EE324: Experiment 1
DC Motor Position Control

Akhilesh Chauhan 21d070010
Uvesh Mohammad 21d070084
Parth Arora 21d070047

August 30, 2023

Contents

1 Overview of the experiment **2**
1.1 Aim . . . 2
1.2 Objectives . . . 2
1.3 PID Control Logic . . . 2

2 Design **3**
2.1 Components Required . . . 3
2.2 Block Diagram . . . 3
2.3 Breadboard Circuit . . . 4
2.4 Arduino Programming . . . 4

3 Experiment **6**
3.1 Results . . . 6
3.2 Observations . . . 7
3.3 Challenges Faced and their Solutions . . . 7

1 Overview of the experiment

1.1 Aim

To design and implement a PID Position Controller using Arduino Mega microcontroller.

1.2 Objectives

This PID controller will find its application as a DC Motor Position Controller, which is expected to do:

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

1.3 PID Control Logic

Our setpoint is the initial angle + 180. So, we use a PID controller in order to achieve this. The use of a PID controller can be justified by the following :

- **Proportional Controller:** It is responsible for driving the output according to the present error. The present error is the difference between the current position and the set point which gets multiplies by a proportional gain $\boldsymbol{k_p}$. Though, it drives the system towards the required output,there always exists some steady state error upon using a proportional controller alone.
- **Integral Controller:** The integral factor integrates the error term until it reaches zero;thereby reducing the steady state error to 0.However, the response speed is very slow. The output is obtained by integrating the error and multiplying it by a constant $\boldsymbol{k_i}$ also called as the integral gain.
- **Differential Controller** It observes the rate at which the system variables are changing and generates the output in correspondance to the rate of change. The output is obtained by multiplying the error's rate of change with a constant $\boldsymbol{k_d}$ also called the derivative gain.

So, we use a PID controller in order to improve the drawbacks faced while using the controllers separately.The equation representing the controller is:

$$\boxed{\boldsymbol{\omega = k_p * \theta + k_i * \int(\theta dt) + k_d * \frac{d\theta}{dt}}}$$

2 Design

We designed the position controller by implementing the block diagram given below.

2.1 Components Required

The following components were required for this experiment:

- DC Motor setup
- Arduino Mega
- A-B Cable
- Power Supply
- L293D IC
- Jumper wires
- Single stranded wires
- Bread board
- Screw driver
- Wire stripper

