로봇학실험3 최종보고서

이름 : 장우현 허정범

학번 :2017741009 2017741047

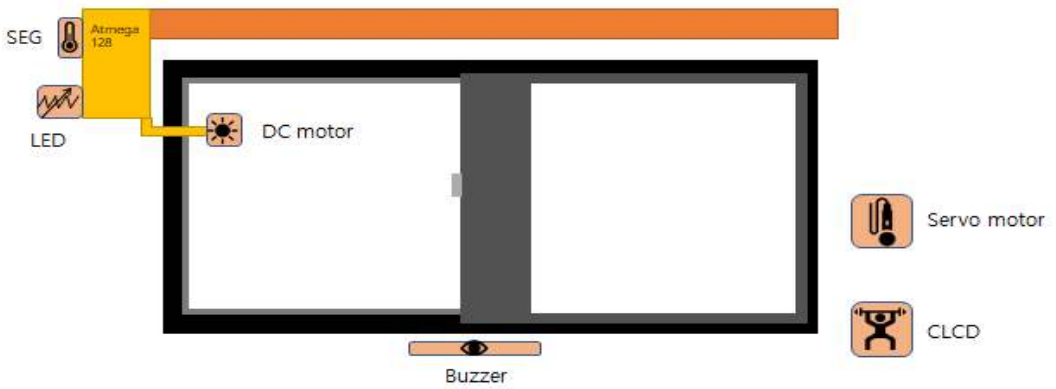
< 목차 >

1. 개요

이번 학기, 로봇학실험3 강의에서는 matlab과 atmega128의 활용, 필터링, 센서 제어 등을 배웠다. 이들을 종합하여 하나의 결과물을 만드는 것이 final term project의 목표다.

2. 서론

여름과 겨울철, 실내 냉난방이 필수적으로 실행되고 있는 요즘 시대. 그에 대한 가스, 전기와 같은 에너지 절약문제도 함께 거론되고 있다. 때문에 냉난방 전자제품을 사용하지 않으면서 비슷한 효과를 내주는 방법들이 심심찮게 추천되고 있다. 그 방법들도 여러 가지인데, 이번에 우리 팀이 소개할 것은 바로 커튼이다. 커튼은 창문 밖 햇빛을 차단하여 실내온도상승을 막아줄 뿐만 아니라, 겨울에는 바깥의 냉기를 물리적으로 막아줘서 실내온도하강 또한 막아주는 대단한 역할을 한다. 하지만 커튼은 사용할 때마다 사용자가 직접 올리고 내려야한다는 번거로움이 있다. 우리는 이를 보완하고자 atmega128을 접목시키기로 했다.



목표하는 시스템에 사용되는 센서는 가변저항, 온도, 조도, 소리, 초음파, 압력 센서로, 총 6가지며, 커튼에 구동될 모터는 DC모터다.

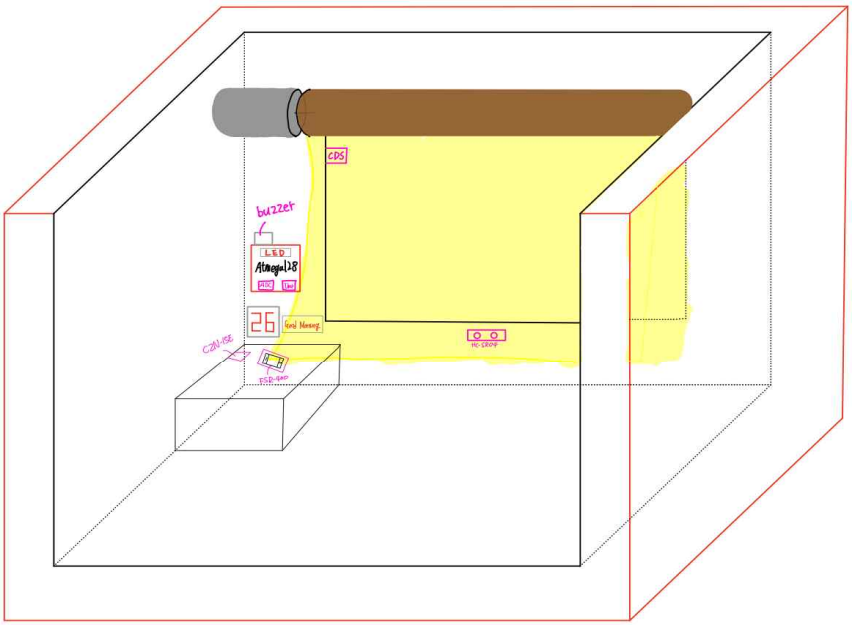
해당 커튼은 밤과 낮으로 mode가 구분된다. 압력센서에 휴대폰이 올려져있으면 밤 mode. 아무것도 올려져있지 않다면 낮 mode로 전환이 된다. mode의 상태는 압력센서에 연결돼있는 CLCD에서 확인이 가능하다. 낮 mode에는 조도센서가 기준치 이상의 값을 읽을 때, 커튼이 내려온다. 밤 mode에는, 아침에 휴대폰에서 울리는 기상 알람소리를 소리센서가

읽어 커튼을 올려준다. 이 때, 소리센서에 연결되어 있는 servo motor가 올라와 기상 알람이 울리고 있다는 것을 가시적으로 나타낸다. 방 안의 온도는 온도센서가 상시로 측정을 하여, 7 segments로 그 값을 출력한다. DC motor가 내려올 때, 초음파센서에 인식이 되면 Buzzer로 작동확인신호를 보내고 DC motor를 정지시킨다. 커튼을 수동으로 작동하고 싶다면 가변저항을 조절하여 DC motor를 켤 수 있으며, 이 때 LED가 켜진다. 이러한 알고리즘으로, 낮에는 햇빛을 자동으로 막아주고, 아침에는 일어나야할 시간에 자동으로 올려주는 기대할 수 있다.

