`

DC Motor Position Control

Date: 2019. 04. 23
Team: 9
Student ID: 21600372
Name: 송윤경

1. Purpose

이번 프로젝트의 목적은 DC 모터의 모델링 및 컨트롤러를 설계하는 것이다. DC 모터를 수학적으로 모델링하여 모터를 회로 및 Arduino 와 연결하여 구동하고, MATLAB 을 통해 모터를 실시간으로 계측한다. 보상기와 PID 컨트롤러 두가지 방법으로 접근하여 컨트롤러를 설계하고 최적의 컨트롤러를 제시하고자 한다.

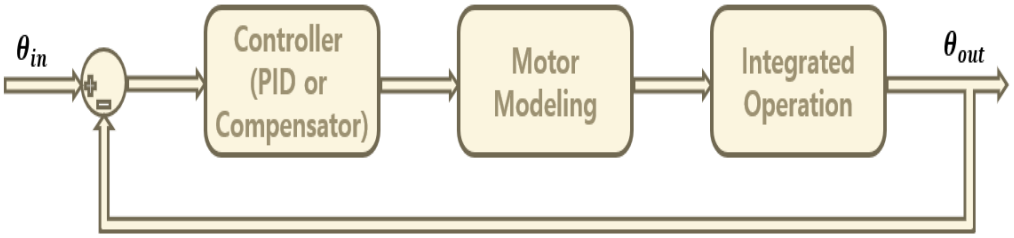
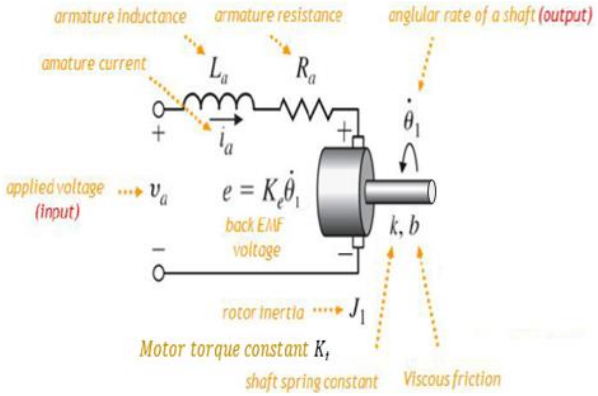


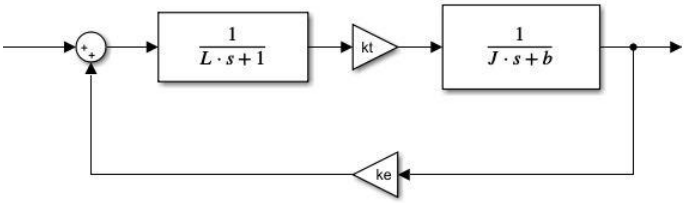
Figure 1. Motor Control System

2. Motor System Modeling

2.1. Mathematical Modeling



(a) Actual Model



(b) Block Diagram

Figure 2. Motor Diagram

· **Electrical domain**

$$-v_a(t) + L_a\frac{di_a(t)}{dt} + R_a i_a(t) + e_m(t) = 0$$

$$e_m(t) = ke\ \omega(t)$$

By Lapalace transform,

$$v_a(s) = (sL_a + R_a)I_a(s) + E_m(s)$$

$$E_m(s) = ke\ \omega(s)$$

· **Mechanical domain**

$$-\tau(t) + J\frac{d\omega(t)}{dt} + b\omega(t) = 0$$

$$\tau(t) = kt\ i(t)$$

By Lapalace transform,

$$-\tau(s) + J_1 s\omega(s) + b\omega(s) = 0$$

$$\tau(s) = kt\ I(s)$$

· **Transfer function**

$$\mathrm{G(s)} = \frac{\omega(s)}{V_a(s)} = \frac{k_t}{JL_a s^2 + (JR_a + L_a b_m)s + k_e k_t + R_a b}$$

기계보다 전기의 반응이 훨씬 빠르므로, $L_a = 0$ 으로 근사시킬 수 있다.

