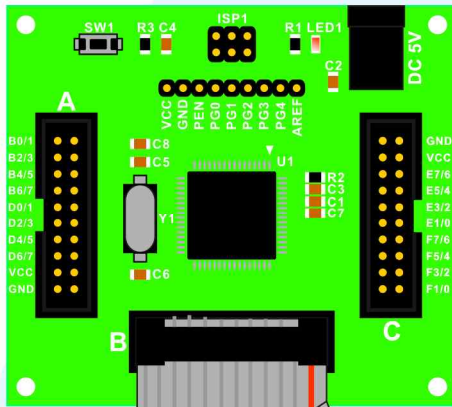
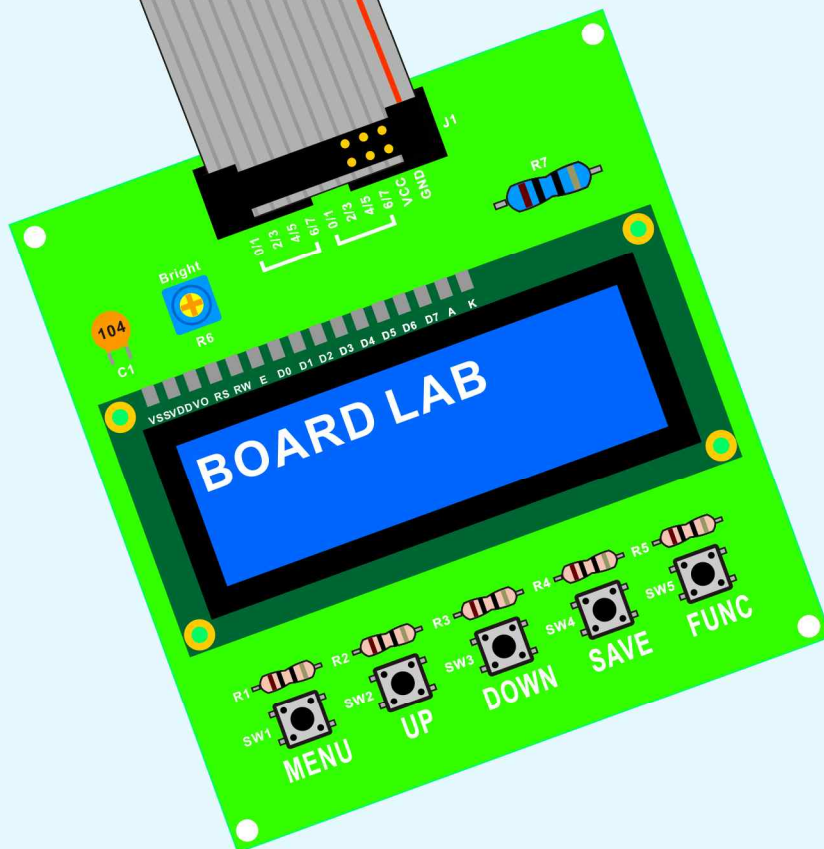


선배가 취업나와 보니까 꼭 필요하더라

전자분야 취준생을 위한 실무 프로젝트



- 실무 프로젝트를 활용한 교육
- 스스로 생각하는힘 형성
- 다양한 예제를 통한 실력향상
- 실무 경험 꿀팁 전달



www.BoardFree.kr

저자:김명수.이소리 공저 / 감수 : 김용필

차례

01. MAIN BOARD

ATmega128 개요
컴파일러
ISP
메인보드
보드 to 보드 연결법

02. LED MATRIX

기본설명, 예제
프로젝트 1 : 미니 전광판
프로젝트 2 : 통신 전광판

03. FND

기본설명, 예제
프로젝트 1 : 전자시계
프로젝트 2 : 모터제어기
프로젝트 3 : 스텝모터제어기
프로젝트 4 : 전자온도계
프로젝트 5 : 온도감지 후드
프로젝트 6 : 온도감지 밸브

04. LCD

기본설명, 예제
프로젝트 1 : 전자시계
프로젝트 2 : 모터제어기
프로젝트 3 : 스텝모터제어기
프로젝트 4 : 전자온도계
프로젝트 5 : 온도감지 후드
프로젝트 6 : 온도감지 밸브

05. 만능기판 사용법

기본설명, 예제
제어 루프 프로그램
Buck타입을 이용한 전압제어
Boost타입을 이용한 전압제어

06. 자료 정리

출처모음

03. FND

기본설명, 예제

프로젝트 1 : 전자시계

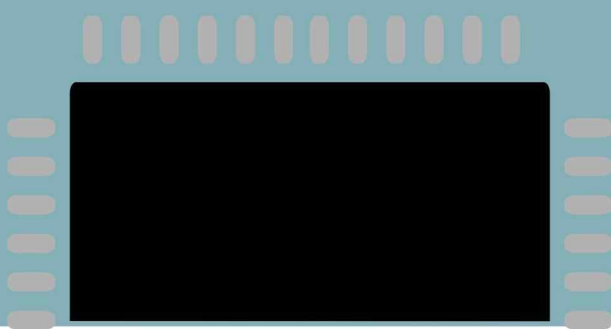
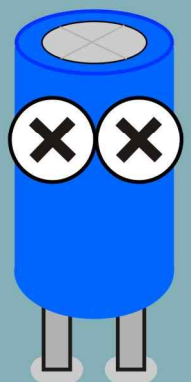
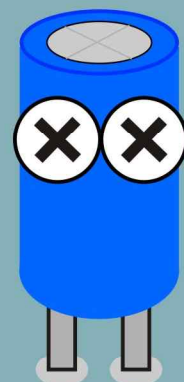
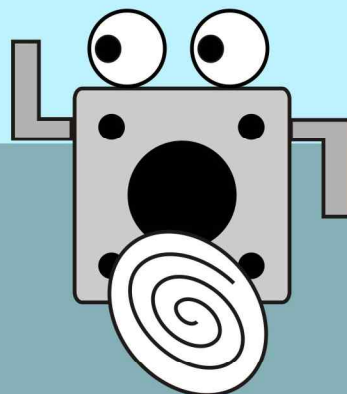
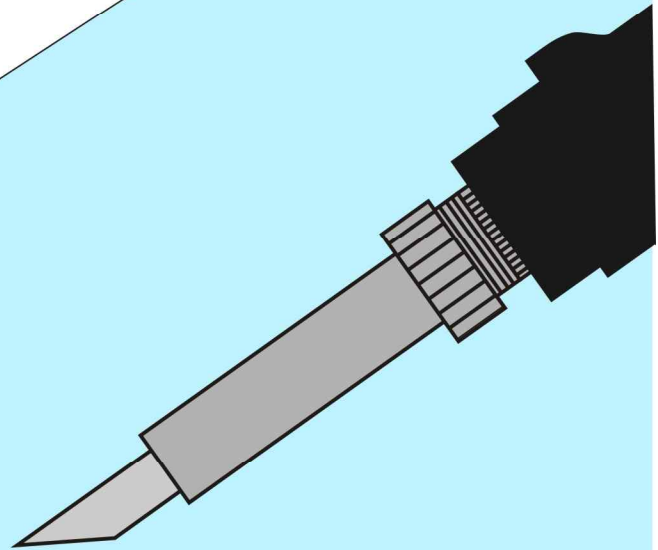
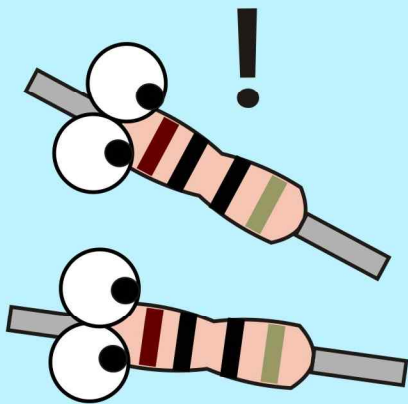
프로젝트 2 : 모터제어기

