Control Systems Lab
Experiment 1: DC Motor Position Control

Soham Rahul Inamdar (210100149)
Satush Parikh (21D070062)
Rohit Rajwansi (18D070025)

August 2023

1 Introduction

In this experiment we had to design a **PID Position Controller** for a DC motor using Arduino Mega.

2 Objectives

The objectives of the experiment were as follows:

(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**

3 Procedure

We started the experiment by setting up the system according to the block diagram in the lab handout which is given below.

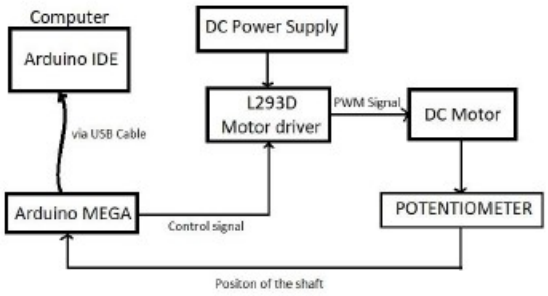

Figure 1: Block Diagram for PID Control of DC Motor

As can be clearly seen in the diagram, we had to use the **L293D Motor driver** interface the e DC Motor with our Arduino MEGA microcontroller. The pinout diagram for the L293D Motor driver is as follows:

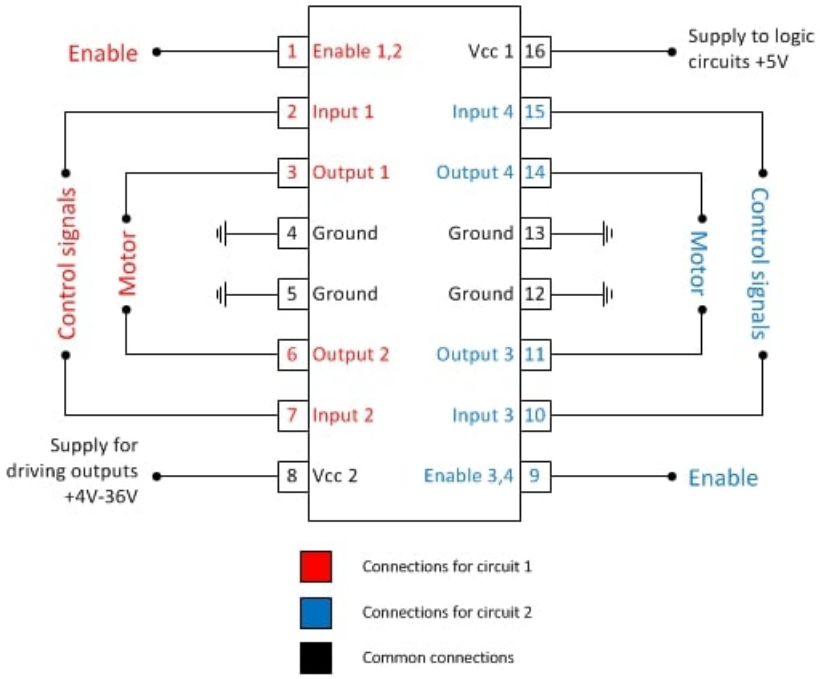

Figure 2: Pinout Diagram of L293D

Analog pin A0 was connected to the potentiometer output. This reads the position of the motor on a scale of **0 to 1023**. Two PWM pins(2 and 3) of the Arduino were connected to the L293D IC to control the speed as well as direction of the motor.
We also identified the non-linear region for the motor by rotating it through 360°. We found this non-linear region to lie between $180° - 190°$. In this region, the potentiometer gives arbitrary outputs for the motor position.

4 PID Control Algorithm

After making the connections, we wrote and dumped the code to implement PID control. Some important elements of our code are as follows:

- The current time is obtained using the function *millis()* for the time-based aspects of the algorithm.

