Speed Control of DC motor with PI control

Project report submitted in partial fulfillment of the degree of
Electronics and Communication Engineering

For the course

Design Lab 1

Asit Gautam
19UEC112

Department of Electronics and Communication Engineering
The LNM Institute of Information Technology, Jaipur

9 November 2020

Abstract

Direct current (DC) motor plays an important role in industries. The speed of DC motor is important in the process control industries so we have to control the speed of DC motor. For this speed control proportional-integral-derivative (PID) and proportional-integral(PI) controllers are used. Here PI controller is employed to control DC motor speed and proteus is used for simulation. This report contains two parts,first part contains PI controller implemented using Opamp and second using arduino. Due to lack of information to find transfer function ,hit and trial method is used to find PI parameters.

1 MOTIVATION

Basically in industries we need everything to be controlled and motor is a basic component for many industrial machines.In this modern industrial age, there is hardly any industrial application in which DC motors are not being used . This is so because of ease of control, low cost maintenance especially of brushless DC motor type, low price, and ruggedness of DC motor over a wide range of applications. Some industrial applications, which are worth mentioning, in which DC motors are being used widely are machine tools, paper mills, textile industry, electric traction, and robotics.
Now due to dc motors industrial importance ,controlling its speed also becomes important.
We can control its speed using 1) armature resistance 2) field flux 3) armature voltage.The flexibility in controller design of DC motors is due to the fact that armature winding and field winding could be controlled separately. So in this report we will see one of the many methods to control speed of motor.

2 PROPOSED SOLUTION

The best-known controllers used in industrial control processes are proportional-integral-derivative (PID) and proportional-integral (PI) controllers because of their simple structure and robust performance in a wide range of operating conditions.So to control speed of dc motor we are going to use PI controller.A P.I Controller is a feedback control loop that calculates an error signal by taking the difference between the output of a system. We are having two implementations of PI controller first using analog components like opamp and second using arduino, we will be going into details of both circuits separately.

2.1 ANALOG CIRCUIT

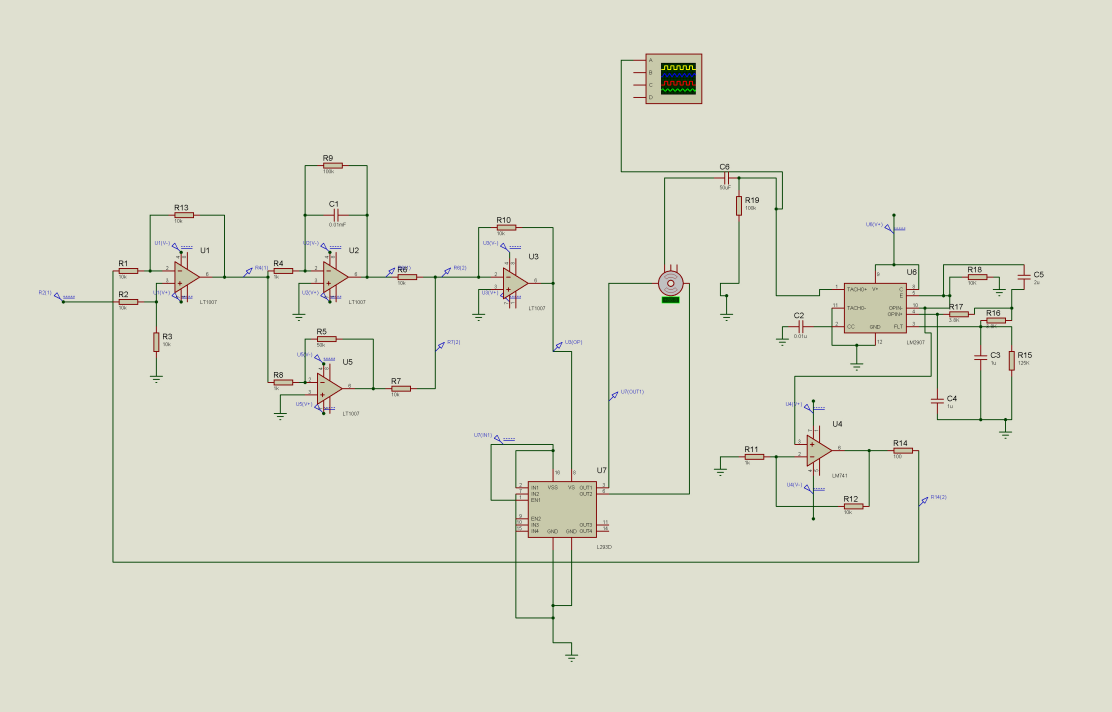

Figure 2.1: Analog Circuit

Components -

- LT1007
- Motor encoder
- Resistors
- Capacitors
- Motor Driver(L239D)
- LM2907
- LM741

LT1007

It was used in place of the most common opamp 'LM741' for formation of PI controller due to its feature of low noise,high speed precision.

