 # 첨부 1) 질량 관성 모멘트의 측정 및 계산

우선 각 부품의 질량과 회전축에서 무게중심간의 거리를 측정하여 각각의 질량관성 모멘트를 구하고, 이를 합하여 총 질량관성 모멘트를 추정하였다. 이 과정을 간단한 MATLAB 프로그램으로 구현하였다. MATLAB m-파일 코드의 리스트는 다음과 같다. 단위계는 물론 MKS이다.

각 요소의 질량관성모멘트를 구하는 MATLAB 프로그램

```
% 각종 질량 관성 모멘트를 구하는 프로그램
clear
% Motor coupler
d1=30e-3;   t1=10-3;
d2=55e-3;   t2=7e-3;
m=60e-3;        dis=7e-3;
m1=((d1/2)^2*pi*t1)/(pi*(t1*(d1/2)^2+t2*(d2/2)^2))*m;
m2=m-m1;
j_Mcoupler=(m1*(d1/2)^2+m2*(d2/2)^2)/2
% Plate
l=165e-3;    b=62.5e-3;
d1=107e-3; d2=58e-3;
m=50e-3;        dis=l/2-d2;
j_plate=(m/12)*(b^2+l^2)+m*dis^2
% Encoder
l=53.7e-3;   d=54e-3;
m=290e-3;  dis=l/2;
j_encoder=(m/12)*(3*(d/2)^2+l^2)+m*dis^2
% Prism 1, 2
t=9.65e-3;   b=50.8e-3;
m=50e-3;        dis1=60e-3;
dis2=102e-3;jj1_prism=m*dis1^2;
jj2_prism=m*dis2^2;
j1_prism=(m/12)*(b^2+t^2)+jj1_prism
j2_prism=(m/12)*(b^2+t^2)+jj2_prism
% Encoder coupler
t=20e-3;d=35e-3;
m=60e-3;    dis=122e-3;
j_Ecoupler=(m/12)*(3*(d/2)^2+t^2)+m*dis^2
j_theta_Ecoupler=(m/2)*(d/2)^2
% Bar
l=1000e-3;  d=8e-3;
m=30e-3;        dis=122e-3;
j_bar=m*dis^2
j_theta_bar=(m*l^2)/3
% J Motor
Jm=2.754e-5
% Total
j_h=j_Mcoupler+j_plate+j_encoder+j1_prism+j2_prism+j_Ecoupler+Jm
j_tot=j_Mcoupler+j_plate+j_encoder+j1_prism+j2_prism+j_Ecoupler+Jm+j_bar
```

실 행 결 과

```
j_Mcoupler =   6.8034e-006          : Motor coupler
j_plate =   1.5973e-004             : Frame
j_encoder =   3.3161e-004           : Encoder
j1_prism =   1.9114e-004            : Prism 1
j2_prism =   5.3134e-004            : Prism 2
j_Ecoupler =   8.9963e-004          : Encoder coupler (φ방향)
j_theta_Ecoupler =   9.1875e-006    : Encoder coupler (θ방향)
j_bar =   4.4652e-004               : Bar (θ방향)
j_theta_bar =      0.0100           : Bar (φ방향)
Jm =   2.7540e-005                  : Motor
j_h =      0.0021                   : Arm
j_tot =      0.0026                 : Arm + Bar
```

첨부 2) DC-Motor 와 Encoder 의 사양

1. DC-Motor

제 품 명	EC-530
생 산 사	Electro Craft co.
장 착 된 엔 코 더	800 p/r Incremental encoder
토 크 상 수	0.05154 Nm/A
역 기 전 력 상 수	0.05241 V/rad/s
전 기 자 저 항	2.27 Ohm
전 기 자 인 덕 턴 스	6.43e-3 H
관 성 모 멘 트	2.754e-5 kgm^2
정 격 전 압	24 V
유 효 출 력	80 W

