

# TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018-06-25 (80 회차)

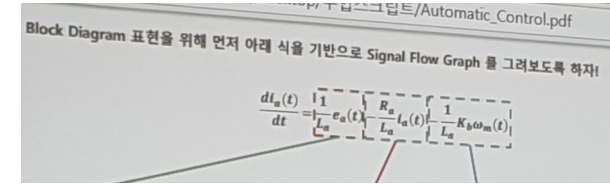
강사 - Innova Lee(이상훈)  
[gcccompil3r@gmail.com](mailto:gcccompil3r@gmail.com)  
학생 - 정유경

## 1. BLDC 모터의 PID 제어

## 2. DSP 영상처리 예제

## PID 제어가 필요한 이유

- 1) 코일은 온도에 따른 특성변화가 크다.
- 2) 엔코더는 무거워서 쿼드콥터에 사용할 수 없다.



## PID 제어란

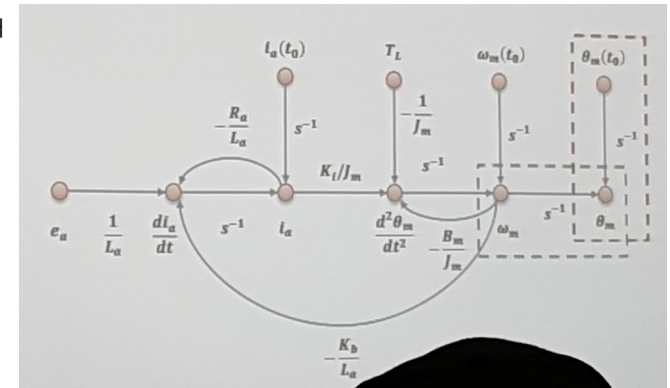
PID 제어는 자동제어 방식 가운데서 가장 흔히 이용되는 제어방식으로 여러 값을 제어하여 목표값에 수렴시킨다.

에러값을 제어하는 방식은 다음과 같은 3 가지가 있다.

P: Proportional(비례), I: Integral(적분), D: Differential(미분)

위와 같은 방식으로 에러를 제어 명령으로 바꾸어 시스템에 전달하게 된다.

위식을 정리한 신호흐름도는 다음과 같다.



## 컴퓨터 상에서 PID 제어를 구현하는 알고리즘

$e(t)$  : 오차값 = 목표값 - 현재값

$MV(t)$  : 제어량, 제어량 + 현재값 = 목표값

$MV(t) = K_p * e(t) + K_i * (\text{integral})e(t) + K_d * (\text{derivative}) * e(t)$

$K_p$ ,  $K_i$ ,  $K_d$  값을 찾는 것이 PID 제어의 핵심이다.

모터는 입력전류의 흐름을 제어하여 속도를 제어해준다.

1)  $K_p$  값이 크면 제어량이 커서 목표값에 빠르게 도달하지만 수렴하지 못하고 진동한다. 즉, 오차가 0에 수렴하지 못함.

2)  $K_i$  를 설정해주면 (정상상태) 오차가 줄어든다. 모터는 오버슈트를 막는 것 보다는 빨리 안정되는 것이 중요하므로  $K_i$  값이 적당히 클수록 좋다.  $K_i$  값이 너무 크면 진동한다.

여기서  $K_t$ 는 단위  $N \cdot m/A$  로 나타낸 토크 상수다.  
제어 입력 전압  $e_a(t)$ 로 시작하여 전동기 회로에 대한 방정식을 구해 보면 아래와 같다.

$$\frac{di_a(t)}{dt} = \frac{1}{L_a} e_a(t) - \frac{R_a}{L_a} i_a(t) - \frac{1}{L_a} e_b(t)$$

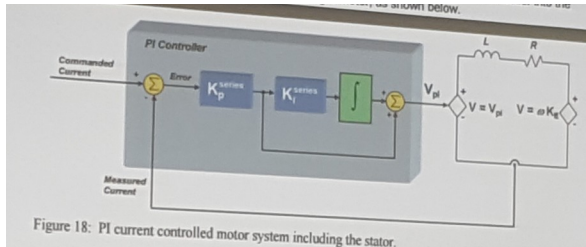
$$T_m(t) = K_t i_a(t)$$

$$e_b(t) = K_b \frac{d\theta_m(t)}{dt} = K_B \omega_m(t)$$

$$\frac{d^2\theta_m(t)}{dt^2} = \frac{1}{J_m} T_m(t) - \frac{1}{J_m} T_L(t) - \frac{B_m}{J_m} \frac{d\theta_m(t)}{dt}$$

여기서  $T_L(t)$ 는 부하 맥달 토크를 나타낸다.  
첫 번째 식의  $di_a(t)/dt$ 는 입력 전압  $e_a(t)$ 로 인한 직접적인 효과에 해당한다.  
두 번째 식의  $i_a(t)$ 는 토크  $T_m(t)$ 를 발생시킨다.  
세 번째 식은 역기전력을 정의하며 마지막 식에서  $T_m(t)$ 는 각속도  $\omega_m(t)$ 와 변위  $\theta_m(t)$ 를 변화시킨다.