$$\therefore \mathrm{G(s)} = \frac{k_t}{JR_a s + R_a b + k_e k_t}$$

2.2. Design DC Motor System

2.2.1. Hardware System

- **Gearbox**

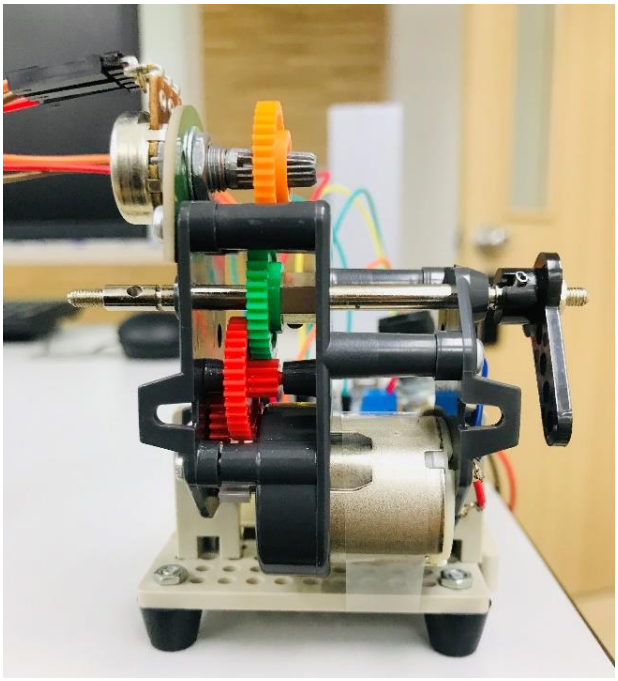


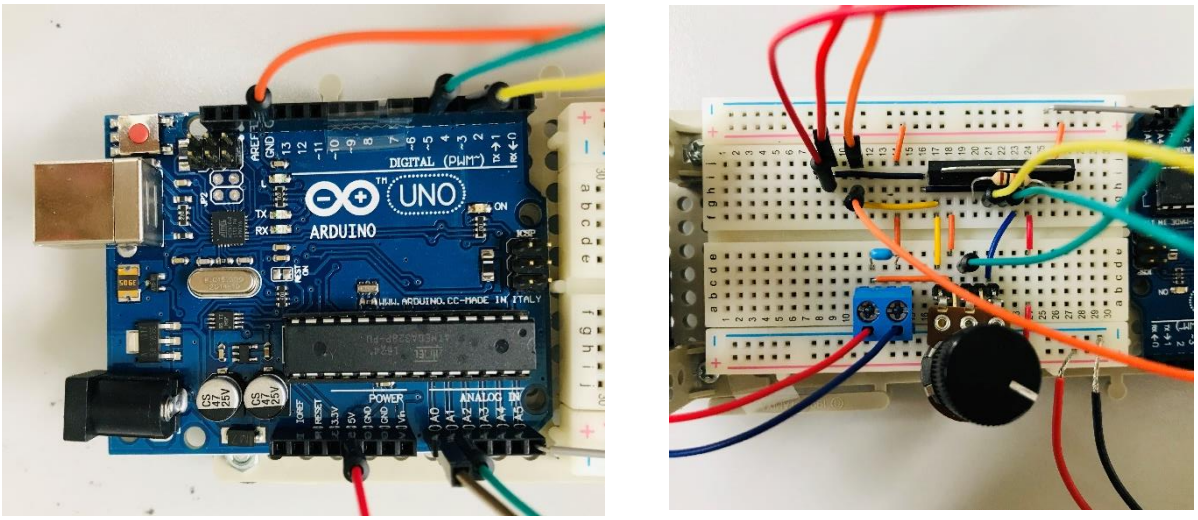
Figure 3. Motor Gearbox

모터는 전기에너지를 회전에너지로 바꾸는 역학을 하며, 모터 내부에 감긴 코일을 통해 전해지는 전기에너지가 자기장의 의해 회전하는 원리를 이용한 것이다. 보통 수만 번 회전하게 되기 때문에 감속기가 필요하다. 외부의 힘이 그대로 전달되면 모터의 수명이 단축하고 부품의 마모도가 증가하여 결국 파괴된다. 감속기는 이러한 문제를 해결하기 위해 사용된다. 감속기는 기어로 이루어져 있는데. 기어비로 인해 모터의 회전을 변속 시키는 것이 목적이다.

- **Two potentiometers**

이번 실험에서 두개의 가변저항을 사용한다. 먼저, 회로에 있는 가변저항은 처음 회로를 만들 때 회로를 확인하는 용도로 회로를 만들고 가변저항을 변화시키며 가변저항의 변화에 의해 바뀌는 출력을 따라오는지 확인하는 용도로 사용되었다. 모터에 있는 가변저항은 모터의 각도를 제어하는 역할을 하며, 가변저항의 값에 따라 모터 회전할 수 있는 각도가 변화한다.

- **Arduino and Peripheral Circuit**



(a) Arduino (b) Peripheral Circuit

Figure 4. Arduino and Peripheral Circuit

아두이노는 오픈소스를 기반으로 한 단일보드인 마이크로컨트롤러로 완성된 보드이다. 아날로그 입력과 디지털 입출력을 통해 하드웨어 회로를 만들지 않고도 프로그램을 통해 쉽게 시스템을 제어할 수 있도록 고안되었다. 아날로그입력을 넣어 아두이노에서 자동적으로 ADC를 변환하여 디지털 출력을 받는다. 아두이노의 주변회로에서 사용한 소자인 TA7291은 모터드라이버로 입력의 신호에 따라 모터의 회전방향과 모터의 동작상태를 결정한다.

- **Constructed Circuit**

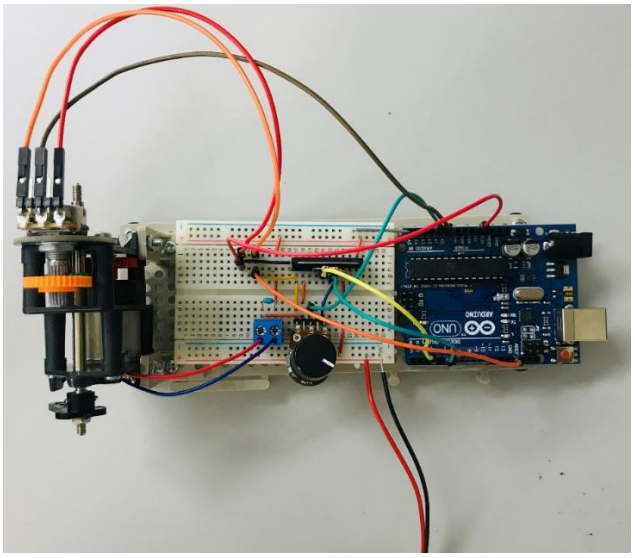


Figure 5. Arduino and Peripheral Circuit

2.2.2. Software System

```
freq=20;
ts=1/freq;              %sampletime
r_cyc=4000*ts;          %Pulse Period
Ncyc=8*4;               %Number of Pulse
tfinal=r_cyc*Ncyc;      %total time
l=tfinal/ts;            %total sample
r=60;                   %Motor angle move range(+-30)
Kp_d=10;                %gain
n=32;
```

Figure 6. Input Parameters in MATLAB

모터 제어를 하기에 앞서 결정한 파라미터 값은 figure 6과 같다. SIMULINK상에서의 시간이 실제 시간과 맞도록 설정해주기 위해 시뮬레이션을 돌려가며 주파수를 찾고, 그 주파수를 사용하여 샘플링 시간간격을 설정하였다. 다른 값들은 시간을 받기위한 파라미터 값이다. 모터의 움직임의 각도를 각 방향으로 30° 만큼 움직이게 제한하였다. K_{p_d}값은 스스로 발진하도록 설정하여 진동이 발생하도록 한다. 진동을 통해 시스템의 특성을 알 수 있어 전달함수를 추정하기 위해 진동하도록 10으로 설정하였다. 또한 이 값을 포함한 채로 모델링을 한다.

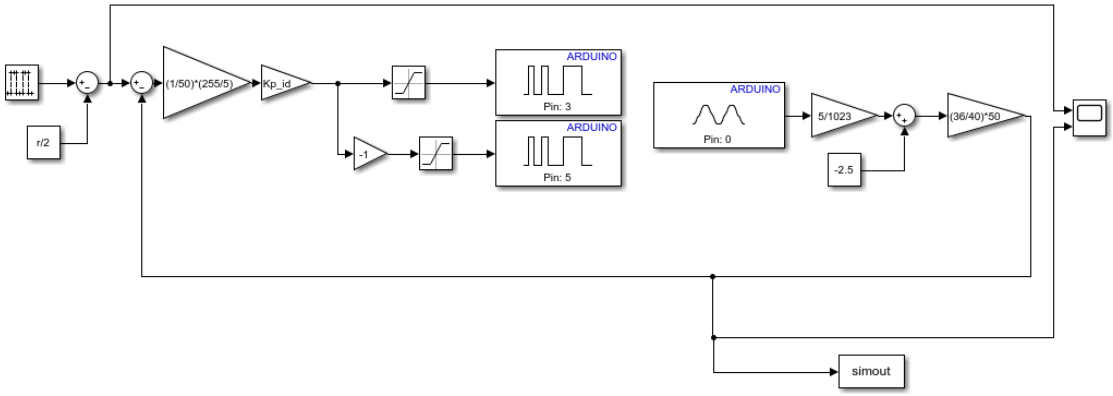


Figure 7. Simulink Block Diagram

아두이노를 연결하여 모터를 작동하기 위한 Block diagram이다. 아두이노에서 아날로그 입력 값을 전압 값으로 변환 시키기 위해 5/1023의 이득을 넣어주고, $(36/40) \times 50$의 이득은 우리가 사용한 기어비로 초록색 기어이가 40, 주황색 기어이가 36, 가변저항에서 전압과 각도를 $50[\mathrm{deg}]/1[\mathrm{V}]$로 변환한다.

