Experiment 1: Motor PID Controller

Siddhant Batra (200070094)
Shivam Patel (200070077)
Harsh Lulla (200070024)

September 2022

1 Objective

The aim of this experiment was to create a PID controlled motor which rotates 180 degrees which can be changed by changing a variable in the code. We used Arduino Mega 2500 board for the controller to control a servo motor as our motor system. Accurate turning of the motor in the desired specifications are achieved.

2 Control Algorithm

In the control algorithm we used PID controller implementation in discrete machine time (t 10us) where differential and integral are calculated at every step.

$$u(t) = K_p[e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{d}{dt}e(t)] \tag{1}$$

3 Design

The motor has 3 individual connection ports which are:

- 5v DC supply for the potentiometer
- 12v control signal for motor
- Potentiometer output in DC value

The controller microchip on the motor converts the discrete value of driving control signal from 0-255 to 0-12V DC voltage. The analog reading pins get the value of the potentiometer output from 0-1023. Intially there is an offset in the potentiometer with a value of 512, which should be taken care for. Hence in the code we take the inout from motor and then decide which direction the motor needs to turn by 180i.e. a half rotation.

```
void setup() {
  // put your setup code here, to run once:
  pinMode(pos, INPUT);
  pinMode(o1, OUTPUT);
  pinMode(o2, OUTPUT);
  Serial.begin(9600);
  initVal = analogRead(pos);
  if(initVal >= offset){
    finVal = initVal - offset;
    err =-offset;
  }
  else{
    finVal = initVal + offset;
    err = offset;
  }
}
```

Above shown code is used for normalising the offset.
We continously update teh total and differential error at every time step by taking the readings from analog pin, then the control signal is sent by the Arduino board to the control microchip which in turn controls the movement of motor in appropriate direction.
A lot of trials were done to obtain the appropriate values of Kp, Kd and Ki which are as follows.

- Kp = 5
- Kd = 0.01
- Ki = 0.019

4 Challenges Faced

1. We initially had written a code which read the output value in degrees, and decide whether it had to rotate clockwise or anticlockweise. This caused continuous oscillation of the system at the target , because either rotate clockwise or rotate anti Clockwise was being called. This meant that the system always had a full high positive or negative turning command, and hence the oscillations, rather than settling down. This was eventually countered by using a continuous output value, rather than a full high or low value. This meant that now the system could settle .

2. At first, we didn't normalise the offset, this caused our motor to rotate continously and not changing direction to stop, the error was very high and the potentiometer values were peaking at 1023, after considering all the above errors we countered the offset by 512 which helped the motor move upto a certain point (180in our case) and then stopped at that place after a few oscillations about the final point.

3. While we were observing the error and input from the motor through the print statement we did not account for the time delay which was caused by so many print statemets, after removing them our code worked much efficiently.

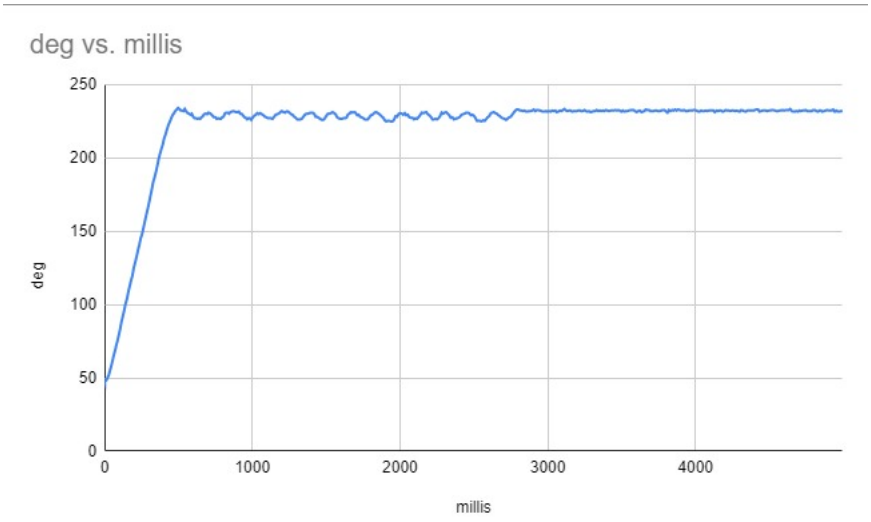
5 Code Snippet

```
#define pos A0
int o1 = 5;
int o2 = 6;

int initVal, currVal, finVal, derv, func;
int offset = 512;
float err;
float kp = 5 ;
float kd =0 ;
float ki =0.0001;
float volt = 0;
int integral = 0;
float prev_err = 0;
```

```
void setup() {
  // put your setup code here, to run once:
  pinMode(pos, INPUT);
  pinMode(o1, OUTPUT);
  pinMode(o2, OUTPUT);
  Serial.begin(9600);
  initVal = analogRead(pos);
  if(initVal >= offset){
    finVal = initVal - offset;
    err = -offset;
  }
  else{
    finVal = initVal + offset;
    err = offset;
  }
}

void loop() {
  // put your main code here, to run repeatedly:
  currVal = analogRead(pos);
  Serial.println(currVal);
//  Serial.print(millis());
//  Serial.print("\t");
//  Serial.println(currVal);
//  Serial.println(integral);
//  Serial.println(derv);
  err = finVal - currVal;
  integral = integral + err;
  derv = err - prev_err;
  func = (int)(kp*err + kd*derv + ki*integral);
  volt = func/4;

  if(abs(volt) > 255){volt = 255;}

  if(err > 0){
    analogWrite(o2, abs(volt));
    analogWrite(o1, 0);
  }
```

```
  else{
    analogWrite(o1, abs(volt));
    analogWrite(o2, 0);
  }

  prev_err = err;
  //Serial.println(err, currVal);
}
```

6 Results



- Kp = 5
- Kd = 0.01
- Ki = 0.019
- TRISE = 394 ms
- TSETTLE = 637 ms
- TPEAK = 497 ms
- % O.S. = 1.18 %