프로젝트 3 : 스텝모터제어기

프로젝트 4 : 전자온도계

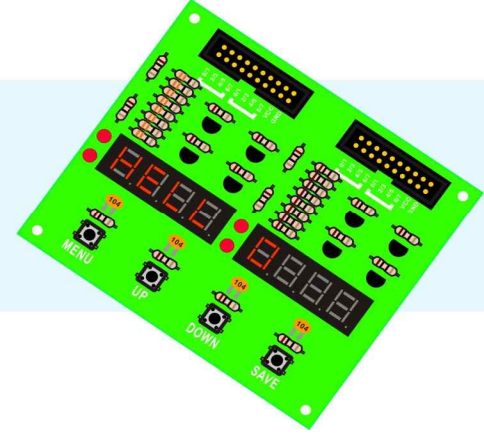
프로젝트 5 : 온도감지 후드

프로젝트 6 : 온도감지 벨브



1

기본설명 및 예제



1.1 FND가 뭘까?

다들 LED는 알고 있지? FND(Flexible Numeric Display)는 쉽게 말해서 LED 7개를 묶어서 아라비아 숫자로 표시할 수 있는 부품이야. LED를 묶는 방법은 anode 끼리 묶거나(common anode) cathode 끼리 묶는(common cathode) 두 종류로 분류해.

LED 7개라고 말하긴 했지만 7개는 아라비아 숫자를 표시하기 위한 거고 사실 점을 찍어 소수점 등을 나타내기 위한 dp라는 LED가 하나 더 있어. 그래서 FND는 총 8개의 led로 이뤄져 있어. 아래 그림은 실제 FND의 내부를 그린 그림이야.

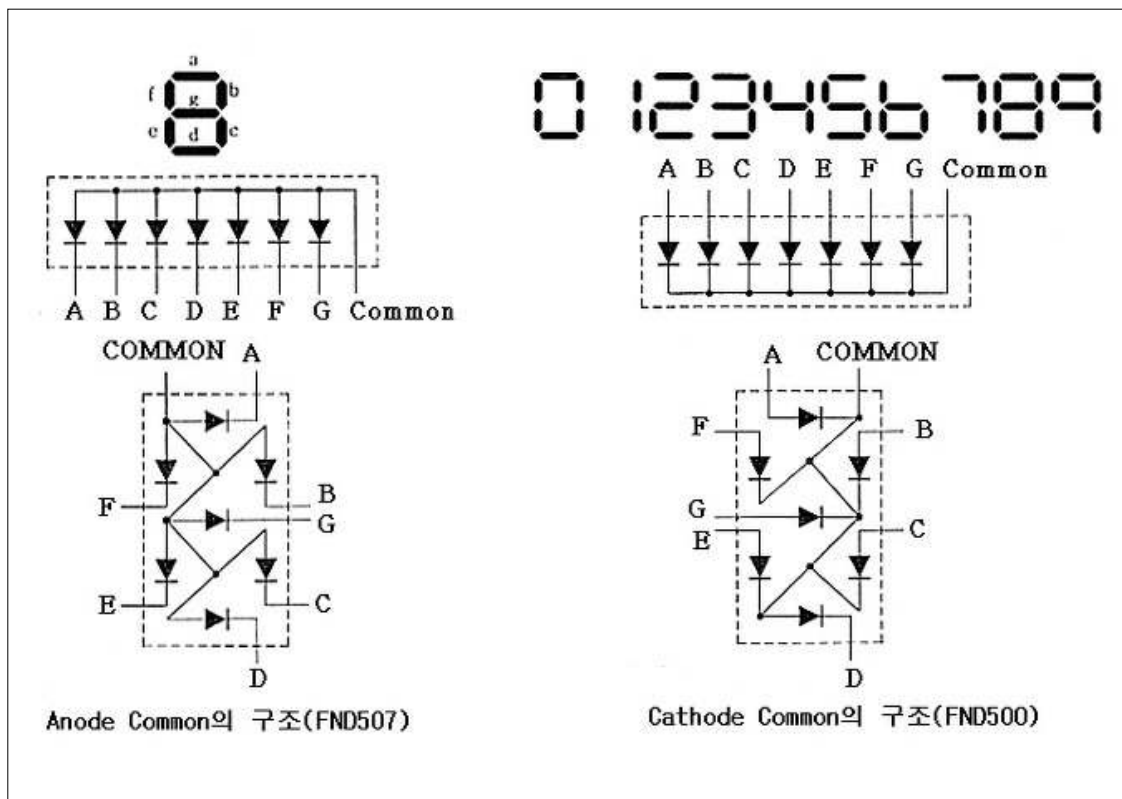
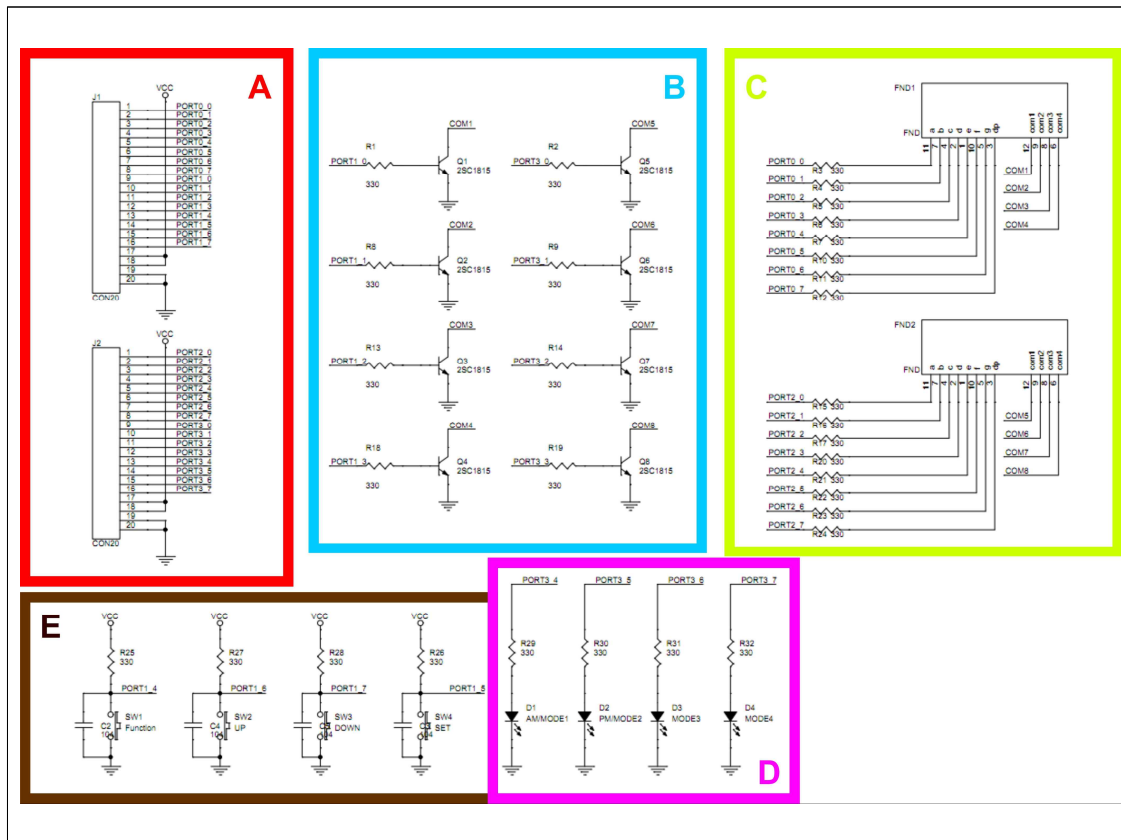


그림 1-1. FND 7-Segment Display 구조

1.2 회로도



A. 박스헤더

박스헤더는 CPU와 연결해주는 거야. 위쪽에 있는 박스헤더만 연결하면 FND 1개와 SW4개를 사용할 수 있어. 아래 있는 박스헤더만 연결하면 FND 1개 LED4개를 사용할 수 있고. 너희가 원하는 기능이 들어있는 박스헤더를 사용하면 된다는 말이지.