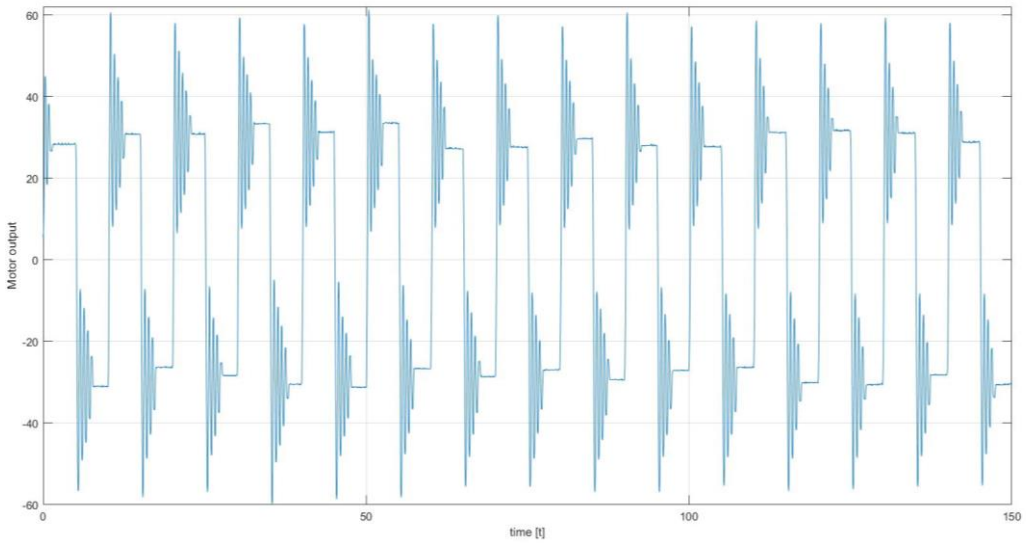


Figure 8. Motor Output

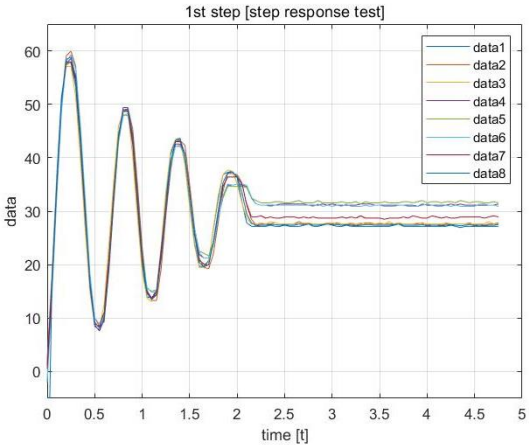
Figure 7 에서 그린 Block diagram을 통해 아두이노로부터 데이터를 받은 모터의 출력이다. 이처럼 진동을 많이 발생시켜 모터의 특성을 통해 전달함수를 추정할 것이다. 시스템이 불안정하기 때문에 진동이 발생하는 횟수가 다르다. 따라서 모터의 출력에서 같은 진동 수를 가지는 데이터를 선별해 하나의 전달함수로 표현하고자 한다.

2.3 Estimation of Transfer Function of Motor

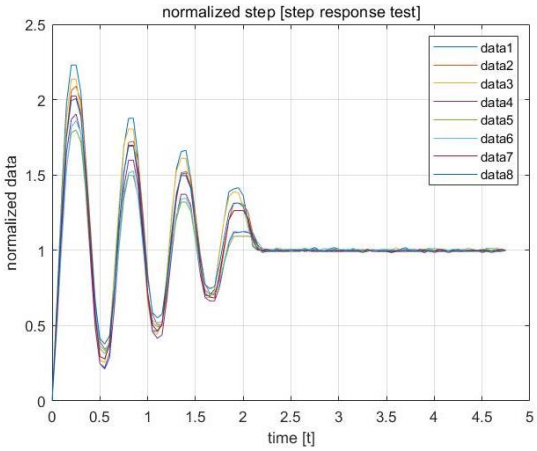
모터의 작동을 전달함수로 표현하여, MATLAB상에서 시뮬레이션을 하고자 한다. 실제 모터작동을 전달함수로 표현하기 위해 모터의 양의 부분을 단위계단입력의 출력으로 보아 모터의 특성을 파악한 후, 최적의 전달함수를 추정하려 한다.

2.3.1. Motor Output Data Processing

PWM의 파형에서 양의 부분만 보기 위해 출력의 행렬에서 양의 부분만을 사용하여 그래프로 나타냈다. 모터의 출력을 단위계단의 출력으로 표현하기 위해 데이터의 시작 값을 0, 최종 값을 1로 표현해준다. 이 과정을 Normalization이라 한다. 모터의 양의 부분 출력과 Normalization을 한 데이터를 그래프로 그리면 figure 7에서와 같이 나타난다.



(a) Step Data



(b) Normalization data

Figure 9. Step Data and Normalization Data

8개의 데이터의 평균을 구하여 전달함수의 일반적인 작동 출력을 보고자 한다. Normalization 한 데이터의 평균을 통해 모터의 작동을 하나의 일반적인 단위계단의 출력으로 나타낸다.

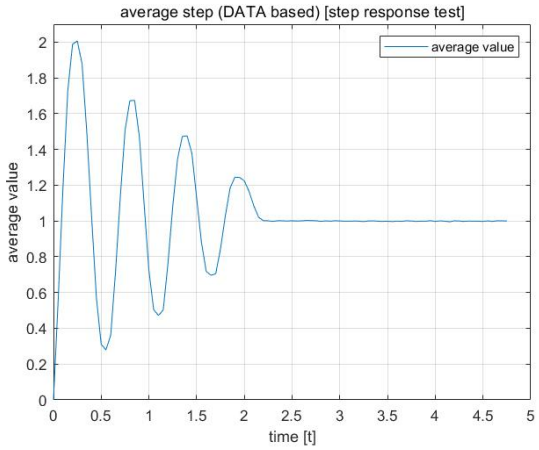


Figure 10. Average Output from Data

2.3.2. Optimization

위에서 구한 일반적인 평균 출력으로 모터의 특성을 알아보려 한다. 모터의 수학적 모델링을 통해 2.1에서 모터의 일반적인 전달함수를 구하면 근사 되어 1차시스템의 전달함수로 표현할 수 있다. 일반적인 1차 시스템의 전달함수는 다음과 같이 표현된다.