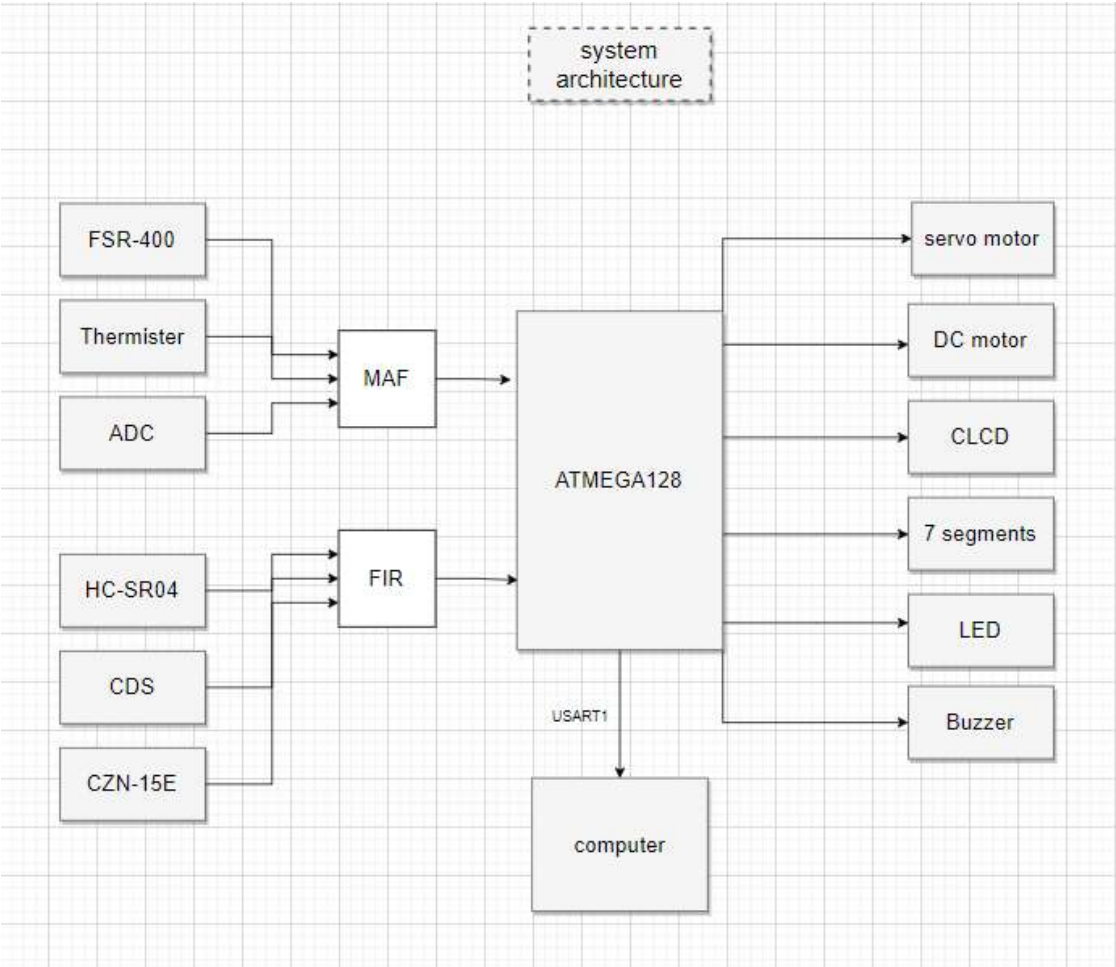
3. 실험 환경 구성

가. H/W 시스템 구조

- H/W outline



- system architecture



4. 실험 결과

0. 공통 코드

- usart

```
///USART=============================================================================================
void USART1_TX_vect(unsigned char data) // Transmit 함수 설정
{
    while(!(UCSR1A & (1<<UDRE1)));  //UDRE = 1 -> buffer empty -> ready to write
    UDR1 = data;
}
void USART1_TX_INT(int nNum)        // ascii code 변환
{
    USART1_TX_vect((nNum % 10000) / 1000 + 48);     // 천의 자리
    USART1_TX_vect((nNum % 1000) / 100 + 48);       // 백의 자리
    USART1_TX_vect((nNum % 100) / 10 + 48);         // 십의 자리
    USART1_TX_vect((nNum % 10) / 1 + 48);           // 일의 자리
}
//===================================================================================================
```

```
///SENSOR=============================================================
double Get_ADC(unsigned char ch)        // ADC값 받아오기
{
    ADMUX = ch;
    ADCSRA |= (1<<ADSC);
    while(!(ADCSRA & (1<<ADIF)));

    return ADC;
}
```

- MAF

```
///Filter
//MEM함수 배열을 저장하기 위한 함수. memmove를 이용하여 배열0번을 없애고 좌로 한칸씩 당기고 최신값을 마지막에 대입.
void MEM_Var(double data)
{
    memmove(Var_MAF_Buffer-1,Var_MAF_Buffer,sizeof(Var_MAF_Buffer));
    Var_MAF_Buffer[19]=data;
}
void MEM_Fsr(double data)...
void MEM_Sound(double data)...
void MEM_Ther(double data)...
void MEM_CDS(double data)
{
    memmove(CDS_sim_y-1, CDS_sim_y ,sizeof(CDS_sim_y));
    CDS_sim_y[51]=data;
}
void MEM_Wav(double data)
{
    memmove(Wav_sim_y-1, Wav_sim_y ,sizeof( Wav_sim_y));
    Wav_sim_y[51]=data;
}
```

```
//MAF  배열로 과거값과 미래값을 저장하여 평균값을 출력한다.
//MEM함수를 통해서 MOVING을 하게 해준다. 오래된 값을 삭제하고 최신값을 넣어 준다.
double MAF_Var(double out)
{
    int sum_data = 0;
    MEM_Var(out);
    for (int j=0; j<20; j++)
    {
        sum_data = sum_data + Var_MAF_Buffer[j];//배열의 합을 구해서 크기 20으로 나누어 평균을 구한다.
    }
    return sum_data/20;
}
```

