Experiment No 1

Position Control of DC Motor

AIM

To Design and implement an embedded (PID) feedback controller using Arduino Mega, which uses the motor driver IC (L293D) to drive the DC motor.

OBJECTIVE

a) To rotate the dc motor by an angle of 180 degrees from any given point.
b) To ensure that the task is constrained by the design specifications such as 0.5 second rise time, 1 second's settling time and 10% overshoot.

BLOCK DIAGRAM

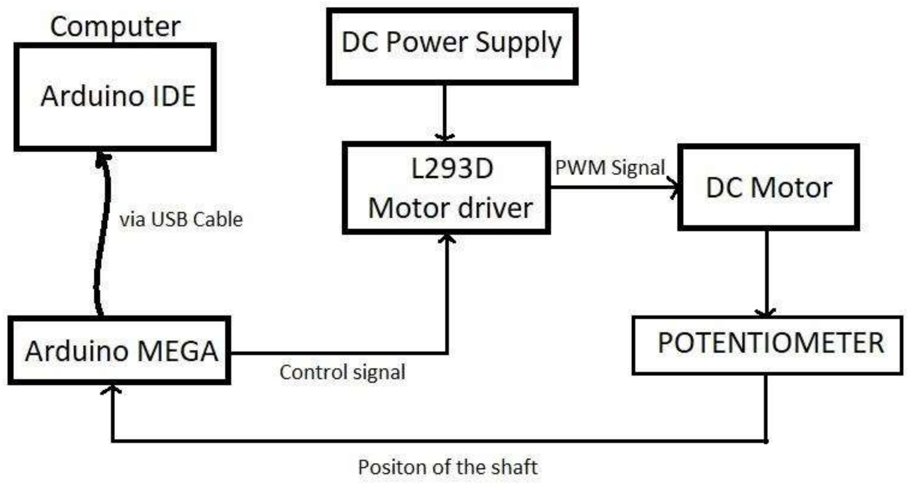


Fig 1: Block diagram for DC motor position control

EQUIPMENTS USED

a) DC MOTOR

The DC motor used in this experiment has a rated RPM of 6000 at 12V, which has been geared down to 65 RPM. The shaft of the motor is connected to a rotary potentiometer, which records the position and gives a feedback voltage between 0-5 V. The potentiometer has a small window of nonlinearity, where the output exhibits random values.

The shaft is also connected to a disc which has degrees printed from 0 to 360, and is concentric with the shaft.