$$G_m = \frac{k}{Ts + 1}$$

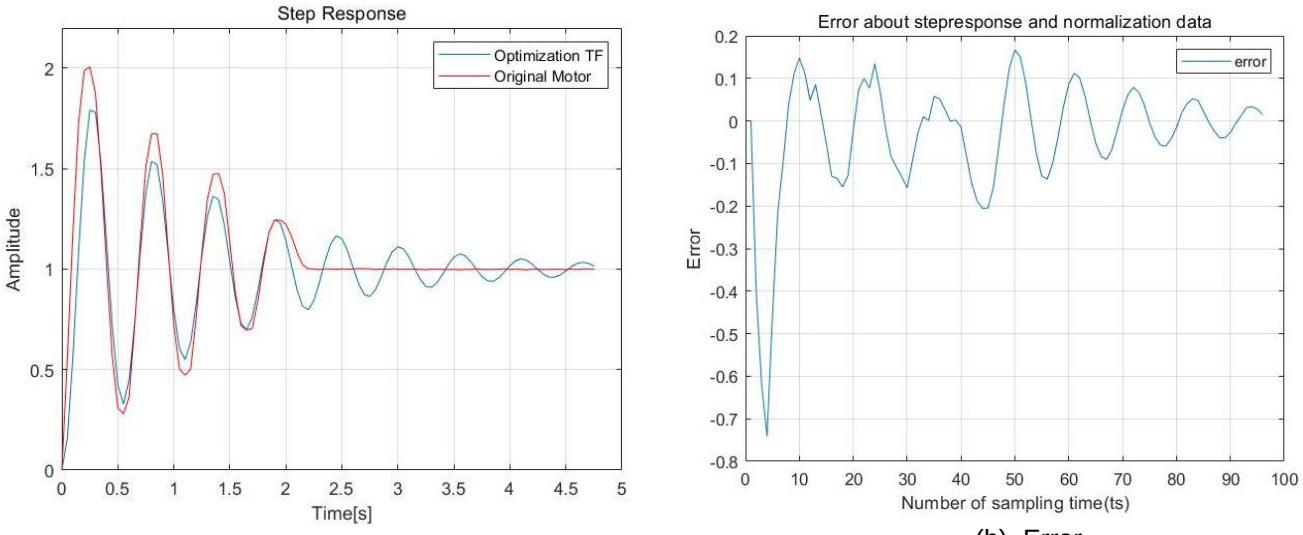
우리는 모터의 출력을 통해 1차 전달함수의 T와 k값을 추정하여 최적화된 모터의 전달함수를 찾으려 한다. (T, k) 값을 찾기 위해 MATLAB에서 최적화된 값을 찾아주는 함수(fminsearch)를 사용하여 (T, k)쌍을 찾는다. Fminsearch를 사용하기 위한 목적함수로 매개변수(T, k)를 가지는 함수를 만들어야 한다. 일반적인 폐 루프 1차 전달함수를 만들어 모터의 출력과 같은 샘플시간을 갖는 함수로 이산화 시킨 후, 이산화 시킨 행렬과 평균모터출력의 차로 오차를 계산한다. 가장 작은 오차 값을 구하기위해 norm으로 계산하여 오차를 구한다. 이 오차행렬을 fminsearch의 목적함수로 넣어 오차가 가장 작을 때의 (T, k) 값을 반환 받는다.

$$(\mathrm{T}, \mathrm{k}) = (0.2024, 141.8790)$$

반환한 T, k값을 전달함수에 넣어 최적화된 모터의 전달함수를 구하면 다음과 같이 표현된다.

$$G_m = \frac{1419}{0.2024s + 1}$$

추정된 전달함수의 단위계단함수 응답과 Normalization한 실제 모터작동의 평균값을 한 그래프로 나타내고 오차를 계산해보면 figure11와 같이 나온다.



(a) Step Response (b) Error

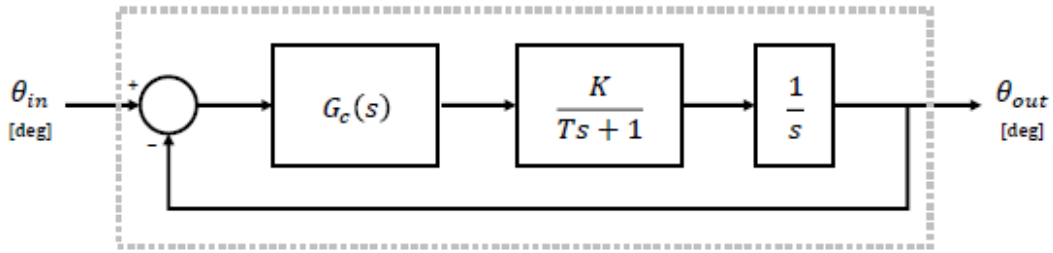
Figure 11. Optimization Transfer function

figure 11 (a)에서 전달함수의 단위계단응답이 모터의 작동을 추정함을 보인다. 이를 오차 계산해보면 최소 0에서 최대 -0.74까지의 오차를 갖고, 평균을 구하면 약 -0.0342의 오차를 갖는다. 이는 우리가 해석하는 s 영역에서 해석하는 시스템은 LTI(Linear Time Invariant) 시스템을 가정하는데, 실제 모터의 동작은 비선형적으로 작동하여 발생하는 것이다. 완벽하게 실제 모터를 추정할 수 없으므로 오차를 허용하여 전달함수가 유효하다고 간주한다.

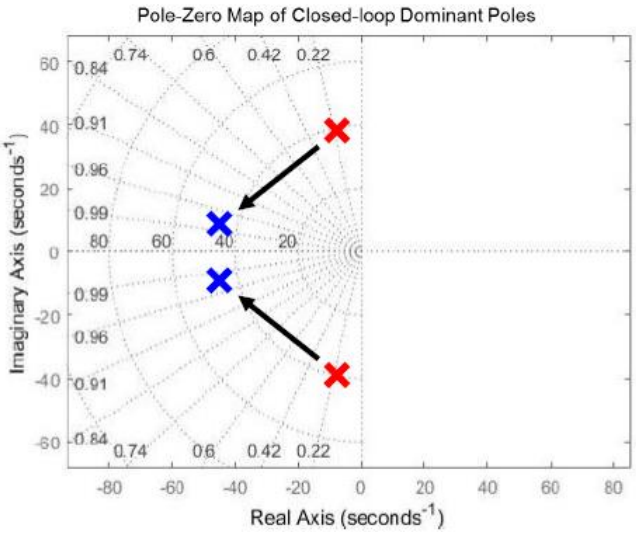
3. Motor Controller Design

3.1. Preliminary Concerning Controller Design

제어기는 보통 제어하고자 하는 플랜트의 앞에 달아주어 시스템을 안정하게 하는 목적을 가진다. 일반적으로 제어기를 설계하는 방법은 pole과 zero의 위치를 이동시켜 원하는 시스템의 특성을 만들어 주는 것이다. pole과 zero는 전달함수에서 특성방정식에 의해 구해지는 해로, pole과 zero에 의해 시스템의 특성이 결정된다. 제어기를 설계하여 시스템을 특성을 바꿔 엔지니어가 원하는 사양을 가진 특성을 가진 시스템이 된다.