Features -

- Guaranteed 4.5nV/$\sqrt{Hz}$ 10Hz Noise
- Guaranteed 3.8nV/$\sqrt{Hz}$ 1kHz Noise
- 0.1Hz to 10Hz Noise, 60nVP-P Typical
- Guaranteed 7 Million Min Voltage Gain, RL = 2k
- Guaranteed 3 Million Min Voltage Gain, RL = 600Ω
- Guaranteed 25µV Max Offset Voltage
- Guaranteed 0.6µV/°C Max Drift with Temperature
- Guaranteed 11V/µs Min Slew Rate (LT1037)
- Guaranteed 117dB Min CMRR

Motor Encoder

The MOTOR-ENCODER model tracks the angular position value at the output of the DC motor model schematic, and generates the appropriate output pulses. This model is parameterized in terms of the number of pulses per revolution (PPR), and uses some sophisticated interpolation techniques to generate the digital pulses at the correct times without unduly limiting the speed of the simulation.

Features -

- Nominal Volatage-12V
- Coil Resistance-12
- Coil Inductance-100mH
- Zero Load RPM-360
- Load Max Torque-50
- Effective mass-10g
- Pulses per Revolution-24

Motor Driver

Features -

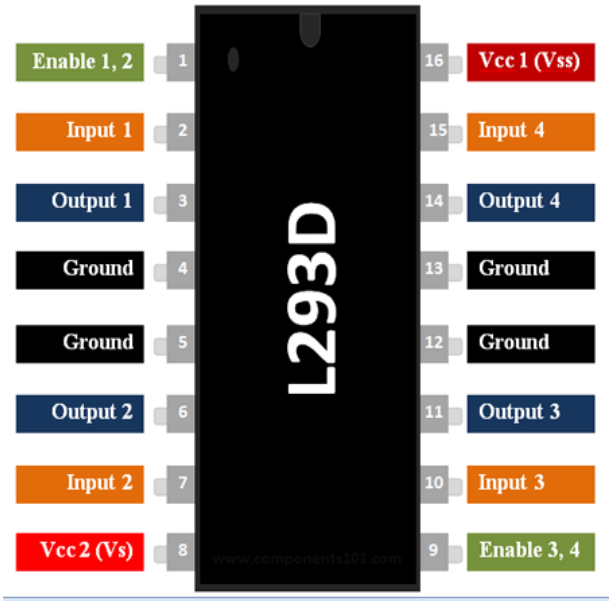

Figure 2.2

- Can be used to run Two DC motors with the same IC.
- Speed and Direction control is possible
- Motor voltage Vcc2 (Vs): 4.5V to 36V
- Maximum Peak motor current: 1.2A
- Maximum Continuous Motor Current: 600mA
- Supply Voltage to Vcc1(vss): 4.5V to 7V
- Transition time: 300ns (at 5Vand 24V)
- Automatic Thermal shutdown is available
- Available in 16-pin DIP, TSSOP, SOIC packages

LM2907

The LM2907 device is monolithic frequency-to-voltage converter with a high gain op amp designed to operate a relay, lamp, or other load when the input frequency reaches or exceeds a selected rate.The op amp is fully compatible with the tachometer and has a floating transistor

as its output. This feature allows either a ground or supply referred load of up to 50 mA. The collector may be taken above VCC up to a maximum VCE of 28 V.
Features -

- Ground Referenced Tachometer Input Interfaces
- Directly With Variable Reluctance Magnetic Pickups
- Op Amp Has Floating Transistor Output
- 50-mA Sink or Source to Operate Relays,Solenoids, Meters, or LEDs
- Frequency Doubling For Low Ripple
- Tachometer Has Built-In Hysteresis With Either Differential Input or Ground Referenced Input
- Ground-Referenced Tachometer is Fully Protected From Damage Due to Swings Above VCC and Below Ground
- Easy to Use; VOUT = fIN × VCC × R1 × C1