B. COM 단자

CPU에서 신호를 주면 원하는 트랜지스터를 동작시키면 COM 단자가 GND에 연결될 거야. 그럼 그 COM단자에 해당하는 FND가 켜지게 되는 거지.

C. FND

이번 단원의 핵심이야. 데이터 및 COM단자 선택은 CPU에서 받아서 사용될 거야. 자세한 동작 방법은 뒤에 나와!

D. LED

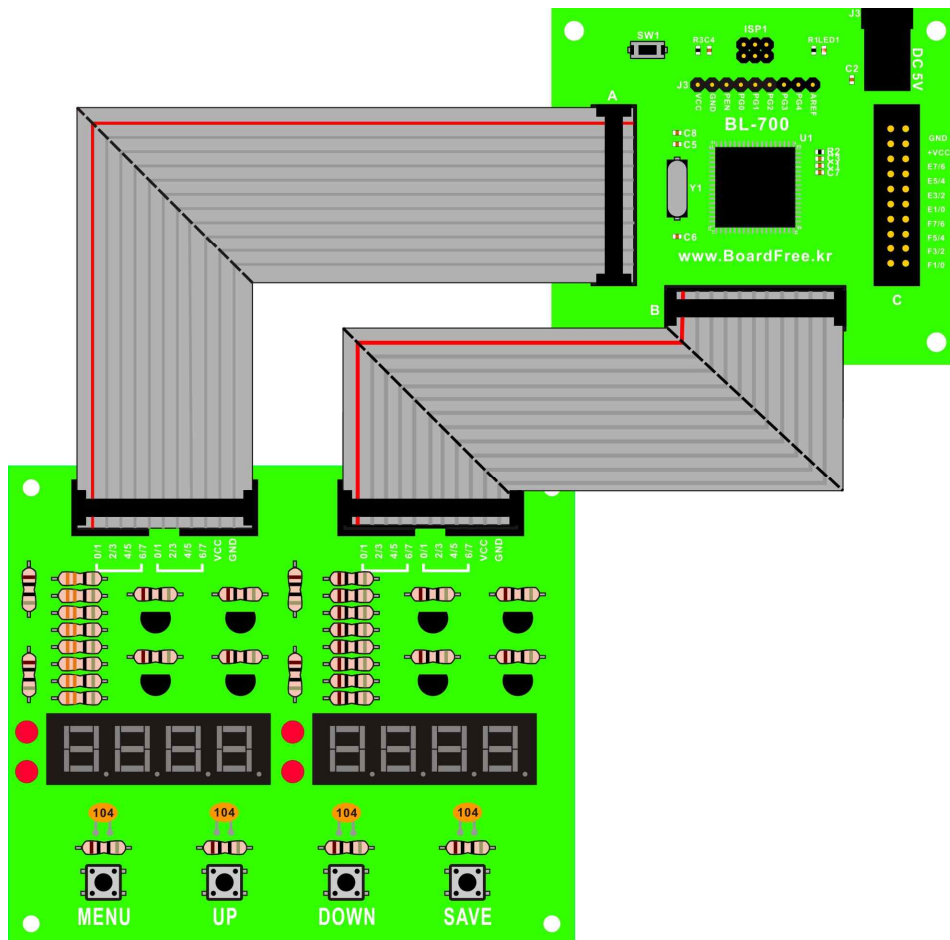
FND 말고 어떠한 표시를 하고 싶을 때 사용할 수 있도록 4개의 LED를 추가해줬어. 프로젝트들을 진행하다 보면 쓸 일이 많을 거야.

E. SW

기본 예제 말고 응용할 때 자주 사용될 거야. 총 4개의 스위치가 있어서 이것저것 많은 기능을 추가할 수 있어.

1.3 동작 해보기

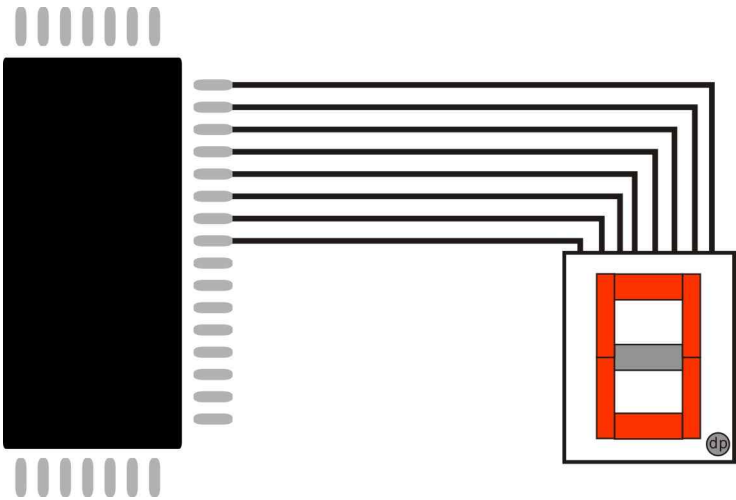
우선 CPU보드와 연결을 해야겠지? 너희 마음대로 연결해도 무관하지만 나는 어떻게 연결했는지 알려줄게. 그리고 코드도 지금 보여주는 그림을 따라서 짤 거라 너희가 다른 방법으로 연결하면 그에 따라 조금씩 변화가 있을 거야.



FND의 기본 동작은 쉽게 따라 할 수 있을 거야. 잘 생각해보면 LED 몇 개를 ON/OFF 하는 거니까 너무 어렵게 생각할 필요 없어.

먼저 C.A (Common Anode)와 C.C (Common Cathode)의 차이를 알려줄게. C.A는 anode가 묶여있는 거야. 그래서 원하는 자리에 LED를 키고 싶다면 그 자리를 GND에 연결하면 켜지는 거야. 반대로 C.C는 cathode가 묶여있으니 원하는 자리에 LED에 VCC를 연결하면 켜지는 거지.