2. Encoder

제 품 명	MS50-1000-ABZ-Line Driver
생 산 사	LG 산전 (주)
전 원 전 압	5 - 30 V
소 비 전 류	100 mA 이하
분 해 능	1000 p/r
출 력 특 성	위상차 T/4 +- T/8, 듀티비 T/2 +- T/8
응 답 주 파 수	5 - 50 kHz max
사 용 온 도 범 위	-10 - +55 도(섭씨)
최 고 회 전 수	5000 rpm
기 동 토 크	100 gfcm max at 25 도(섭씨)
축 방 향 진 폭	0.02 mm 이내
축 방 향 허 용 하 중	2.5 kg
반 경 방 향 허 용 하 중	10 kg
허 용 진 동	진폭 1.5 mm (10 - 55 Hz)
허 용 충 격	50 g , 11 ms (3 방향, 3 회)

첨부 3) DC-Motor의 K_m과 C_ϕ를 구하는 과정

본 연구에서는 모터와 Arm을 연결한 상태에서의 Motor torque 상수 K_m을 실험적으로 구하여 사용하였다. 다음은 그 과정을 나타낸 것이다.

1) Motor에 연결된 엔코더는 800펄스의 분해능을 가진 것으로, 전형적인 인크리멘탈(증분형) 엔코더이다. 여기에 8비트까지 데이터를 표현해 줄 수 있는 카운터 회로를 연결해 800 pulse 의 분해능을 모두 살려 주었다.

2) 이 Motor에 우리가 장착할 Arm을 연결한 상태에서 7 Volt (1.5 A) 의 스텝 입력을 가하였다.

3) 그 결과를 데이터로 저장하여 다음 MATLAB 프로그램 (프로그램 4)으로 속도 그래프를 도시하였다. 동시에 그 속도 그래프에 근접한 DC-Motor의 일반적인 거동을 겹쳐 그려, 계수의 값을 추정하였다. 만일 일반화된 Curve-fitting 과정을 거쳤다면 더욱 정밀한 값을 얻을 수 있었을 것이다.

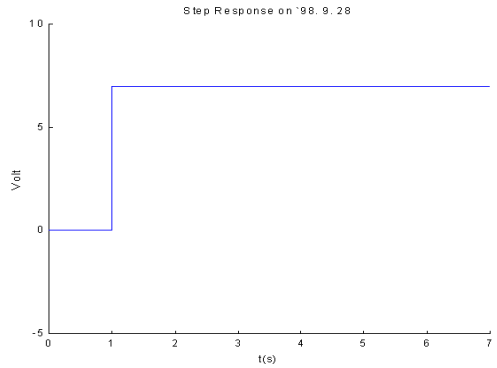

그림 1) 스텝입력 그래프

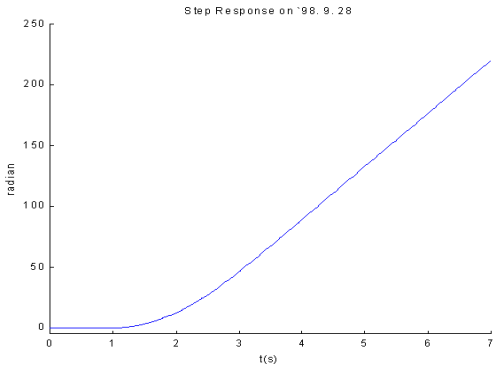

그림 2) 응답변위 그래프

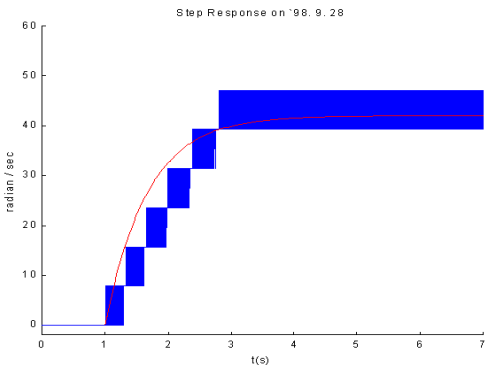

그림 3) 스텝응답곡선

4) 해 석

DC-Motor의 운동방정식은 일반적으로 다음과 같이 간략하게 나타내어진다.

$$J_h\ddot{\Phi} + C_\phi\dot{\Phi} = K_m u$$

위 식은 DC 모터가 전압에 비례하여 선형적으로 토크를 발생하는 장치라는 가정에서 나온 것이다. 여기서 $\dot{\Phi} = \omega$ 라고 하면 다음 등식을 얻는다.

$$\dot{\omega} + \frac{C_\phi}{J_h}\omega = \frac{K_m}{J_h}u$$

여기서 $\frac{C_\phi}{J_h} = a$ 이고 $\frac{K_m}{J_h} = b$ 라고 하면, 다음을 얻는다.