LM741

The LM741 series are general-purpose operational amplifiers
Features -

- Overload Protection on the Input and Output
- No Latch-Up When the Common-Mode Range is exceeded

Explaination

Lets see circuit in two parts, first comes the PI controller part-

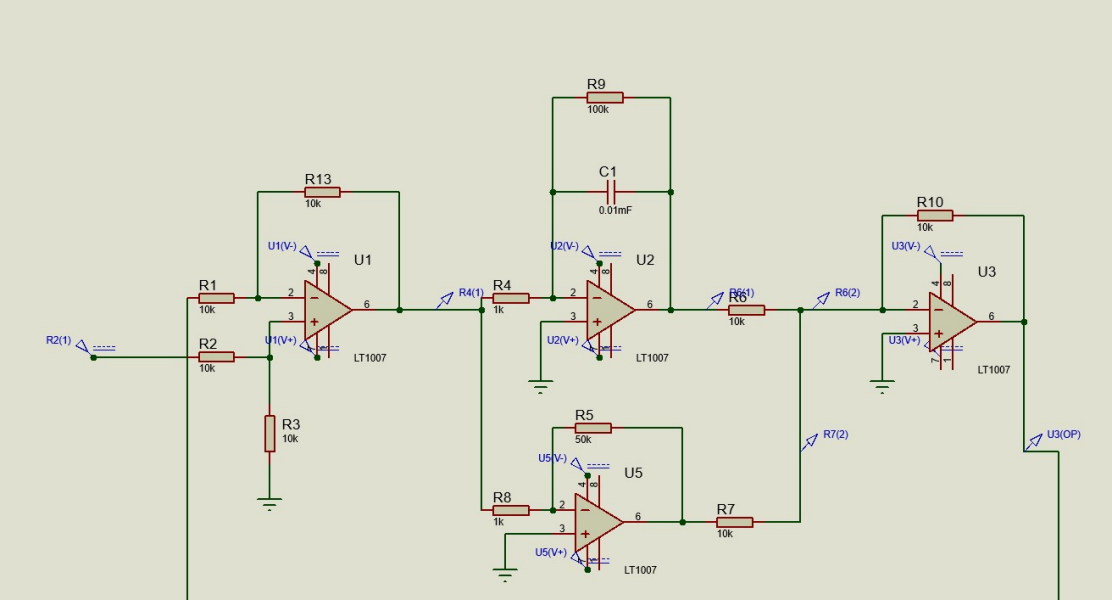

Figure 2.3: PI controller

U1 is the differential amplifier or subtractor which gives the error betweeen the set voltage and the voltage we are getting after f/v conversion,this error voltage then is given to integrator(U2) and amplifier(U5),the gain of amplifier is actually kp,i.e proportionality constant.So we can define the output of both opamp as-

P Amplifier: Vo = (R5 / R8) Verr

I Integrator: Vo = 1/(R9C1) $\int Verr\,dt$

Then there is summer(U3) which adds the two P,I terms.Now this summed voltage is given to motor driver which further drives the motor.Motor encoder depending on rpm gives the square output pulse.