우리가 사용하는 FND는 C.C야. 그러니 원하는 자리를 켤 땐 COM단자엔 GND를, 원하는 자리엔 VCC를 주면 되는 거지. 이해했다면 아라비아 숫자를 한번 써보자.



```
#include <mega128.h>

unsigned char fnd_num[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,
                          0x7c,0x27,0x7f,0x67};

//FND에 표시할 숫자들을 배열로 정리한 것
int fnd_sel_data; //FND 자릿수 설정
int fnd_data = 0; //FND 표시할 숫자 정하기
```

```

void Fnd_Display(int fnd_sel, int num); //FND Display 함수

void main(void)
{
    DDRB = 0xff; //포트 설정
    DDRD = 0x0f;
    while (1)
    {
        Fnd_Display(1,fnd_data); //함수 호출 “1번째 자리에 fnd_data 값 출력”
    }
}

void Fnd_Display(int fnd_sel, int num)
{
    if(fnd_sel == 1) fnd_sel_data = 0x08;
    else if(fnd_sel == 2) fnd_sel_data = 0x04;
    else if(fnd_sel == 3) fnd_sel_data = 0x02;
    else if(fnd_sel == 4) fnd_sel_data = 0x01;

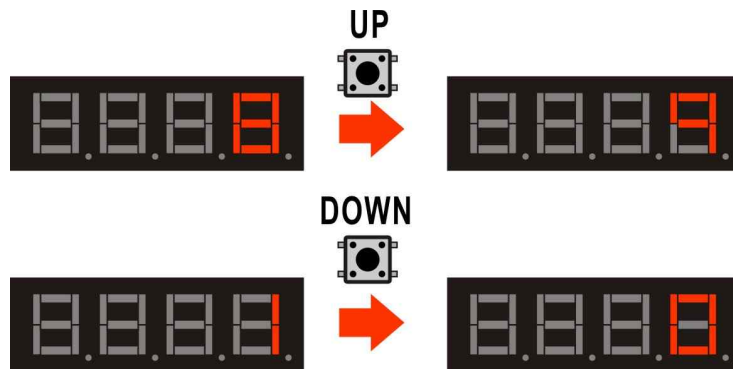
    PORTD = fnd_sel_data;

    PORTB = fnd_num[num];
}

```

위에처럼 하면 아라비아 숫자 0이 나오는 거 확인했지? 그럼 FND에는 A~G까지의 위치가 있는 거 알고 있지? 이 위치를 잘 생각해서 켜고 끈다면 쉽게 아라비아 숫자를 표시할 수 있을 거야. 예를 들면 1을 표시한다면 B,C의 해당하는 LED만 키고 다른 것들은 전부 끄면 되겠지? 이런 식으로 너희가 알아서 0~9까지의 숫자를 FND에 한번 띄어봐.

이제 스위치를 이용해 0~9를 UP/DOWN 하는 것을 해보자. UP을 누르면 +1, DOWN을 누르면 -1로 동작하게끔 FND 한자리에 표시해봐.



```
#include <mega128.h>

unsigned char fnd_num[10]={0x3f,0x06,0x5b,
                           0x4f,0x66,0x6d, 0x7c,0x27,0x7f,0x67};
//FND에 표시할 숫자들을 배열로 정리한 것
int fnd_sel_data; //FND 자릿수 설정
int fnd_data = 0; //FND 표시할 숫자 정하기

void Fnd_Display(int fnd_sel, int num); //FND Display 함수

void main(void)
{
    DDRB = 0xff; //포트 설정
    DDRD = 0x0f;
    while (1)
    {
        if(!PIND.6) //UP버튼을 누르면 fnd_data+1
        {
            fnd_data = fnd_data +1;
            if(fnd_data == 10) fnd_data =9; //10이 되면 더 이상 올라가지 않음
        }
        if(!PIND.7) //DOWN버튼을 누르면 fnd_data-1
        {
            fnd_data = fnd_data-1;
            if(fnd_data <= 0) fnd_data=0; //0이하가 되면 더 이상 내려가지 않음
        }
    }
}
```

```

    }
    Fnd_Display(1,fnd_data); //함수 호출 "1번째 자리에 fnd_data 값 출력"
}
}

void Fnd_Display(int fnd_sel, int num)
{
    if(fnd_sel == 1) fnd_sel_data = 0x08;
    else if(fnd_sel == 2) fnd_sel_data = 0x04;
    else if(fnd_sel == 3) fnd_sel_data = 0x02;
    else if(fnd_sel == 4) fnd_sel_data = 0x01;

    PORTD = fnd_sel_data;

    PORTB = fnd_num[num];
}

```

이번엔 다이내믹 디스플레이를 이용해서 해보자. LED MATRIX에서 해서 이미 알고 있지? 그 방법을 사용해서 4개의 FND에 간단하게 같은 숫자를 먼저 띄워보자. 다 했다면 숫자의 순서대로 1234 , 5678 등의 숫자를 표현해봐.



```

#include <mega128.h>

unsigned char fnd_num[10]={0x3f,0x06,0x5b,0x4f,
                          0x66,0x6d, 0x7c,0x27,0x7f,0x67};

//FND에 표시할 숫자들을 배열로 정리한 것
int fnd_sel_data; //FND 자릿수 설정
int fnd_data = 0; //FND 표시할 숫자 정하기

void Fnd_Display(int fnd_sel, int num); //FND Display 함수

```

```

void main(void)
{
    DDRB = 0xff; //포트 설정
    DDRD = 0x0f;
    while (1)
    {
        Fnd_Display(1,4);
        Fnd_Display(2,3);
        Fnd_Display(3,2);
        Fnd_Display(4,1);
    }
}

void Fnd_Display(int fnd_sel, int num)
{
    if(fnd_sel == 1) fnd_sel_data = 0x08;
    else if(fnd_sel == 2) fnd_sel_data = 0x04;
    else if(fnd_sel == 3) fnd_sel_data = 0x02;
    else if(fnd_sel == 4) fnd_sel_data = 0x01;

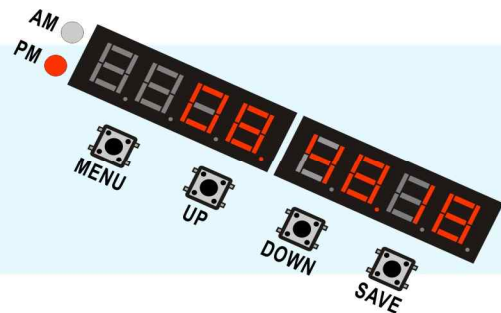
    PORTD = fnd_sel_data;

    PORTB = fnd_num[num];
}

```

2

프로젝트 1 : 전자시계



2.1 전자시계를 만들어 보자

갑자기 LED ON/OFF하다가 전자시계라고 하니까 ‘헉’하지? 걱정하지 마. 생각보다 쉬울 거야! 생각해보면 1초를 셀 수 있다는 의미는 1분

을 만들 수 있다는 거잖아? 더 나아가 1분을 만들 수 있다면 1시간을 만들 수 있고. 결국, 최종적으로 그게 시계가 되겠지.

그럼 이제부터 FND에 간단하게는 초시계부터 심화 과정으로는 우리가 자주 보는 전자시계를 만들어 볼 거야. 시계를 만들기 전에 기본부터 알아야겠지?

