Experiment 1

DC Motor Position Control

Vinay Sutar, 21d070078
Prajapati Kishan K , 21D070048
Shounak Das , 21D070068

August 2023

Contents

1 Aim

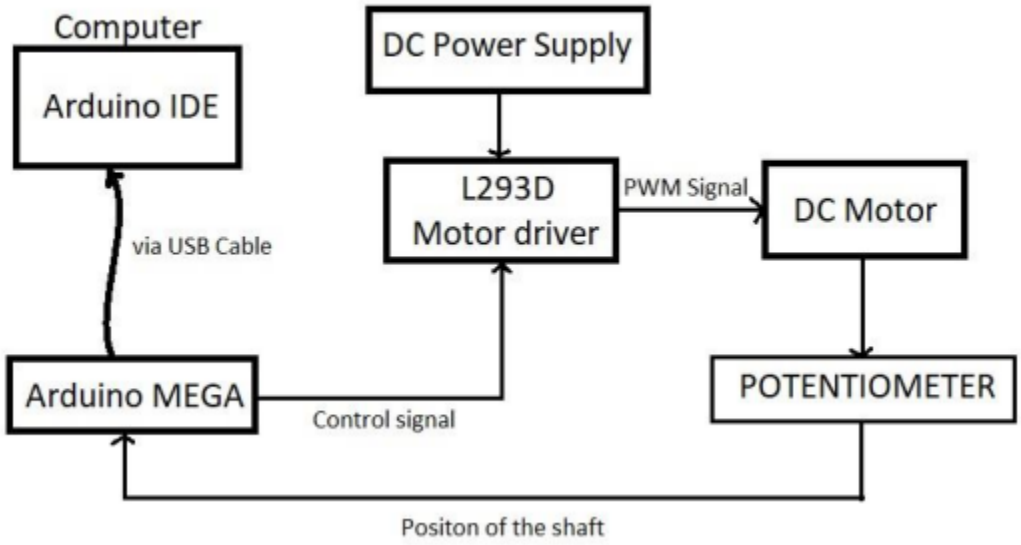
Design and implement a PID position controller of a motor using Arduino Mega.

2 Getting Started

2.1 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 percent overshoot.