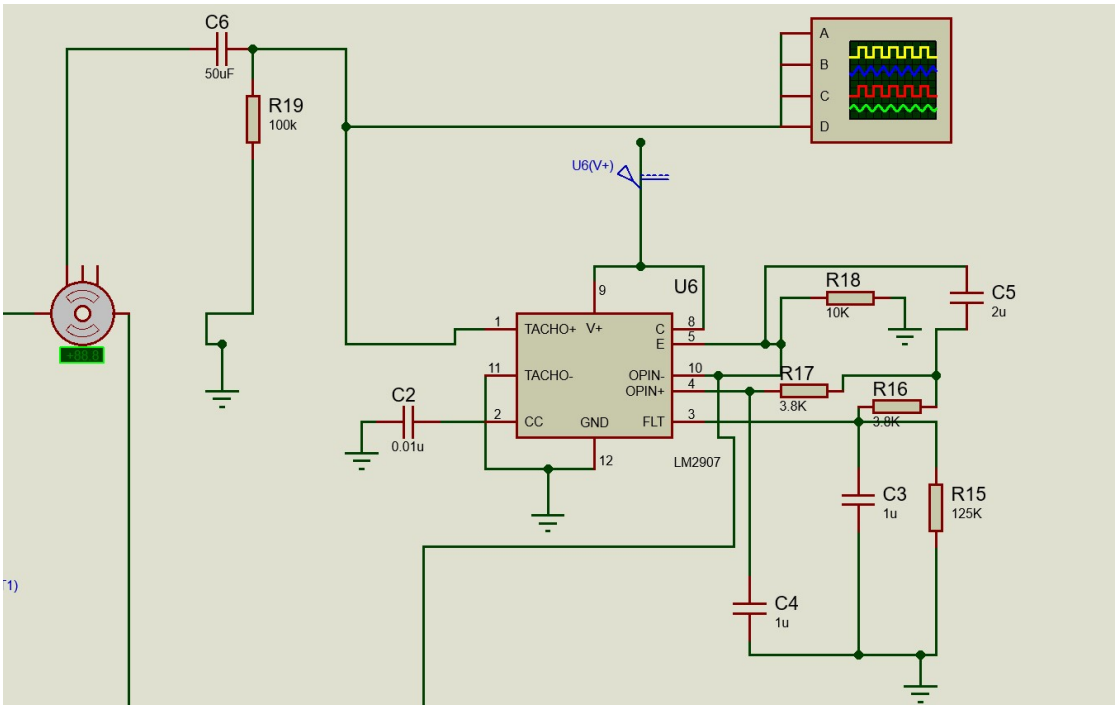

Figure 2.4: F/V converter

Now comes the F/V converter part.LM2907 receives the square pulse from the motor encoder and according to frequency converts it into voltage.High pass filter is used brfore sending the pulse to LM2907 as motor doesn't starts revolving instantaneously,so to cut off that starting straight constant dc part of square wave high pass filter was used.

Internal working of LM2907(content source-[1])-

The internal schematic of LM2907N is shown in figure 2.5. The input stage of LM2907N is a charge pump where the input frequency is converted to a dc voltage. To do this it requires one timing capacitor C2, one output resistor R15, and one integrating or filter capacitorC6. When the input stage changes state (due to a suitable zero crossing or differential voltage on the input) the timing capacitor is either charged or discharged linearly between two voltages whose difference is 1/2 Vcc. Then in one half cycle of the input frequency or a time equal to 1/2 Fin the change in charge on the timing capacitor is equal to 1/2Vcc $*$ C2.
Then the average amount of current pumped into or out of the capacitor is,

Q/T= Ic(avg) = C2 $*$ 1/2Vcc $*$ 2Fin = Vcc $*$ C2 $*$ Fin

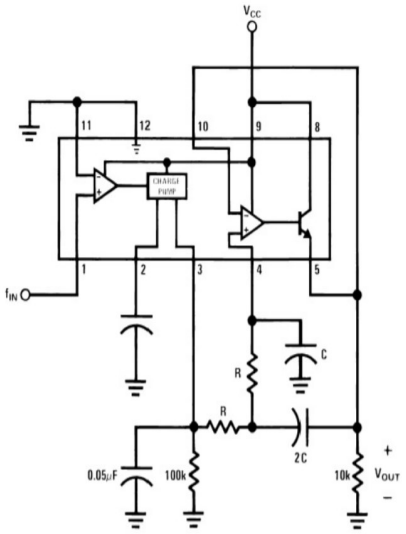

Figure 2.5: Internal schematic

The output circuit mirrors this current very accurately into the load resistor R1, connected to ground, such that if the pulses of current are integrated with a filter capacitor, then Vo = Ic ∗ R1 and the total conversion equation becomes,

Vo = Vcc ∗ Fin ∗ C2 ∗ R15 ∗ K

where K is gain constant, typically K=1. The size of C6 is dependent only on the amount of ripple voltage allowable and the required response time.

