EE324: Experiment 1
DC Motor Position Control

Batch 4
Ankith R, 200070006
Anmol Saraf, 200070007
Anubhav Bhatla, 200070008

September 2, 2022

1 Aim of the experiment

Design and implement a PID position controller using Arduino Mega

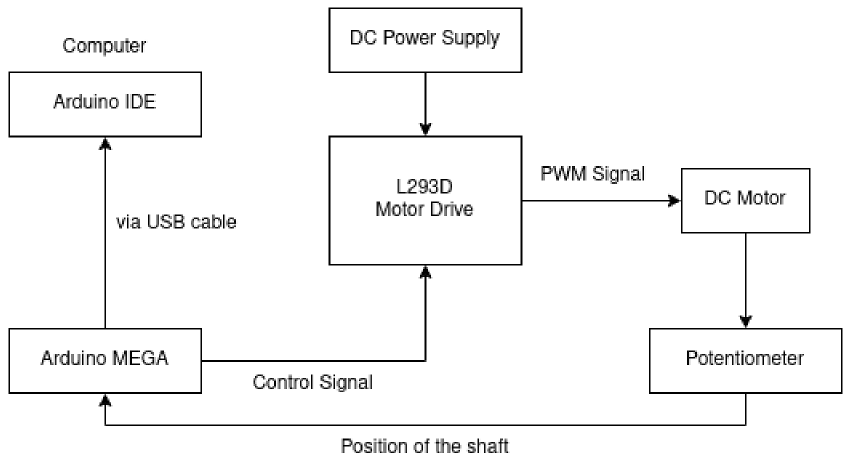
2 Objectives

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

DC Motor setup, Arduino Mega, A-B cable, Power supply, L293D IC, Jumper wires, Single stranded wires, Bread board, Screw driver and Wire stripper

4 Block Diagram



5 Control Algorithm

We are required to implement a PID position controller to rotate the motor to the required set point angle. The control signal output of a PID controller is given as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t) \tag{1}$$

where $e(t) = y_{sp}(t) - y(t)$ is the error between the set point and the current angle, and K_p, K_i, K_d are the respective PID constants.

6 Challenges Faced

- One of the primary challenges we faced was understanding the **non-linear region** for the motor. The potentiometer output from the motor varies between 0 and 1023 based on the angle it has rotated. Now when the potentiometer jumps from 0 to 1023 or vice-versa, the error readings would get shoot up and the motor starts behaving non-ideally. Therefore we first need to identify this region. In our case, we marked down 1000-1023 and 0-50 as the non-linear for our motor and we modified our code so as to avoid passing through this region.

- While we were trying to test to test our code, we were getting an unexpected error wherein the motor was not at all moving. Upon checking the voltage values in our circuit using a DMM, we realized that one of the Arduino ports was not giving the correct output to the Motor Controller IC. We decided to use another PWM port and it fixed the problem.

- Since none of us was accustomed to using an Arduino, it was a steep learning curve to get accustomed to the basics of Arduino programming and being able to implement the PID controller.

7 Observations

Given below are the readings we got for the Angle (from 0 to 1023) value read using the potentiometer on the motor:

Time (in ms)	Angle
0	292
60	324
120	391
182	459
242	532
303	605
363	674
424	740
485	789
545	808
606	809
667	809
728	810
788	811
848	809
910	809
970	809

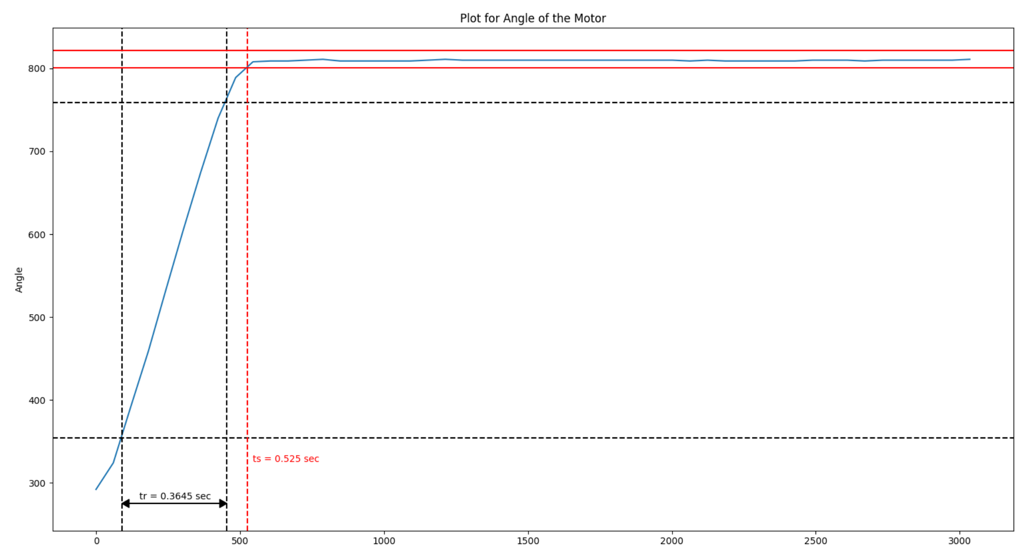
Time (in ms)	Angle
1031	809
1092	809
1153	810
1213	811
1273	810
1335	810
1395	810
1456	810
1517	810
1577	810
1638	810
1699	810
1760	810
1820	810
1881	810
1942	810
2002	810

