EE324: Control Systems Lab

Experiment 1: DC Motor Position Control

Thursday batch - Table 13

Jatin Kumar (22B3922)
Aayush Kumar (22B0769)
Archisman Bhattacharjee (22B2405)

August 29, 2024

1 Aim

Design and implement a PID position control system using Arduino Mega.

2 Objective

- To rotate the DC motor by an angle 180 degrees from an arbitrary initial position.
- To ensure that the rotation happens successfully within the constraints of a 0.5-second rise time, a 1-second settling time, and less than 10% overshoot.

3 Control Algorithm and Arduino Code

In this experiment, we implement the PID control algorithm (Proportional-Integral-Derivative). It is a basic control strategy consisting of 3 main terms, each of which plays a role in adjusting the output response of the system. The proportional term uses the difference/error between the setpoint of 180 degrees and the current angle. The integral term accumulates the past errors and over time responds based on the duration and magnitude of the errors. Finally, the derivative term operates based on the rate of change of the error. The parameters are carefully selected after trial and error so as to get the desired response based on the given constraints.

The working of the algorithm is as follows:

- Reading the potentiometer output value (which will be proportional to the current position) and defining a final position value, which will be either 512 (proportional to a 180-degree rotation of the motor) more or less than the current potentiometer value.
- Calculating the error value, which is the difference between the current and final position.
- Computing the proportional term of feedback by taking the product of the error and the proportional gain value. The derivative term of feedback is computed by taking the product of the derivative gain with the difference between the current and previous error. The integral term is computed by taking the product of the integral gain with the sum of the current and last two error values.
- The total feedback value is computed by taking the sum of the proportional, derivative, and integral terms. This feedback acts as a control signal, feeding back to the system as a PWM value.
- In the case of overshoot due to the system's inertia, a conditional statement is incorporated that reverses the direction of the motor's rotation if the error value's sign changes, which causes damping.

3.1 Arduino Code

The following Arduino code implements the PID control algorithm:

```
int enablepin = 5;
int pot = A0;
int potreading =0;
int initial = 0;
int fin=0;
//int bound2=0;
const float kp=5;
const float ki=2;
const float kd=7;
int pwm_val=0;
float feedback=0;
int error=0;
int old=0;
int old1=0;
int old2=0;

void setup() {
  Serial.begin(9600);
  initial = analogRead(pot);
  old = initial;
  if (initial>512){
      fin = initial-512;
   }
   else{
    fin = initial+512;
    }
}

void loop() {
  potreading = analogRead(pot);
  Serial.print("Final: ");
  Serial.println(fin);
  Serial.print("Current: ");
  Serial.println(potreading);
  error = fin - potreading;
  feedback =
      (kp*abs(error)/2)+(kd*(abs(error)-abs(old))/2)+(ki*(abs(old)+abs(old1)+abs(old2))/3);
  if(feedback>255){
    pwm_val = 255;
    }
  else{
    pwm_val = (int)feedback;
    }

  if (error>0){
    digitalWrite(2, LOW);
    digitalWrite(3, HIGH);
    analogWrite(enablepin, pwm_val);
    }
  else{
    digitalWrite(2, HIGH);
    digitalWrite(3, LOW);
    analogWrite(enablepin, pwm_val);
    }
  old2=old1;
  old1=old;
  old = error;

    delay(5);
}
```

Listing 1: Arduino Code for PID Control

Table 1: PID controller parameters

Parameter	**Value**
Kp	5
Kd	7
Ki	2

4 Challenges Faced

The following challenges were faced by us during the course of this experiment.

- **Speed control** - Our initial approach for speed control involved supplying the enable pin of the motor driver with a PWM input, but this unfortunately did not provide a good enough response. To solve this issue, we instead applied PWM inputs to the control signal pins of the motor driver.
- **Parameter selection** - After selecting the parameters K_P, K_D, and K_I which gave a satisfactory response, on adding code to record the time and angle of the motor we observed that the behavior of the system changed. We then continued adjusting the constants to get the required response again.
- **Deadband-(Non Linear Region)** - After identifying the deadband region, we observed that the response of the potentiometer was not linear or predictable in that region. In order to achieve the desired motion using PID control, we decided to ensure that this region would not be encountered during the rotation. To do so, we identified whether it was ideal to move in the clockwise or anti-clockwise direction based on the starting position.