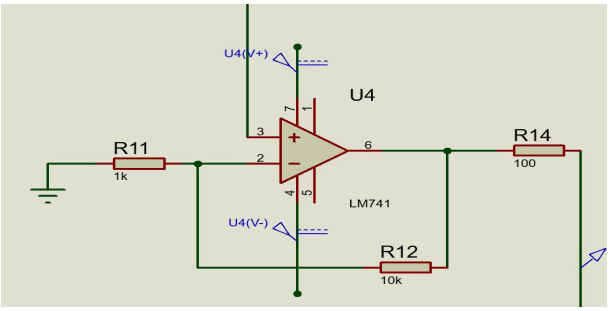

Figure 2.6: Amplifier

Now the voltage from converter is supplied to the amplifier(U4) as the output voltage is very less,however we could make change in the above circuit of LM2907 also to increase the output voltage.Gain of amplifier is 10.This final voltage is then supplied to the subtractor which gives the error or difference between set voltage and this output voltage as feedback to PI controller ,and this process keeps on repeating until error doesn't becomes zero.

However this circuit doesn't works according to the expectations.Problem occurred-Both PI controller and frequency to voltage converter are working individually fine but when we combine these two, things doesn't work,it seems PI controller is not working on feedback and so the voltage at the end of PI controller(the voltage being fed to the motor) is coming constant(and very less).

2.2 DIGITAL CIRCUIT

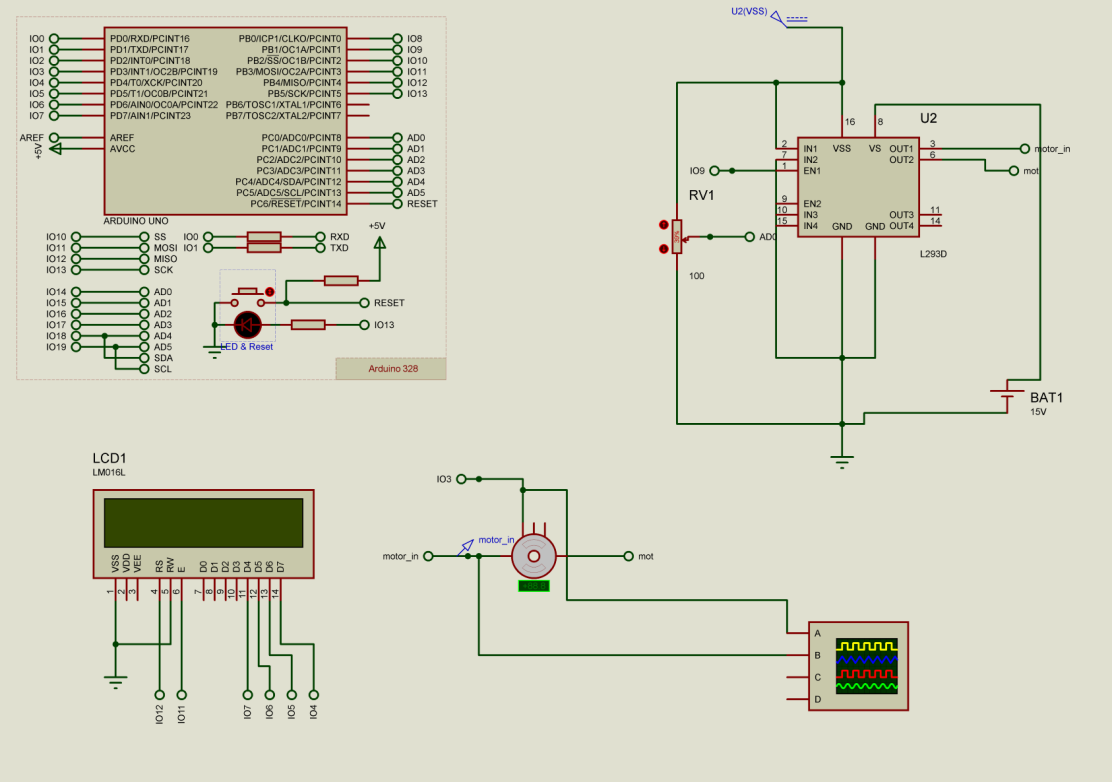

Figure 2.7: Digital Circuit

Components -

- Arduino UNO
- 16 X 2 LED
- Motor Encoder
- Motor Driver(L239D)