(a) Block diagram with Controller



(b) Pole's Movement

Figure 12. Basic Concept of Controller

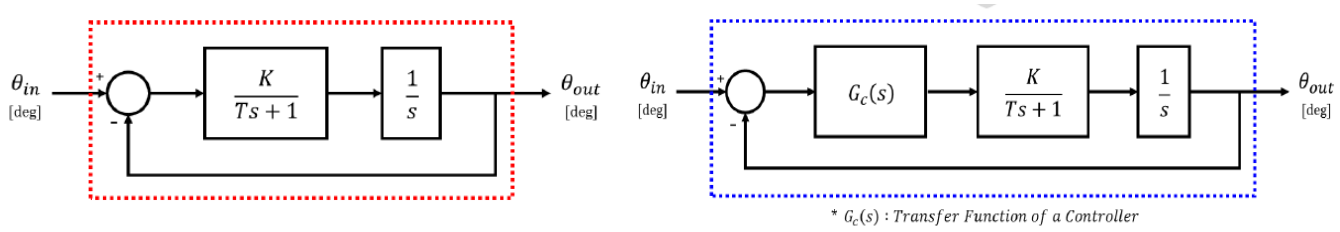
3.2. Compensator

3.2.1. Design Concept

보상기는 시스템의 pole과 zero를 상쇄하여 원하는 pole의 위치를 구현하는 컨트롤러로 원래 전달함수의 pole과 zero를 상쇄하고 새로운 pole과 zero를 생성하여 시스템의 안정도를 높인다. 1차시시스템의 보상기는 다음과 같이 정의된다.

$$G_c = \frac{k_c(Ts+1)}{T_c s+1}$$

보상기의 분자는 원래 전달함수의 특성방정식을 분자에 넣어 기존의 pole, zero를 상쇄시켜주고 새로운 파라미터인 T_c, k_c값을 통해 시스템의 특성을 결정한다.



(a) Without Compensator

(b) With Compensator

Figure 13. Compensation Block Diagram

보상기를 넣은 후 폐 루프 시스템의 전달함수는 다음과 같이 구해진다.

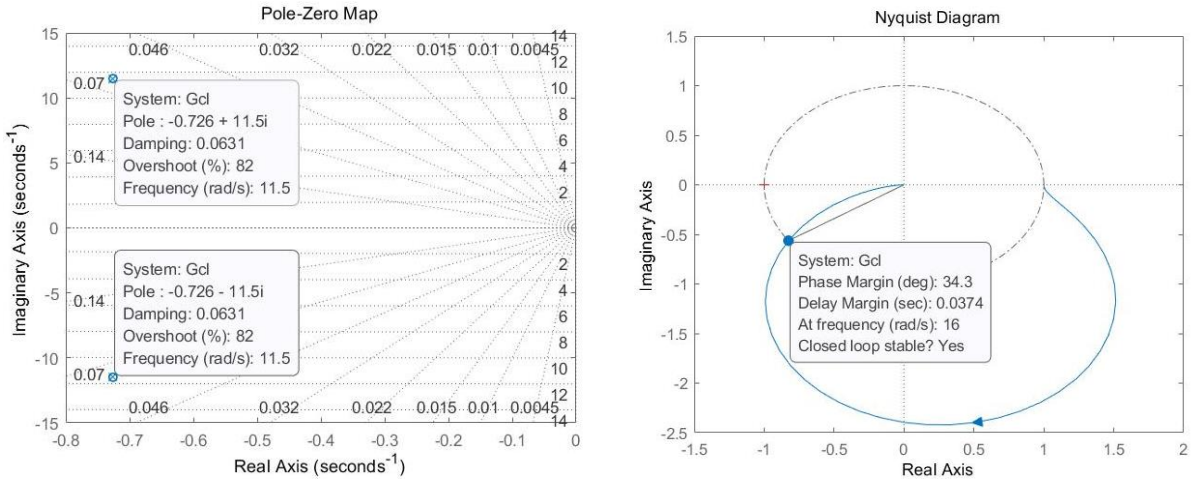
$$G_{cl}(\mathrm{s}) = \frac{\frac{k_c(Ts+1)}{T_c s+1} \times \frac{k}{s(Ts+1)}}{1+\left(\frac{k_c(Ts+1)}{T_c s+1} \times \frac{k}{s(Ts+1)}\right)} = \frac{k_c k}{T_c s^2 + s + k_c k}$$

$$\therefore G_{cl}(\mathrm{s}) = \frac{k_c k}{T_c s^2 + s + k_c k}$$

전달함수 (H(s))를 보면 T_c, k_c값을 통해 임의로 pole의 위치를 개선할 수 있음을 알 수 있다.

3.2.2. Control Specification

원하는 모터의 사양을 결정하기 전에 먼저 모터의 사양을 알아보려 한다. 추정한 전달함수를 통해 본 모터의 pole-zero위치와 Nyquist plot이다.



(a) Pole-zero map

(b) Nyquist plot

Figure 14. Original Motor Specification

Pole-zero map을 통해서 모터의 pole의 위치를 확인할 수 있다.

$$\mathrm{p1} = -0.726 + \mathrm{j}11.5, \qquad \mathrm{p2} = -0.726 - \mathrm{j}11.5$$

지금의 모터보다 더 안정하게 시스템이 작동하기 위해서는 pole의 실수부가 -0.726보다 작아야 하고, 허수부의 절대값이 11.5보다 작아야 한다.

Nyquist plot을 통해 gain margin과 phase margin을 확인하여 모터의 안정도를 확인한다.

$$\text{Gain Margin} = \infty[\mathrm{dB}], \qquad \text{Phase Margin} = 34.4[\text{degree}]$$

Gain margin은 무한대이므로 매우 안정하여 제어기를 설계할 때 고려하지 않고 phase margin은 -180°에서 멀어질수록 시스템이 안정해지므로 여유를 더 두어 phase margin이 60°보다 크게 만들고자 한다.

