REPORT

-결과 보고서-

제출일 :	2018/12/3
과목 :	센서응용시스템
교수님 :	이 응혁 교수님
학과 :	임베디드 시스템
학번 :	2014146042
이름 :	주 성 민

1. 과 제 명 : 스마트 커튼

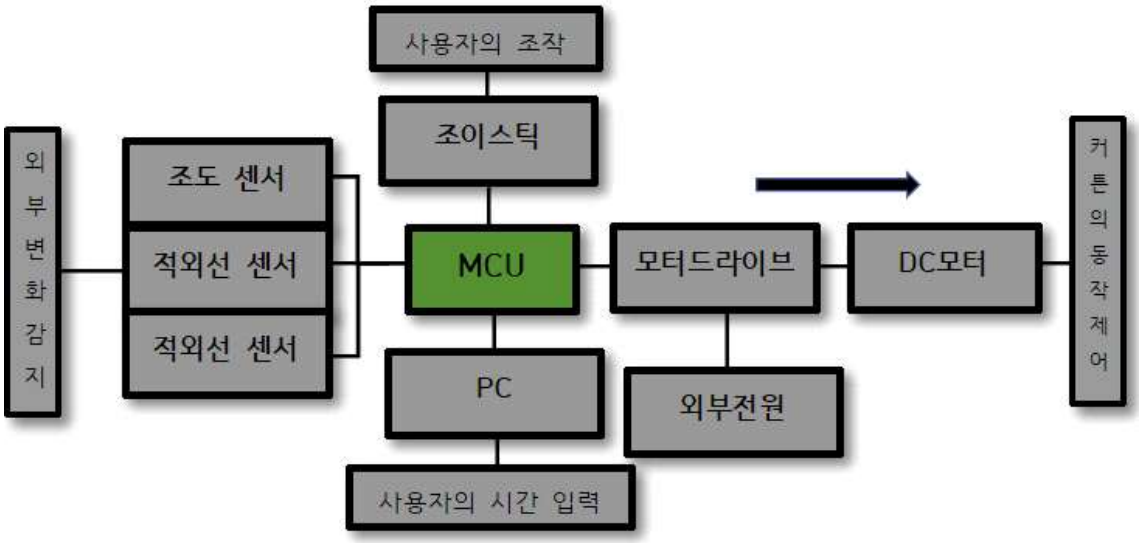
2. 시스템 구현 목표

<다양한 기능을 갖는 똑똑한 커튼 개발>

- **조도 센서를 이용하여 조도에 따른 커튼 제어를 구현**
- **적외선 센서를 이용하여 커튼의 동작범위 제한을 구현**
- **모터드라이브와 DC모터를 이용하여 커튼의 동작을 구현**
- **UART통신과 8비트 타이머를 이용하여 Real Time Clock구현**
- **조이스틱을 이용하여 커튼의 메뉴 조작을 구현**

3. H/W 시스템 구성

-블록 다이어그램



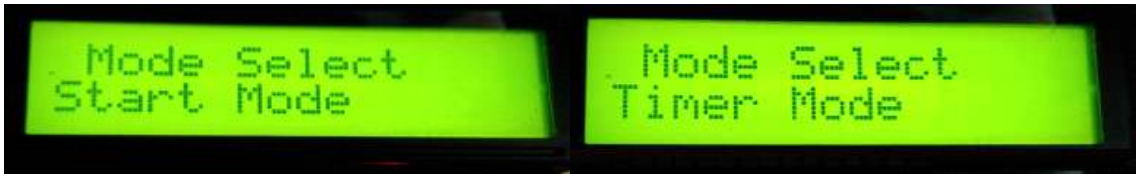
- 이어서 블록별로 상세한 설명이 있습니다.

● 외부의 입력

사용자의 조작, 센서들의 변화 감지, 사용자의 시간 입력으로 나눌 수 있다. 조이스틱을 통해 사용자는 수동모드에서 커튼의 제어와 메뉴나 옵션 등을 선택할 수 있다. 타이머 모드를 사용하게 되면 PC와 UART통신을 통해 시간을 입력받게 된다. 이러한 입력들을 하나의 MCU가 처리하여 모터드라이브와 모터를 제어하게 된다. 기본적으로 조이스틱의 좌, 우 입력에의해 모드를 선택하게 된다. 조이스틱에 대해서는 뒤에서 자세히 설명하기로 하고 먼저 전체적인 흐름이 담긴 코드를 보면 다음과 같다.

```
void Mode_select()
{
    int Switch_val = 0;
    LCD_Pos(1,0);
    LCD_Str(Erase);
    while(1)
    {
        Switch_val =  Get_ADC(Mode_Sel);
        if(Switch_val > 1000)
        {
            if(Mo_count >= 3)
            {
                Mo_count = 1;
            }
            else
            {
                Mo_count++;
            }
            Count = 0;
            while(Count<125)
            {SSound(Buzz);}
            delay_ms(100);
        }
        else if(Switch_val < 100)
        {
            if(Mo_count <= 1)
            {
                Mo_count = 3;
            }
            else
            {
                Mo_count--;
            }
            Count = 0;
            while(Count<125)
            {SSound(Buzz);}
            delay_ms(100);
        }
        if(Mo_count == 0)
        {
            LCD_Pos(1,0);
            LCD_Str("Start Mode ");
        }
        else if(Mo_count == 1)
        {
            LCD_Pos(1,0);
            LCD_Str("Auto Mode  ");
        }
        else if(Mo_count == 2)
        {
            LCD_Pos(1,0);
            LCD_Str("Manual Mode");
        }
        else if(Mo_count == 3)
        {
            LCD_Pos(1,0);
            LCD_Str("Timer Mode ");
        }
        if(Mo_exit == 1)
        {
            Mode_flag = Mo_count;
            Mo_exit = 0;
            break;
        }
    }
}
```

초기화면에서 모드를 선택을 구현하는 함수. 간단하게 3진 카운터의 알고리즘을 사용하여 구현하였으며, 외부인터럽트에 조이스틱의 클릭 스위치를 매핑해서 편리한 메뉴선택을 구현하였다. 아래는 초기화면에서 모드 선택을 위해 조이스틱으로 '좌' 입력을 넣어 준 예시이다.



- 조도 센서 - CdS 포토레지스터

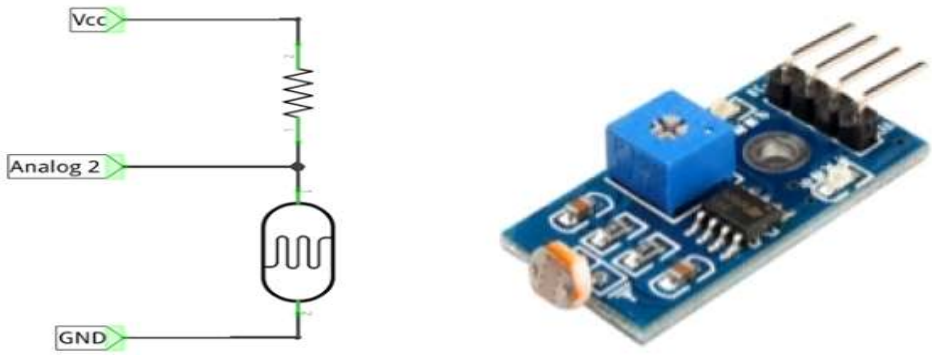
자동모드에서 밝기의 변화를 감지한다. 초기 설정은 임계값만 정해준 뒤 값을 기준으로 커튼을 제어하였는데 조금 더 현실성 있는 작품을 만들기 위해 외부 인터럽트를 이용한 간단한 알고리즘을 통해 사용자가 임계값을 설정할 수 있는 기능을 추가하였다.

```
interrupt [TIM0_COMP] void timer0_out_comp(void)    //타이머 CTC 비교일치시 수행되는 함수 정의
{
    bright_count++;
    Count++;
    if(bright_count > 499)
    {
        if(B_on == 1)
        {
            average[index_bri]  = Get_ADC(Bright_check); //조도는 시도때도없이 확인할 필요가 없기때문에 일정시간동안만 하는것을 구현
            index_bri++;
            if(index_bri >= 9)
            {
                index_bri = 0;
                for(i = 0; i < 10 ; i++)//10개의 입력받은 데이터 값을 내림차순으로
                {                       //정렬하는 알고리즘 입니다.
                    for(j = i+1; j < 10 ; j++){
                        if(average[i]>average[j])
                        {
                            Temp_sort = average[i];
                            average[i] = average[j];
                            average[j] = Temp_sort;
                            j = i+1;
                        }
                    }
                }
                pre_Bright = Bright_val;
                Bright_val =  average[4];

            }
        }
        bright_count = 0;
    }
}
```

