3.1 DC motor PID control

PID Controller

비례(P)제어기

-기준신호와 귀환신호 사이의 차인 오차신호에 적당한 비례상수 이득을 곱해서 제어신호를 만들어내는 제어기법.
-구성이 간단함
-이득의 조정만으로는 시스템의 성능을 여러 가지 면에서 함께 개선시키기는 어려움

비례적분(PI)제어기

-오차신호를 적분하여 제어신호를 만들어내는 적분제어를 비례제어와 병렬로 연결하여 사용하는 제어기법
-시스템의 정상상태 오차가 개선되지만, 응답속도가 늦어짐

비례미분(PD)제어기

-오차신호를 미분하여 제어신호를 만들어내는 미분제어를 비례제어에 병렬로 연결하여 사용하는 제어기법
-미분제어는 오차신호의 변화를 억제하는 역할을 하기 때문에 감쇠비를 증가시키고 초과를 억제-과도응답특성을 개선시킬 수 있지만, 정상상태 응답특성은 개선되지 않음

비례적분미분(PID)제어기

례(P), 적분(I), 미분(D) 제어의 세 부분을 병렬로 조합하여 구성하는 제어기 정상상태 응답과 과도상태 응답을 모두 개선할 수 있음

PID CONTROL THEORY

PID control scheme

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

where

K_p: Proportional gain, a tuning parameter

K_i: Integral gain, a tuning parameter

K_d: Derivative gain, a tuning parameter

e: Error $= SP - PV$

t: Time or instantaneous time (the present)

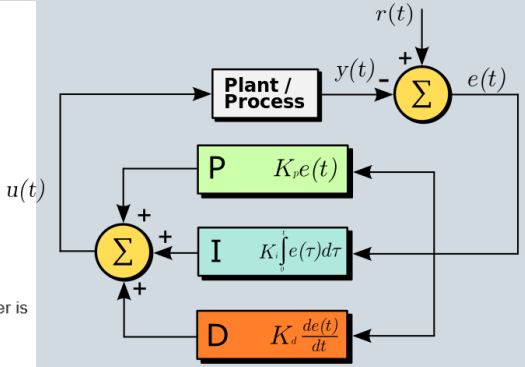
τ: Variable of integration; takes on values from time 0 to the present t.

Equivalently, the transfer function in the Laplace Domain of the PID controller is

$$L(s) = K_p + K_i/s + K_d s$$

where

s: complex number frequency

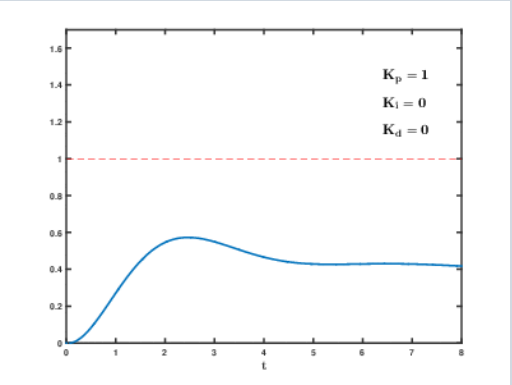


PID CONTROL THEORY

□ P, I, D gains :

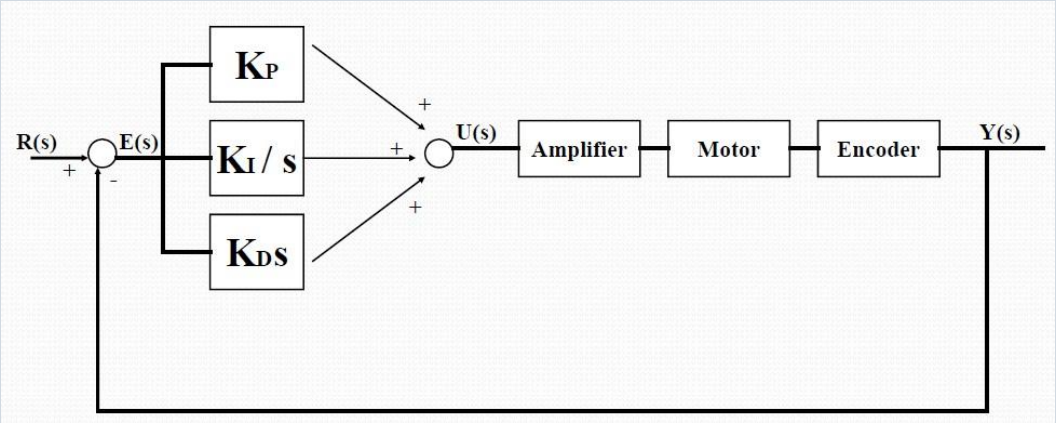
- P gain: 현재 오차값에 지배
- I gain: 누적된 과거의 오차에 지배
- D gain: 미래 오차의 예측, 오차의 차이를 이용

