EE 324 : Lab 1 Report

Bhavik Yadav   Sujeet Mehta   Jayveer Singh Sikarwar

August 28, 2023

Batch no. 12

1 Objective

- To design and implement a PID position controller using Arduino Mega
- 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 settling time and 10% overshoot

2 Control Algorithm

We used the below code to read the potentiometer values at various angles. This helped us to determine the region in which non-linearity of voltage and angle is present.

```
int volt = A0;
int read_val;
void setup() {
    Serial.begin(9600);
    pinMode(volt,INPUT);
}
void loop() {
    read_val = analogRead(volt);
    Serial.println(read_val);
}
```

The below code helped us to control the DC motor and adjust its various parameters according to our needs.

```
const int potPin = A0;
const int motorPinPositive = 6;
const int motorPinNegative = 3;
// Constants for motor control
const int targetThreshold = 560;
const int minMotorVelocity = 20;
const int maxMotorVelocity = 255;
float kp = 2.5, kd = 0, ki = 0;

void setup() {
  Serial.begin(9600);
  pinMode(potPin, INPUT);
  pinMode(motorPinPositive, OUTPUT);
  pinMode(motorPinNegative, OUTPUT);
  // Calculate target based on potentiometer value
  float potValue = analogRead(potPin);
  int target;
  if (potValue > 512) {
    target = potValue - targetThreshold;
  } else {
    target = potValue + targetThreshold;
  }
}

void loop() {
  float potValue = analogRead(potPin);
  Serial.println(potValue);
  int error = target - potValue;
  int velocity = calculateVelocity(error);
  applyVelocity(velocity);
  delay(10);
}

int calculateVelocity(int error) {
  static float previousError = 0;
  static float integralPart = 0;
  float differentialPart = error - previousError;
  integralPart += error;
  int velocity = kp * error + kd * differentialPart + ki * integralPart;
  // Clamp velocity within limits
  if (velocity > maxMotorVelocity) {
    velocity = maxMotorVelocity;
  } else if (velocity < -maxMotorVelocity) {
    velocity = -maxMotorVelocity;
  }
  if (velocity < 70 && velocity > 0) {
    velocity = 70;
  } else if (velocity > -70 && velocity < 0) {
    velocity = -70;
  }
  if (velocity > -minMotorVelocity && velocity < minMotorVelocity) {
    velocity = 0;
  }
  previousError = error;
  return velocity;
}

void applyVelocity(int velocity) {
  if (velocity >= 0) {
    analogWrite(motorPinPositive, velocity);
    analogWrite(motorPinNegative, 0);
  } else {
    analogWrite(motorPinPositive, 0);
    analogWrite(motorPinNegative, -velocity);
  }
}
```

3 Challenges faced and their Solutions

We faced various challenges throughout this experiment.

- Adjusting the values of kp, ki and kd were challenging. It took a lot of time and experimentation to get the motor to behave as we wanted it to.
- We couldn't find the non-linearity region at the start as the potentiometer wouldn't show a linear graph at all. Later, it came to light that the code we had written had the reading of the pin as output instead of input.
- The DC motor would twitch and rotate a little even after reaching 180°. This was creating an unnecessary overshoot of around 4-5°. The solution that we found to this was to limit the speed of the motor in the code. Instead of having the velocity vary with error everytime, we edited it such that at very low errors it would stop and at low errors, the velocity would be some constant.

- Finding the potentiometer value corresponding to a change of 180° was also another challenge. Since the non-linearity region took up a part of region of the 1024 divided parts, the value wasn't coming out to be exactly 512. After experimenting with various values and reading the corresponding errors on the computer, we came upon a value that helped to make the error almost 0.

4 Results

- The final values of kp, ki and kd that we had were 2.5, 0 and 0 respectively.
- The non-linearity region was found to be in the range of 110°-130°.
- The analog reading of potentiometer corresponding to a change of 180° in the angle and an error of 0 was found to be 560.
- There was no overshoot in the final experimentation. Therefore, % overshoot is 0. The rise time was determined to be 330 ms. The settling time was found to be 380 ms.

5 Observations and Inference

- There was no overshoot in the system because we had set kd and ki to 0. This meant that only the variable kp would control the system. Setting high kp would increase the speed of the system but care had to be taken to not make it too large.
- The potentiometer reading corresponding to a change of 180° wasn't exactly 512 because the non-linearity region would take up some amount of the range.
- The Arduino board was used to control the setup. An external voltage source was used to provide voltages of 5V and 12V to the board and DC motor respectively. The Arduino board had to be supplied with a voltage so as to enable the control signals.