이번 과제에서 조도 센서는 10비트 ADC를 통해 MCU와 통신하게 된다. ADC에 대한 설명은 ADC를 사용하는 모든 센서의 설명이 끝난 후에 하려고 한다. 위 코드는 조도 센서와 직접적으로 관련이 있는 코드이다. 타이머를 사용하여 10개의 값에 대해 필터를 적용하여 최종적으로 하나의 값만 Brigtht_val 변수에 저장한다. demo를 위해 샘플링 주기를 0.5초로 설정해두고, 5초마다 10개의 샘플 중 한 개의 값을 추론하여 반환하게 된다. 추론 알고리즘은 입력받은 10개의 값을 크기순으로 나열하여 가운데 값을 받아왔다. 실제 기상변화를 고려한다면, 샘플링 주기를 약 30초 정도로 설정해주는 것이 바람직하다고 생각한다. 후에 알고리즘을 구현하기 위해 pre_Bright 변수에 최종 조도 측정값이 변할 때마다 이전 값을 저장해 주었다.

필터의 알고리즘에서 가운데 값을 받아온 이유는 햇빛이 구름에 잠시 가리거나, 구름 사이로 햇빛이 잠시 나오는 변칙적인 기상 상황이 조도 센서에 영향을 줄 수 있기 때문이다. 기상변화로 조도 센서의 값이 급격하게 변했다가 돌아오는 순간이 ADC 오차(오류)인지 기상의 변화인지 판단하기 힘들어서 확률의 개념을 적용하여 조도값을 측정하는 알고리즘을 구현하였다.



위 사진은 조도센서(Cds 포토레지스터)의 회로도와 외관 사진이다. 포토레지스터의 특성은 어두운 곳에서는 저항이 높아지고, 가시광선이 닿으면 저항 값이 낮아지게 된다. 가시광선이 닿아 센서의 저항값이 낮아지게 되면 Analog 2 지점의 전압값이 전압 분배법칙에 의해 감소하게 된다. 반대로 어두운 곳에서는 저항이 높아져 Analog 2의 전압값이 높아지게 된다.

회로도에서 조도 센서 앞단의 저항은 조도 센서의 민감도를 조절하는 역할을 하며, 센서에서 가변저항을 조절하면 원하는 민감도를 얻을 수 있다.

다음은 조도센서를 사용한 Auto_mode의 코드이다.

```
void Auto_mode(void)
{
    short gab,temp_val;
    LCD_Pos(1,0);
    LCD_Str(Erase);
    B_on = 1;
    Bright_val = Get_ADC(Bright_check);// 초기 한번 측정 후 b
    pre_Bright = Bright_val;
    gab  = 50;
    while(1)
    { //Auto_mode
        if(gab >= 50)
        {
            if(Bright_val < Treshold)
            {
                while(1)
                {
                    delay_ms(5);
                    if(Treshold_count == 0)
                    {
                        Treshold_count = 1 ;
                        Treshold = Treshold + 100;
                    }
                    B_on = 0;
                    temp_val = Get_ADC(Down_max);
                    B_on = 1;
                    if(temp_val < 400)
                    {
                        delay_ms(5);
                        motor_down();
                        LCD_Pos(1,0);
                        LCD_Str(Erase);
                        LCD_Pos(1,0);
                        LCD_Str("Curtain Down");
                        delay_ms(20);
                        motor_stop();
                    }
                    else
                    {
                        delay_ms(5);
                        LCD_Pos(1,0);
                        LCD_Str(Erase);
                        LCD_Pos(1,5);
                        LCD_Str("No move");
                        delay_ms(200);
                        break;
                    }
                }
            }
            else{
                while(1){
                    if(Treshold_count == 1){
                        Treshold_count = 0 ;
                        if(Treshold > 100)
                        {Treshold = Treshold - 100;}
                    }
                    delay_ms(5);
                    B_on = 0;
                    temp_val = Get_ADC(Up_max);
                    B_on = 1;
                    if(temp_val > 600){
                        delay_ms(5);
                        motor_up();
                        LCD_Pos(1,0);
                        LCD_Str(Erase);
                        LCD_Pos(1,0);
                        LCD_Str("Curtain up ");
                        delay_ms(30);
                        motor_stop();
                    }
                    else{
                        delay_ms(5);
                        LCD_Pos(1,0);
                        LCD_Str(Erase);
                        LCD_Pos(1,5);
                        LCD_Str("No move");
                        delay_ms(200);
                        break;
                    }
                }
            }
        }
        if(pre_Bright > Bright_val){
            gab = pre_Bright - Bright_val;
        }
        else{
            gab = Bright_val - pre_Bright;
        }
        if(Mode_flag != 1){//Auto Mode 가 아닐시 함수탈출
            LCD_Clear();
            Treshold = 170;
            B_on = 0;
            break;
        }
    }
```

(1)

위 코드에서 주목해야 할 변수는 gab이다. 이전의 코드에서 조도의 측정값이 변할 때마다 이전 측정값을 저장해 두었었다. (1)번 구간에서 이전 측정값과 현재 측정값의 차이를 계산하여 gab 변수에 저장하게 된다. 이후 while 반복문 안에서 조도의 변화량이 analog 문턱값으로 설정해 둔 50이 넘는다면, 조건에 맞게 커튼을 동작하게 구현하였다. 이는 외부에서 조도값이 변하지 않을 때 필요 없는 동작을 생략하게 하기 위함이다. gab변수는 Auto_mode에 진입하는 순간에 50으로 설정되기 때문에, 진입 후 (1)번 알고리즘에 도달하기 전에 적어도 한번은 현재 조도값에 대해 커튼이 동작하게 설정했다. (이 과정이 없으면 햇빛이 아주 강한 순간에 Auto_mode에 진입해도 커튼이 올라가지 않는 오류가 생긴다.)

● 적외선 센서 - SHARP 0A41SK0F

물체와 거리가 가까워질수록 높은 아날로그 전압값을 출력한다. 두 개의 적외선 센서를 이용하여 아래쪽 적외선 센서는 높은 전압값을 감지하여 커튼이 센서 앞쪽에 위치하는 순간을 감지하고, 반대로 위쪽 적외선 센서는 낮은 전압값을 감지하여 커튼이 센서 앞쪽에서 사라지는 순간을 감지한다. 간단한 알고리즘을 통해 커튼의 동작범위를 제어하게 된다. SHARP사에서 가장 짧은 거리를 잘 측정하는 센서를 사용하여 오차율을 줄였다.

