPort1.Close()
        End
```

End Sub

Private Sub AboutToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles AboutToolStripMenuItem.Click

```
      aboutwindow.Show()
```

End Sub

Private Sub datareception(ByVal [text] As String)

```
        If Me.receiveddata.InvokeRequired Then
            Dim x As New settextcallback(AddressOf 
datareception)
            Me.Invoke(x, New Object() {(text)})
        Else
            Me.receiveddata.Text &= [text]
        End If
```

End Sub

Private Sub received_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles receiveddata.TextChanged

```
        receiveddata.SelectionStart = receiveddata.TextLength
        receiveddata.ScrollToCaret()
```

End Sub

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles freerunbtn.Click

        If SerialPort1.IsOpen = True Then
            SerialPort1.Write("!")
            freerunbtn.Enabled = False
            onloadbtn.Enabled = True
            modestatus.Text = "Free-run mode is activated"
        End If

End Sub




Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles onloadbtn.Click

    If SerialPort1.IsOpen = True Then
       SerialPort1.Write("#")
       freerunbtn.Enabled = True
       onloadbtn.Enabled = False
       modestatus.Text = "On-load condition mode is activated"
    Else
       MessageBox.Show("Hardware is not found")
    End If

End Sub

Private Sub clrbtn_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles clrbtn.Click

    receiveddata.Clear()

End Sub

Private Sub Button2_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles refreshbtn.Click

        If SerialPort1.IsOpen = False Then
            newcomportlist.Items.Clear()
            baudratelist.Items.Clear()
            If SerialPort1.IsOpen = True Then
                serialrefresh()
            End If
        End If
End Sub

Private Sub serialrefresh()

        myport = IO.Ports.SerialPort.GetPortNames()
        For i = 0 To UBound(myport)
            For j = i + 1 To UBound(myport)
                comi = myport(i)
                comj = myport(j)
                If comj < comi Then
                    myport(j) = comi
                    myport(i) = comj
                End If
            Next
        Next
        For i = 0 To UBound(myport)
            newcomportlist.Items.Add(myport(i))
        Next
        baudratelist.Items.Add(9600)
        baudratelist.Items.Add(19200)
        baudratelist.Items.Add(38400)
        baudratelist.Items.Add(57600)
        baudratelist.Items.Add(115200)
        baudratelist.Text = baudratelist.Items.Item(0)
        newcomportlist.Text = newcomportlist.Items.Item(0)
        disconnectbtn.Enabled = False
        DisconnectToolStripMenuItem.Enabled = False
        TrackBar1.Value = "0"
        TextBox1.Text = TrackBar1.Value
        freerunbtn.Enabled = False
        onloadbtn.Enabled = True
        serialstatus.Text = "Disconnected!"
```