Arduiuno UNO

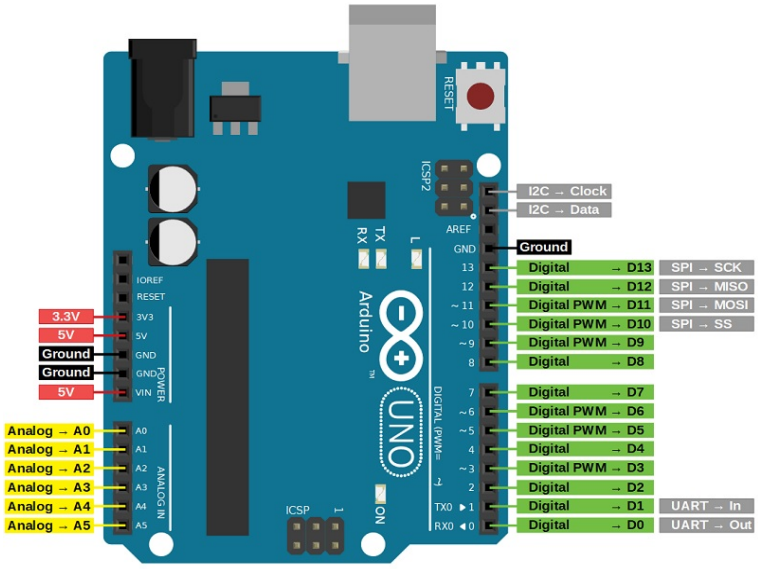

Figure 2.8: Arduino UNO

Arduino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button.The Arduino Uno ATmega328 also offers UART TTL-serial communication, and it is accessible on digital pins like TX (1) and RX (0).
Features -

- The operating voltage is 5V
- The recommended input voltage will range from 7v to 12V
- The input voltage ranges from 6v to 20V
- Digital input/output pins are 14
- Analog i/p pins are 6
- DC Current for each input/output pin is 40 mA
- DC Current for 3.3V Pin is 50 mA
- Flash Memory is 32 KB
- SRAM is 2 KB

- EEPROM is 1 KB
- CLK Speed is 16 MHz

In this project our main focus will be one time function so lets see timers in arduino.A timer or to be more precise a timer / counter is a piece of hardware builtin the Arduino controller (other controllers have timer hardware, too). It is like a clock, and can be used to measure time events.The Arduino Uno has 3 timers: Timer0, Timer1 and Timer2. Timer0 is a 8bit timer.In the Arduino world timer0 is been used for the timer functions, like delay(), millis() and micros(). Timer1 is a 16bit timer.In the Arduino world the Servo 1.1k uses timer1 on Arduino Uno.Timer2 is a 8bit timer like timer0. In the Arduino work the tone() function uses timer2.We will be using millis() and micros() so we only need to know about Timer0.The Arduino clock runs at 16MHz, this is the fastest speed that the timers can increment their counters. At 16MHz each tick of the counter represents 1/16,000,000 of a second (63ns), so a counter will take 10/16,000,000 seconds to reach a value of 9 (counters are 0 indexed), and 100/16,000,000 seconds to reach a value of 99. Timer0 is an 8-bit that counts from 0 to 255 and generates an interrupt whenever it overflows. It uses a clock divisor of 64 by default to give us an interrupt rate of 976.5625 Hz (close enough to a 1KHz for our purposes) hence 1 milli second.

16 X 2 LED

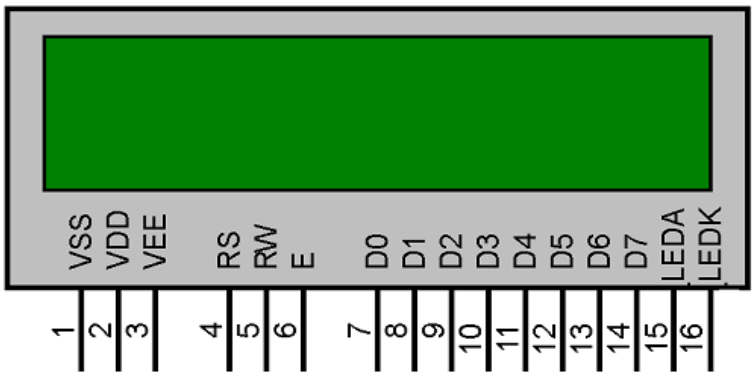

Figure 2.9: LED 16 X 2

This has 16 pins and can be operated in 4-bit mode (using only 4 data lines) or 8-bit mode (using all 8 data lines). Here we are using the LCD module in 4-bit mode. To facilitate communication between Arduino and LCD module, we make use of a built in library in Arduino <LiquidCrystal.h> – which is written for LCD modules making use of the Hitachi HD44780