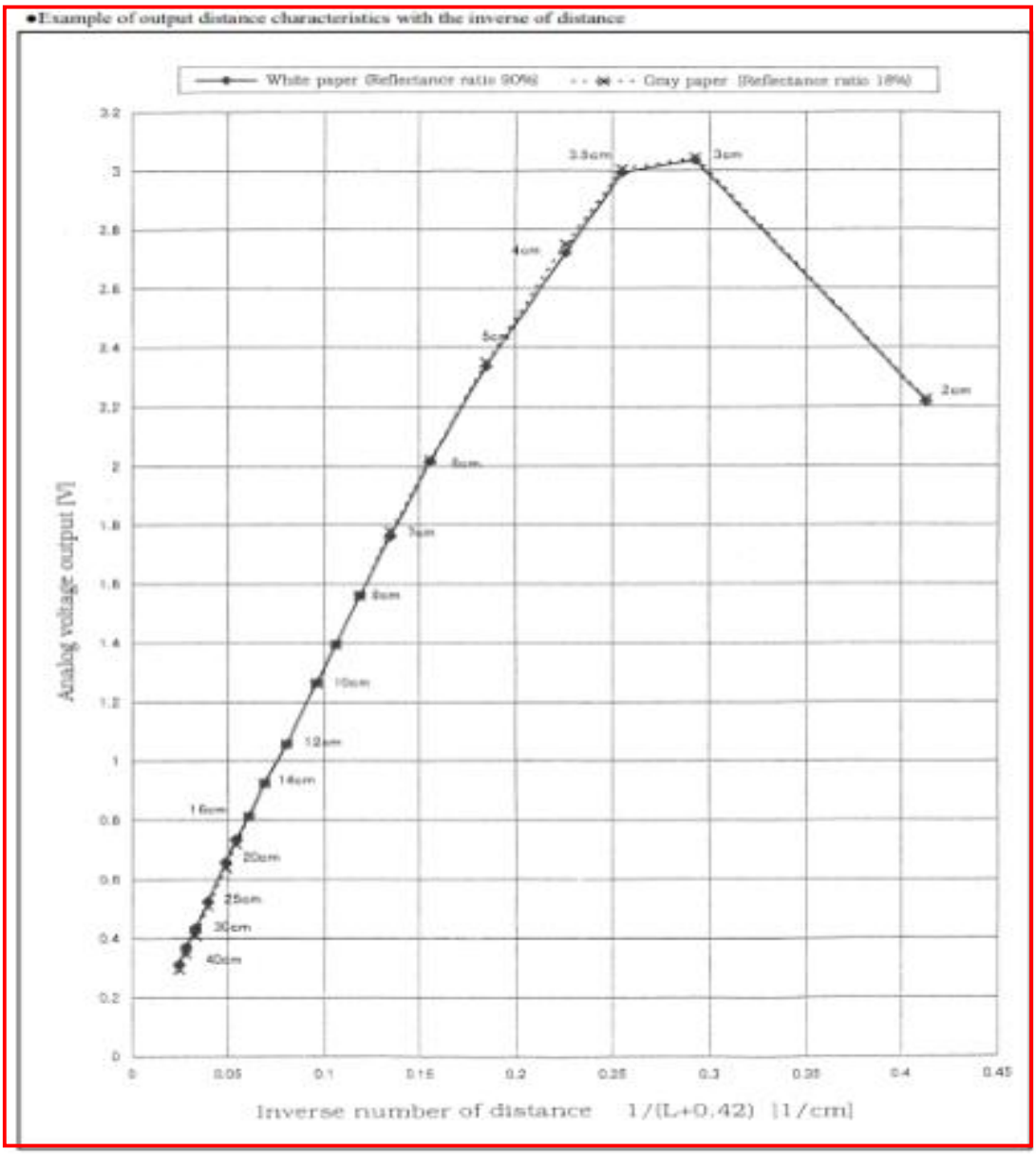
```
if(Get_ADC(Up_max) > 600){
    motor_up();
    LCD_Pos(1,0);
    LCD_Str(Erase);
    LCD_Pos(1,0);
    LCD_Str("Curtain up ");
    Count = 0;
    while(Count<125)
    {SSound(Buzz);}
    delay_ms(20);
    motor_stop();
}
else
{
    LCD_Pos(1,0);
    LCD_Str(Erase);
    LCD_Pos(1,5);
    LCD_Str("Up_Max");
    Count = 0;
    while(Count<125)
    {SSound(warning);}
    delay_ms(200);
}
```

```
if(Get_ADC(Down_max) < 600){
    motor_down();
    LCD_Pos(1,0);
    LCD_Str(Erase);
    LCD_Pos(1,0);
    LCD_Str("Curtain Down");
    Count = 0;
    while(Count<125)
    {SSound(Buzz);}
    delay_ms(20);
    motor_stop();
}
else
{
    LCD_Pos(1,0);
    LCD_Str(Erase);
    LCD_Pos(1,5);
    LCD_Str("Down_Max");
    Count = 0;
    while(Count<125)
    {SSound(warning);}
    delay_ms(200);
}
```

적외선 센서와 직접적인 관련이 있는 코드이다.

상단의 적외선 센서는 센서 앞에 커튼이 없어지는 순간을 감지하여 그 순간보다 위로는 커튼이 올라가지 못하게 막아주는 역할을 하고, 하단의 적외선 센서는 센서 앞에 커튼이 나타나는 순간을 감지하여 그 순간보다 아래로는 커튼이 내려가지 못하게 막아주는 역할을 한다.

600이라는 수치는 10bit ADC를 사용하여 1.8V (7cm)근처를 나타내는 수치인데, 뒷장에서 데이터 시트와 함께 상세히 설명하려고 한다.



다음 사진은 적외선 센서의 데이터 시트 중 일부이다. 앞서 설명한 대로, 감지범위는 2.5cm ~ 40cm 로 물체와 거리가 가까워질 때 마다 아날로그 전압값이 선형적으로 증가하는 것을 볼 수 있다.

필요한 감지 거리는 5cm ~ 10cm 이기 때문에 2.56V의 전압을 기준으로 오차 범위를 고려한 ADC값 600을 임계치로 설정하였다.

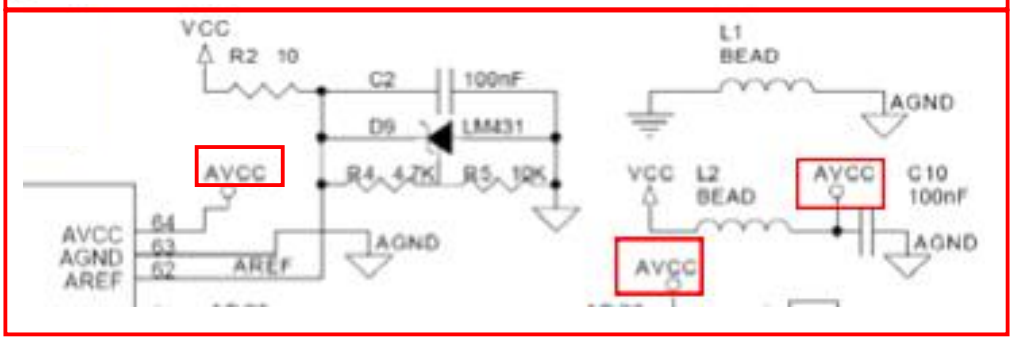
-600 이상 반환 시 커튼이 전방에 존재.
-600 미만 반환 시 전방에 존재하지 않음.

위 두 가지 경우로 나눠서 커튼의 동작 범위를 제어하게 된다.

● 조이스틱

조이스틱은 초기 계획에는 없던 센서(하드웨어)이다. 간단한 스마트 커튼에 견고한 메뉴인터페이스를 구현하기 위해 텀프로젝트 중간에 추가하게 되었다. 조이스틱은 ADC를 사용하는 다른 센서들과 다르게 한 개의 디지털 포트와 x축과 y축에 각각 ADC를 한 개씩 사용하는 특징을 가지고 있다. 다음은 조이스틱을 이해하기 위한 ADC에 대한 설명이다.

```
short Get_ADC(int Number)
{
    ADMUX = Number;
    ADMUX |= (1<<REFS0)|(1<<REFS1);
    ADCSRA |= (1<<ADEN)|(1<<ADSC)|(1<<ADFR)|(1<<ADPS2)|(1<<ADPS1);
    delay_ms(5);
    while(!(ADCSRA&(1<<ADIF)));
    return ADCW;
}
```



조이스틱과 직접적인 연관이 있는 부분이 많이 흩어져있어서 실습에서 센서들에 공통 적으로 사용된 함수와 ADC의 전원부 회로도를 통해 조이스틱의 구현방법을 설명하고자 한다.