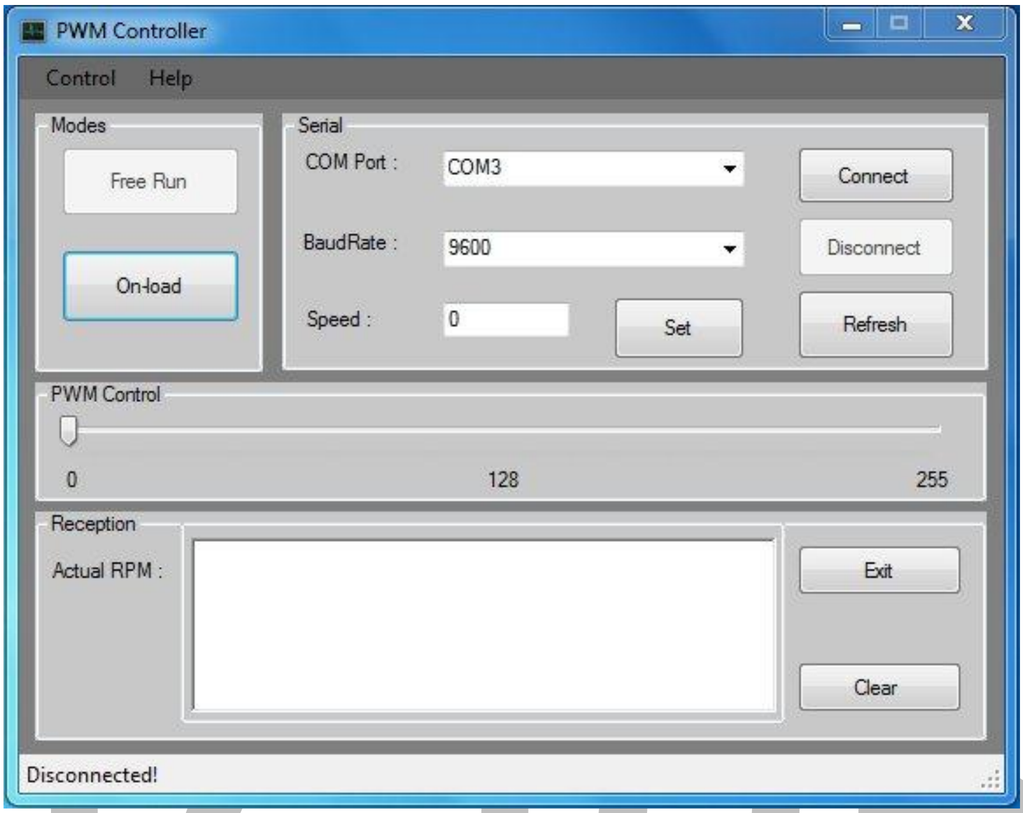
End Sub

Private Sub setbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles setbtn.Click

```
       fp1 = TextBox1.Text
        If fp1 >= 0 And fp1 <= 255 Then
            If SerialPort1.IsOpen = True Then
                TrackBar1.Value = TextBox1.Text
                cj = TrackBar1.Value()
                D = cj.Length()
                SerialPort1.Write(D)
                SerialPort1.Write(cj)
                SerialPort1.Write("%")
            End If
        Else
            MessageBox.Show(" Value of PWM must be between 0
and 255")
        End If
```

End Sub

End Class



COM Port : COM3

BaudRate : 9600

Speed : 0

Actual RPM :

Disconnected!

Figure 4.13 Screenshot of PWM Controller software build by using Visualbasic 2010

The screen shot of the developed software is shown in above figure having facility to display received data, instant change in speed, and Speed variation by sliding track bar. Two buttons free run and no-load condition are used for mode selection. Refresh button is used to refresh the whole software by making speed equal to zero and re-detect all COM Ports and update them on the display.

CHAPTER 5

SIMULATION RESULTS

5.1 Introduction

This chapter will discuss about result of project. The result is considered on the basis of the application of load on the DC motor. We will discuss results for No-Load condition and On-load conditions.

The procedure for taking result for various inputs is described below:

- Start the PWMController software, which is made by using Visual basic 2010.
- Set proper baud rate and COM Port for Serial communication between PC and Arduino Board.
- Set the proper speed of DC motor using PWMController software.
- Do not apply any load for No-Load conditions, while for On-Load condition apply proper load.
- Read the pulses generated by Schmitt trigger circuit and calculate Rpm.
- Get the difference between desired Rpm and Actual Rpm.

5.2 Results on No-Load Condition

Here generated Pulses for Various Speed and Rpm is shown in below table:

The table represents the values of pulses generated for particular rang of PWM value. This value is taken as reference to maintain constant speed. It is considered as Desired RPM and for maintaining constant speed actual RPM is compared with this desired RPMs.

Speed Range (PWM)	**Pulses generated**	
	Min No of pulses generated	**Max No of pulses generated**
0-9	0	0
10-19	0	0
20-29	0	8
30-39	0	34
40-49	35	65
50-59	66	97
60-69	97	123
70-79	124	145
80-89	147	163
90-99	164	176
100-109	176	187
110-119	188	197

120-129	198	204
130-139	205	211
140-149	212	216
150-159	217	221
160-169	221	226
170-179	227	229
180-189	229	233
190-199	233	233
200-209	233	233
210-219	234	239
220-229	240	242
230-239	242	243
240-249	243	246
250-255	247	255

Table 5.1 Result of generated pulses for different PWM (Speed) value for no-load condition

5.3 Results on On-load Condition

The section describes results of on-load condition. Here as shown in table the desried speed is given as 105, for the free running condition the rpm is vary in range of 176 to 187. The actual rpm is 184.

Now if we applied the load the actual speed will be reduced to the 51, In on-load condition mode the Arduino tries to catch the desired speed. As shown in figure the RPM will increase from 51 to 183 continuously.

Input speed (PWM Value)	105
Desired RPM	176-187
Actual RPM on No-load condition (in free running mode)	184
Actual RPM on load condition (in free running mode)	51
In On-load Mode : Variation in RPM to catch the Desired RPM When load is applied the rpm goes to the 51 then in on-load mode it increasing continuously to reach to the 184.	64
	87
	104
	119
	135
	145
	152
	162
	170
	177

	178
	177
	186
	186
	180
	182
	182
	183
In On-load Mode : Variation in RPM to catch the Desired RPM. When load is removed from the motor the rpm goes to the 51 then in on-load mode it increasing continuously to reach to the 184 value.	220
	233
	233
	224
	220
	214
	209
	202
	193
	183
	183
	183
	182

Table 5.2 Result of Actual RPM on Applied-load condition for desired speed of 105 (PWM)

After reaching to the 184 RPM, if we remove the load then the actual rpm will suddenly increase up to 233 and then it will reduce and maintain to the 182.

So, from the above reading we can conclude that by applying load or removing load the feedback will try to maintain rpm at the desired RPM. Thus, we can get constant RPM over variation of the load.