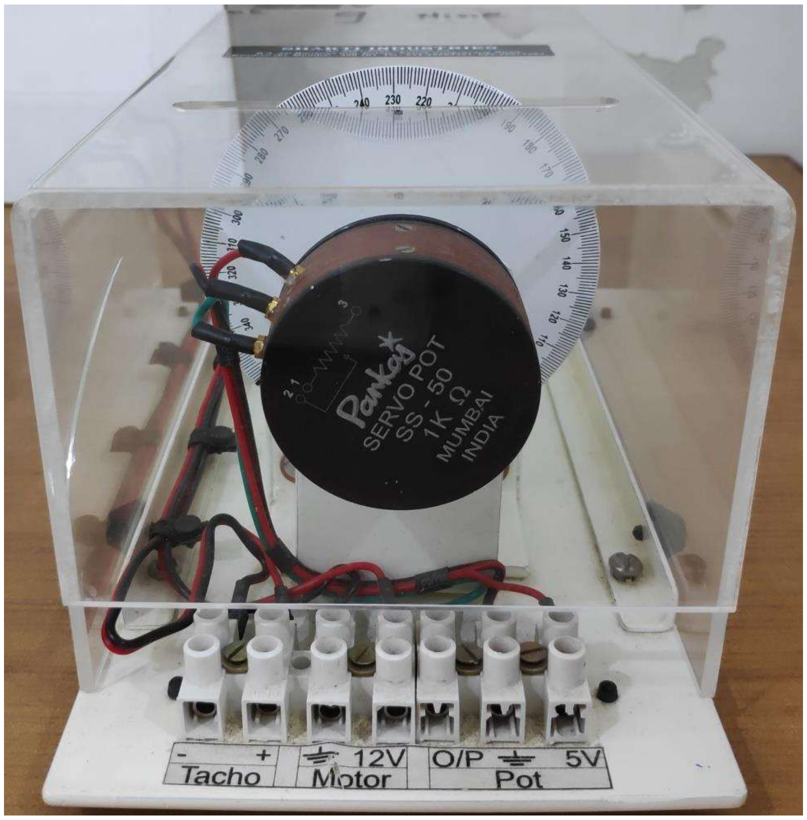


Fig 2: DC motor front view

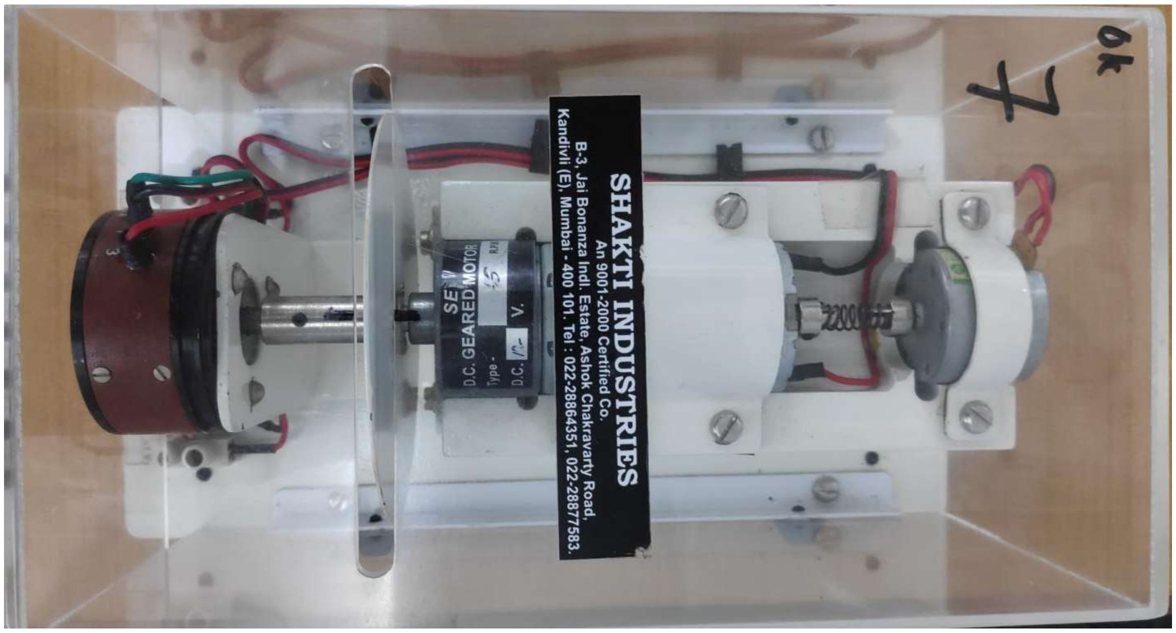
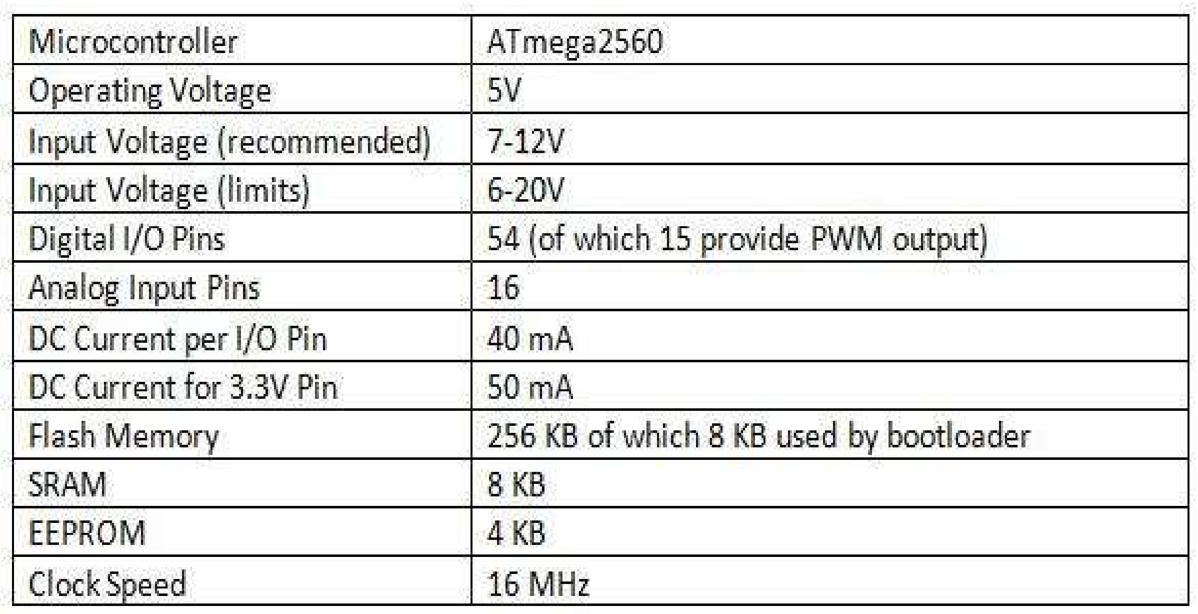