$$\dot{\omega} + a\omega = bu$$

이를 라플라스 변환하면 다음 식과 같다.

$$s\bar{\omega}(s) + a\bar{\omega}(s) = b\bar{u}(s)$$

여기서 다음의 전달함수를 구한다.

$$\frac{\bar{\omega}}{\bar{u}} = \frac{b}{s+a}$$

그리고, 스텝 입력값인 u_0가 상수이므로

$$\bar{\omega} = \frac{u_0 b}{s(s+a)} = \frac{u_0 b}{a}\left(\frac{1}{s} - \frac{1}{s+a}\right)$$

이다. 이를 다시 역라플라스 변환하면

$$\dot{\phi} = \frac{u_0 b}{a}(1 - e^{-at})$$

를 얻는다. 이 결과는 곧 모터 거동의 이론적인 모형이므로 실제 모터를 구동하여 얻은 데이터의 그래프와 비교하여 계수 a와 b를 결정할 수 있다. (그림 26)에 a와 b를근사화시킨 곡선과 실제 실험데이터가 겹쳐 그려져 있다.

5) 결과는 다음과 같다.

$$a \simeq 1.5$$
$$b \simeq \frac{\omega_{term} \cdot a}{u_0} = \frac{42 \cdot a}{7} = 9$$

앞에서 미리 구한 Arm의 질량관성 모멘트 값 J_h를 사용하면,

$$C_\phi = 3.15\times10^{-3} N\cdot m$$
$$K_m = 18.9\times10^{-3} J/V$$

이다.

첨부 4) MATLAB 프로그램

LQR을 위한 프로그램

```
% LQ  제어기 설계 프로그램
% 물성치
clear
g=9.81;
mb=30e-3;
L=1000e-3;
r=122e-3;
km=0.0189;
Cphi=3.15e-3;
Ctheta=0.0;
Jtot=0.0026;
Jb=(mb*L^2)/3;
% 치환
Q=-0.5^2*mb*L*r;
R=-0.5*mb*L*g;
S=Jb*Jtot-Q^2;
% 치환
k1=(-Jtot*R)/S;
k2=(-Jtot*Ctheta)/S;
k3=(Q*Cphi)/S;
k4=(Q*R)/S;
k5=(Q*Ctheta)/S;
k6=(-Jb*Cphi)/S;
k7=(-Jtot*Q*km)/S;
k8=(Jb*km)/S;
% 상태공간
A=[zeros(2) eye(2); k1 0 k2 k3; k4 0 k5 k6];
B=[0; 0; k7; k8];
C=[eye(4)];
D=zeros(4,1);
% 안정도판별
pzmap(A,B,C,D)
% 가중행렬
Q=[eye(4)];
R=0.05;
% 이산화
[Ad,Bd]=c2d(A,B,0.001);
Cd=C;
Dd=D;
% 제어기 생성
[K,S,E]=dlqr(Ad,Bd,Q,R)
printsys(Ad,Bd,Cd,Dd)
```

PID 제어기를 위한 프로그램

```
clear
Jphi = 0.0026;
Jthe = 0.0105;
N = -0.0018;
G=1.472*10^-1;
Cphi=3.15/1000;
km = 18.9/1000;
Cthe=0.0;
M = [ Jphi N; N Jthe]
C = [ Cphi 0; 0 Cthe]
K = [ 0 0; 0 -G]
Bs = [km; 0]
A=[ zeros(2,2) eye(2); -inv(M)*K -inv(M)*C]
B = [ 0; 0; inv(M)*Bs ]
C = [ 0 1 0 0]
D = [ 0 ]
sys=ss(A,B,C,D);
systf=tf(sys)
eig(systf)
Kd=30.0011; Kp=1500; Ki=40
pidc=tf([Kd Kp Ki],[1 0])
Cs=feedback(systf,pidc)
eig(Cs)
pzmap(Cs)
```

첨부 5) 사용된 제어용 C 프로그램 (데이터 획득용)