첫 번째 줄에서 ADMUX 레지스터에 입력받은 숫자를 넣어주면서 초기화를 해준다. ADMUX를 초기화해주지 않으면 여러 개의 ADC를 사용하기가 힘들어진다. ADC의 기준 전압을 초기에는 AVCC로 설정 해주었지만 텀프로젝트 검토 과정에서 내부 기준 전압 2.56V로 변경하였다. 전원부 회로도를 보면 기준 전압으로 설정 가능한 (ADC에 공급되는)AVCC는 VCC를 사용하지만, 인덕터를 통해 조금 더 안정적인 전압을 만들어 준다. 이미 안정화 된 전압이기 때문에 안정적인 값이 나올 것이라고 예상했지만, 이번 텀프로젝트에서는 작은 수치의 아날로그 전압값들을 검출해서 그런지 생각보다 오차가 많이 생겼다. 결과적으로 내부의 2.56V 기준 전압을 사용하여 작은 값들에 대해 분해능을 높여서 ADC변환을 하게 되었다. REFS1~0을 [1,1]로 해주면, 기준 전압이 2.56V로 설정된다.

초기에는 ADCSRA 레지스터의 설정을 따로 하고 ADC포트만 변경 하면서 여러가지 센서들을 사용하려고 시도했는데, 생각보다 오차가 많이 생기게 되어 센서값을 읽을 때 마다 위의 함수를 통해 ADC enable비트 ADC 변환 시작비트, 프리러닝 비트를 1로 설정해주고 분주비를 64분주로 설정했다. 여러 가지 설정으로 ADC값을 받아보았는데 프리러닝 모드가 가장 오차가 적어 이러한 방법을 선택하게 되었다. 이 부분에 대해서는 조금 더 깊이 있는 공부가 필요하다고 생각한다.

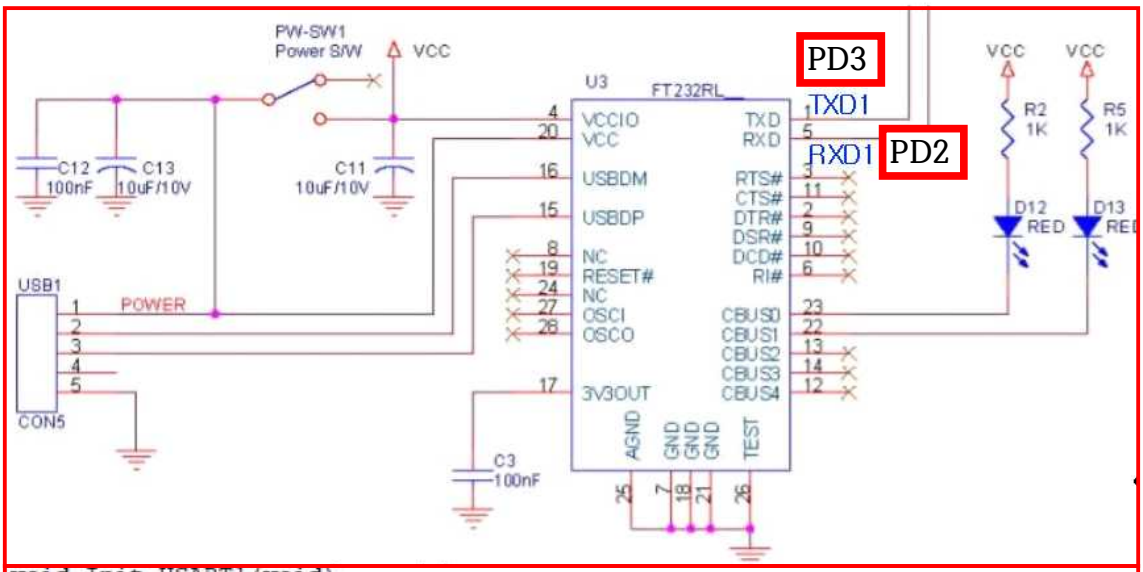
조이스틱을 사용할 때 고려됐던 부분은 x축과 y축이 서로 다른 ADC를 사용한다는 점이다. 동시에 x축과 y축을 사용하게 되면 오차가 심하게 발생하여 커튼을 제어에는 y축을, 메뉴선택에는 x축을 사용하였다.

조이스틱의 마지막 특징은 I/O Digtial 포트로 사용되는 클릭 신호가 있다는 점이다. 상, 하, 좌, 우가 아닌 클릭을 하게 되면 내부 회로가 연결되어서 신호가 그대로 포트로 흘러가게 되는데, 아트메가의 내부 풀업 저항 기능을 사용하여 조금 더 견고한 시스템이 되도록 하였다.

- PC(UART)

실제로 쓰이는 가정용 IOT 제품들을 보면 하나의 메인 컴퓨터(서버)에 연동되어 제어되고 있다. 작품을 조금 더 현실성 있게 구현하고자, PC와의 인터페이스를 구현하였다. PC와 UART통신을 이용하여 사용자에게 키보드로 시간입력을 받아낸 후, 해당 시간부터 타이머를 작동하여 시간이 00:00:00이 되면 커튼이 동작하도록 구현하였다. 마지막에 조이스틱으로 커튼의 제어 옵션을 설정할 수 있도록 했다.

다음은 프로젝트에 사용한 ATmega128의 USART 관련 회로도와 초기화 함수이다.




```
void Init_USART1(void)
{
    UCSR1B |= (1<<RXCIE1)|(1<<RXEN1)|(1<<TXEN1);
    UCSR1A = 0x00;
    //UCSR1C |= (1<<UCSZ11)|(1<<UCSZ10);
    UBRR1H = 0x00;
    UBRR1L = 0x07;
    SREG |= 0x80;
}
```

두 개의 U(S)ART 채널 중 USB를 이용한 1번 채널을 사용했다. UCSR1B 레지스터에서 RX, TX enable 비트와 RX 송신완료 인터럽트를 사용하기 위해 해당 허가 비트를 1로 설정해주었다. UCSR1A 레지스터는 대부분 상태를 확인하는 Flag bit이거나 사용하지 않는 모드에 대해 설정하기 때문에 0x00으로 초기화만 해주었다. UCSR1C 레지스터에서 UCSZ10 ~ UCSZ12 비트는 전송 문자 비트 수를 정해주는데 초기 값이 8비트 전송설정이기 때문에 따로 건들지 않고, 주석처리만 해주어 코드의 가시성을 높였다. 이전에 1번 채널을 사용하면서 PD2,3을 사용 시 오류가 발생한 적이 있어서 이번 텀프로젝트 에서는 해당 핀을 U(S)ART 통신목적 외에는 사용하지 않았다. UBRR1L 에 숫자 7을 입력해줌으로써 전송속도 115,200bps, 비동기 일반모드를 사용하였다.

다음은 U(S)ART 통신에 사용된 인터럽트 코드이다.

```
interrupt [USART1_RXC] void usart1_receive(void)
{
    unsigned char str;
    str = UDR1;
    if(str >= 0x30 && str <= 0x39)
    {
        if(index >5)
        {
            index = 6;
        }
        else
        {
            if(index == 0 || index == 2 ||index == 4)
            {
                if(str >= 0x30 && str <= 0x35)
                {
                    Time[index] = str;
                    index++;
                }
                else
                {
                    puts_USART1("\r\nerror retry\n\r");
                    index = 0;

                }
            }
            else
            {
                Time[index] = str;
                index++;
            }
            if(index == 2 || index == 4)
            {
                putch_USART1(':');
            }
        }
    }
}
```

```
enter hour : 01:01:01

 Time Setting Ok!
```

비동기 통신을 구현하기 위해 전용 인터럽트를 사용하였다. 숫자만 입력받고 시, 분, 초를 명확하게 구분하기 위해 ':'를 추가해주는 알고리즘까지 구현해 보았다.