2.1 딜레이 사용

우선 우리는 딜레이 함수를 사용 해볼 거야. 이 함수는 칩 내부적으로 의미 없는 사이클을 만들어 시간을 딜레이를 시켜주는 함수야. 선언과 사용은 아래 코드와 같아. 이때 괄호 안에 넣은 ‘숫자 ms’ 만큼 딜레이 시켜주는 거야.

```
#include <delay.h> //delay함수 선언
-----
delay_ms(100);    //delay함수 호출 (100msec)
```

이제 딜레이 함수를 이용해 0000부터 너희가 딜레이 함수를 이용해 9999까지 가는 초시계를 만들어봐.



```
#include <mega128.h>

unsigned char fnd_num[10]={0x3f,0x06,0x5b,
                           0x4f,0x66,0x6d, 0x7c,0x27,0x7f,0x67};
//FND에 표시할 숫자들을 배열로 정리한 것
int fnd_sel_data; //FND 자릿수 설정
```

```

int fnd_data = 0; //FND 표시할 숫자 정하기

void Fnd_Display(int fnd_sel, int num); //FND Display 함수

void main(void)
{
    DDRB = 0xff; //포트 설정
    DDRD = 0x0f;
    while (1)
    {
        Fnd_Display(1,(fnd_data/1)%10);
        Fnd_Display(2,(fnd_data/10)%10);
        Fnd_Display(3,(fnd_data/100)%10);
        Fnd_Display(4,(fnd_data/1000)%10);

        delay_ms(1000);          //1초 딜레이
        fnd_data = fnd_data + 1;
    }
}

void Fnd_Display(int fnd_sel, int num)
{
    if(fnd_sel == 1) fnd_sel_data = 0x08;
    else if(fnd_sel == 2) fnd_sel_data = 0x04;
    else if(fnd_sel == 3) fnd_sel_data = 0x02;
    else if(fnd_sel == 4) fnd_sel_data = 0x01;

    PORTD = fnd_sel_data;

    PORTB = fnd_num[num];
}

```

위 예제를 해보면 뭔가 잘 안된다는 느낌이 드는 게 맞아. 이제 다음 단원을 배우면 절대 저런 일이 생기지 않고 깨끗하게 출력될 거야.

2.2 인터럽트 사용

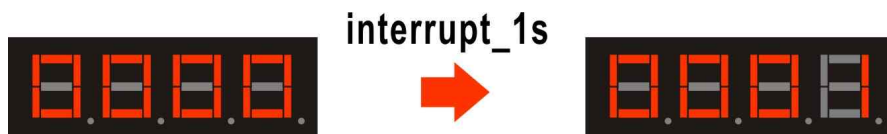
다음 단계로 넘어가기 전에 유용하게 쓰일만한 걸 하나 알려줄게. 인터럽트라고 들어봤니? 먼저 인터럽트가 뭔지 간단하게 설명해줄게. 도서관에서 공부하다가 잠시 커피를 마시고 다시 도서관에 와서 공부했어. 이 상황에서 커피를 마시러 갔던 게 바로 인터럽트야. 코드에서로 얘기하자면 MAIN문을 돌다가 어떤 조건이 충족되면 잠시 인터럽트에 가서 그 일을 하고 다시 MAIN문으로 돌아오는 게 인터럽트인 거지.

이제 데이터 시트로 들어가서 심화한 내용을 얘기해볼까? 우선 데이터 시트 92쪽부터 보자. Timer/Counter 8bit부터 레지스터를 하나씩 설명할 건데, PWM에 관련된 것들은 다음 프로젝트에 있으니까 거기에서 설명하도록 할게.

TCCR	여기선 CSxx를 제외하곤 전부 PWM에 관련된 것들이니 CSxx를 설명할게. CS는 Clock Select 의 약자로 표에 나온 것처럼 입력하면 클럭을(16Mhz) 8,32,64,128 등등 분주 할수 있어.	
TCNT	타이머 카운터 값	
OCR	카운터 값과 일치하는지 비교하여 인터럽트를 생성하는 것	
TIMSK	OCIE	Output Compare 인터럽트를 사용하겠다는 비트
	TOIE	Overflow 인터럽트를 사용하겠다는 비트

위에 내용은 8bit 설명이었고 ATmega128은 16bit도 사용 가능해. 16bit는 위에 내용과 비슷하니까 너희가 데이터 시트를 보고 이해해봐.

자, 이제 인터럽트 사용법도 알았으니 이제 인터럽트를 이용해 시계를 만들어보아야지? 0000~9999까지 1초에 1씩 올라가는 초시계를 만들어 보자.



```
#include <mega128.h>
#include <delay.h>

unsigned char fnd_num[10] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,
0x7c,0x27,0x7f,0x67};
int sel_num = 0;
int fnd_sel_data;
int fnd_sel;
void Fnd_Display(int fnd_sel, int num);
void Fnd_Display_isr();
int fnd_data = 0;
int sec_cnt = 0;
void main(void)
{
    DDRB = 0xff;
    DDRD = 0x0f;
    DDRC = 0xff;
    DDRA = 0xff;

    TCCR0 = 0x0c;
    TCNT0 = 0x00;
    OCR0 = 0xF9;
    TIMSK=0x02;
    #asm("sei")

    while (1)
```

```

    {
        Fnd_Display(1,(fnd_data/1)%10);
        Fnd_Display(2,(fnd_data/10)%10);
        Fnd_Display(3,(fnd_data/100)%10);
        Fnd_Display(4,(fnd_data/1000)%10);
    }
}

void Fnd_Display(int fnd_sel, int num)
{
    if(fnd_sel == 1) fnd_sel_data = 0x08;
    else if(fnd_sel == 2) fnd_sel_data = 0x04;
    else if(fnd_sel == 3) fnd_sel_data = 0x02;
    else if(fnd_sel == 4) fnd_sel_data = 0x01;

    PORTD = fnd_sel_data;

    PORTB = fnd_num[num];

}

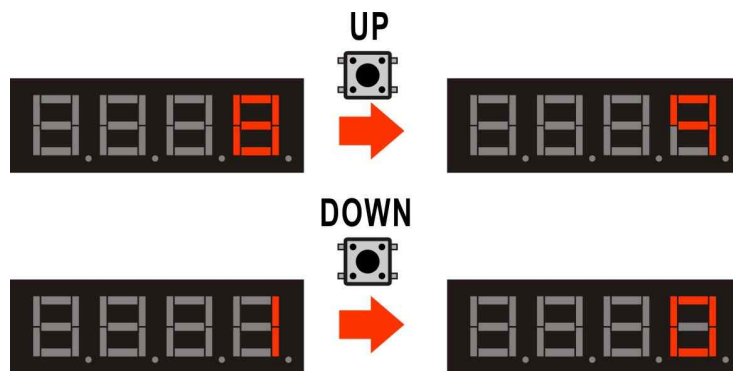
interrupt [TIM0_COMP] void Timer0_1ms_Interrupt (void)
{
    sec_cnt=sec_cnt+1;
    if(sec_cnt == 1000)
    {
        sec_cnt = 0;
        if(fnd_data >=10000) fnd_data = 0;
        else fnd_data = fnd_data + 1;
    }
}

```

딜레이를 사용했을 때와는 느낌이 다르지 않아? 이전에 while() 문을 돌면서 fnd_data를 1초씩 딜레이하고 FND에 표시했던거 기억하지? 그 방법은 사람의 눈을 속일만큼이 아니여서, 즉 자연스럽지 않아서 뭔가 이상하다는 생각이 들었던거야. 하지만 이제 인터럽트를 사용해 디스플레이를 하던 중에도 fnd_data를 변경할 수 있어서 문제가 없을거야.

그리고 전에 했던 스위치를 이용해 값이 변하게 하는 예제를 해보면 스위치를 한 번만 눌렀을 뿐인데 이상하게 여러 번 숫자가 변하거나 값이 끝까지 올라가는 일이 있었을 거야. 왜 그런건지 눈치챘어?