- FIR

FIR filter를 쓰는 센서들은 공통적으로 주 주파수가 0Hz로 확인되었으므로, n = 50차식, Fc(차단 주파수) = 5Hz로 설정하였다.

```
//FIR 필터
//차수를 50으로 하였고, matlab을 통해서 b값을 가져와 배열로 저장하여 사용하였다.
double FIR_Sound(double out)
{
    int n =50;
    double FIR_y=0;
    Sound_FIR_BUFF[0] = 0;
    MEM_Sound(out);
    for(int j=0; j<n; j++)
    {
        Sound_FIR_BUFF[j+1] = 0;
        Sound_FIR_BUFF[j+1] = b[j+1] * Sound_sim_y[n-j];
        FIR_y += Sound_FIR_BUFF[j+1];
    }
    return FIR_y;
}
```

- main

```
int main(void)
{
    DDRF = 0x00;//DDRX = 1 출력 0 입력
    DDRB = 0xff;
    DDRA = 0xff; //LED
    DDRD = 0xff; //LCD 상위 4비트
    DDRE = 0xf7; //LCD RS RW E 하위 3비트
    DDRC = 0xff; //segment

    PORTD = 0x00;
    PORTA = 0x00; //LED 초기값
    PORTE = 0xff;

    ADCSRA = 0b10000111;        // ADC enable

    //Timer 1
    TCCR1A = (1<<COM1A1)|(1<<WGM11)|(1<<COM1C1);    // 타이머 셋팅1A: 10000010 -> OCR이면 low, bottom이면 high
    TCCR1B = (1<<WGM13)|(1<<WGM12)|(1<<CS11)|(1<<CS10); // 타이머 셋팅1B: 00011011 -> mode: Fast PWM, prescale: clk/64
    ICR1 = 4999;    // top 값인 ICR1을 4999로 함.
    OCR1A= 125;     // 초기상태를 0도로 하기 위한 OCR 초기값.

    //Timer 2
    TCCR2 = (1<<WGM21)|(1<<WGM20)|(1<<COM21)|(1<<CS22)|(1<<CS20);       // fast PWM mode, 1024 prescale
    TIMSK = (1<<TOIE2);                                                 // timer2 overflow interrupt enable
    TCNT2 = 255- 234;

    //Timer 3
    TCCR3A = 0x00;            // Set timer up in CTC mode
    TCCR3B = 0x80;

    UBRR1L = (unsigned char) UBRR;
    UBRR1H = (unsigned char) (UBRR>>8);
    UCSR1B = (1<<TXEN1);                        //Transmitter Enable
    UCSR1C = (1<<UCSZ11)|(1<<UCSZ10);           //비동기, non-parity mode, stop bit : 1 bit data: 8bit

    init_lcd();//LCD setting
    sei();      // 전체 interrupt enable
    while (1)
    {
    }
}
```

가. CDS

- CDS는 주변 빛의 밝기의 변화로 저항값이 변하면서 생기는 전압값으로 조도를 측정하는 센서다. CDS는 주변 밝기가 일정해도 그 센서값은 항상 진동하는 형태를 보였다. 때문에 DC motor를 구동하기 위한 조도의 기준을 명확하게 잡고 제어하기 위해서 센서값을 비교적 선형적이게 만들 필요가 있었다. 이에, 신호값을 선형적으로 변환시켜주는 FIR 필터를 사용하여 신호를 완만하게 만들었다.
DC motor를 구동하기 위한 조도의 기준은, 모의 상 휴대폰의 불빛을 비췄을 때로 측정하였고 그 값은 필터링 된 값으로 600 이다. 따라서 CDS에서 해당 값을 측정하게 되면 DC motor가 구동하여 커튼이 내려오게 된다.