## Motor PI Control Example



$$G(s) = \frac{1 + \frac{s}{K_p^{series}}}{\left( \frac{L}{K_p^{series} \cdot K_i^{series}} s^2 + \left( \frac{R}{K_p^{series} \cdot K_i^{series}} + \frac{1}{K_i^{series}} \right) s + 1 \right)} = \frac{1 + \frac{s}{K_i^{series}}}{\left( 1 + \frac{R}{K_p^{series} \cdot K_i^{series}} s \right) \left( 1 + \frac{s}{K_i^{series}} \right)}$$

$$\frac{R}{K_p^{series} \cdot K_i^{series}} = \frac{L}{K_p^{series} \cdot K_i^{series}} \Leftrightarrow \frac{R}{K_i^{series}} = L \Leftrightarrow K_i^{series} = \frac{R}{L}$$

$$G(s) = \frac{1 + \frac{s}{K_i^{series}}}{\left( 1 + \frac{R}{K_p^{series} \cdot K_i^{series}} s \right) \left( 1 + \frac{s}{K_i^{series}} \right)} = \frac{1 + \frac{Ls}{R}}{\left( 1 + \frac{Rs}{K_p^{series} R} \right) \left( 1 + \frac{Ls}{R} \right)} = \frac{1}{1 + \frac{L}{K_p^{series}} s}$$

$$G(s) = \frac{1}{1 + \frac{L}{K_p^{series}} s} = \frac{1}{1 + \frac{L}{K_p^{series}} s} = \frac{K_p^{series}}{K_p^{series} + Ls}$$

$$C \cdot D = \frac{L}{K_p^{series} \cdot K_i^{series}} = \frac{L}{K_p^{series} R}, \quad C + D = \frac{R}{K_p^{series} \cdot K_i^{series}} + \frac{1}{K_i^{series}} = \frac{L}{K_p^{series}} + \frac{L}{R}$$

$$C = \frac{R}{K_p^{series} \cdot K_i^{series}} = \frac{L}{K_p^{series}}, \quad D = \frac{1}{K_i^{series}} = \frac{L}{R}$$

$$\frac{R}{K_p^{series} \cdot K_i^{series}} = \frac{L}{K_p^{series} \cdot K_i^{series}} \Leftrightarrow \frac{R}{K_i^{series}} = L \Leftrightarrow K_i^{series} = \frac{R}{L}$$

$$L \frac{di}{dt} \Rightarrow Ls I(s), \quad iR \Rightarrow RI(s), \quad iR + L \frac{di}{dt} \Rightarrow Ls I(s) + RI(s) \Rightarrow \frac{RI(s)}{(Ls + R)I(s)} = \frac{R}{Ls + R} = \frac{1}{1 + \frac{L}{R}s} = G(s)$$

$$G(s) = \frac{1}{1 + \frac{L}{K_p^{series}} s} \Rightarrow K_p^{series} = L \cdot \text{Bandwidth}$$