SHAKTI INDUSTRIES
An 9001-2000 Certified Co.
B-3, Jai Bonanza Indl. Estate, Ashok Chakravarty Road,
Kandivli (E), Mumbai - 400 101. Tel : 022-28864351, 022-28877583.
D.C. GEARED MOTOR
ok
7

Fig 3: DC motor top view

b) ARDUINO MEGA 2560

The technical specifications of the microcontroller used are:



Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 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 bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Fig 4: Arduino Mega specifications

The pin out diagram for the same is:

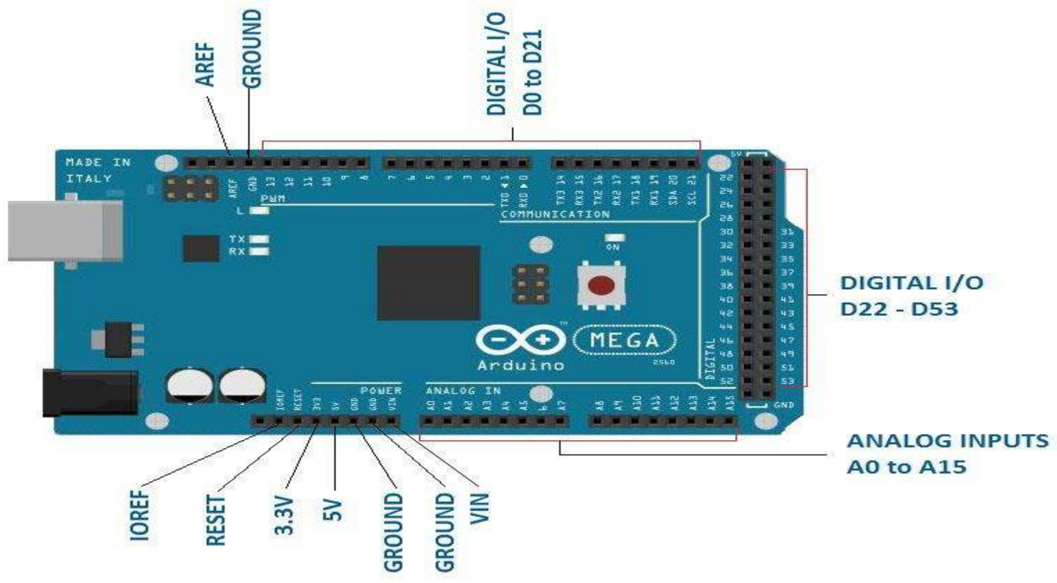


Fig 5: Arduino Mega pin out diagram

c) L293D (Motor driver IC)

The motor driver is used for controlling the speed and direction of the motor. The pin out diagram is as follows:

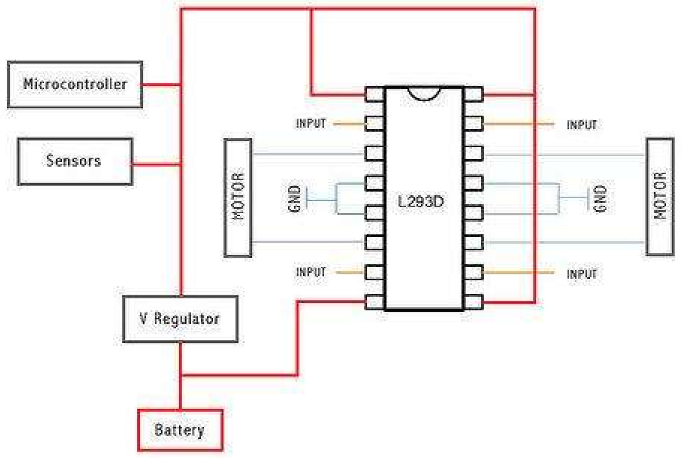


Fig 6: L293D pin out diagram

The link to the datasheet for the same is attached in the appendix.

MATERIALS REQUIRED

Jumper wires, single stranded wires, bread board, screw driver and wire stripper.