LQ 최적제어기법을 위한 프로그램

```
// LQ control with 1/1000 second sampling time
// Sampling Data
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#include <math.h>
#include <graphics.h>
// Address for each ports
#define PA8255    0x300
#define PB8255    0x301
#define PC8255    0x302
#define CW8255    0x303
// Motor stop value
#define STOP      0x00
// For the interrupt
#define PIC1 0x20
#define INTR 0x8
// Time for Initial state
#define TOCK      100
// Noise voltage
#define NOISE     24
// Time for displaying data
#define PTIME     20
// Matrix gain
#define K1        -809.6896
#define K2        -4.3723
#define K3        -197.5417
#define K4        5.4039
// Define functions
void interrupt sampling(void);
void interrupt (*oldvect)();
int i=0;                          //Clocking for plotting timing
unsigned int times=0;             //Clocking for real time
unsigned char char_a=0,char_b=0;//Source input
double volt=0.;
// smapling data
short int huge data[30001][4];
void main()
{
   //    Variables for LQ controller
   int a=0,b=0,l_a=0,l_b=0,s_a=0,s_b=0;
   double k1=0.,k2=0.,k3=0.,k4=0.;
   //    Variables for output
   unsigned char char_c=0,dir=0;
   //    Initialization of graphic mode & file stream
   FILE *stream;
   //    For Interrupt Sampling :
   //    0x4A6 => 0.001s, 0x2E7C => 0.01s
   disable();
   oldvect=getvect(INTR);
   setvect(INTR, &sampling);
   outportb(0x43, 0x36); //8253 : Mode 3, Binary counter, counter
#0
   outportb(0x40, 0xa6); //Under byte:   Timer #0 => 1.19Mhz
   outportb(0x40, 0x04); //Over byte :   T=(Load Number)/1190000
   enable();
   //    Initialization of 8255 & Motor
   outportb(CW8255,0x92);
   outportb(PC8255,STOP);
   //    Inital gain
   k1=(K1*2.*3.14*0.001);
   k2=(K2*2.*3.14*0.005);
   k3=(K3*2.*3.14);
   k4=(K4*2.*3.14*0.002);
   while(!kbhit())
   {
         //    Converting Encoding Data to Radian
         a=char_a-128;
         b=char_b-128;
         //    LQ Controler
         s_a=a-l_a;
         l_a=a;
         s_b=b-l_b;
         l_b=b;
         // 1/1000s
         volt=(int)((k1*a+k2*b+k3*s_a+k4*s_b)*100.)*0.01;
         //    Direction of Motor
         if        (volt<0.) dir=0;
         else if (volt>0.) dir=1;
         //    Boundary of Voltage
         if        (volt<-24.) volt=-24.;
         else if (volt> 24.) volt= 24.;
         //    Output
         if (times<TOCK)
         {
           volt=NOISE;
           char_c=abs((int)(volt*127./24.))*2+dir;
           outport(PC8255,char_c);
         }
         else if (times>=TOCK)
         {
           char_c=abs((int)(volt*127./24.))*2+dir;
           outportb(PC8255,char_c);
         }
         //    For Graphical output
         if (i>PTIME)
         {
           printf("A=%6d,  B=%6d,  C=%6.2f,  t=%5d\r",a,b,volt,times);
         }
// Close Loop of "while(!kbhit())"
   }
// Return to Standard Environment
   disable();
   setvect(INTR, oldvect);
   enable();
   outportb(PC8255, STOP);
// For Output Stream of Data
   stream=fopen("c:\\bc\\bin\\lq103.dat","wt");
   i=0;
   while(i<30001)
   {
         fprintf(stream,"%5d %5d %5d %3d\n"
,data[i][0],data[i][1],data[i][2],data[i][3]);
         i++;
   }
// Close function of "main()"
}
// Interrupt rutine
void interrupt sampling()
{
// Time Cloacking
   if (times>30000) {times=0;}
   times++;
   if (i>(PTIME+1)) {i=0;}
   i++;
//       Sampling Data : 1/1000 Sec
         char_a=inportb(PA8255);
         char_b=inportb(PB8255);
//       Sampling Data for Stream : 1/1000 Sec
         data[times][0]=(short int)times;
         data[times][1]=(short int)char_a;
         data[times][2]=(short int)char_b;
         data[times][3]=(short int)volt;
// End of Interrupt Rutine
   outportb(PIC1, 0x20);
}
// End of Program ~~~
```