2.2 Block Diagram

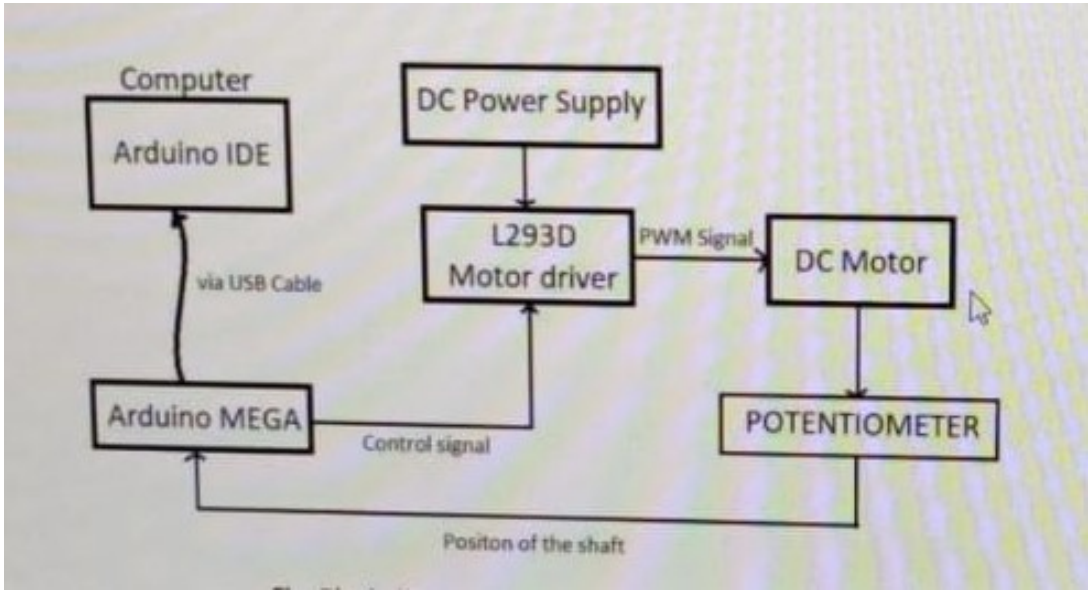

Figure 1: Block Diagram

2.3 Breadboard Circuit

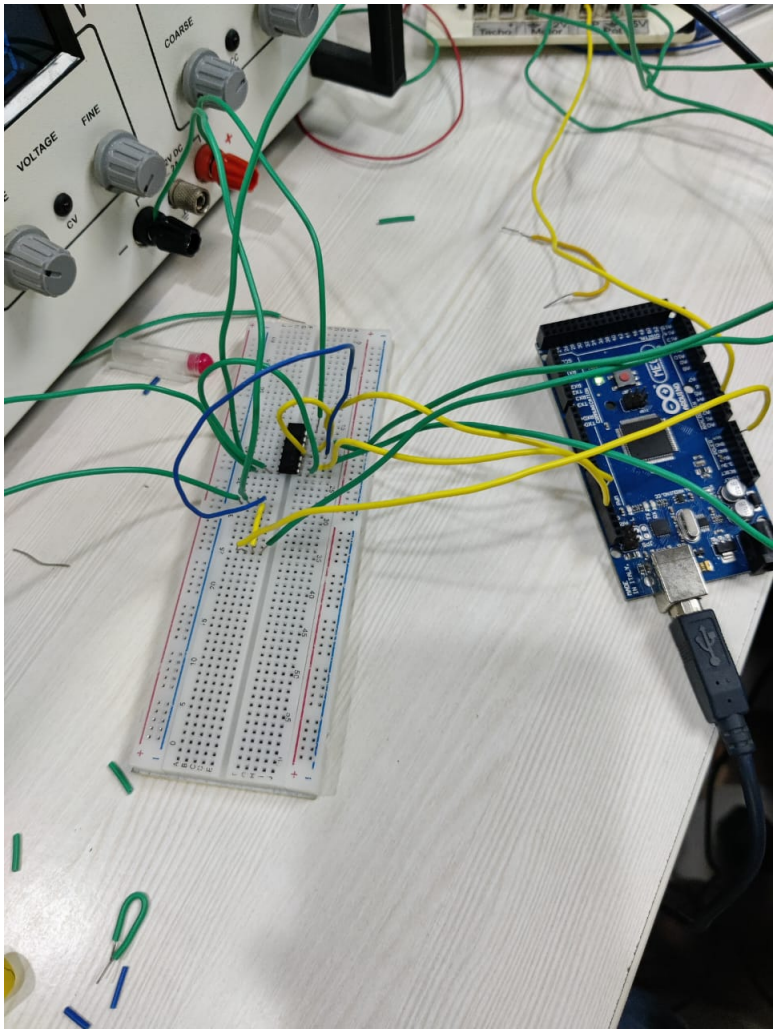

Figure 2: Breadboard circuit

2.4 Arduino Programming

Code is given below:

```
int input = A0;
int out2 = 12;
int out1 = 13;

int v =0, target, output;

double Kp = 9.5 , Kd = 0.5, Ki = 0.001;
double error = 0, errpre=0, errsum = 0;
double h=40;
```

```
void setup() {
pinMode (input,INPUT);
pinMode(out1,OUTPUT);
pinMode(out2,OUTPUT);

v = analogRead(input);
if(v > 511)
  target = v - 532;
else
  target = v + 532;

Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
v = analogRead(input);
error = target - v;
errsum = errsum + error;
output = error*Kp + Ki*errsum + Kd*(error - errpre);

errpre = error;

Serial.println(v);

if(error > 0)
{
  analogWrite(out1, 0);
  analogWrite(out2, min(255, output));
}

else
{
  analogWrite(out1, min(255, abs(output)));
  analogWrite(out2, 0);
}

}
```

3 Experiment

3.1 Results

The time response of the designed PID Position controller is shown below. This system has the following gain constants.