Time (in ms)	Angle
2063	809
2124	810
2185	809
2245	809
2307	809
2367	809
2427	809
2489	810
2549	810
2610	810
2670	809
2732	810
2792	810
2852	810
2914	810
2974	810
3035	811

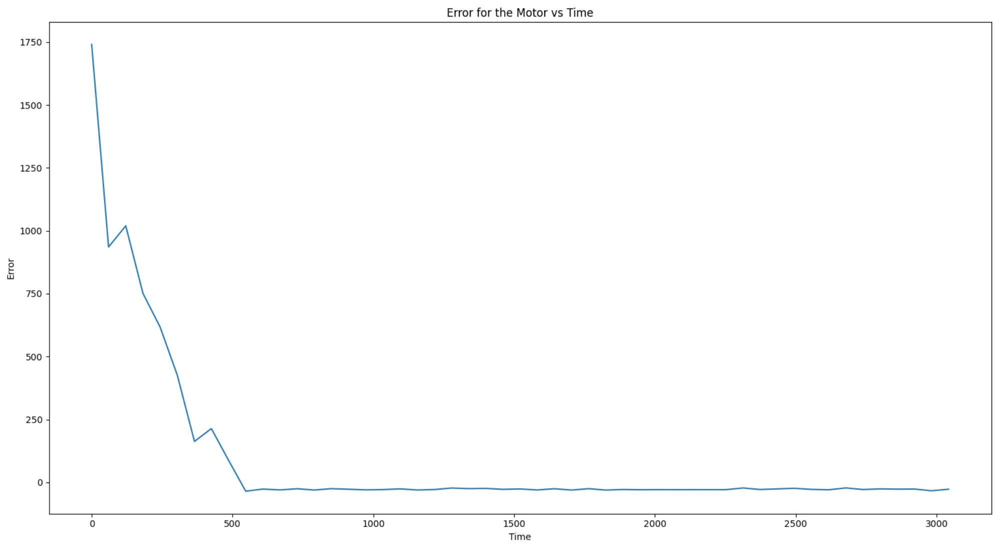
Given below are the readings we got for the Error $e(t)$ i.e. $y_{sp}(t) - y(t)$:

Time (in ms)	Error	Time (in ms)	Error	Time (in ms)	Error
0	1740.80	1034	-28.86	2069	-29.18
60	935.68	1095	-25.86	2129	-29.13
121	1019.73	1156	-30.46	2191	-29.15
182	751.51	1217	-28.62	2251	-29.14
243	617.40	1277	-22.55	2313	-22.34
304	426.24	1338	-24.98	2373	-28.46
365	162.70	1399	-24.01	2435	-26.02
425	213.72	1460	-27.80	2495	-23.59
486	87.91	1521	-26.28	2556	-27.96
547	-35.17	1582	-30.29	2617	-29.62
608	-26.73	1643	-25.29	2678	-22.15
669	-30.11	1703	-30.69	2739	-28.54
730	-25.36	1765	-25.13	2799	-25.98
790	-30.66	1825	-30.75	2861	-27.01
851	-25.14	1887	-28.50	2921	-26.60
912	-27.35	1947	-29.40	2982	-33.56
973	-29.86	2008	-29.04	3043	-27.38

Given below is the plot for Angle (in 0 to 1023) vs Time:



Given below is the plot for Error vs Time:



8 Results and Conclusion

Given below are values of the PID constants we used:

K_p	3
K_i	10
K_d	0.4

Given below are the calculated values for the achieved design specifications:

Rise Time	364.5ms
Settling Time	525ms
Overshoot	0.123%

These values satisfy the required design specifications. Therefore we were able to successfully design the PID controller with the correct specifications.