- 코드

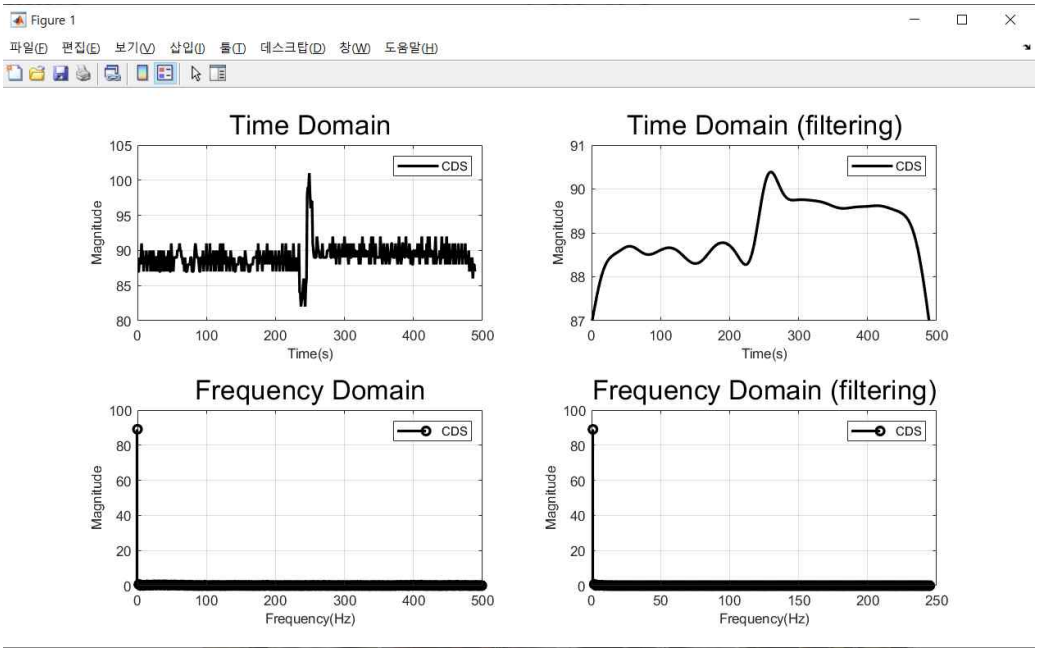
(CDS 수치화)

```
double CDS(int sen)             // CDS 수치화 함수
{
    double Vout = Get_ADC(sen) * (5.0/1024);
    double R9 = 4.7*1000;                       // CDS에 연결된 저항 값
    double Rcds, x;                             // CDS 저항 값, 수치화된 조도

    Rcds = (R9*AVCC)/Vout-R9;
    x = pow(10,(1-(log10(Rcds)-log10(40000))/0.8));

    return x;
}
```

```
void CDSMotion(double Rout, double Cout)//DC모터 종합 함수
{
    if(FSR_flag == 1)//FSR flag는 물체가 올려져있을때 flag 가 1이 된다.
    {
        if (SNflag ==1)//FSR flag가 1이면 자고 있는 상태이고 그때 알람이 울리면 커튼을 올리게 된다.
        {
            PORTD = 0b00000001;//커튼을 올린다
        }
    }
    else//물체가 없을경우 조도와 가변저항으로 커튼을 조절한다.
    {
        if(Rout < 340)
        {
            PORTD = 0b00000001;
        }
        else if (Rout >680)
        {
            PORTD = 0b00000010;
        }
        else
        {
            if(Cout > 600)
            {
                PORTD = 0b00000010;//커튼을 내린다
            }
            else
            {
                PORTD = 0b000000000;
            }
        }
    }

}
```

```
//CDS
CDSOut = CDS(0x01);
F_CDSOut = FIR_CDS(CDSOut);                                    // CDS 송신
USART1_TX_INT((int)F_CDSOut);   USART1_TX_vect(',');
CDSMotion(F_ResOut,F_CDSOut);//DC모터들과 관련된 각종 동작들
```

- 결과 그래프 (Matlab)



위 결과를 보면, CDS의 주 주파수 0 Hz만 존재하는 것을 알 수 있다. 때문에 0 Hz 성분을 살리면서 신호의 파형을 변경하기 위해 FIR의 Low Pass Filter를 적용시켰고, 50차수, 5 Hz의 차단 주파수로 필터링하였다. 결과는 이전 결과보다 선형적으로 정리된 모습을 볼 수 있다.

나. HC-SR04 초음파 센서

- HC-SR04(이하 초음파 센서)는 센서에서 보낸 초음파가 물체를 맞고 다시

돌아올 때 까지의 시간을 세어서 빛의 속도와 곱한 값으로 그 거리를 측정하는 센서다. 이를 활용하여, CDS에 의해 내려오는 커튼이 감지가 되면 Buzzer를 울려 커튼이 성공적으로 내려온 것을 알린다. 하지만 해당 센서는 물체가 스치기만 해도 인식을 잘하는 탓에, 아무조치 없이 motor를 멈추게 하는 센서값을 기준 잡는다면, 잠깐 스쳐가서 순간적으로 기준값을 충족시킬 때 오작동을 일으킬 우려가 있다. 때문에, 지속적인 신호가 아닌, 순간적으로 튀는 신호를 제거하기 위해서, FIR Low 필터를 사용하였다. 처음에는 MAF를 사용하려 하였으나, MAF는 미래의 값까지 필터링 계산에 사용되면서 시간지연이 생기므로 사용하지 않기로 했다. FIR은 과거의 값만을 계산에 사용하므로 해당 문제는 발생하지 않는다고 판단하였다. 그렇게 필터링 된 값으로, 커튼이 위치할 곳의 신호값을 기준값으로 설정하여 Buzzer의 모션을 제어한다.