5 Results

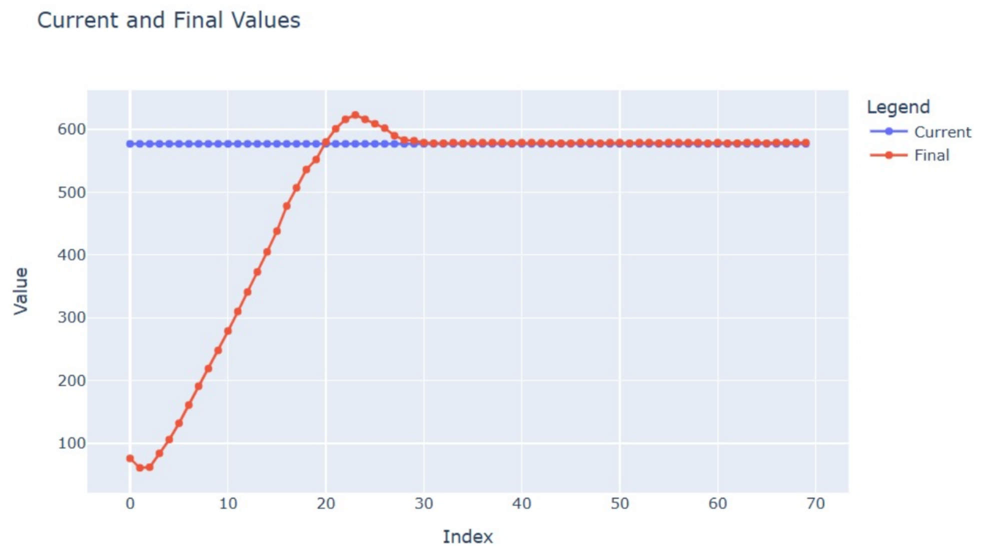

Figure 1: Step response of the Motor

The final implementation of the PID controller achieved the following design constraints:

- **Rise Time :** 150 miliseconds
- **Settling Time :** 532 miliseconds
- **Overshoot :** 7.97%

6 Calculations

1. **Rise Time**
   The rise time of the system is defined as the time taken for the system to go from 10% to 90% of the target final position. In this case, it is the time taken by the DC motor to rotate from 18 to 162 degrees.
   The rise time was observed to be **150 ms**.

2. **Settling Time**
   The settling time is the time taken for the system's response to stabilize within a small percentage of the setpoint or final value, which in this case is 180 degrees from the initial position. We choose the settling time to be within 5% of the setpoint.
   The settling time was observed to be **532 ms**.

3. **Overshoot Percentage**
   The overshoot is the maximum value that the system exceeds the setpoint before it stabilizes.
   The Overshoot Percentage was calculated using the following formula:

$$\text{Overshoot (\%)} = \left(\frac{y_{\text{max}} - y_{\text{final}}}{y_{\text{final}}} \right) \times 100\%$$

   On substituting the peak value of 191.95 degrees and setpoint 180 degrees:

$$\text{Overshoot (\%)} = \left(\frac{194.346 - 180}{180} \right) \times 100\% \approx 7.97\%$$

These results indicate that the design specifications were successfully met.

7 Observation and Inference

Through this experiment, we were able to understand the working of a PID controller and how it can be used to achieve a good level of precision when used to control the DC motor. In order to achieve the desirable response and to meet the design constraints, one needs to be careful while tuning the parameters for each of the terms.

After carefully selecting the parameters, we observed the impact of each of the constants on the behavior of the system.

- K_p
  On increasing K_p, we are able to reduce the rise time, but only at the cost of also increasing the overshoot percentage and increasing oscillations.
- K_I
  K_I was found to be vital in reducing the steady-state error, but if not tuned properly, it could lead to more oscillations.
- K_D
  K_D helps to control the oscillations and overshoot and adjust the response time of the system.