chipset (or a compatible chipset). This library can handle both 4 bit mode and 8 bit mode wiring of LCD.We will be using 8 pins.Pin 1 and pin 5 will be grounded.Pin 4 will be connected so that when we feed an input to the data lines (D4 to D7), this input can be treated as data to display on LCD screen or as command depending on input.Other 4 pins(D4 to D7) are simple data pins to feed the data to LCD.[2]

Code-

```
#include<LiquidCrystal.h>

LiquidCrystallcd(12,11,7,6,5,4);
float t1=0,t2=0,t3=0,t4=0,sum=0,In=0.0,rpm,des_rpm;
float kp=1,ki=2;

voidsetup()
{
pinMode(9,OUTPUT);
pinMode(14,INPUT);
pinMode(3,INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(3),ON,RISING);
lcd.begin(16,2);
}

voidloop()
{
des_rpm=analogRead(14)/5.0;
float error;
t3=micros()/1000;
lcd.setCursor(0,0);
lcd.print("DES_RPM-");
lcd.setCursor(11,0);
lcd.print(des_rpm);
lcd.setCursor(0,1);
lcd.print("RPM-");
```

```
lcd.setCursor(6,1);
lcd.print(rpm);
floatdt=(float)(t3-t4);
error=des_rpm-rpm;
In=In+error*dt/1000;
sum=(kp*error+ki*In);
if(sum>255)
sum=255;
if(sum<0)
sum=0;
analogWrite(9,sum);
t4=t3;
}

voidON()
{
t1=micros();
rpm=60000000/((t1-t2)*24);
t2=t1;
}
```

Explaination -

We provide desired rpm by changing voltage value(0 to 5V) using potentiometer then according to the desired value PI controller algo works on the error(difference between desired rpm and current rpm) and provide us the desired rpm.

Analog pin 14 is read and divided by 5 so that desired rpm value can only vary between 0-205 as maximum rpm of motor with 15V battery is 210rpm.To find rpm of the motor we will use interrupt function; we are using the fastest method to calculate rpm that is finding the time between two rising edges of square wave coming from encoder and then multiplying it by 24 because 24 pulses means one revolution as pulses are generated whenever light passes through one of the 24 slots present in the rotating code wheel of encoder.

After calculation of rpm PI controller does its work,firstly error is calculated and then integral part and proportional part is calculated depending on parameters kp and ki(which can be changed later on to obtain observations),their sum is provided to the enable pin of motor driver as duty cycle of pwm wave(ranging from 0 to 255).This process keeps on repeating un-

til error becomes 0 ideally or negligible,therfore proportional part becomes zero and integral part becomes constant.
So speed of motor is controlled by changing the duty cycle of pwm as the power applied to the motor can be controlled by varying the width of these applied pulses and thereby varying the average DC voltage applied to the motors terminals. By changing or modulating the timing of these pulses the speed of the motor can be controlled, ie, the longer the pulse is “ON”, the faster the motor will rotate and likewise, the shorter the pulse is “ON” the slower the motor will rotate.

3 Results

Changing values of kp and ki gives us different results,all the results are summarised in below table.