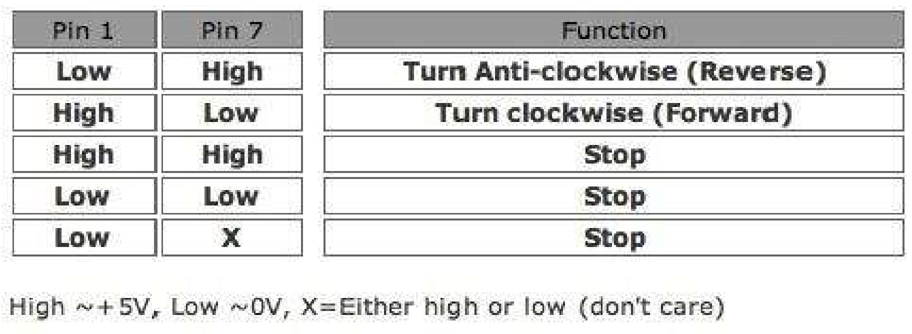
PROCEDURE

1) The potentiometer of the DC motor is powered from a 5 V supply. The output is connected to the analog pin of the Arduino, where the ADC has 10 bit resolution.

2) The disc connected to the shaft is manually rotated and the relation between the potentiometer feedback values and degrees is noted. Which is further used to check the linear region of the potentiometer.

3) The digital PWM pins of the Arduino are connected to the control pins of the motor driver (pins 1 and 7). They are supplied with PWM signals (whose duty cycle determines the speed of the motor).

The direction control logic for the motor is:



Pin 1	Pin 7	Function
Low	High	Turn Anti-clockwise (Reverse)
High	Low	Turn clockwise (Forward)
High	High	Stop
Low	Low	Stop
Low	X	Stop

High ~+5V, Low ~0V, X=Either high or low (don't care)

Fig 7: Motor driver control logic

4) A mapping function is used in the program to convert the feedback values to their corresponding values in degrees.

5) The PID control logic is made with the error as the difference between the initial position and the desired angle, which is separated by 180 degrees initially.

6) The motor driver is powered from a 12 V supply at pin 8. The DC motor’s terminals are connected to pins 3 and 6 of the motor driver. With pins 4 and 5 grounded.

7) The PID output is scaled to a value which fits the 8 bit DAC, so that the speed can be controlled accordingly. This scaled value determines the duty cycle of the PWM signals fed to the motor driver’s control pins.

8) The error is put through an IF statement, where the direction of rotation is determined by the sign of the error, also the motor must not rotate through the nonlinear region of the potentiometer (else it will corrupt the PID output).

9) The motor stops when the error is between the threshold and the motor stops at the desired angle.

10) The PID gain values are tuned until the required constraints in the problem statement are met.

11) The graph of angle (degrees) vs time (seconds) is plotted with the values in the serial monitor of the Arduino IDE.

12) The plot is examined to make sure that the position control is carried out with the design specifications met simultaneously.

APPENDIX

CODE

```
*************************************************************************

float kp=35,ki=0.001,kd=30,error,lasterror=0,kP=0,kI=0,kD=0,timeseries;//INITIALIZATION

int a=A0,act,angle,pid,ref=180,pidout,pidcont,pidc;//REFERENCE ANGLE SET AS 180


void setup()

{

 Serial.begin(9600);//SETTING UP THE BAUD RATE

 pinMode(11,OUTPUT);//ASSIGNING THE I/O PINS AS OUTPUT FOR GENERATING PWM SIGNALS

 pinMode(9,OUTPUT);//ASSIGNING THE I/O PINS AS OUTPUT FOR GENERATING PWM SIGNALS

}


void loop()

{

 timeseries=millis();//SETTING THE STOPWATCH

 act=analogRead(a);//READING THE POTENTIOMTER'S OUTPUT

 angle=map(act,1010,0,0,330);//MAPPING IT TO DEGREES


//PID LOGIC

 error=ref-angle;

 kP=(kp*error);

 kI+=(ki*error);

 kD=(kd*(error-lasterror));

 pid=kP+kI+kD;


 pidout=map(pid,630,-630,255,-255);//MAPPING PID OUTPUT FOR THE 8 BIT DAC

 pidcont=abs(pidout);

 pidc=constrain(pidcont,95,255);//MOTOR STOPS IF SUPPLY FALLS BELOW 4 VOLTS,SO IT IS CONSTRIANED
//PRINTING THE ANGLE AND TIME IN THE SERIAL MONITOR
  Serial.print(angle);
  Serial.print("\t");
  Serial.println(timeseries);


//CHECKING THE ERROR TO DETERMINE THE DIRECTION OF ROTATION
  if (error>1)
  {
  analogWrite(11,pidc);
  analogWrite(9,0);
  }
  else if (error<-1)
  {
  analogWrite(11,0);
  analogWrite(9,pidc);
  }
  else
  {
  analogWrite(11,0);
  analogWrite(9,0);
 }
  lasterror=error;
}
```

****

For beginners in Arduino the following tutorial link can be used:

https://www.youtube.com/watch?v=fCxzA9_kg6s&list=PLA567CE235D39FA84

The link for the datasheet of L293D is:

http://www.ti.com/lit/ds/symlink/l293.pdf