Control Systems Lab - Experiment 1
DC Motor Position Control

Rohan Kalbag 20D170033
Pulkit Paliwal 20D100021
Yuvraj Singh Tanwar 20D100030

August 2022

1 Introduction

In this experiment, we design and implement a PID position controller using Arduino Mega to rotate a DC motor by 180 degrees.

2 Aim

At the end of this experiment, we aim to achieve the following objectives:

- 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 Procedure

The block diagram provided with the handout was set up. An Arduino MEGA microcontroller was used to program the PID controller, the logic is explained in further detail in the PID Algorithm section. The potentiometer output was connected to an Analog Pin of the Arduino. Two output pins of Arduino were connected to the L293D IC to control the direction of the DC motor. A third PWM pin was also connected to the enable pin of the L293D to control the speed of the DC motor. The connections were made as per the Block Diagram and Circuit shown below.

We also identified a non linear region for the motor where the potentiometer reading obtained by `analogRead()` transitions from 1024 to 0. This occurs between the angles of 30-40 degrees.

Thus we require the logic to avoid this non linear region when we perform the rotation of 180 degrees.

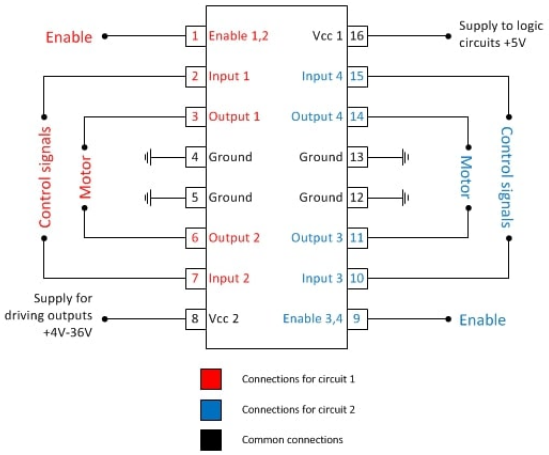

Figure 1: Circuit

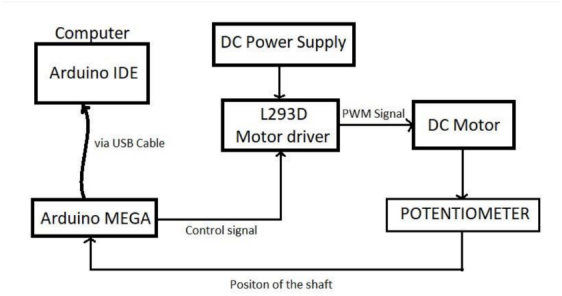

Figure 2: Block Diagram

4 PID Algorithm

The above circuit was set up and the procedure above was followed after dumping the PID Algorithm code in the Arduino microcontroller.

- The current timestamp was obtained and measured with every reading to compare with the time aspect of the constraints.
- A conditional block was used to obtain a value called target_pos which is the target destination to rotate the motor to, the conditional block is designed in such a way that the non linear region corresponding to 30-40 degrees is avoided.
- A variable called `control` was defined to hold a value corresponding to the correction to be applied to the motor controller IC as a PWM signal. This variable has a proportional, derivative as well as an integral term to account for the PID control logic.

```
curr_pos = analogRead(analogPin);
e_pos = target_pos - curr_pos;
e_der = (e_pos - prev_e)/delta_t;
e_int += e_pos*delta_t;
```

```
control = k_p*e_pos + k_d*e_der + k_i*e_int;
prev_e = e_pos;
```

- If the control value obtained was positive then the motor is made to rotate clockwise and if the control value obtained is made negative then the motor is made to rotate in counter clockwise direction.
- A variable called `PWM_val` was defined to be proportional to the value of `control` lying in 0 to 255 to apply as a PWM to the enable signal of the motor controller IC.

```
if (control < 0){
    motor_dir = -1;
}
else if (control > 0){
    motor_dir = 1;
}
PWM_val = min((int)fabs(control), 255);
```

- The motor was made to rotate and the value of the error, the value of PWM and also time was printed out in order to make plot for the response.
- This was plotted to study the response of the PID controller and the values of Ki, Kd, Kp were adjusted accordingly via trial and error to meet the desired control contraints.