	Rise Time	Over shoot	Settling time	SSE
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Change	Decrease	Decrease	Small Change



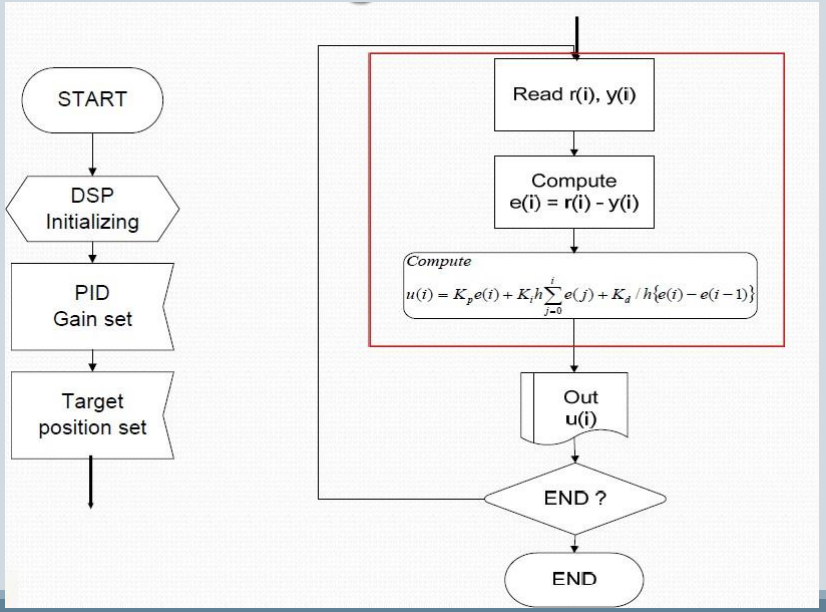
PID control block diagram

PID Controller in a Laplace Domain



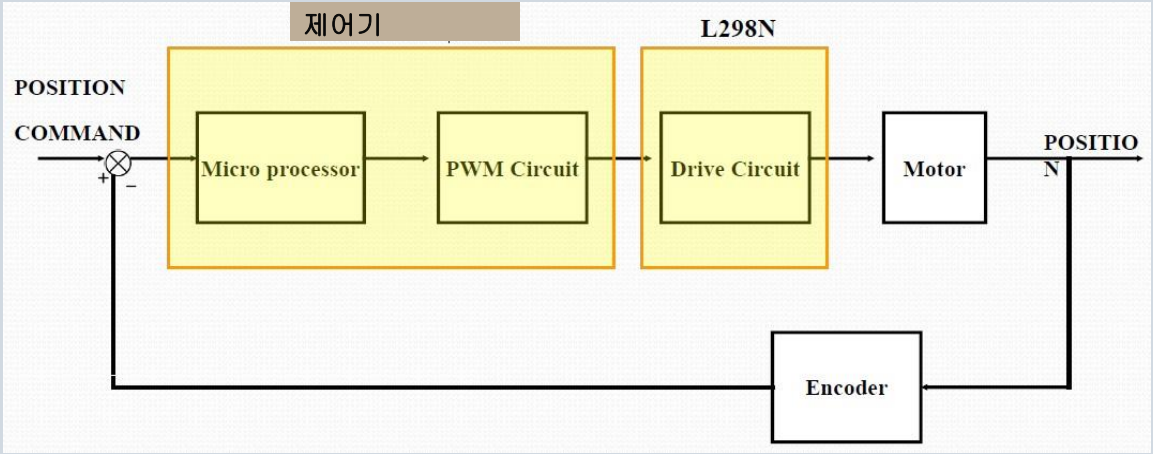
PID algorithm

Flow Chart of Digital PID Controller

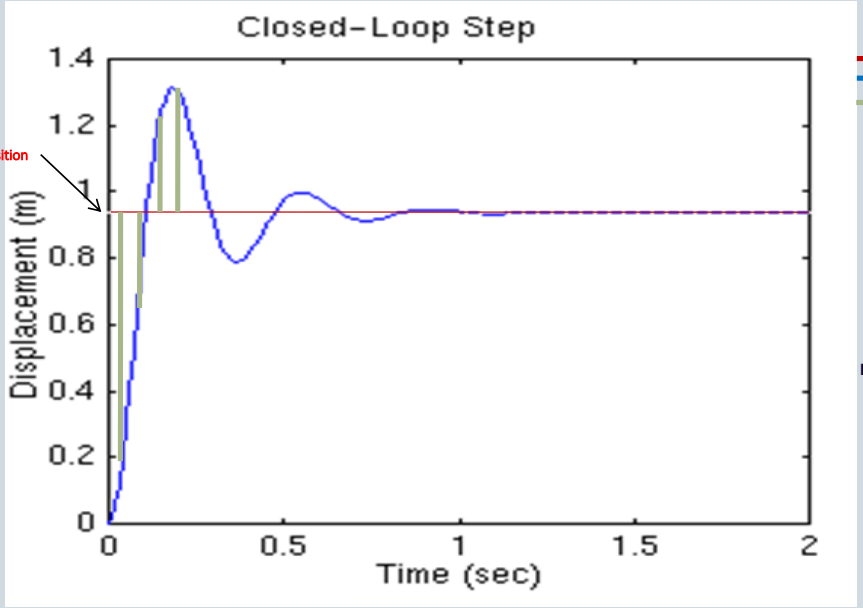


PID control system

System block diagram using digital circuit



PID control performance



Target position
Current position
P_error

P_error = Target position –current position

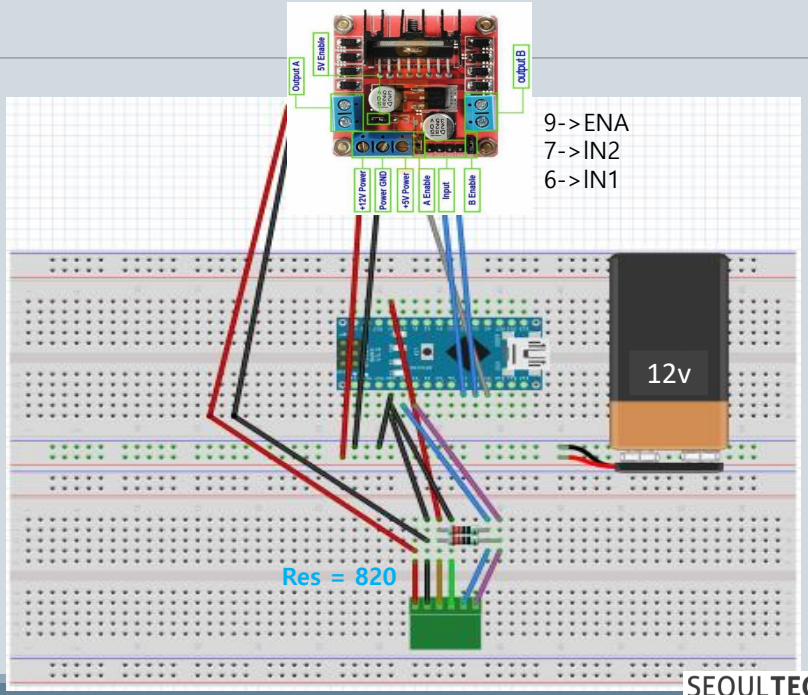
I_error = $\sum$ P_error

D_error =P error(current)- P error(pre)

DC motor PID position Control experiment

☐ PID control

- Line map



DC motor PID position Control codes

□ PID control

```
// motor control pin
const int motorDirPin1 = 6; // L298 Input 1
const int motorDirPin2 = 7; // L298 Input 2
const int motorPWMPin = 9; // L298 Input 3

// encoder pin
const int encoderPinA = 2;
const int encoderPinB = 3;

int encoderPos = 0;
const float ratio = 0.2;

// P control
double Kp = 25;
double Ki = 2;
double termI = 0;
double Kd = 0.05;
float targetDeg = 360;

// Tracks the time since last event fired (sampling time)
unsigned long previousMillis=0;
unsigned long currentMillis=0;
float motorDegPrev=0;
float error;
void doEncoderA(){  encoderPos +=