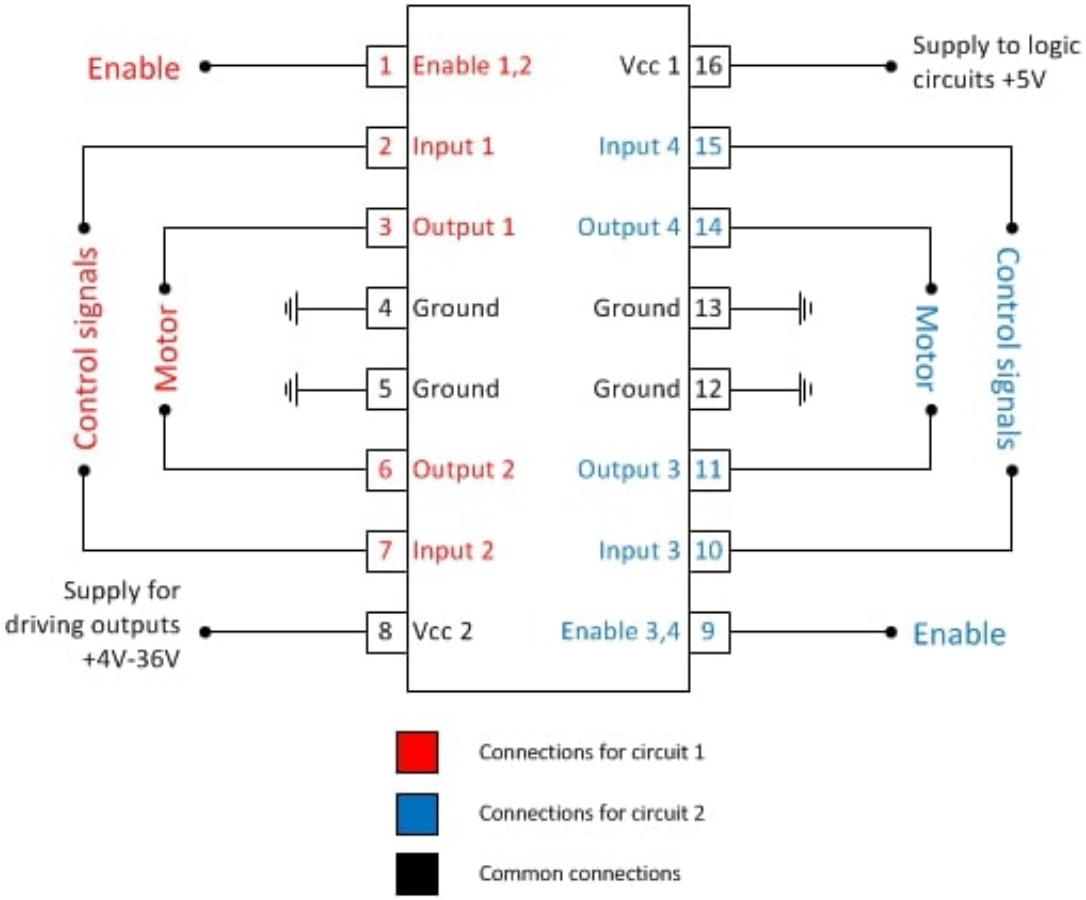
2.2 Materials Used



2.3 Components

1. *L293D Motor Driver* 2. *Bread Board* 3. *Arduino Mega* 4. *A-B cable* 5. *Wires* 6. *Wire Stripper* 7. *Voltage supplier* 8. *Motor*

- L293D - Motor Driver:



- Arduino Mega:
- Motor and Potentiometer:

2.4 Setup

Setup was made according to the above figure, Motor was connected to the IC, Control signals are taken from Arduino which was controlling as we write the code.

3 Experiement

3.1 Finding Non-linear region

```
const int motorPin1 = 12;  // PWM pin for motor 1
const int motorPin2 = 4;   // PWM pin for motor 2
const int analogPin = A4; // Analog pin to read voltage from

void setup() {
  Serial.begin(115200); // Initialize serial communication
}

void loop() {

  int sensorValue = analogRead(analogPin); // Read analog value (0-1023)
  float voltage = sensorValue * (5.0 / 1023.0);

  Serial.print("Raw ADC Value: ");
  Serial.print(sensorValue);
  Serial.print(", Voltage: ");
  Serial.print(voltage, 2); // Print voltage with 2 decimal places
  Serial.println("V");

  delay(750); // Delay for a second before taking another reading
}
```

The provided code reads an analog voltage from a sensor connected to pin A4. It calculates the voltage value from the sensor's raw analog reading and prints both the raw reading and the corresponding voltage to the serial monitor. The purpose of this code is to manually rotate a motor to explore its non-linear region, where the motor's behavior deviates from linearity. This exploration is done by observing how the analog voltage changes as the motor is turned by hand. The code then introduces a delay of 750 milliseconds before taking the next reading.

From the data obtained on the Serial Monitor the non-linear region of the Potentiometer is 290-320 degrees.

3.2 Implementing PID controller to rotate the Motor by 180 degrees

```
void setup() {
```

```
  Serial.begin(9600); // Initialize serial communication
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);

float x1 = analogRead(analogPin); // Read analog value (0-1023)

if(x1>540)
{
 target = x1 - 540;
}
else if(x1<540)
{
   target = x1 + 540;
}
}
void loop() {
  float sensorValue = analogRead(analogPin); // Read analog value (0-1023)
  float voltage = sensorValue * (5.0 / 1023.0);
  float curr=sensorValue;

  error_p = target - curr;

  e_der=(error_p-prev_err);
  e_int  += error_p;
  velocity = kp*error_p + kd*e_der + ki*e_int;

  if (velocity>255)
  {
    velocity = 255;
  }
  if (velocity < -255)
  {
    velocity = -255;
  }
  if (velocity >=0){
    analogWrite(motorPin2, (velocity));
    analogWrite(motorPin1, LOW);
  }
  else{
    analogWrite(motorPin1, -1*(velocity));
    analogWrite(motorPin2, LOW);
```

```
  }

  prev_err=error_p;
}
```

- The provided code implements a PID (Proportional-Integral-Derivative) controller to rotate a motor by 180 degrees based on an analog sensor's input. The analog sensor's output voltage is read from the specified analog pin (A4) and converted to a corresponding value. The code initializes various constants for the PID control (kp, kd, ki) as well as variables to track errors, integrals, derivatives, and control signals.
- Upon startup, the code calculates a target value based on the initial analog reading. If the reading is greater than 540, the target is set to the difference between the reading and 540. If it's lower than 540, the target is set to the sum of the reading and 540.
- In the loop, the code computes the error between the target and the current sensor reading. Using PID control principles, it calculates a control value (velocity) to adjust the motor's rotation speed. The PID components (proportional, integral, and derivative) are computed and combined to determine this control value.
- The calculated velocity is constrained within the range of -255 to 255 to ensure it doesn't exceed the motor's operational limits. Depending on the velocity's sign, the appropriate PWM signals are sent to the motor's pins using the 'analogWrite()' function, controlling its rotation direction and speed.
- The PID controller continuously refines the control signal based on the error and its derivatives and integrals, allowing the motor to gradually and accurately rotate to the desired position of 180 degrees. The controller's parameters (kp, kd, ki) can be adjusted to fine-tune the motor's response and minimize any overshooting or oscillations.