3.2.3. Designed Compensator

설계를 하기 위해 각 파라미터들을 일반적인 2차전달함수의 기본형태로 표현해준다. 2차 전달함수의 일반적인 형태는 다음과 같다.

$$G(\mathrm{s}) = \frac{k{\omega_n}^2}{s^2 + 2\zeta\omega_n s + {\omega_n}^2}$$

위의 3.2.1에서 구한 보상기가 달린 폐 루프 시스템의 전달함수를 ζ, ω_n로 나타내려 한다.

$$G_{cl}(\mathrm{s}) = \frac{k_c k}{T_c s^2 + s + k_c k} = \frac{\frac{k_c k}{T_c}}{s^2 + \frac{1}{T_c}s + \frac{k_c k}{T_c}} = \frac{\frac{k_c k}{T_c}}{(s - p_1)(s - p_2)}$$

$$T_c = -\frac{1}{p1 + p2}, \qquad k_c = \frac{T_c}{\mathrm{k}} p_1 p_2$$

$$p_1 = -\zeta\omega_n + j\omega_n\sqrt{1 - \zeta^2}, \qquad p_2 = -\zeta\omega_n - j\omega_n\sqrt{1 - \zeta^2}$$

$$\therefore\ T_c = \frac{1}{2\zeta\omega_n}, \qquad k_c = \frac{T_c}{k} p_1 p_2$$

가장 이상적인 ζ의 값인 $\zeta = 0.707[-]$로 ζ를 고정한 후, 시스템의 특성이 좋아지는 방향으로 ω_n의 범위를 설정하여 ζ, ω_n값을 결정하는 방향으로 보상기를 설계하였다. 먼저, pole의 실수부인 $-\zeta\omega_n$은 모터의 pole의 실수부보다 작아야 한다. 이때의 ζ는 고정되어 있으므로 ω_n의 최소범위가 결정된다.

$$-\zeta\omega_n < -0.726$$

$$\therefore\ \omega_n > 1.027[\mathrm{rad/s}]$$

pole의 허수부인 $\omega_n\sqrt{1 - \zeta^2}$은 모터의 pole의 허수부인 11.5보다 작아야 하므로 ω_n의 최대범위 또한 결정된다.

$$\omega_n\sqrt{1 - \zeta^2} < 11.5$$

$$\therefore\ \omega_n < 16.261[\mathrm{rad/s}]$$

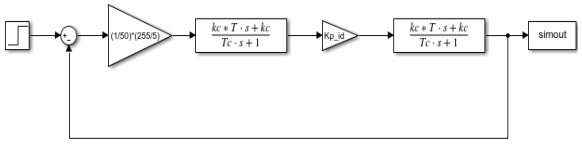
이 결과로 인해 ω_n의 범위가 $1.027 < \omega_n < 16.261$으로 결정된다. ω_n의 값이 너무 작으면 전달함수가 모터를 따라가지 못하므로, $\omega_n = 8[\mathrm{rad/s}]$ 로 설계하였다.

$$\therefore\ T_c = 0.0884, \qquad k_c = 0.1994$$

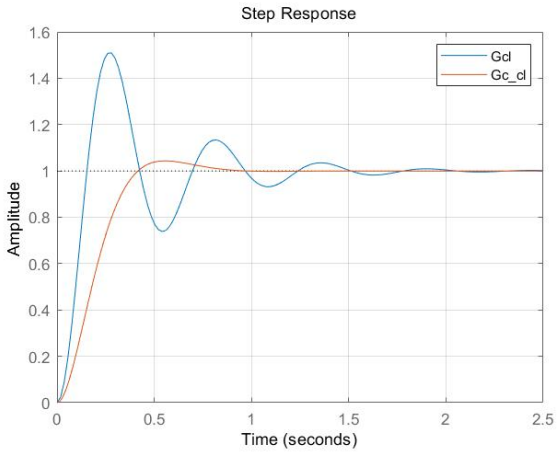
$$\therefore\ Gc(\mathrm{s}) = \frac{0.04036 + 0.1994}{0.0884s + 1}$$

위에서 구한 보상기의 전달함수와 모터의 전달함수로 피드백을 받아 폐 루프 시스템의 전달함수를 구할 수 있다.

$$Gc_{cl}(s) = \frac{1.145s + 5.5658}{0.01789s^3 + 0.2908s^2 + 2.145s + 5.658}$$



(a) Block diagram



(b) Step response

Figure 15. Motor Specification with Compensator

보상기가 달린 폐 루프 전달함수(Gc_{cl})와 모터의 폐 루프 전달함수(G_{cl})의 단위계단 응답을 비교하며 시스템의 응답 특성의 변화에 볼 수 있다. Figure 14. (a), (b)를 보며 두 시스템의 응답 특성을 비교하면 Rising time은 0.1029[s]에서 0.2686[s]으로 0.1657[s]증가하였고, Setting time은 1.4440[s]에서 0.7454[s]로 0.6965[s]만큼 감소한다. 또한 overshoot이 50.9154[%]에서 4.3251[%]로 46.5903[%] 감소함을 보인다. 따라서 rising time은 증가했지만 진동을 잡아주어 overshoot이 작이지고, 시스템이 안정한 범위에 들어가는 setting time이 감소하였으므로 시스템이 전보다 안정됨을 보인다.

RiseTime: 0.1029
SettlingTime: 1.4440
SettlingMin: 0.7383
SettlingMax: 1.5092
Overshoot: 50.9154
Undershoot: 0
Peak: 1.5092
PeakTime: 0.2796

(a) G_{cl} response characteristic

RiseTime: 0.2686
SettlingTime: 0.7454
SettlingMin: 0.9036
SettlingMax: 1.0433
Overshoot: 4.3251
Undershoot: 0
Peak: 1.0433
PeakTime: 0.5537

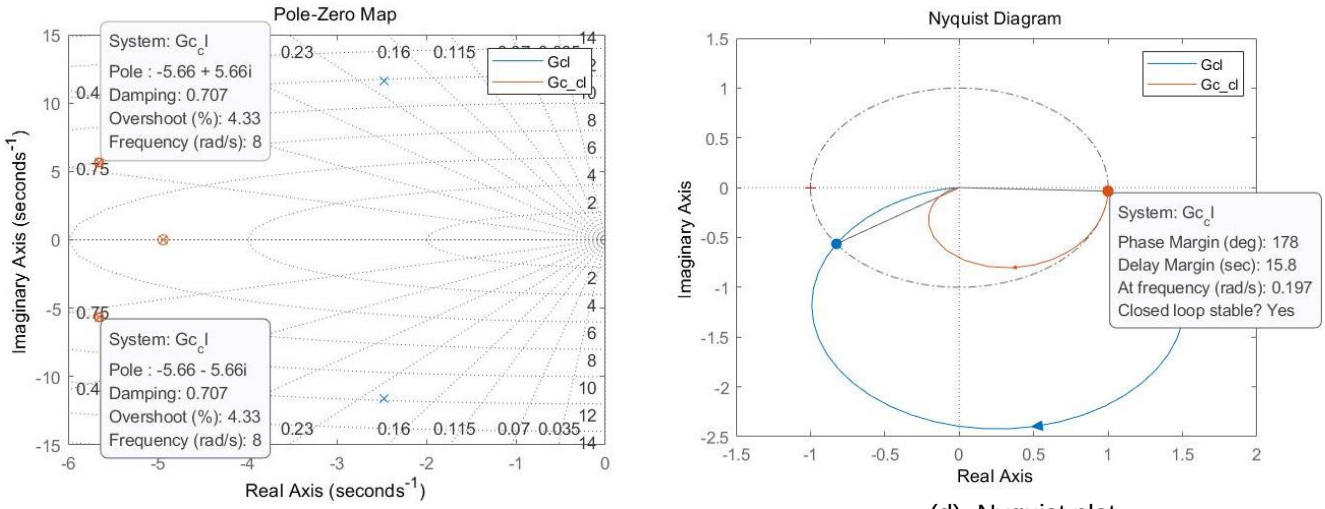
(b) Gc_{cl} response characteristic

Figure 16. Motor Specification with Compensator

보상기가 달린 폐 루프 전달함수(Gc_{cl})와 모터의 폐 루프 전달함수(G_{cl})의 pole-zero map과 Nyquist plot을 통해 control specification에서 설정한 조건을 만족하는지 확인하려 한다. Gc_{cl}의 전달함수에서 pole-zero의 값을 계산한다.