- $k_p = 9.5$
- $k_d = 0.5$
- $k_i = 0.001$

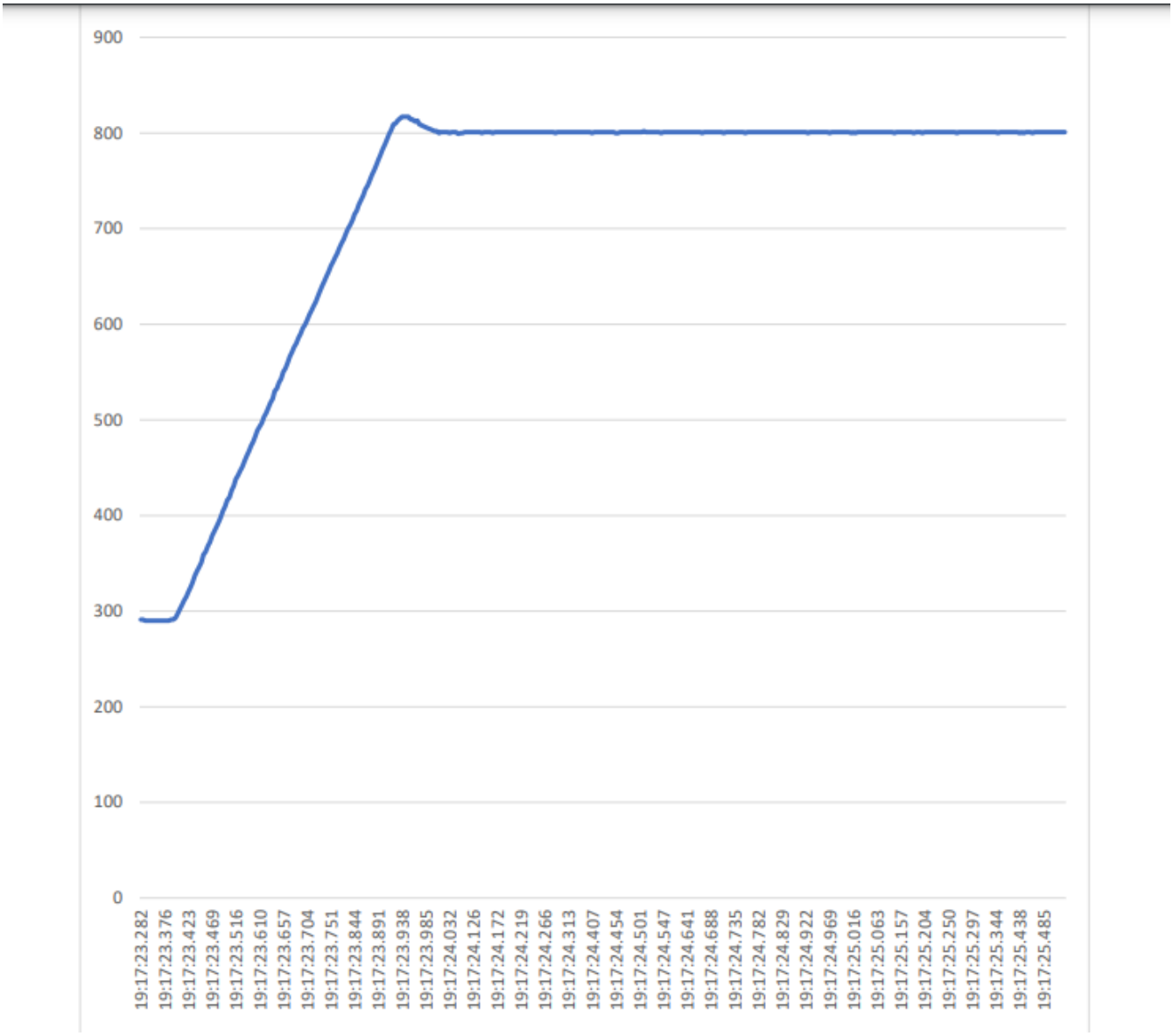

Figure 3: Time Response of the designed PID Position controller

3.2 Observations

By observing the above shown time response of the PID Controller, the following transient characteristics are noted.

- **Rise Time:**The time required for the waveform to go from 0.1 of the final value to 0.9 of the final value.

$$\boxed{\boldsymbol{T_r = 0.42s}}$$

- **Settling Time:**The time required for the transient's damped oscillations to reach and stay within 2 percent of the steady-state value.

$$\boxed{\boldsymbol{T_s = 0.81s}}$$

- **Overshoot:**The amount that the waveform overshoots the steadystate, or final, value at the peak time, expressed as a percentage of the steady-state value

$$\boxed{\boldsymbol{OS\% = 2.1\%}}$$

Thus, it can be inferred that the system we designed is well within the required transient response of **0.5** second rise time, **1** second's settling time and **10 %** overshoot.

3.3 Challenges Faced and their Solutions

- $\boldsymbol{K_p, K_d, K_i}$ **Fine Tuning:** To get the desirable transient response the gain constants should be fine tuned. So we sought the method of Trial and Error for this purpose. In this method, we first increase the value of kp until system reaches to oscillating response but system should not make unstable and keep value of kd and ki zero. After that, we set value of ki such that, oscillation of the system stops. After that the value of kd is set for fast response.

- **Potentiometer Irregularity:** Another minor challenge that we faced is regarding the potentiometer reading not getting synchronized perfectly with the angle i.e. each time the motor rotates;the angle corresponding to a particular potentiometer reading changes. Hence, we dealt with relative variables such as initial angle and the setpoint.

- **Non-linearity:** When the initial angle is around 340-350 the motor did not stop at the desired angle. This occurs due to the discontinuity in the potentiometer reading as it drops off from 1023 to 0 and consequently, the motor enters the non-linear region.