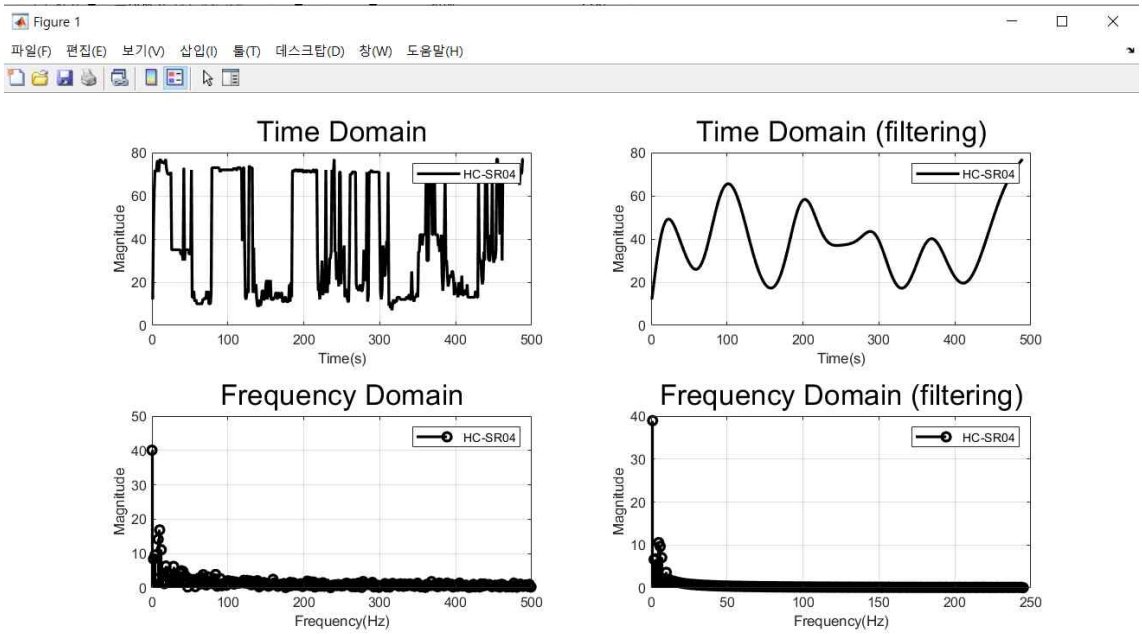
-코드

```
//UltraWave
unsigned int getEcho(void)  // 초음파 센서
{
    Trigger_ON; _delay_us(10); Trigger_OFF;  // Trigger에 10us 신호 출력, 이후 초음파 센서 자체적으로 8cycle의 초음파 출력
    while(Echo==0x00); TCCR3B=0x02; TCNT3=0x00; // 초음파가 도착한 후 echo rising, 이때부터 echo가 low로 떨어질 때 까지 카운팅할 카운터3 셋팅
    while(Echo!=0x00); TCCR3B=0x08;             // echo가 low로 떨어졌을 때, 타이머3 off
    return (TCNT3/58);  //카운팅한 값 == 센서부터 물체까지 빛의 속도로 걸린 시간
                        //1cm에 58us가 걸리므로 물체와의 거리 == (카운팅한 값 = TCNT3)/(58)
}
```

```
//buzzer
void WaveMotion(unsigned int out)//초음파 센서로 거리를 인식하여 일정 거리안에 물체가 들어오면 소리를 낸다.
{
    if(out < 20)      //거리가 20cm안에 물체가 인식이 되면 buzzer flag를 1로하여 소리를 낸다.
    {

        if (Buzzer_flag==1)
        {
            PORTB |= 0x10;
            _delay_ms(10);
            Buzzer_flag = 0;
        }
        else
        {
            PORTB &= ~0x10;
        }

    }
    else
    {
        PORTB &= ~0x10;
        Buzzer_flag = 1;
    }
}
```

```
//UltraWave
WavOut = getEcho();
F_WavOut = FIR_Wav(WavOut);
USART1_TX_INT(F_WavOut);   USART1_TX_vect(',');
WaveMotion(F_WavOut);//일정거리 안에 물체가 있으면 buzzer
```

- 결과 그래프 (Matlab)



다. FSR-400 압력센서

- FSR-400(이하 압력센서)는 압력의 변화로 인한 저항변화로 발생하는 전압값을 측정하는 센서다. 이 센서는 시스템의 mode를 제어하는 역할을 하는데, 이 때 사용할 매개체로는 휴대폰을 생각하였다. 즉, 일어날 때, 커튼이 자동으로 올라가는 mode로 바꿀 때는 휴대폰을 압력센서 위에 놓는다. 아침 기상 후, 휴대폰을 들면 낮 mode로 바뀌어서, 그 때부터는 조도센서가 DC motor를 제어하게 된다. 그리고 각 mode마다 CLCD를 통해 해당 상태를 알려준다. 이 때, 밤 mode를 위해 휴대폰을 센서 위에 올려놓게 되면 신호가 측정되는데, 밑의 왼쪽 결과사진을 보면 순간적으로 신호값이 잠시 떨어졌다가 복구되는 상황이 발생하는 것을 알 수 있다. mode를 정하는 압력센서의 기준값을 정했을 때, 만일 이와 같은 상황이 발생하면 의도치 않게 mode가 바뀔 수 있기 때문에 신호에서 갑작스런

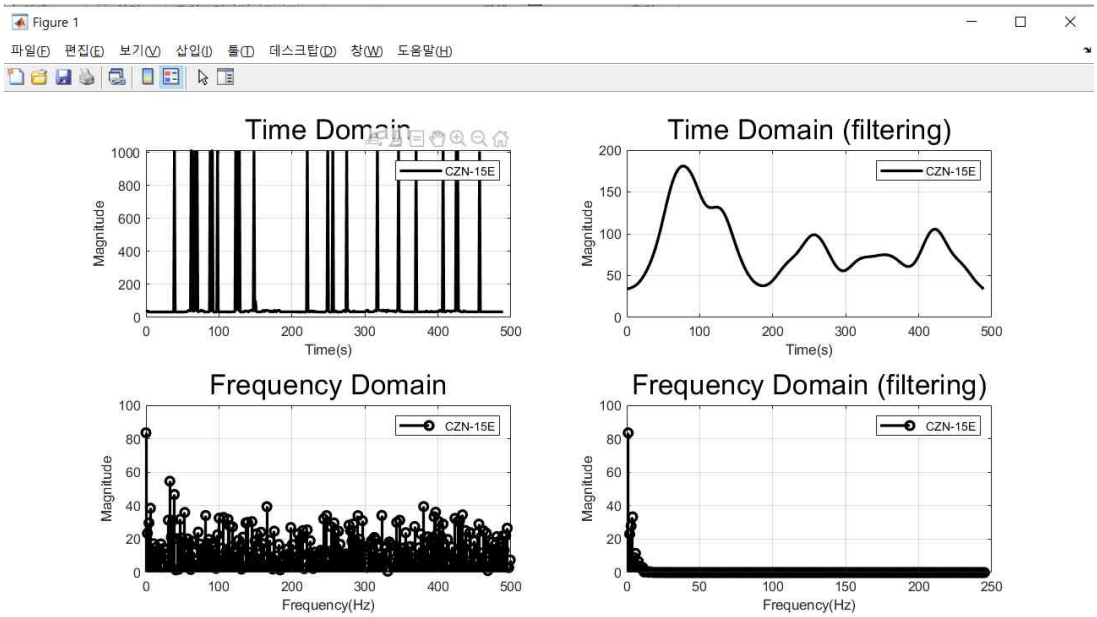
peak를 없애기 위해 MAF(moving Average Filter)를 사용한다. MAF는 일정 과거값, 미래값과의 평균을 신호의 값으로 내보내는 필터인데, 우리는 과거값과 미래값 사이의 크기, width 값을 20으로 하였다. 그렇게 필터링한 신호에서 찾은 기준값은, 절반정도 되는 200으로 설정하였다.