PID 제어기법을 위한 프로그램

```
//      PID control with 1/1000 second sampling time
//      Sampling data
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#include <math.h>
#include <graphics.h>
//      Addresses for each ports
#define PA8255    0x300
#define PB8255    0x301
#define PC8255    0x302
#define CW8255    0x303
//      Motor stop value
#define STOP      0x00
//      For the interrupt
#define PIC1      0x20
#define INTR      0x8
//      Time for Initial state
#define TOCK      100
//      Noise
#define NOISE     24
//      Time for displaying data
#define PTIME     20
//      gain
#define K1                501000.
#define K2                -1001000.
#define K3                500000.
//      Define functions
void interrupt sampling(void);
void interrupt (*oldvect)();
int i=0;                  //Clocking for plotting timing
unsigned int times=0;             //Clocking for real time
unsigned char char_a=0,char_b=0;  //Source input
double volt=0.;
//      smapling data
short int huge data[30001][4];
void main()
{
        //      Variables for PID controller
        int b=0,x=0,xm1=0,xm2=0;
        double um1=0.;
        double k1=0.,k2=0.,k3=0.;
        //      Variables for output
        unsigned char char_c=0,dir=1;
        //      Initialization of graphic mode & file stream
        FILE *stream;
        //      For Interrupt Sampling :
        //      0x4A6 => 0.001s, 0x2E7C => 0.01s
        disable();
        oldvect=getvect(INTR);
        setvect(INTR, &sampling);
        outportb(0x43,0x36);   //Mode3,Binarycounter,counter#0
        outportb(0x40,0xA6);   //Underbyte:Timer #0 => 1.19Mhz
        outportb(0x40,0x04);   //Overbyte:T=(LoadNumber)/1190000
        enable();
        //      Initialization of 8255 & Motor
        outportb(CW8255,0x92);
        outportb(PC8255,STOP);
        //      Initial gain
        k1=(K1*2.*3.14*0.001);
        k2=(K2*2.*3.14*0.001);
        k3=(K3*2.*3.14*0.001);
        while(!kbhit())
        {
                //      Converting Encoding Data to Radian
                x=char_a-128;
                b=char_b-128;
                //      PID Controler
                volt=(int)((um1+k1*x+k2*xm1+k3*xm2)*100.)*0.01;
                xm2=xm1;
                xm1=x;
                um1=volt;
                //      Direction of Motor
                if      (volt<0.) dir=1;
                else if (volt>0.) dir=0;
                //      Boundary of Voltage
                if      (volt<-24.) volt=-24.;
                else if (volt> 24.) volt= 24.;
                //      Output
                if (times<TOCK)
                {
                        volt=NOISE;
                        char_c=abs((int)(volt*127./24.))*2+dir;
                        outport(PC8255,char_c);
                }
                else if (times>=TOCK)
                {
                        char_c=abs((int)(volt*127./24.))*2+dir;
                        outportb(PC8255,char_c);
                }

                //      For Graphical output
                if (i>PTIME)
                {
                        printf("A=%6d,  B=%6d,  C=%6.2f,
t=%d\r",x,b,volt,times);
                }
//      Close Loop of "while(!kbhit())"
        }
//      Return to Standard Environment
        disable();
        setvect(INTR, oldvect);
        enable();
        outportb(PC8255, STOP);
//      For Output Stream of Data
        stream=fopen("c:\\bc\\bin\\pid.dat","wt");
        i=0;
        while(i<30001)
        {
                fprintf(stream,"%5d  %5d  %5d  %3d\n"
,data[i][0],data[i][1],data[i][2],data[i][3]);
                i++;
        }
//      Close function of "main()"
}
//      Interrupt rutine
void interrupt sampling()
{
//      Time Cloacking
        if (times>30000) {times=0;}
        times++;
        if (i>(PTIME+1)) {i=0;}
        i++;
//              Sampling Data : 1/1000 Sec
                char_a=inportb(PA8255);
                char_b=inportb(PB8255);
//              Sampling Data for Stream : 1/1000 Sec
                data[times][0]=(short int)times;
                data[times][1]=(short int)char_a;
                data[times][2]=(short int)char_b;
                data[times][3]=(short int)volt;
//      End of Interrupt Rutine
        outportb(PIC1, 0x20);
}
//      End of Program ~~~
```