$$\mathrm{p1} = -5.66 + \mathrm{j}5.66, \qquad \mathrm{p2} = -5.66 - \mathrm{j}5.66, \qquad \mathrm{z} = 4.94$$

pole의 실수부를 보면 원래 모터의 pole의 실수부보다 4.943만큼 작아졌고, pole의 허수부는 모터의 pole의 실수부보다 5.84만큼 작아졌다. pole-zero map에서 zero가 폐 루프 모터 전달함수의 pole을 상쇄하는 것을 보인다. Nyquist plot에서 gain margin은 무한대이고 phase margin은 178°임을 보여 원했던 모터의 사양인 60°보다 크다. 따라서, 3.2.2.에서 설정한 모터의 control specification을 만족한다.

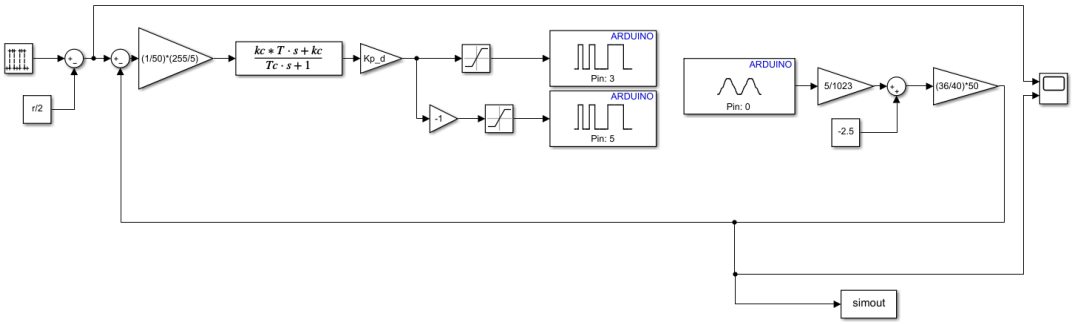


(c) Pole-zero map (d) Nyquist plot

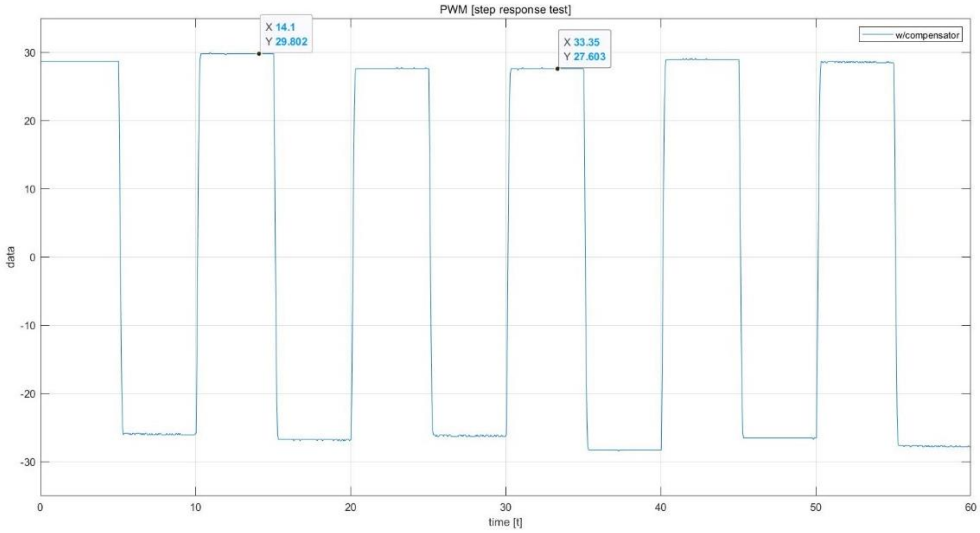
Figure 17. Motor Specification with Compensator

3.2.4. Performance Analysis

설계한 보상기를 달아준 Block diagram을 SIMULINK에 그리고 ADUINO에 연결하여 모터로부터 실제의 출력을 받으면 다음과 같이 출력된다.



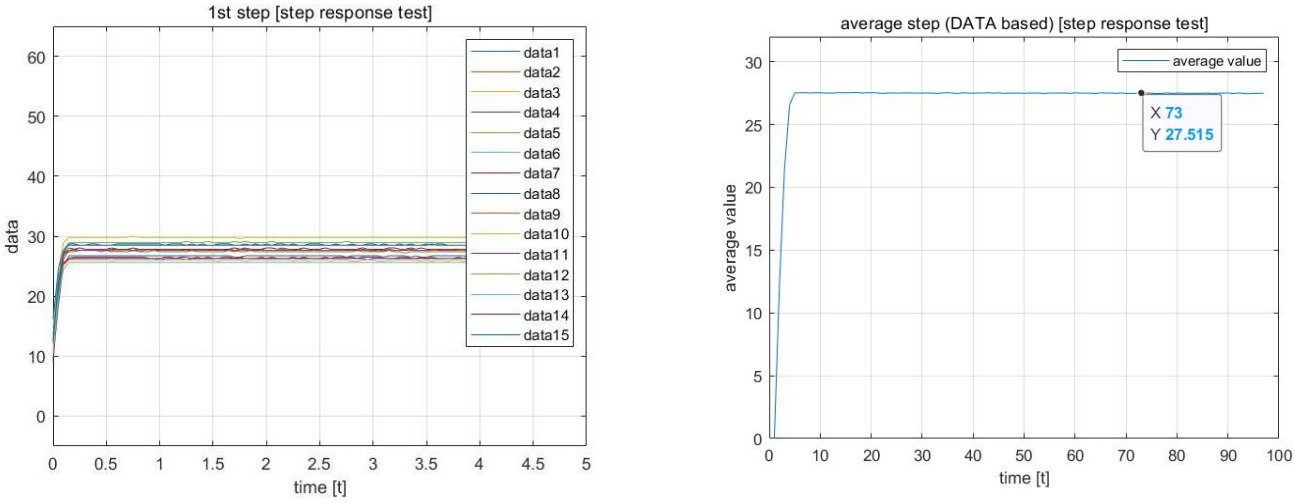
(a) Block diagram



(b) Motor output with Compensator

Figure 18. Motor with Compensator

직관적으로만 봐도 보상기를 달기 전보다 시스템이 안정된 것이 보인다. 시스템을 정확하게 분석하기위해 전달함수를 추정할 때 와 같이 여러 PWM의 positive부분만을 보아 step 입력처럼 보려 한다. 데이터의 신뢰성을 위해 총 15번의 step을 받아 평균을 구해 step입력을 보려 한다.