- 코드

```
//LCD - 누르면 LCD에서 NIGHT 평소 MORNING
void Lcd(double data)
{
    if(data > 50){           //센서를 누르면 올라감 data올라감
        lcd_str(str2);       //LCD 문자열 WRITE
        lcd_iw(0x01);        //LCD 명령 실행
        FSR_flag=1;

    }
    else{
        lcd_str(str1);
        lcd_iw(0x01);
        FSR_flag=0;
    }

}
```

```
//FSR
FsrOut = Get_ADC(0x02);
F_FsrOut = MAF_Fsr(F_FsrOut);
USART1_TX_INT((int)F_FsrOut);  USART1_TX_vect(',');
Lcd(F_FsrOut);//LCD로 밤, 아침 상태 표시
```

- 결과 그래프 (Matlab)



라. Thermister 온도센서

- Thermister는 온도가 올라갈수록 저항이 감소하는 센서로, 주변 온도를 측정한다.

Thermister는 현재 실내의 온도를 측정하여 사용자에게 알리는 역할을 하는데, 알리는 방법으로는 segments를 사용하기로 하였다. 이 센서는 애초에 큰 노이즈도 없고 갑자기 값이 크게 변하는 현상도 측정되지 않았다. 때문에, 혹시 모를 상황을 대비하여 MAF를 사용하여 만일의 상황을 대비하였다. 그렇게 필터링된 온도는 2개의 7-segments로 섭씨온도를 나타내어, 사용자가 온도를 확인한 후, 본인이 추위를 느끼는 온도일 때, 가변저항을 사용하여 커튼을 수동으로 조작한다.

- 코드

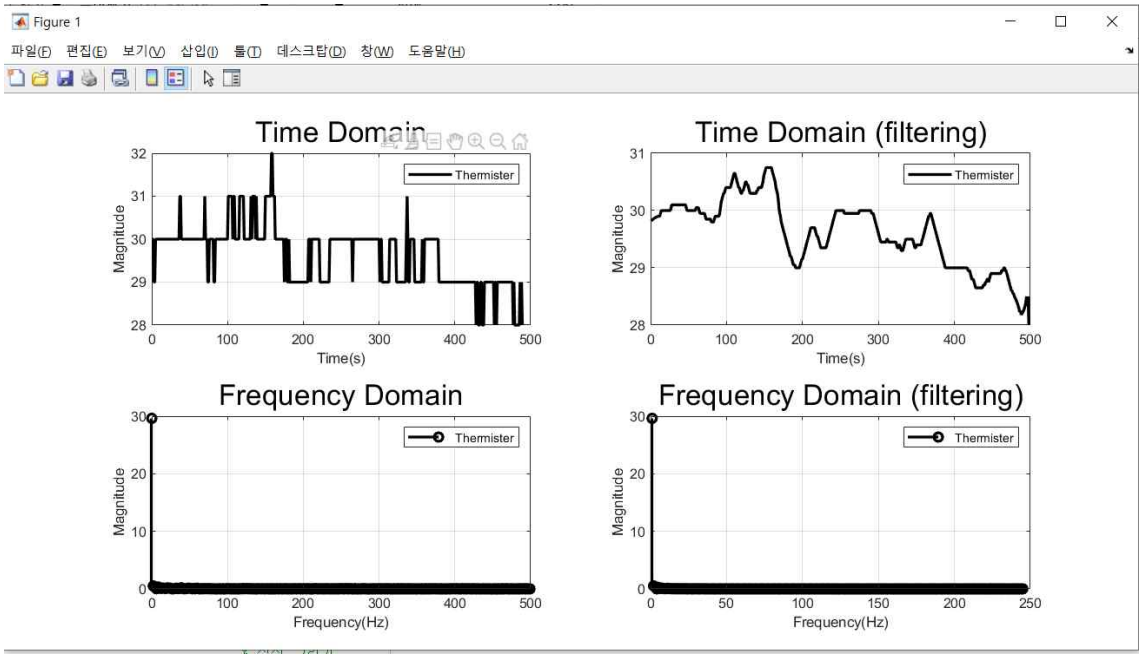
(Thermister 수치화)

```
double Thermistor(int sen)      // 서미스터 수치화 함수
{
    double Vout = Get_ADC(sen) * (5.0/1024);   // ADC값은 0~1023. 이를 5v이내의 v값으로 얻기위한 식이다.
    double T0 = 25 + 273.15;                 // T 초기값
    double R0 = 1000;                       // R 초기값
    double B = 3650;                       // Beta Value
    double R10 = 4.7*1000;                    // 서미스터에 연결된 저항 값
    double Rth;                             // 서미스터 저항 값
    double Tk,Tc;                          // 수치화된 켈빈 온도, 섭씨 온도

    Rth = (AVCC/Vout)*R10-R10;
    Tk = 1/((1/T0)+(log(Rth/R0)/B));
    Tc = Tk - 273.5;

    return Tc;
}
```

```
  //SEGMEMT
void seven_segment(double data)  //써미스터의 온도를 segment 로 출력
{
    char temp;
    int second;
    int first;
    second = (int)(data/10); //앞자리
    first = (int)(data - (second*10)); //뒷자리
        switch(second){
            case 0: PORTE |= 0b11100000;
            break;
            case 1: {temp = PINE; temp &= 0b11011111; temp |=11000000; PORTE=temp;}
            break;
            case 2: {temp = PINE; temp &= 0b10011111; temp |=10000000; PORTE=temp;}
            break;
            case 3: {temp = PINE; temp &= 0b00011111; PORTE=temp;}
            break;
        }
    //PORTB = ~second_num[second];
    PORTC = ~first_num[first];
    _delay_ms(20);
}
```

```
//TERMISTER
 ThOut = Thermistor(0x03);
 F_ThOut = MAF_Ther(ThOut);                  // 서미스터 송신
 USART1_TX_INT((int)F_ThOut);   USART1_TX_vect(',');
 seven_segment(F_ThOut);//segment로 현재 온도 표시
```