4 Observations and Inference

Time	Error	Time	Error	Time	Error
10	540	400	-3	770	-3
20	537	410	-3	780	-3
30	530	420	-3	790	-4
40	520	430	-4	800	-4
50	509	440	-4	810	-3
60	498	450	-3	820	-3
70	485	460	-3	830	-3
80	474	470	-3	840	-4
90	461	480	-4	850	-4
100	448	490	-3	860	-3
110	435	500	-3	870	-3
120	423	510	-3	880	-4
130	411	520	-4	890	-4
140	392	530	-4	900	-3
150	373	540	-3	910	-3
160	354	550	-3	920	-3
170	335	560	-3	930	-4
180	315	570	-4	940	-4
190	297	580	-3	950	-3
200	278	590	-3	960	-3
210	258	600	-3	970	-3
220	240	610	-4	980	-4
230	221	620	-4	990	-3
240	201	630	-3	1000	-3
250	183	640	-3	1010	-3
260	164	650	-3	1020	-3
270	145	660	-4	1030	-4
280	127	670	-4	1040	-4
290	106	680	-3	1050	-3
300	88	690	-3	1060	-3
310	70	700	-4	1070	-3
320	53	710	-4	1080	-3
330	34	720	-3	1090	-4
340	17	730	-3	1100	-4
350	-1	740	-3		
360	-15	750	-4		
370	-15	760	-4		
380	-14	770	-3		
390	-5	780	-3		
400	2	790	-4		
410	6	800	-4		
420	7	810	-3		
430	4	820	-3		
440	-2	830	-3		
450	-6	840	-4		

Table 1: Variation of Error with respect to time

5 Results

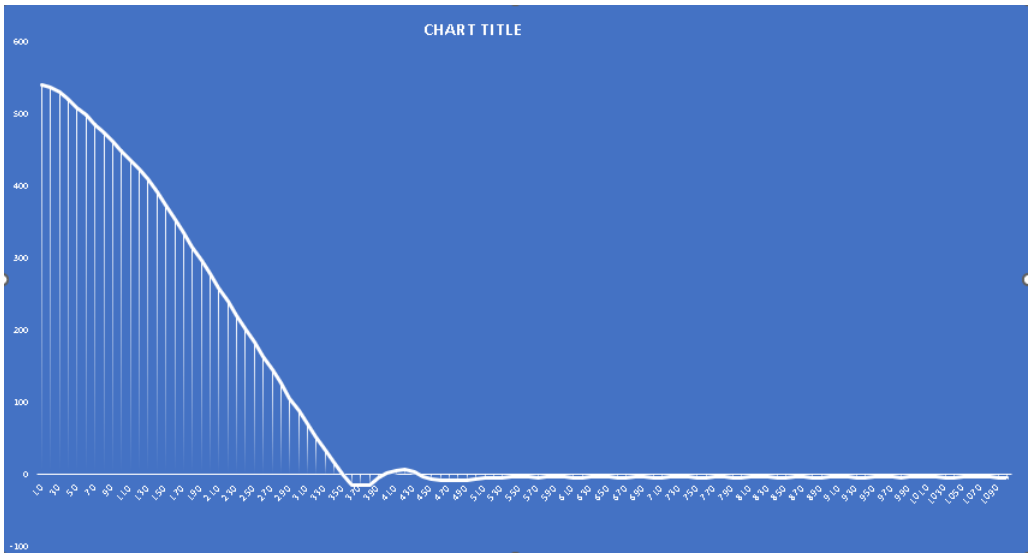


Figure 1: Plot of Error vs Time

The results for kp = 5, ki = 0.0001 and kd = 0.01 were observed as follows:

- **Percent Overshoot 6.98%** : It was calculated using the curr values. Basically it is overshoot/finalvalue multiplied by 100.
- **Rise Time 0.25sec** : It is the time between 10% to 90% of the final value. Which can easily calculated from the taken data.
- **Settling time 0.47sec** : It is the time for reaching 2-5% of the final value. Which also easy to derive from the taken data.

6 Challenges faced and Solutions

- 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 30-35 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 kp, ki and kd values. respectively.