(digitalRead(encoderPinA)==digitalRead(encoderPinB))?1:-1;}
void doEncoderB(){  encoderPos +=
(digitalRead(encoderPinA)==digitalRead(encoderPinB))?-1:1;}  // 4체배

void doMotor(bool dir, int vel){
  digitalWrite(motorDirPin1, dir);
  digitalWrite(motorDirPin2, !(dir));
  analogWrite(motorPWMPin, min(vel,255));
}
```

```
void setup() {
  pinMode(encoderPinA, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(2), doEncoderA, CHANGE);

  pinMode(encoderPinB, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(3), doEncoderB, CHANGE);

  pinMode(motorDirPin1, OUTPUT);
  pinMode(motorDirPin2, OUTPUT);

  Serial.begin(115200);
}

void loop() {
  while(currentMillis - previousMillis<50)currentMillis = millis();  // 50 msec sampling time
  float motorDeg = float(encoderPos)*ratio;

  error = targetDeg - motorDeg;
  termI += Ki * error * 0.001;
  float control = Kp*error + termI - Kd*(motorDeg-motorDegPrev)/0.001;
  if(abs(error)<1) control=0;
  doMotor( (control>=0)?HIGH:LOW, min(abs(control), 255));

  Serial.print("encoderPos : ");
  Serial.print(encoderPos);
  Serial.print("   motorDeg : ");
  Serial.print(float(encoderPos)*ratio);
  Serial.print("   error : ");
    Serial.print(error);
  Serial.print("    control : ");
  Serial.println(control);
  motorDegPrev = motorDeg;
  previousMillis = currentMillis;
}
```