- 결과 그래프 (Matlab)



마. CZN-15E 사운드 센서

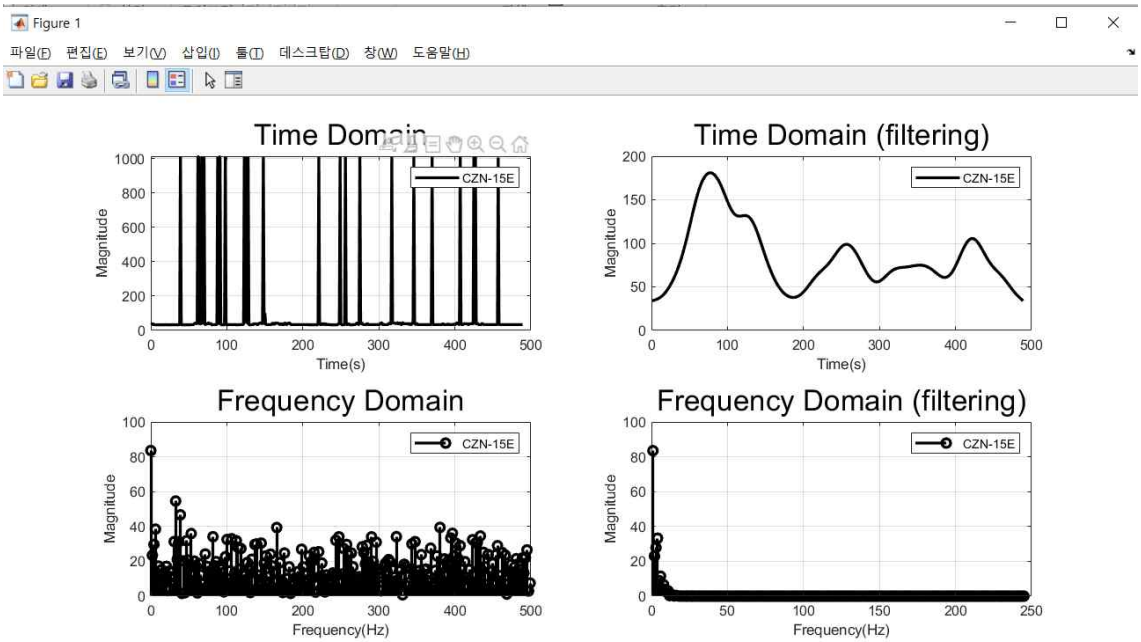
- CZN-15E(이하 사운드 센서)는 소리의 크기를 측정하는 센서로, 소리의 음정은 측정하지 못한다. 사운드 센서는 밤 mode일 때, 휴대폰에서 나오는 기상알람 소리를 읽는데, 알람 소리의 평균되는 크기인 90이상인 값이 20번 측정되면 DC모터를 구동시켜 커튼을 올린다. 그리고 servo motor를 구동시켜 servo motor에 달린 "아침입니다."라는 문구를 보여준다. 사용자가 알람을 끄면, 조용한 상태일 때의 신호 크기인 40에 맞춰서, 50보다 작은 값이 30번 측정되면 servo motor를 초기상태로 되돌려 문구를 내린다. 이 모션의 기준이 되는 값을 필터링 없이 정하기에는 밑의 왼쪽 결과처럼, 신호가 비선형적으로 너무 극단적인 값이 나온다. 때문에, 원활한 모션제어를 위해 신호를 선형적으로 만들기로 하였고, 이에 가장 적합한 필터는 FIR필터이므로, FIR의 low필터를 사용하여 그 신호를 만들었다.

  - 코드

```
//servo motor
unsigned int set_servo(double angle)//모터의 각도를 조절하는 함수
{
    //0: 0.5ms, 90:1.5ms 180 : 2.5ms
    double width;
    double duty;

    width = angle/90.0 + 0.5;

    duty = (width/20)*100;  // 서보모터 주기가 20ms

    OCR1A = (duty/100)*ICR1;
    return OCR1A;
}

void servoMotion(double out)//알람소리가 계속 울리게 되면 서보모터를 작동하게 된다.
{
    if(SNflag == 0)//조용해서 플래그가 0일때 큰 소리가 나는 것을 카운트 한다.
    {
        angle = 0;
        if(out > 100)
        {
            i++;
            if(i==30)
            {
                SNflag = 1;//벨소리가 계속 울리다는 flag를 1로 해준다
            }
        }
    }
    else if(SNflag == 1)//flag가 1일때 조용함이 지속되면, 알람을 끄면 flag를 0으로 해준다.
    {
        angle = 90;
        if(out < 70)
        {
            i--;
            if(i==0)
            {
                SNflag = 0;
            }
        }
    }
```

```
//sound
SNOut = Get_ADC(0x05);
F_SNOut = FIR_Sound(SNOut);
USART1_TX_INT((int)F_SNOut);    USART1_TX_vect(',');
servoMotion(F_SNOut);//서보모터
```