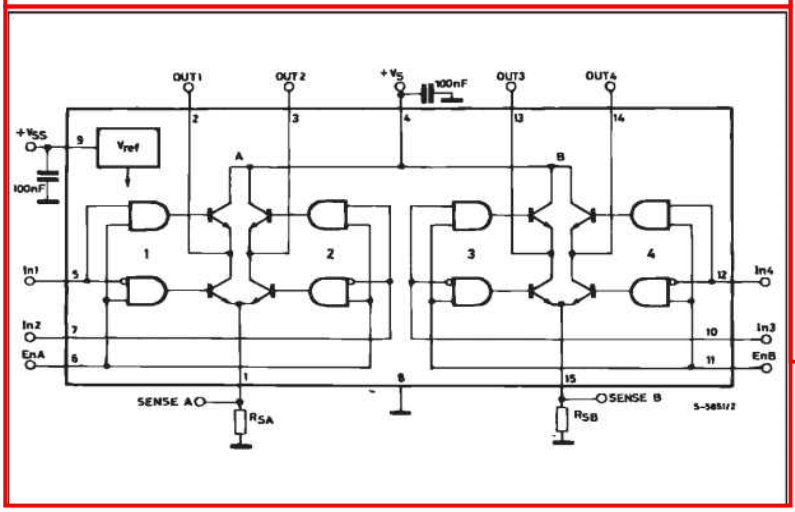
위쪽 파란색 작은 사진은 터미널 창에서 시간을 입력받는 모습이다. 단순하게 숫자만 눌러도 입력규격이 맞춰지면서 가시성을 높였다. 입력받은 시간 값들은 시, 분, 초 의 가중치를 갖고 타이머 모드로 진입하여 타이머값으로 사용된다.

간단한 알고리즘이라고 생각할 수 있지만 견고한 시스템을 설계하기 위해 한가지 알고리즘을 추가하였다. 시간 입력 시 10의자리 수에는 0~5의 숫자만 입력할 수 있게 하여 실제로는 없는 시간 단위는 입력받지 않도록 구현하였다.

● 모터드라이브(L298N), DC모터, 외부전원

커튼의 제어와 물리적으로 연관된 부분이다. MCU만으로는 부족한 전류를 외부전원을 사용하여 해결하였다. 모터드라이브는 MCU의 제어 신호를 받아서 DC모터의 동작과 방향을 제어하게 된다. 추가적으로 pwm을 사용하여 저전력 모드를 구현할 수 있다. 실제 데모에서는 전원공급장치를 사용하겠지만, 텀프로젝트 진행단계에서는 건전지를 사용했기 때문에 pwm을 50%로 설정하여 건전지를 절약하였다.

```
void Timer2_Init(void)
{
    TCCR2 = 0x00;
    TCCR2 |= (1<<WGM21)|(1<<WGM20)|(1<<COM21);
    OCR2 = 128;
}
```




```
void motor_up(void)
{
    PORTB.0 = 1;
    PORTB.1 = 0;
    PORTB.2 = 1;
}
void motor_stop(void)
{
    PORTB.0 = 0;
    PORTB.1 = 0;
    PORTB.2 = 0;
}
void motor_down(void)
{
    PORTB.0 = 0;
    PORTB.1 = 1;
    PORTB.2 = 1;
}
```

모터제어에 직접적인 관련이 있는 함수들과 데이터시트 이다. 실제 커튼은 demo버전 커튼의 무게보다 훨씬 더 무겁다. demo에서는 저전력을 구현함과 동시에 전류의 양을 줄이기 위해 pwm을 50%로 설정한 후 PORTB.7로 출력하여 enable신호로 사용한다. 좌측 상단 코드가 pwm 50%를 구현하는 타이머 코드이다. 실제 커튼처럼 무거운 커튼으로 이 프로젝트를 구현한다면 enable포트에 PORTB.2의 신호(PWM100%)를 넣어주면 된다. 코드상에 사용하지도 않는 PORTB.2가 정의된 이유이다.

모터드라이브(L298N)의 데이터 시트에서 in1,in2(PORTB.0, PORTB.1) 두 신호는 DC모터의 방향을 결정하게 된다. 두 포트의 입력값이 같아진다면 모터는 회전을 멈추게 되고 서로 값이 다를 때만(모터드라이브에 반전 입력이 주어질 때만) 회전을 하게 된다.

- MCU

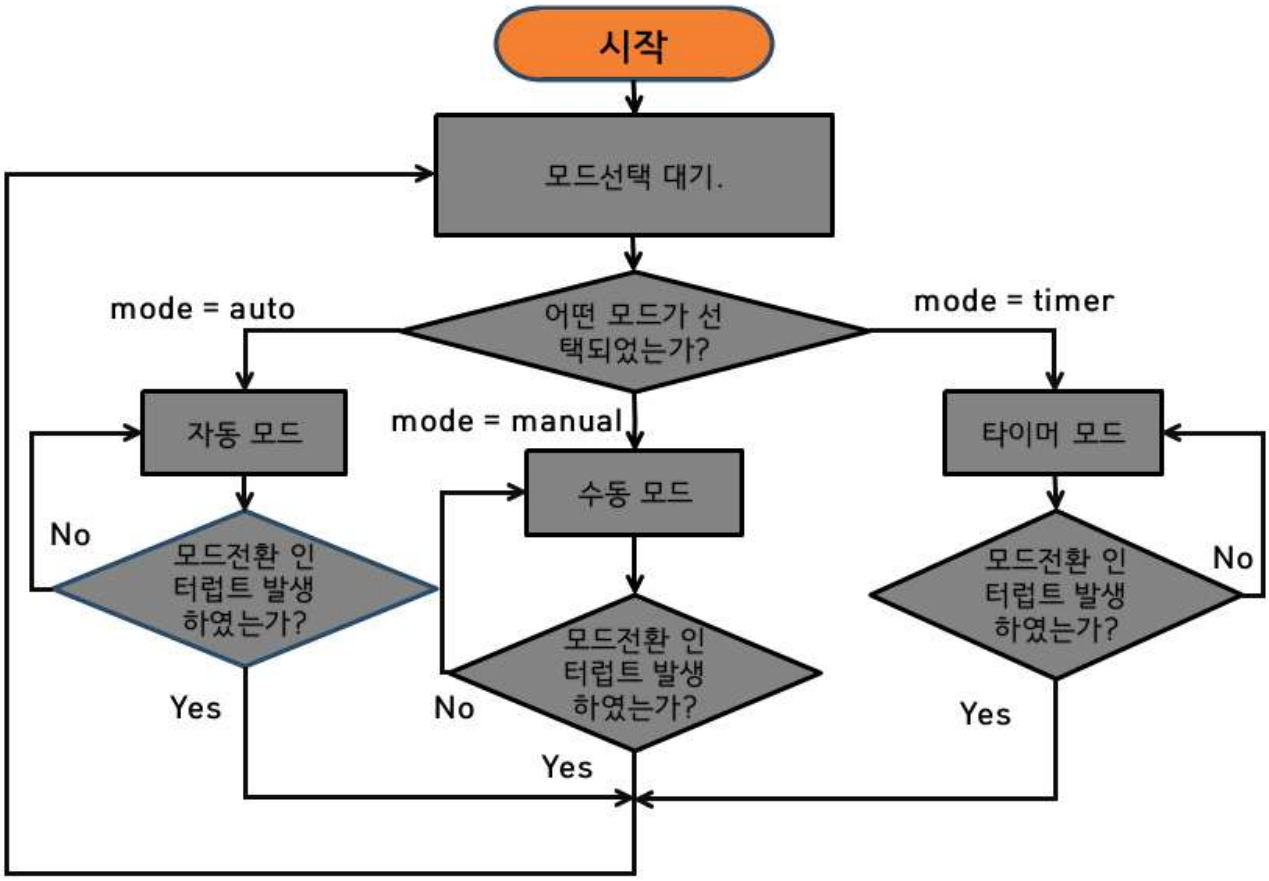
텀프로젝트의 심장부이다. 실제 IOT시스템처럼 메뉴인터페이스를 구현하여 사용자의 입력에 따라 여러 가지 제어를 할 수 있게 구현하였다. 스마트커튼의 소프트웨어 특징은 어떤 상태에서도 동작모드 변경과 같은 명령들을 간단한 조작으로 구현했다는 것이다. 또한 필요하지 않은 입력에 대해서는 잡음이라고 판단하여 무시하도록 알고리즘을 구현하였다. 다음은 이번 프로젝트에서 사용한 ATmega128의 기능들이다.