스위치의 접점이 불안정해서 한번 누른 것이 여러 번 눌리는 거라 인식 돼서 그런거야. 그래서 SW를 인식하는 if()문에 계속 걸려서 값이 변하는 거지. 그걸 막는 것이 채터링 이라는 거야. 이제부터 인터럽트를 이용해 채터링을 어떻게 하는지 알려줄게.



```
#include <mega128.h>
#include <delay.h>

#define UP PIND.6
#define DOWN PIND.7
#define CHT_value 200

unsigned char fnd_num[10] = {0x3f,0x06,0x5b,0x4f,
                             0x66,0x6d, 0x7c,0x27,0x7f,0x67};

int sel_num = 0;
int fnd_sel_data;
int fnd_sel;
void Fnd_Display(int fnd_sel, int num);
void Fnd_Display_isr();
int fnd_data = 0;
int sec_cnt = 0;
```

```

int UP_CHT, DOWN_CHT;

void main(void)
{
    DDRB = 0xff;
    DDRD = 0x0f;
    DDRC = 0xff;

    DDRA = 0xff;

    TCCR0 = 0x0c;
    TCNT0 = 0x00;
    OCR0 = 0xF9;
    TIMSK=0x02;
    #asm("sei")

    while (1)
    {
        Fnd_Display(1,(fnd_data/1)%10);
        Fnd_Display(2,(fnd_data/10)%10);
        Fnd_Display(3,(fnd_data/100)%10);
        Fnd_Display(4,(fnd_data/1000)%10);
    }
}

void Fnd_Display(int fnd_sel, int num)
{
    if(fnd_sel == 1) fnd_sel_data = 0x08;
    else if(fnd_sel == 2) fnd_sel_data = 0x04;
    else if(fnd_sel == 3) fnd_sel_data = 0x02;
    else if(fnd_sel == 4) fnd_sel_data = 0x01;

    PORTD = fnd_sel_data;

    PORTB = fnd_num[num];
}

interrupt [TIM0_COMP] void Timer0_1ms_Interrupt (void)
{

```

```

if(UP == 0)
{
    if(UP_CHT==CHT_value)
    {
        UP_CHT=0;
        fnd_data = fnd_data + 1;
    }
    else
    {
        UP_CHT+=1;
    }
}
else UP_CHT=0;

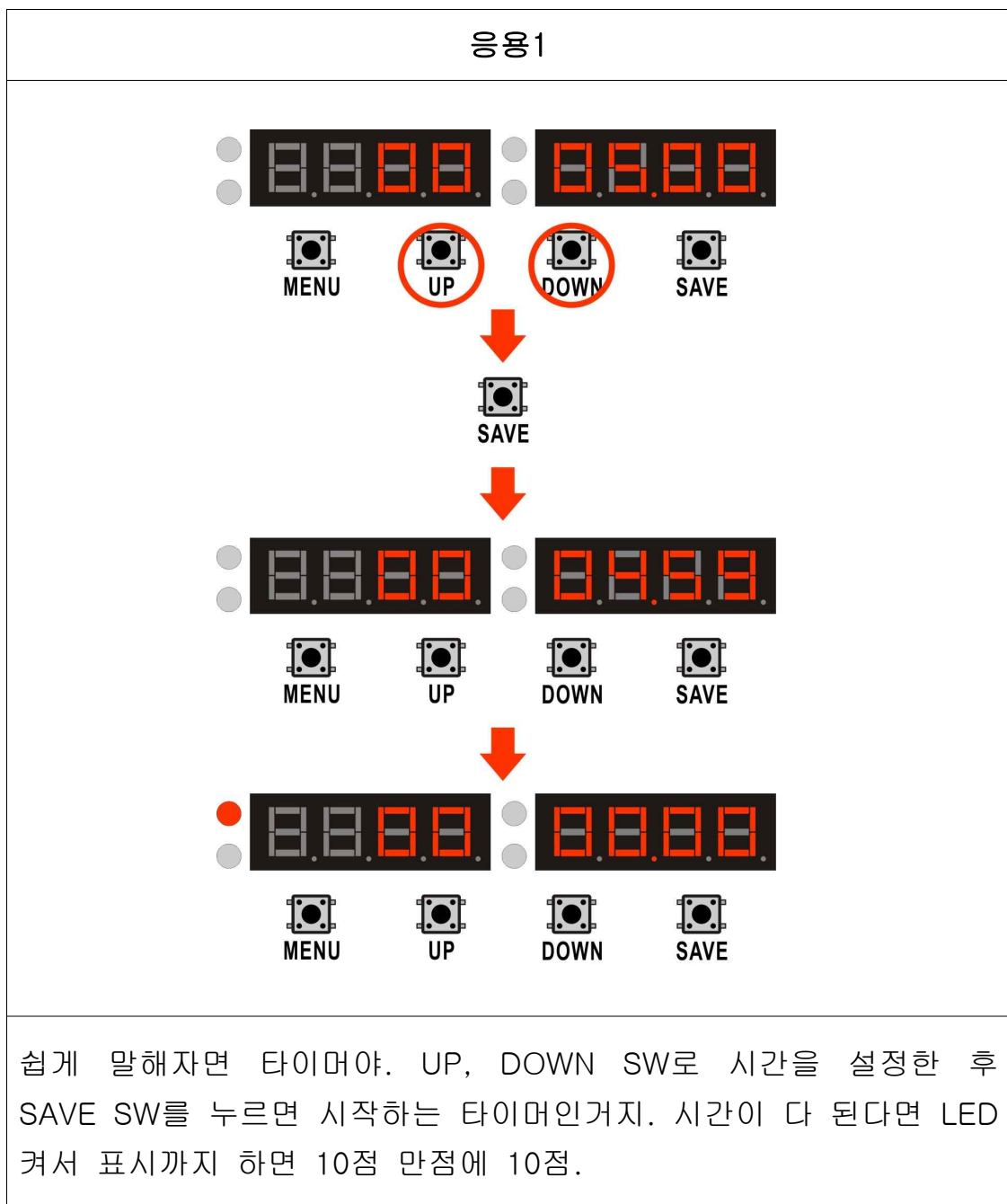
if(DOWN == 0)
{
    if(DOWN_CHT==CHT_value)
    {
        DOWN_CHT=0;
        fnd_data = fnd_data - 1;
    }
    else
    {
        DOWN_CHT+=1;
    }
}
else DOWN_CHT=0;
}

```

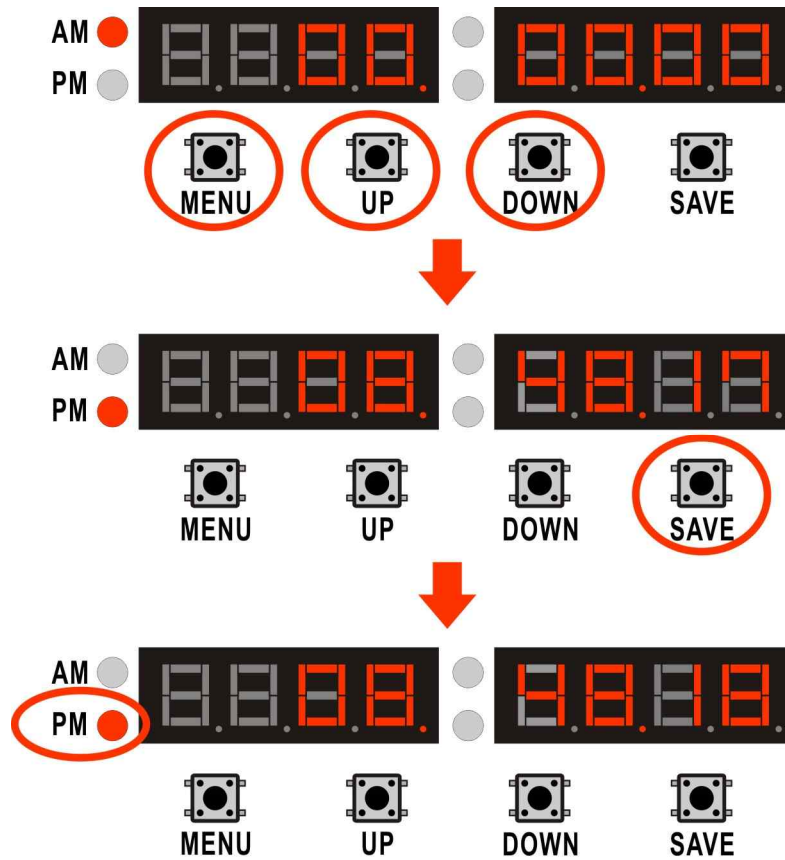
위에 예제를 했다면 아까 했던 STEP1, STEP2있지? 그걸 꼭 하고 다음으로 넘어가.