(a) data (b) average data

Figure 19. Motor Step Response with Compensator

15개의 데이터로 평균을 구한 데이터로 분석하면 진동이 억제되어 settling time이 작아 시스템이 steady state 상태에 진입하는 시간이 빨라진다. steady state상태에서의 출력은 27.515[degree]로, 시스템이 목표 값인 30[degree]까지 따라가지 못함을 보이고 있다. PRE를 계산하면 -8.33[%]의 오차를 가진다. 하지만 앞에서 언급한 것처럼 우리는 선형을 가정하고 시스템을 분석하지만 실제 모터는 dead zone과 같은 비선형적 특징을 가지기 때문에 처음 전달함수를 추정한 데이터조차 steady state error가 존재하는 시스템이다. 따라서, 실제 모터에서 보상기의 유무에 따른 출력을 통해 거시적으로 보면 시스템의 출력이 안정되었다고 할 수 있다.

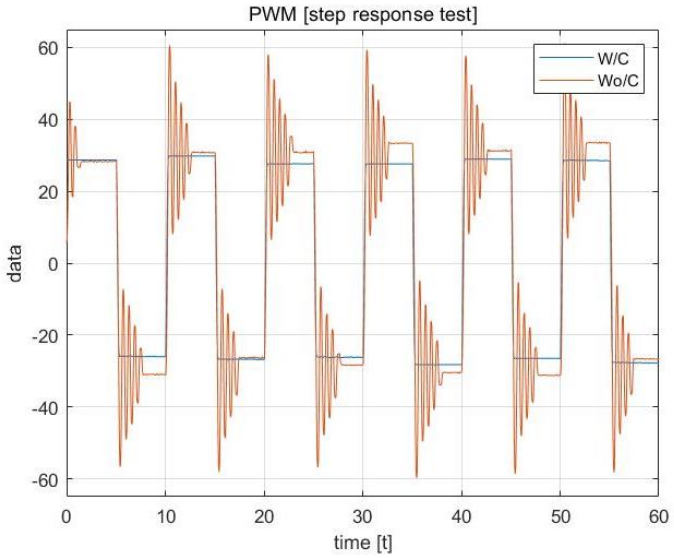
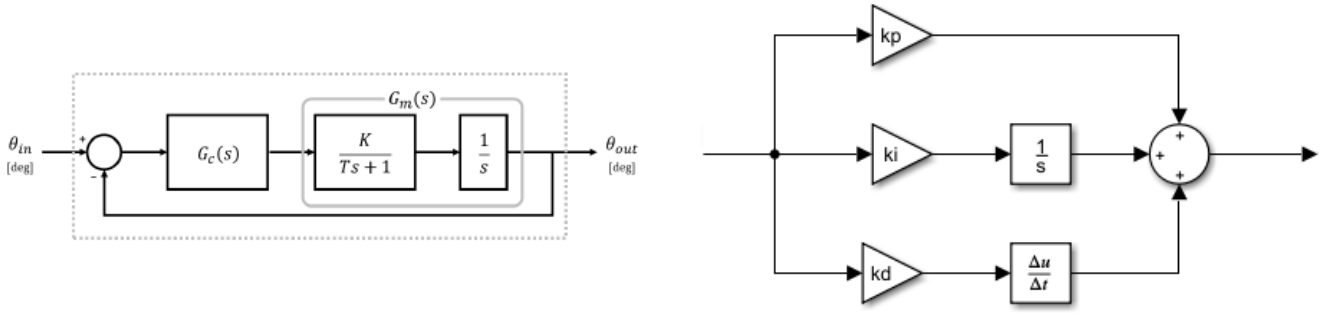


Figure 20. Actual Motor Output

3.3. PID Controller Design using Ziegler-Nichols Method

3.3.1. Design Concept

PID 제어기는 K_P, K_I, K_D 의 이득 값과 적분기와 미분기를 사용하여 pole-zero의 위치를 변화시켜주는 제어기이다. PID 제어기 역시 pole의 위치를 변화시켜 원하는 시스템의 특성을 갖게 만들어 준다. P제어기는 proportional operation으로 응답속도를 빠르게 향상시켜주고, I제어기는 Integrated operation으로 steady state error를 개선해준다. D제어기는 derivative operation으로 진동을 억제해준다.



(a) Block Diagram

(b) Internal structure

Figure 21. PID Controller

$$G_c(s) = K_P + \frac{K_I}{s} + K_D s = \frac{s^2 K_D + sK_P + K_I}{s}$$

$$G_{cl}(s) = \frac{G_c(s)G_m(s)}{1 + G_c(s)G_m(s)} = \frac{K_D K s^2 + K_P K s + K_I K}{Ts^3 + (K_D K + 1)s^2 + K_P K s + K_I K}$$

3.3.2. Designed PID Controller

Ziegler-Nichols Method의 방법은 K_P, T_I, T_D 이 세개의 파라미터를 T_C, K_C의 값을 통해 표로 정리하여 일반적인 상황에서 사용되는 각각의 K_P, T_I, T_D값을 찾는다. 이때의 T_C, K_C는 모터의 K_P값을 시스템이 진동할 때까지 증가시켜 응답에서 나타나는 파라미터로 모터의 특성을 나타내는 값이다. T_C는 진동하는 주기를 의미하고, K_C는 시스템의 목표 값을 의미한다.

	K_P	T_I	T_D
P	$0.5K_C$	–	–
PI	$0.45K_C$	$0.85T_C$	–
PID	$0.6K_C$	$0.5T_C$	$0.125T_C$

Figure 22. Ziegler-Nichols Method Table

3.4. Optimal Controller Suggestion

두가지 컨트롤러 설계방법을 비교하고 어느 것이 더 최적의 컨트롤러인지 선정하려 했으나 이번 프로젝트에서 PID제어기를 실험하지 못해 비교하지 못하였다. 그러나 보상기의 Nyquist plot을 통해 선도가 단위 원 안으로 들어온 것으로 보아 시스템의 안정되었고, 실제로 아웃풋을 통해 안정하게 제어되는 것을 보았다. 보상기가 Optimal Controller라고 할 수 있다.

5. Conclusion

아두이노를 사용하여 모터를 제어하는 프로젝트를 진행하였다. 시스템의 전달함수를 알지 못하기 때문에 블랙박스 모델링 즉, 모터의 입력과 출력의 특성을 비교하여 전달함수를 추정하였다. 추정한 전달함수를 이용하여 pole의 위치를 변경하는 보상기를 직접 설계하여, 보상기를 달기 전과 후의 출력을 비교했다. 보상기의 유무가 시스템 안정에 크게 기여함을 보였고, 프로젝트를 통해 제어기의 목적과 제어기 설계의 흐름을 보았다.