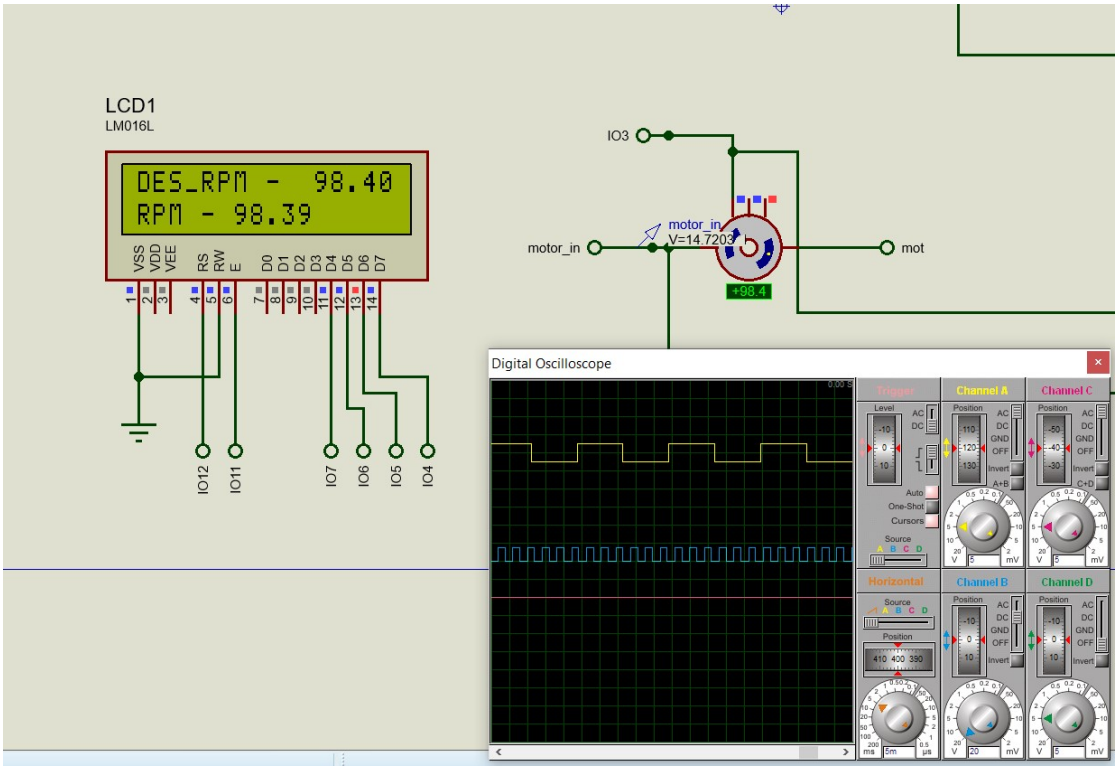

Figure 3.1

kp	ki	overshoot(%)	steady state fluctuation	peak_rpm	rising time(sec)	settling time(sec)
1	1	0.04	81.90−82.04	82.04	16	–
1	5	14.63	81.92−82.04	94	0.8	2.8
1	20	58.78	81.968−82.026	130.2	0.4	4.2
5	1	0.04	81.96−82.04	82.04	12	–
5	5	4.87	81.98−82.04	86	4.2	0.46
5	20	36.9	81.98−82.028	112.29	0.35	1.9

Table 3.1: des_rpm=82

With increase in ki overshoot percentage increases but steady state error decreases,settling time also increases,and with increase in kp rising time decreases so steady state is reached fast. If kp is increased more then the results are not clear as the rpm increases very fast in starting but then it takes a large amount of time to reach the desired rpm.So the best values of kp and ki comes out to be 1.
This behaviour is because kp determines how much change the output will make due to a change in error. This mainly corrects the output based on upsets as they happen.ki determines how much to change the output over time due to the error (regardless of the direction of movement of the error). This brings a stable measured value that is off set point toward the set point. When the measure value is approaching the set point, proportional and integral work in opposite directions to cause the measured value of a properly tuned loop to get to the set point quickly without excessively overshooting.For a typical reverse-acting loop, the proportional will try to close the output as the measured value rises toward the set point, but the integral will try to open the output because measured value is below set point. As the measured value gets closer to the set point, integral action decreases resulting in the measured value smoothly decelerating into the set point.[3]

4 Conclusion and future work

So above digital circuit was able to give us satisfactory results.
This was a hit and trial method to find appropriate so for future work, transfer function of motor can be calculated using open loop response and then using methods like Ziegler-Nichols we can tune PI controller parameters kp and ki to find a pricise value of kp and ki which gets us best results to control speed of motor or to get desired rpm.

REFERENCES

[1] Mahesh Bhaganagare Nikunj A. Bhagat. Dc motor speed control using pid controllers, 2009. http://bhagatnikunj.weebly.com/uploads/1/0/4/6/10469153/esd_report.pdf.

[2] Interfacing lcd to arduino. https://www.circuitstoday.com/interfacing-lcd-to-arduino.

[3] How to tune a pid loop. https://www.crossco.com/resources/technical/how-to-tune-pid-loops/.