- 입/출력 포트 내부 풀업저항 및 외부 인터럽트.
- 8비트 타이머를 이용한 PWM 및 Real Time Clock 구현.
- LCD 출력을 통해 메뉴인터페이스 구현.
- USART를 이용한 pc와 의 통신.
- ADC를 이용한 Analog data 처리 및 다중 ADC 사용.
- C언어를 이용하여 메뉴알고리즘 구현.

4. S/W 및 시나리오.

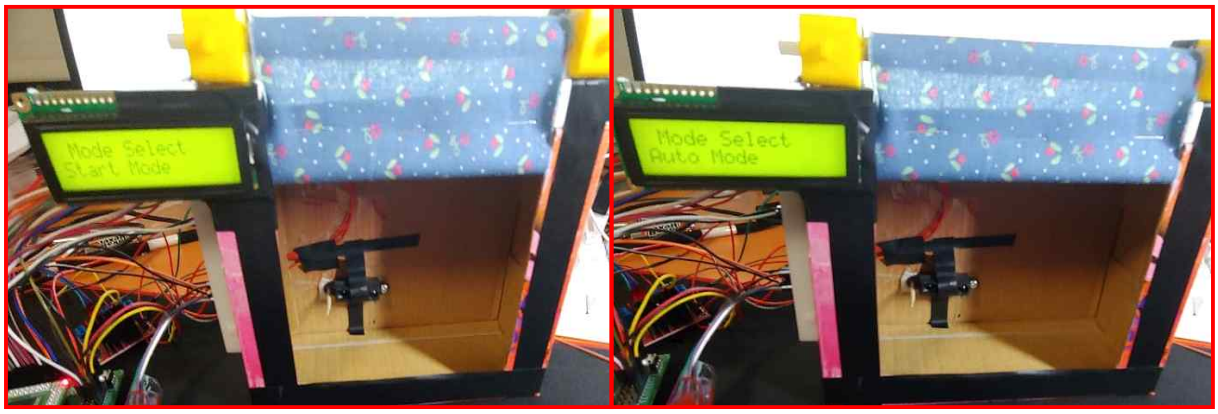
- S/W에 대한 설명

앞서 H/W설명을 하면서 어느정도 S/W에대한 설명을 했었지만, 결론적으로 목표는 실제로 사용 가능할 정도의 S/W를 구현하는 것이다. 구현 과정 중 가장 어려웠던 점은 인터럽트를 사용하여 어떤 상황에서도 사용자의 입력에 응답하면서, 커튼 자체의 알고리즘을 수행하는 알고리즘을 설계하는 것이였다. 완벽한 S/W를 구현하기 위해 아트메가의 여러 가지 기능을 사용하다 보니 결과적으로는 내부 풀업 저항 설정부터 ADC와 ATmega128의 통신까지, 배웠던 거의 모든 기능을 사용하게 되었다. 다음은 S/W에대한 흐름도이다.



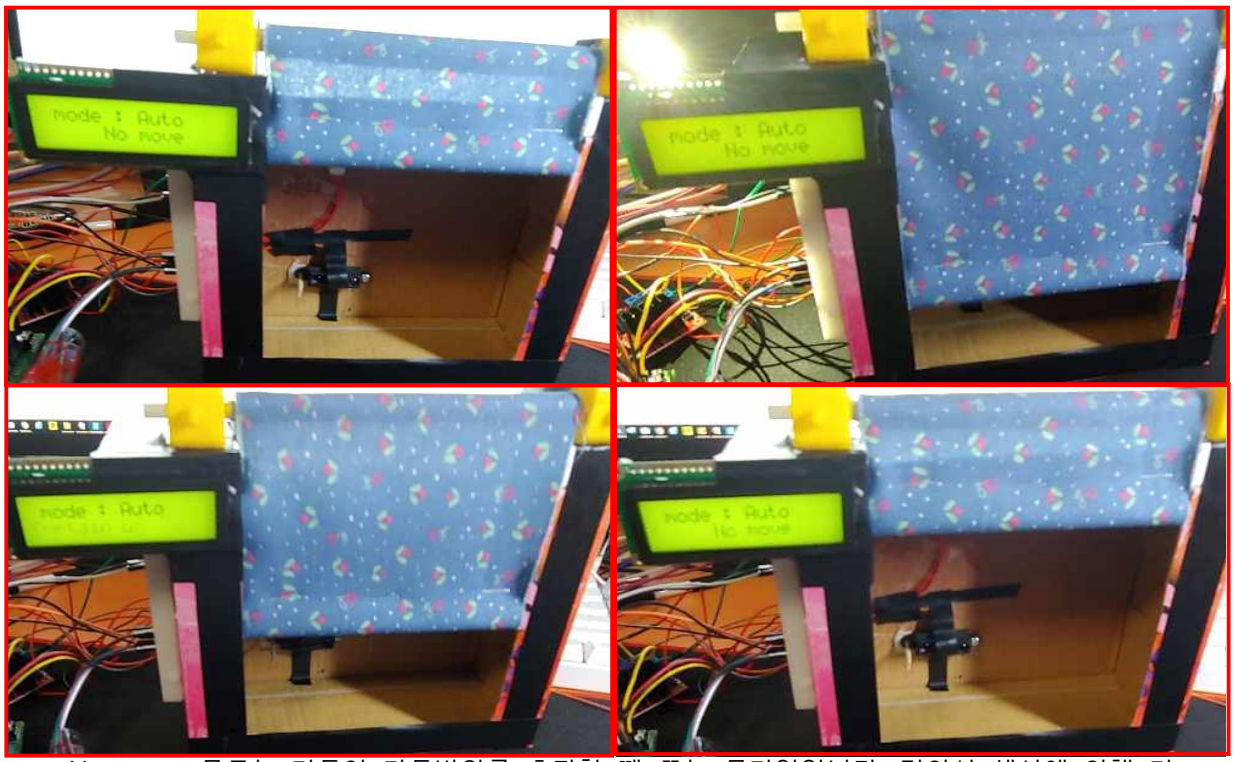
- 동작 시나리오와 동작 설명

● 1. 사용자는 전원을 켜고 조이스틱으로 모드를 선택한다.