- We have a conditional block to make sure that the non-linear region of $180° - 190°$ is avoided at all times.
- A variable called 'control' is defined to compute and store the value of the PID signal which is done as follows:

```
e_pos = target_pos - current_pos;
e_der = (e_pos - e_prev) / delta_t;
e_int += e_pos * delta_t;
control = kp * e_pos + kd * e_der + ki * e_int;
```

- The direction of rotation of the motor depends on the value of control. A positive value of control rotates the motor in a clockwise direction whereas a negative value of control rotates it in an anticlockwise direction. The code used to implement the above functionality is:

```
if (control > 0)  //10 - 250{
analogWrite(PIN_output1, 0);
analogWrite(PIN_output2, min(control + 30, 255));
}
else if (control <= 0)  //60-190{
analogWrite(PIN_output1, min(-control - 30, 255));
analogWrite(PIN_output2, 0);
}
```

- The motor was thus, made to rotate and the instantaneous value of the error was printed in order to plot the error vs time graph.

5 Observations

5.1 Readings

Time	Error
0	-535
1	-553
1	-563
2	-569
3	-572
4	-574
5	-576
5	-572
15	-573
27	-558
38	-550
50	-538
61	-523
72	-516
83	-493
96	-486
107	-465
119	-449
132	-438
144	-412
157	-401
169	-385
182	-364
194	-355
206	-335
219	-320
232	-307
244	-284
257	-274
269	-257
281	-239

Time	Error
294	-228
307	-205
319	-192
331	-181
344	-157
357	-147
369	-128
381	-113
394	-104
406	-84
418	-83
432	-73
444	-55
456	-52
468	-40
481	-27
494	-25
506	-13
519	-20
531	-15
543	-4
557	-6
569	0
581	2
593	7
606	14
618	2
631	3
644	7
657	3
670	0

Table 1: Instantaneous values of error

At the initial time instants, the values of error are slightly askew. This is because the derivative error is initialized to zero which is not the case in a PID controller.

5.2 Plot

The following plot depicts the variation of error with time when the PID controller is running:

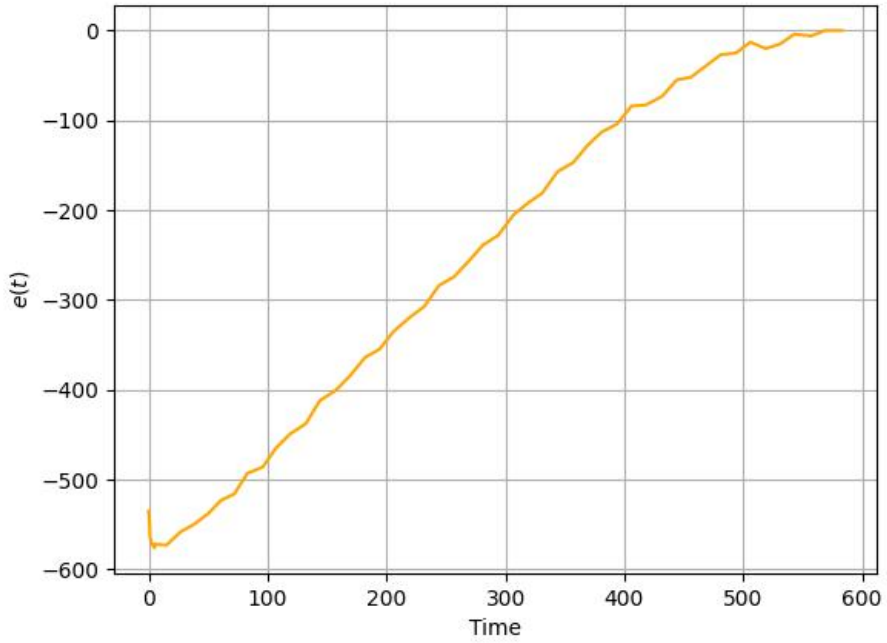

Figure 3: Error vs Time plot

6 Results

We have used the following values of proportional gain(K_p), integral gain(K_i) and derivative gain(K_d) for our PID controller:

K_p	1.20
K_i	0.000002
K_d	0.00001

Table 2: PID parameters

We obtained the following values for the various performance metrics of the PID controller:

Rise Time	0.372s
Settling Time	0.506s
% Overshoot	0%

Table 3: Performance Metrics

7 Challenges Faced

We identified the non-linear region wrongly in our first run, which led to the oscillatory behavior of the motor. To avoid the non-linear region, we had to add a conditional block in our code