2.3 프로젝트를 해보자 : 전자시계

이제 인터럽트를 사용하는 이유는 이전 내용을 충분히 해봤다면 이해했을 거라고 생각해. 그럼 이제 인터럽트를 이용해 두 개의 시계를 만들어볼 거야. 위에 예제를 이해하고 응용할 줄 안다면 너희도 간단하게 전자시계를 만들어서 볼 수 있을 거야.



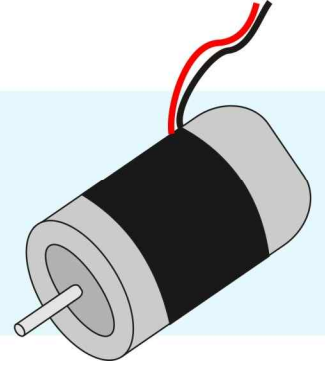
응용2



AM, PM이 표시되어 11시 59분 59초 까지 표시되는 시계야. MENU SW를 누르면 시계가 멈추고 시간을 마음대로 바꿀 수 있고, 다시 SAVE를 누르면 다시 설정된 시간부터 흐르는 시계인거지.

3

프로젝트 2 : 모터제어기



3.1 DC 모터

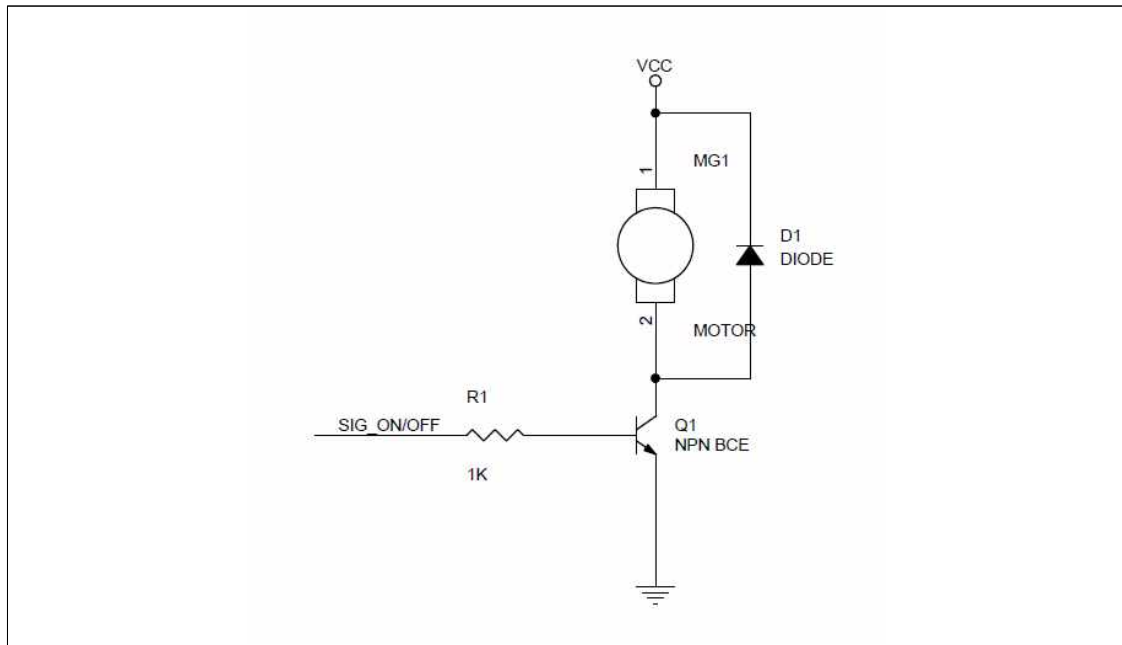
제목의 모터 제어기는 너희가 상상한 대로 모터의 속도, 방향 등을 조절할 수 있는 것을 말해. 모터 제어기를 만들기 전에 우선 DC모터가 어떤 것인지부터 알아야겠지? DC모터는 전압을 넣어주면 회전하고 극성을 바꾸면 반대의 방향으로 회전하는 거야.



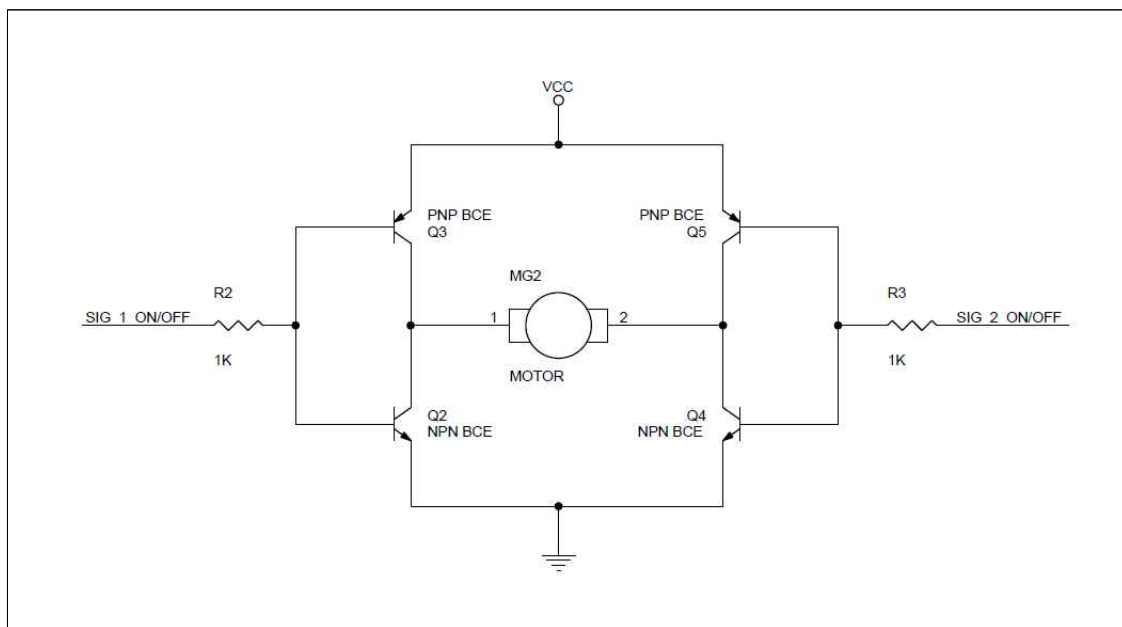
그림3-1.실제 모터 사진

DC모터를 그냥 ATMEGA128에 연결하고 사용해도 문제 없을 거라 생각하겠지만 ATMEGA128이 출력할 수 있는 전류가 한정적이라 모터를 회전시키기엔 부족할 수가 있어. 그래서 우리는 출력 전류를 증폭해 사용할 거야.

