Position Control of DC Motor

Ankit Singh
213070029

1 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.

2 OBJECTIVE

- To rotate the dc motor by an angle of 180 degrees from any given point.
- 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.

3 MATERIALS & EQUIPMENTS REQUIRED

- DC Motor
- Arduino MEGA 2560
- L293D (Motor driver IC)
- Jumper wires, single stranded wires, bread board, screw driver and wire stripper

4 PROCEDURE

- 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.
- 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.
- 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).

- A mapping function is used in the program to convert the feedback values to their corresponding values in degrees.
- 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.
- 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.
- 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.
- 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).
- The motor stops when the error is between the threshold and the motor stops at the desired angle.
- The PID gain values are tuned until the required constraints in the problem statement are met.

5 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;
}
```