Controls Lab Experiment 1
DC Motor Position Control

Friday, Table No. 16

Rohan Nafde, 210070068
Geetesh Kini, 210070041
Aakarsh Chaudhary, 210070002

Friday, Table No. 16

Contents

1 Objectives of the experiment

- 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

  1. 0.5 second rise time
  2. 1 second's settling time
  3. 10% overshoot

2 Control Algorithm

The control algorithm used was **Proportional Integral Derivative Controller**

For below theory, please refer SP-PV error as

$$\text{error} = \text{Set Value (SV)} - \text{Process Variable (PV)}$$

2.1 PID Controller Theory

- Term P is proportional to the current value of the SP-PV error e(t). For example, if the error is large, the control output will be proportionately large by using the gain factor K_p. Using proportional control alone will result in an error between the set point and the process value because the controller requires an error to generate the proportional output response. In steady state process conditions an equilibrium is reached, with a steady SP-PV offset.
- Term I accounts for past values of the SP-PV error and integrates them over time to produce the I term. For example, if there is a residual SP-PV error after the application of proportional control, the integral term seeks to eliminate the residual error by adding a control effect due to the historic cumulative value of the error. When the error is eliminated, the integral term will cease to grow. This will result in the proportional effect diminishing as the error decreases, but this is compensated for by the growing integral effect.
- Term D is a best estimate of the future trend of the SP-PV error, based on its current rate of change. It is sometimes called "anticipatory control", as it is effectively seeking to reduce the effect of the SP-PV error by exerting a control influence generated by the rate of error change. The more rapid the change, the greater the controlling or damping effect.

Friday, Table No. 16

Control action: The mathematical model and practical loop above both use a direct control action for all the terms, which means an increasing positive error results in an increasing positive control output correction.
For instance, if the valve in the flow loop was 100–0% valve opening for 0–100% control output, meaning that the controller action has to be reversed. Some process control schemes and final control elements require this reverse action. An example would be a valve for cooling water, where the fail-safe mode, in the case of signal loss, would be 100% opening of the valve; therefore 0% controller output needs to cause 100% valve opening.

2.2 Mathematical Form

The Control Function is:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)\,d\tau + K_d \cdot \frac{d(e(t)}{dt}$$

where K_p, K_i, K_d are all non-negative, and denote the coefficients for the proportional, integral, and derivative terms respectively

The Laplace transform of the same yields:

$$U(s) = K_p \cdot 1 + K_i \cdot \frac{1}{s} + K_d \cdot s$$

2.3 Simulation in Code

```
prevError = error; // update it with the previous value of error
error = finalValue - sensorValue; // calculate the new error
integral += error; // integral from 0 to t is just summation
delta = error - prevError; // differential term

// now input this value into the motor
// Scale is - 0 is 0V and 255 is 5V
if(sensorValue < finalValue)
{
    outputValue1 = 0;
    outputValue2 = min(kp*error + ki*integral + kd*delta, 255);
}

else
{
    outputValue1 = min(-kp*error - ki*integral - kd*delta, 255);
    outputValue2 = 0;
}
```

Friday Table No. 16

3 Challenges Faced

3.1 Conversion of logic to code

- We know that in C coding, functions for integral and derivative are not directly available to be used for the PID. Hence, We needed to use equivalent arithmetic operations to make the PID
- We looped our algorithm over small intervals of time (say 5ms) over a period of 1.5 seconds to show 0.5s rise time + 1s settle time
- Now for integral term, we took the summation of errors we calculated and multiplied with the time step and then K_i. We then took a $K_i' = K_i *$ (time step) in our code to simplify logic
- For differential term, we kept track of previous error value calculated, subtracted that from the current value, and divided it by time step. Similar to above logic, we took effective $K_d' = K_d * 1/(\text{timestep})$

3.2 Non-linear Region

- The non-linear region of the motor came at a random angle region.
- But it was identified using the sensor readings itself (region where sensor reading transitioned from 1023 to 0)
- The algorithm written was keeping this in mind, as to never go through this region

3.3 Tuning

- After implementation of logic, obviously the controller wasn't going to work out of the blue. Tuning was necessary
- We calculated approximate orders of the coefficients using root locus and simulated it on Matlab to verify our values
- Seeing it work out, we implemented it in our code, finding the effective K_i' and K_d' for our purpose
- Obviously the values calculated were for an ideal scenario, and we had to further tune it (not by orders of magnitude, just slightly) to get appropriate functionality

3.4 Debugging

- This includes both code and circuit debugging
- The code was debugged using the debugger provided by arduino IDE itself
- For the circuit, we used many equipments like DMM, sample inputs from supply for detection, etc. to figure out the core problem

Friday, Table No. 16

4 Observations, Inferences and Results

4.1 Observation

The motor turned 180° with the following observations

- **Rise Time**: 440ms
- **Settling Time**: 843ms
- **% Overshoot**: 2.8%

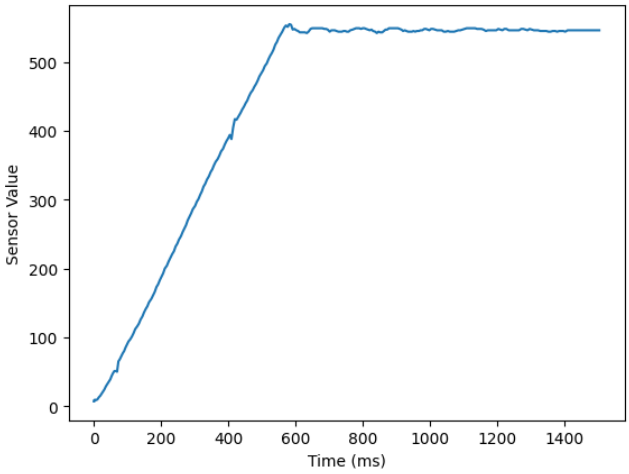
The above satisfy the question constraints

4.2 Inferences

- The experiment perfectly rotated the motor by 180°
- The motor never entered the non-linear region
- All constraints were satisfied

4.3 Results

The following graph shows the variation of sensor value with time. The rise time, settle time and % overshoot can also be observed from the same



Sensor Readings vs Time

Friday Table No. 16