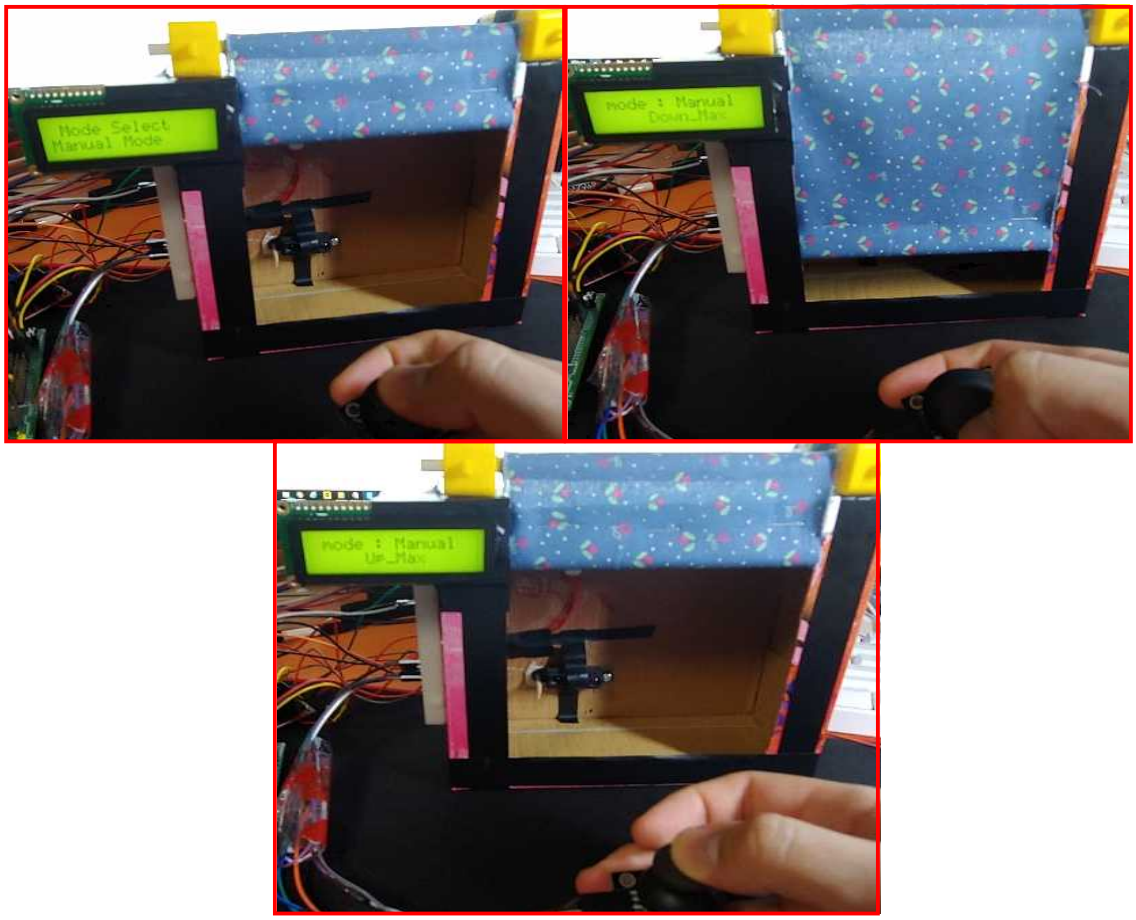
● 2. 다음 세 가지 모드 중 한 가지 모드로 실행

- (1) 자동모드를 사용하여 조도의 임계값에 따라 커튼의 동작을 자동으로 제어. 사용자는 현재의 밝기를 임계치로 사용하여 적절한 임계값 설정 가능. 햇빛이 구름에 잠시 가리게 되면 주변이 잠시 어두워지는 순간을 밝기변화로 인지하지 않고 커튼은 내려가 있는 상태로 유지됨(사진으로는 표현이 힘들어 demo로 구현 예정).



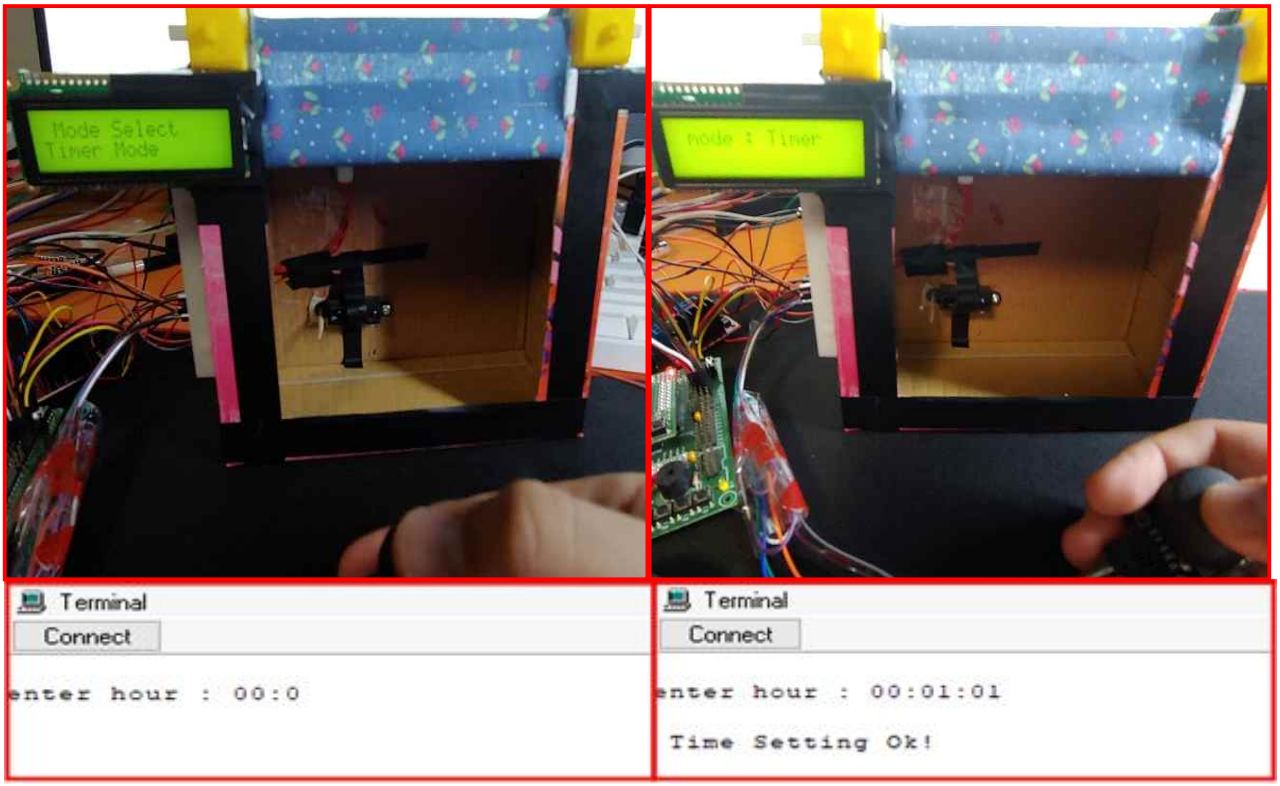
* No move 문구는 커튼이 가동범위를 초과할 때 뜨는 문자열입니다. 적외선 센서에 의해 커튼의 가동범위가 제한되는 모습도 볼 수 있습니다.

- (2) 수동모드를 사용하여 조이스틱으로 커튼을 제어. 타이머나 조도에 영향을 받지 않는 상태. 모드 이름은 수동이지만 조이스틱으로 쉽게 커튼을 제어

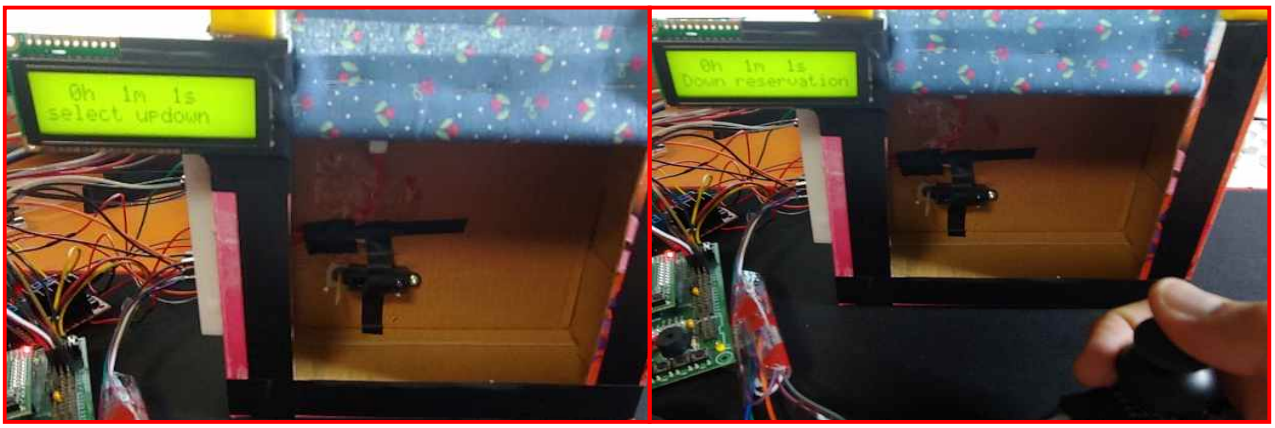


* 수동 모드에서도 적외선 센서는 항상 커튼의 동작범위를 감지하게 됩니다. 추가로 수동모드일때는 부저를 사용하여 동작범위 내에서 동작할 때와 동작범위 초과가 감지될 때 각각 다른 효과음을 구현하였습니다.