 # 첨부 6) Routh-Hurwitz의 안정도 판별법

선형 시불변 계통의 특성방정식을

$$F(s) \; + \; as^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n = 0$$

라고 하자. 위의 특성 방정식의 모든 근이 s-평면의 좌반부에 놓이게 하는 필요충분조건은, 방정식의 Hurwitz행렬식 D_k (k = 1,2, …,n)가 모두 양수(+)이어야만 한다는 것이다. 위의 특성방정식의 Hurwitz 행렬식은 다음과 같이 주어진다.

$$D_1 = a_1$$

$$D_2 = \begin{vmatrix} a_1 & a_3 \\ a_0 & a_2 \end{vmatrix}$$

$$D_3 = \begin{vmatrix} a_1 & a_3 & a_5 \\ a_0 & a_2 & a_4 \\ 0 & a_1 & a_3 \end{vmatrix}$$

$$\vdots$$

여기서, n보다 큰 차수나 또는 부(-)의 지수의 계수는 영으로 대치된다.

Routh의 표의 작성(Tabulation)
Routh-Hurwitz 판별법을 간단히 하는 방법은 말한다.
첫단계,다음과 같이 계수들을 나열한다.

$$\begin{array}{ccccccc} a_0 & a_2 & a_4 & a_6 & a_8 & \cdots \\ a_1 & a_3 & a_5 & a_7 & a_9 & \cdots \end{array}$$

두 번째 단계, 지시된 연산에 의하여 다음 수의 배열을 구성하는 것이다. 여기서는 3차 방정식에 대하여 설명하겠다.(a_4, a_5는 3차식이기 때문에 0 이라고 한다.)

s^3	a_0	a_2
s^2	a_1	a_3
s^1	$A = \frac{a_1 a_2 - a_0 a_3}{a_1}$	$B = \frac{a_1 a_4 - a_0 a_5}{a_1}$
s^0	$C = \frac{A a_3 - a_1 B}{A}$	$D = \frac{A a_5 - a_1 a_6}{A}$

여기서 첫 번째 열의 부호가 항상 양이 되도록 값을 계산한다. 이를 본 도립진자시스템에 적용하면

$$K_p > 11.25$$
$$K_i > 1.35$$

이 주어지게 된다. 이 범위안에 이득값이 존재함으로써 우반부에 존재하는 영점이나 극점을 좌반부로 이동 시킬 수 있다. 즉 위의 범위에 있을 때에만 안정하다고 생각할 수 있다. 시스템이 안정권에 들어가면 응답의 몇가지 특성을 주목하여 원하는 응답으로 만들어주게 된다. K_p, K_i의 비율에 주목하여 시뮬레이션을 통하여 응답을 본 다음 적절한 형태의 시뮬레이션상에서의 그래프가 나올때까지 K_p, K_i의 비를 증가시킨다. 즉 K_p의 값을 $1.35 \times \alpha$로 α를 1에서 10까지 정도화 시킨다음 가장 적절한 α를 지정한다. 이후에 적당한 결과가 나오도록 K_d값을 변화 시킨다.

 # 첨부 7) 참고자료 목록

1. C언어로 구현한 IBM PC 인터페이스 회로 설계 (도서출판 한독)
   정기철, 이형찬, 이복구, 이은욱, 임동균, 민병석, 이장무, 조병섭 공저

2. 기계공학 실험 1 (연세대학교 공과대학 기계공학과)
   과내 실험 교재

3. PC 인터페이스 제작과 실제 (크라운 출판사)
   오재광 저

4. 터보 C 정복 (가남사)
   임인건 저

5. 디지탈 제어시스템 제 2 판 (화성출판사)
   Benjamin C. Kuo 저
   박진배, 어진우, 백윤수, 최윤호 공역

6. 최신 제어시스템 제 7 판 (반도출판사)
   Richard C. Dorf, Robert H. Bishop 저
   박홍배, 이균경 공역

7. MATLAB 제어공학 (멀티정보사)
   Katsuhiko Ogata 저
   황우현, 김경숙 역

8. Automatic Control Systems 제 7 판 (Prentice Hall)
   Benjamin C. Kuo 저

9. 도립진자 시스템의 자세제어에 관한 연구 (95년 성균관대 전기공학과 석사논문)
   강기원

10. 기타