- 결과 그래프 (Matlab)



바. ADC 가변저항

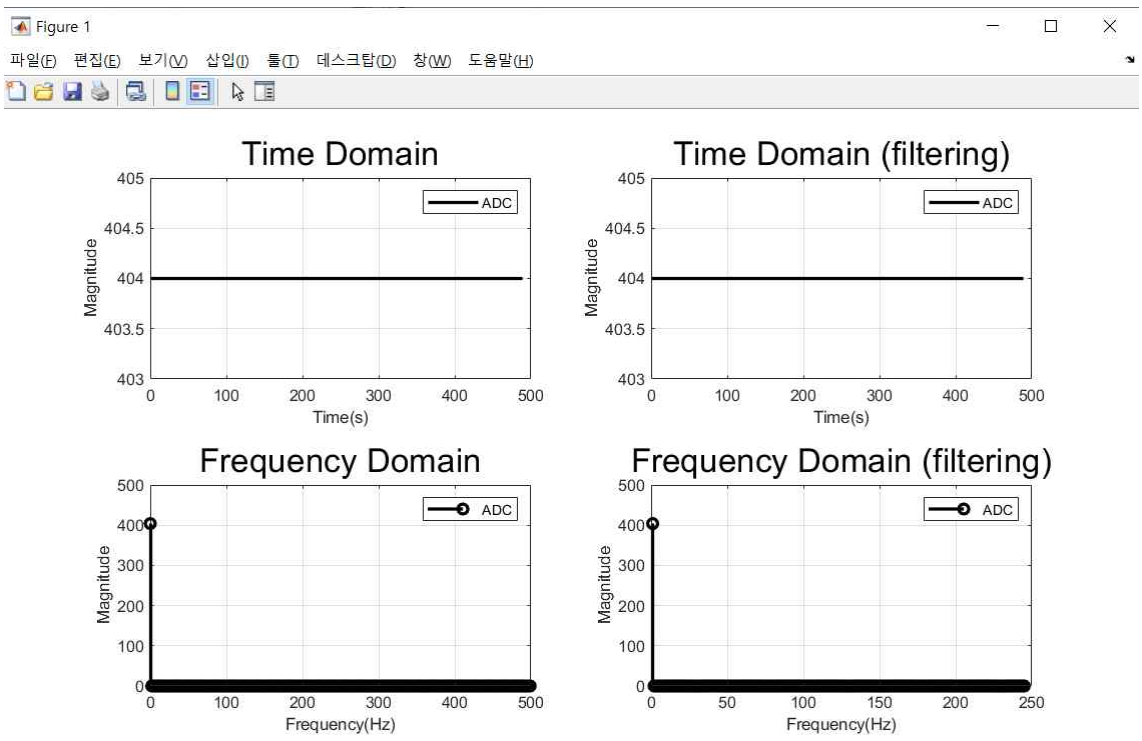
- 가변저항은 크기를 임의로 변경이 가능한 저항이다. segments를 통해 실내 온도를 파악한 사용자가 수동으로 커튼을 내리거나 올리기 위해 DC motor를 구동하고 싶을 때, 가변저항을 사용한다. 0~1023사이의 값으로 저항의 크기를 표현하여, 0~340까지는 커튼 올리기, 341~680까지는 정지, 681~1023까지는 커튼 내리기로 설정하였다. 그리고 0부터 커지면서, LED의 밝기를 조절하여 저항이 변하고 있는 것을 가시적으로 표현하였다. 가변저항은 큰 신호값의 큰 변화가 없으므로, 만일의 상황을 대비하여 MAF로 필터링하여 매순간 평균값을 출력한다.

- **코드**

```
//LED
void Led(double data)//LED 밝기조절
{
    if(data <= 340)
    {
        PORTA = 0xff;
    }
    else if(data <= 680)
    {
        PORTA = 0x00;
        _delay_ms(100);
        PORTA = 0xff;
    }
    else
    {
        PORTA = 0x00;
    }
}
```

```
//VARiABLE RES
ResOut = Get_ADC(0x00)
F_ResOut = MAF_Var(ResOut);
USART1_TX_INT((int)F_ResOut);  USART1_TX_vect(',');
Led(F_ResOut);//LED 밝기 조절
```

- 결과 그래프 (Matlab)



5. 고찰

장우현: 초반에 한 두 개의 센서를 구현할 때는 큰 문제가 발생하지 않았다. 하지만 다른 센서들이 하나 둘씩 더 추가됨으로 시스템의 속도가 확연히 떨어졌다. 또한 몇몇 센서들은 노이즈가 잡히는 모습을 보여 초기에 계획했던 알고리즘에서 점점 멀어지면서 중간마다 시스템 설계 수정을 반복하였고 필터링을 통해서 안정적인 신호로 나타낼 수 있었다. 각 신호들을 더 제대로 분석할 줄 알고, 각각 나타나는 특징에 어울리는 필터를 사용하면 더 나은 결과가 나왔을 것이다.

허정범: 중간중간 USART를 통해서 값이 어떻게 들어오는지 확인을 하였지만 오실로 스코프나 테스터기는 활용을 하지 못하였다. 오실로 스코프로 제어주기를 확인하며 구현을 했으면 훨씬 나은 결과가 나왔을 것이다. 로봇학실험3 수업뿐만 아니라 지금까지 학과에서 배운 모든 수업들의 내용들이 조금씩이나마 집약되어있는 프로젝트였다는 점에서, 수업 때 배운 이론들을 활용하는 능력을 기를 수 있었던 것 같다.