5 Graph

5.1 Error-Time Graph

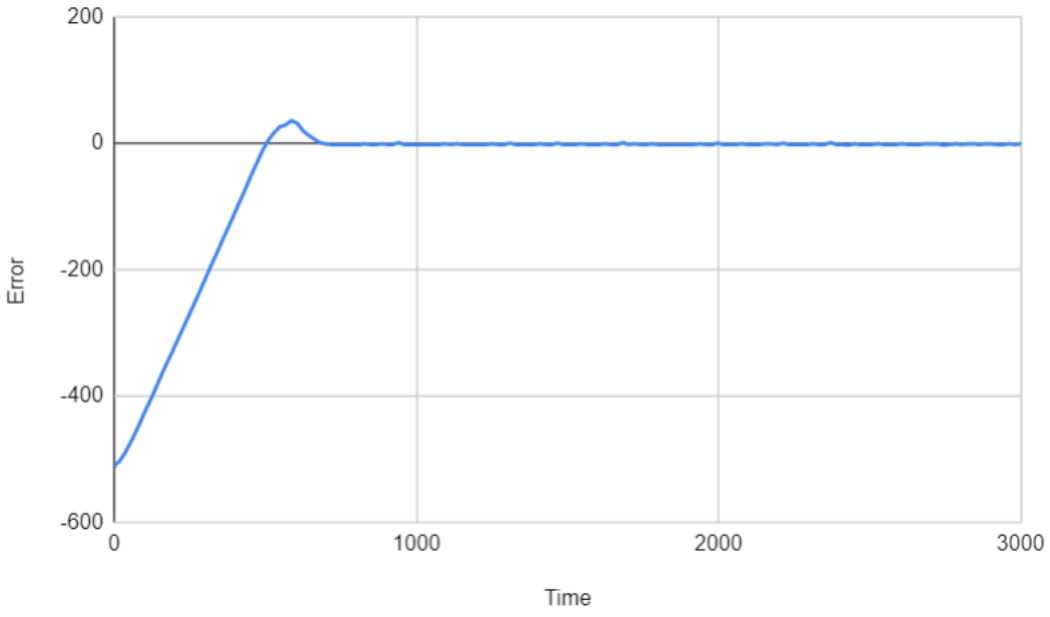

Figure 3: Error Vs Time(s)

5.2 Tabulated Observations

Time (s)	Error	Time (s)	Error	Time (s)	Error	Time (s)	Error
0	-512	773	-2	1546	-1	2332	-2
1	-512	792	-2	1565	-2	2352	-2
2	-509	811	-2	1586	-2	2371	1
18	-503	829	-1	1605	-2	2390	-2
38	-488	848	-2	1625	-1	2409	-2
60	-468	867	-2	1644	-2	2428	-3
80	-448	885	-1	1665	-2	2449	-1
101	-425	904	-2	1684	1	2468	-2
123	-403	923	-2	1701	-2	2488	-2
145	-379	942	1	1722	-1	2507	-2
166	-356	958	-2	1741	-2	2528	-1
189	-333	976	-2	1761	-2	2547	-2
210	-311	996	-2	1781	-1	2567	-2
232	-288	1014	-2	1801	-2	2587	-2
254	-265	1034	-2	1820	-2	2607	-1
276	-242	1054	-2	1840	-2	2626	-2
297	-219	1074	-2	1860	-2	2647	-2
320	-194	1093	-1	1880	-2	2666	-2
342	-171	1114	-2	1899	-2	2685	-1
363	-148	1133	-1	1920	-2	2705	-1
386	-124	1153	-2	1939	-1	2725	-1
407	-101	1172	-2	1958	-2	2745	-3
429	-78	1192	-2	1979	-2	2764	-2
450	-54	1212	-2	1998	0	2785	-1
471	-32	1231	-2	2017	-2	2804	-2
491	-11	1252	-1	2036	-2	2824	-1
510	5	1271	-2	2057	-2	2844	-1
529	17	1291	-2	2076	-1	2864	-2
547	26	1310	0	2096	-2	2883	-1
567	29	1330	-2	2116	-2	2903	-1
587	36	1349	-2	2136	-2	2923	-2
607	31	1369	-2	2155	-1	2942	-2
626	19	1389	-2	2174	-1	2962	-1
645	12	1409	-1	2195	-2	2982	-2
664	6	1428	-2	2214	0	3002	-1
681	1	1448	-2	2233	-2		
698	-1	1468	0	2253	-2		
717	-2	1486	-2	2273	-2		
736	-2	1506	-2	2292	-2		
754	-2	1526	-2	2313	-1		

6 Result

The results for $k_p = 5$, $k_i = 0.0001$ and $k_d = 0.01$ were observed as follows:

- **% Overshoot:** 7.03%
- **Settling Time:** 0.698s
- **Rise Time:** 0.47s

7 Problems faced and their solution

- The initial approach adopted involved using only a proportional controller, which gave a lot of deviation in the angle, much more/less than the desired rotation of 180 degrees. The problem was solved by implementing integral and derivative blocks as well.
- A conditional block was introduced to avoid the non linear region of 30-40 degrees, which was initially a problem
- After tackling the above problem, the next problem encountered was of a very high settling time, which was addressed by fine tuning the k_p, k_i and k_d values to 5, 0.0001 and 0.01 respectively.