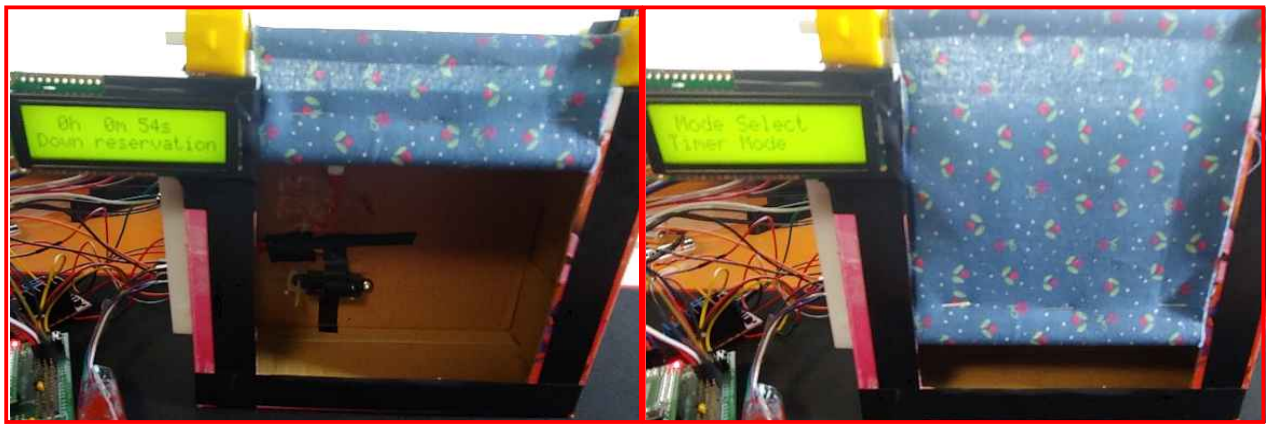
- (3) 타이머모드를 사용하여 일정 시간이 지난 후 커튼을 올리거나 내리도록 동작 예약이 가능



*타이머 모드를 선택 후 PC를 통해 시간을 입력해주는 과정입니다.



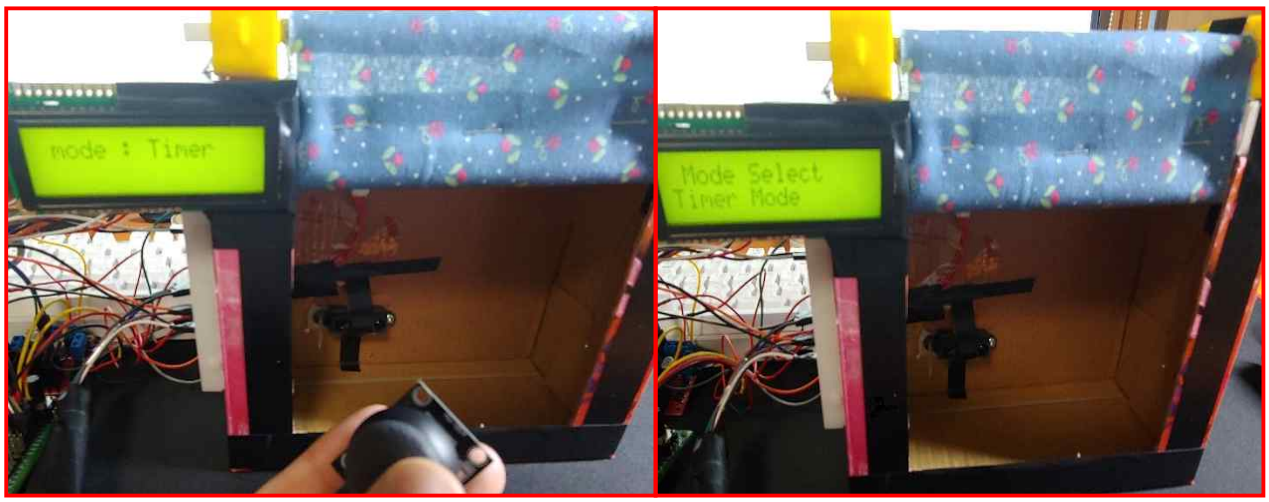
*시간이 입력되면 down, up 중 한가지 동작을 예약할 수 있습니다.



*타이머가 끝나고 0h 0m 0s가 되면 예약된 동작이 수행되고 초기 메뉴선택 화면으로 돌아옵니다.

- **3. 사용자는 어떠한 모드에서도 모드간 전환이 가능.**

사용자가 원할 때 자동모드에서 수동모드로 전환 사용 중 자기 전에는 다시 타이머 모드로 커튼동작 예약할 수 있음.



*예를들어 사용자가 실수로 타이머 모드로 들어갔을 때 다시 클릭 한번으로 초기 메뉴로 돌아갈 수 있는 것처럼, 초기 메뉴와 모드를 자유자제로 이동할 수 있게 설계했습니다.

5. 실험 및 고찰.

메뉴인터페이스를 구현하는 것은 어렵지 않을 것 이라고 생각하고 텀 프로젝트를 시작했는데, 인터럽트를 고려하며 알고리즘을 구현한다는 것은 쉽지만은 않았다. 텀프로젝트 제작과정의 마지막 단계에서 데모시나리오 외 여러 가지 상황에 대해서 실험을 해보며 최종 과제물의 완성도를 높였다. 여러 개의 ADC를 오차 없이 제어하는 단계에서 시간이 꽤 많이 들어갔다.

완성된 텀프로젝트 작품을 기반으로 실제로 내 주제가 사용될 수 있을지 생각해보았다. 타이머 기능을 포함하여 조작부까지 어플리케이션에 연동하고, DC 모터를 더 좋은 성능을 가진 모터로 바꿔준다면 실제에서도 사용 가능하다는 생각이 든다. 이번 데모가 끝나면 방학 기간에 프로젝트 내용 그대로를 어플리케이션을 통해 구현해 볼 예정이다. 결과적으로 동작은 단순하지만, 시스템은 견고한 스마트커튼을 성공적으로 구현했다고 생각한다.

6. 결론

텀프로젝트 결과는 초기 계획했던 것을 모두 구현하고, 추가기능을 더 구현하게 되면서 성공적으로 마무리했다고 생각한다. 시간이 좀 더 있었더라면 앱인벤터를 이용하여 간단한 어플리케이션을 설계한 후 연동하는 과정까지 가능했을 것 같다. 이번 텀프로젝트는 ATmega128의 다양한 기능을 최대한 많이 이용해봤다. 내부 풀업저항 설정으로 외부 스위치를 사용, 타이머인터럽트를 이용한 PWM과 실제 타이머 기능 구현, 여러 개의 ADC를 통한 여러 센서들의 아날로그 출력값 읽기, UART통신을 통한 비동기 통신, 모터드라이브를 사용한 DC모터제어, 외부인터럽트를 사용한 비동기 입력신호 감지 등이 있는데, 과제 구현의 목적을 가지고 시작하게 됐지만, 결과적으로 1년간 ATmega128에서 배웠던 기능들을 복습하는 효과까지 갖게 되었다. 앞서 말한 대로 학기를 마치고 방학 기간에는 어플리케이션에 연동해서 블루투스 모듈을 통한 제어를 추가해볼 예정이다.

제품을 만드는데 개발과정이 복잡하고 장기적인 이유를 알게 되었다. 초기에 생각보다 빠른 진행속도에 이것저것 추가를 많이 했었는데, 알고리즘을 구현하는 단계 이후 여러 가지 오류들을 확인하고 수정하는데 생각보다 많은 시간이 들었었다. 오류 수정의 어려움을 알게된 것은 1년뒤 졸업 후 현장에 나가기 전에 정말 좋은 경험을 했다고 생각한다.

한 학기 중 5주 정도의 시간에 걸쳐 텀프로젝트 과제를 성공적으로 마무리하였다. 직전 학기와 이번 학기에 배운 ATmega128의 기능을 스스로 다시 공부하고, 주변의 많은 질문에 답변하며 복습하는 효과까지, 수동적인 수업보다 훨씬 더 많은 것을 이루는 학기가 되었다.