다음과 같이 게인을 구할 수 있

$K_p = L \times \text{Bandwidth}$

$K_i = R/L$

## DSP Example 1

```
#include "opencv2/opencv.hpp"
```

```
#include <iostream>
```

```
using namespace cv;
```

```
using namespace std;
```

```
int main(void)
```

```
{
```

```
    Mat A1(1,2, DataType<uchar>::type);
```

```
    A1.at<uchar>(0,0) = 1;
```

```
    A1.at<uchar>(0,1) = 2;
```

```
    cout << "A1" << A1 << endl;
```

```
    cout << "depth = " << A1.depth() << ", " << "channels= " << A1.channels() << endl;
```

```
    Mat A2(1,2, DataType<Vec<uchar,3>>::type);
```

```
    A2.at<Vec<uchar,3>>(0,0) = Vec3d(10,20,30);
```

```
    A2.at<Vec<uchar,3>>(0,1) = Vec3d(40,50,60);
```

```
    cout << "A2" << A2 << endl;
```

```
    cout << "depth = " << A2.depth() << ", " << "channels= " << A2.channels() << endl;
```

```
    Mat B(1,2,DataType<float>::type);
```

```
    B.at<float>(0,0) = 10.0f;
```

```
    B.at<float>(0,1) = 20.0f;
```

```
    cout << "B" << B << endl;
```

```
    cout << "depth = " << B.depth() << ", " << "channels= " << B.channels() << endl;
```

```
    Mat C(1,2, DataType<Point>::type);
```

```
    C.at<Point>(0,0) = Point(100,100);
```

```
    C.at<Point>(0,1) = Point(200,200);
```

```
    cout << "C" << C << endl;
```

```
    cout << "depth = " << C.depth() << ", " << "channels= " << C.channels() << endl;
```

```
    Mat D(1,2, DataType<complex<double>>::type);
```

```
    D.at<complex<double>>(0,0) = complex<double>(10.0,20.0);
```

```
    D.at<complex<double>>(0,0) = complex<double>(10.0,20.0);
```

```
    cout << "D" << D << endl;
```

```
    cout << "depth = " << D.depth() << ", " << "channels= " << D.channels() << endl;
```

```
    return 0;
```

```
}
```

## DSP Example 2

```
#include "opencv2/opencv.hpp"
```

```
#include <iostream>
```

```
using namespace cv;
```

```
using namespace std;
```

```
int main(void)
```

```
{
```

```
    Point2f pt1(0.1f, 0.2f), pt2(0.3f, 0.4f);
```

```
    Point pt3 =(pt1 + pt2) * 10.0f;
```

```
    Point2f pt4=(pt1 - pt2) * 10.0f;
```

```
    Point pt5 =Point(10, 10);
```

```
    Point2f pt6=Point2f(10.0f, 10.0f);
```

```
    cout << "pt1: " << pt1 << endl;
```

```
    cout << "pt2: " << pt2 << endl;
```

```
    cout << "pt3: " << pt3 << endl;
```

```
    cout << "pt4: " << pt4 << endl;
```

```
    cout << "pt5: " << pt5 << endl;
```

```
    cout << "pt6: " << pt6 << endl;
```

```

if(pt1 == pt2)

cout << "pt1 is equal to pt2" << endl;

else

cout << "pt1 is not equal to pt2" << endl;

float fValue = pt1.dot(pt2);

cout << "fValue " << fValue << endl;

double normValue = norm(pt1);

cout << "normValue = " << normValue << endl;

Point pt(150, 150);

Rect rect(100, 100, 200, 200);

if(pt.inside(rect))

cout << "pt is an inside point in rect" << endl;

else

cout << "pt is not an inside point in rect" << endl;

return 0;

}

```

### DSP Example 3

```

#include "opencv2/opencv.hpp"

#include <iostream>

using namespace std;

using namespace cv;

int main(void)

{

    Rect rt1(100,100,320,240), rt2(200,200,320,240); // 상대위

    Point pt1(100,100);

    Size size(100,100);

```

```

Rect rt3 = rt1 + pt1;

Rect rt4 = rt1 + size; // 시작점이 이동한게 아니라 가로세로 크기가 100 씩 커짐

cout << "rt1 : (" << rt1.x << ", " << rt1.y << ", " << rt1.width<< ", " << rt1.height <<
")" <<endl;

cout << "rt1 : " << rt1 << endl;

cout << "rt2 : " << rt2 << endl;

cout << "rt3 : " << rt3 << endl;

cout << "rt4 : " << rt4 << endl;

Point ptTopLeft = rt1.tl();

Point ptBottomRight = rt1.br();

cout << "ptTopLeft in rt1: " << ptTopLeft << endl;

cout << "ptBottomRight in rt1: " << ptBottomRight << endl;

Point pt2(200,200);

if(rt1.contains(pt2))

    cout << "pt2 is an inside point in rt1." << endl;

```

```

Rect rt5 = rt1 & rt2;

Rect rt6 = rt1 | rt2;

cout << "rt5 : " << rt5 << endl;

cout << "rt6 : " << rt6 << endl;

Mat img(600,800,CV_8UC3);

namedWindow("image",WINDOW_AUTOSIZE);

rectangle(img, rt1, Scalar(255,0,0),2); // Blue 100 100 320 240

rectangle(img, rt2, Scalar(0,255,0),2); // Red 200 200 320 240

rectangle(img, rt5, Scalar(0,0,255),2); // Green 64 64 320 240

```

```

imshow("image",img);

waitKey();

rectangle(img, rt6, Scalar(0,0,0),1);

circle(img,pt2,5,Scalar(255,0,255),2);

imshow("image",img);

waitKey();

return 0;

}

```