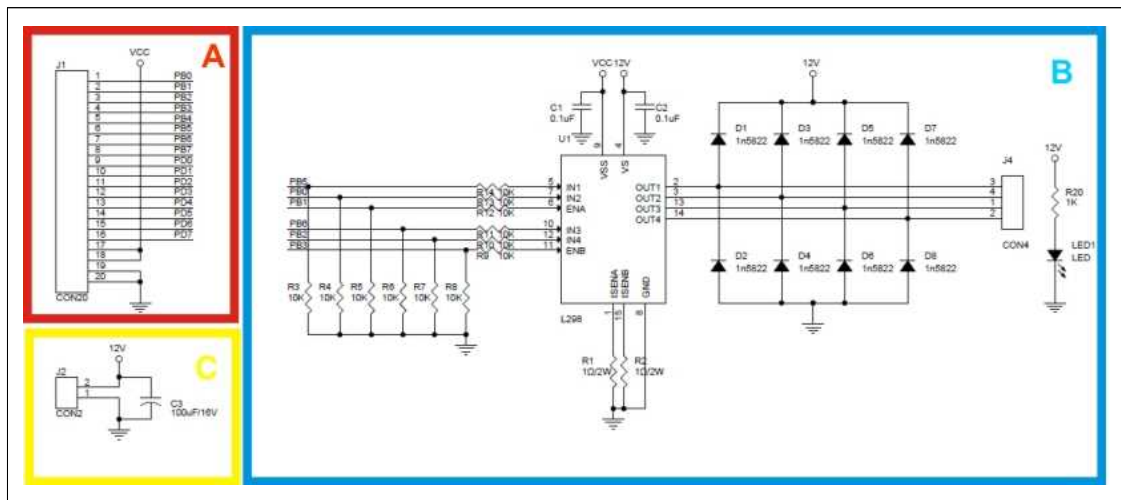
DC 모터를 제어하는 방법은 여러 가지가 있을 수 있는데, 우선 트랜지스터의 컬렉터의 모터를 연결해서 하는 방법을 설명해줄게. 아래 그림을 보면서 이해해봐.



트랜지스터 베이스에 전압을 ON/OFF해서 모터를 제어하는 방법이야. 모터 옆에 다이오드의 기능은 환류 다이오드로, 역기전력이 발생하는 것을 방지해주는 역할을 해. 하지만 이 방법의 문제는 모터의 방향을 바꿀 수 없다는 것이야. 이 방법이 개선되어 있는 방법은 바로 H-브릿지 회로야.



3.2 회로도



CPU로부터 L298에 데이터를 보내기 위한 박스헤더야.

L298을 간단하게 설명 해줄 건데, 너희도 데이터 시트를 꼭 한 번 봐 봐. 큰 도움이 될 거야. L298은 2개의 브리지 드라이버가 내장되어 있어서 2개의 모터까지 컨트롤이 가능해. 최대 전류는 2.5A이고 고속 구동이 가능한 칩이야.

이 커넥터는 너희가 5V의 모터를 돌릴지 12V의 모터를 돌릴지 몰라서 만들어졌어. 사용할 때 너희 상황에 알맞은 전원을 공급하면 돼.

3.3 PWM

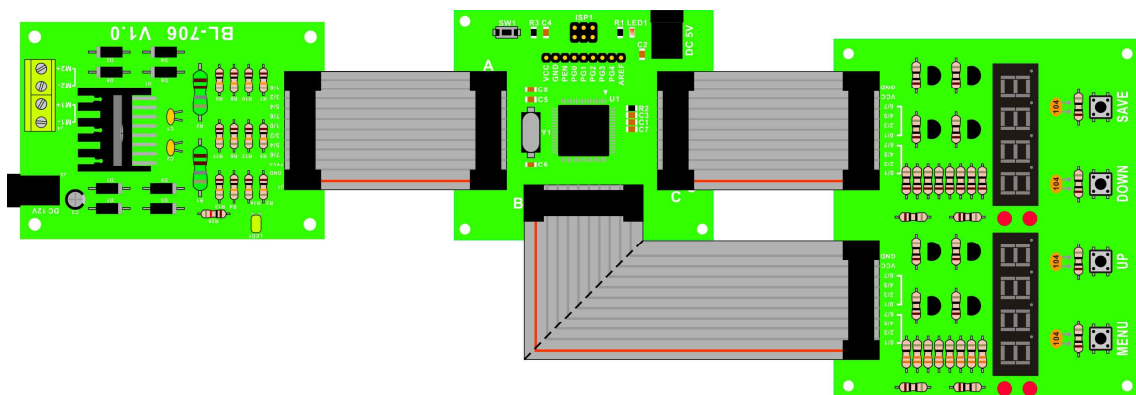
PWM은 보통 너희가 모터의 속도를 제어하거나 LED의 밝기를 제어하는 등의 역할로 사용하게 될 거야. 데이터 시트를 보면서 PWM을 설명

해줄게. 인터럽트 때처럼 8bit를 기준으로 설명해줄게. 92쪽부터 쪽 봐
봐.

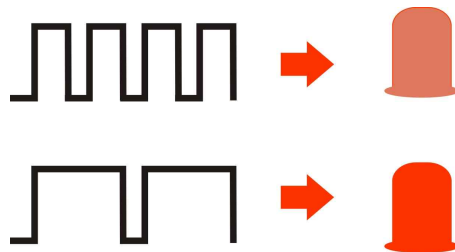
TCCR은 CS를 제외한 bit들을 설명해줄게. 우선 WGM은 PWM의 모드를
설정한다고 생각하면 편해. Normal, PWM Phase Correct, CTC,
Fast PWM의 종류가 있지? 이것들은 너희가 데이터 시트를 보는 게
더 잘 설명되어 있어서 읽어보면서 이해하는 걸 추천할게. 이해가 다
되었다면 너희 상황에 알맞은 걸 사용할 수 있을 거야.

다음으로 설명해줄 건 COM bit야. 위에 WGM으로 모드를 설정했지?
그럼 그 모드에서 PWM을 어떻게 사용할지 설정할 수 있는 bit야. 여
기까지 읽고 데이터 시트를 읽었다면 PWM을 어느 정도 이해했을 거
라고 생각해.

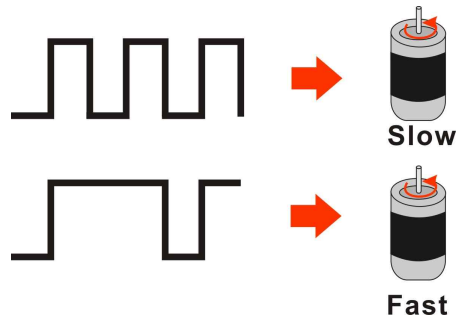
보드 간의 연결은 아래 그림처럼 했어.



이제 PWM을 설정해서 LED의 밝기를 변화 시키는 것을 해보자.



LED로 해봤다면 이제 PWM을 이용해 모터의 속도를 변화시켜보자.



```
#include <mega128.h>

void main()
{
    TCCR1A = 0x83;
    TCCR1B = 0x0B;
    TCNT1 = 0;

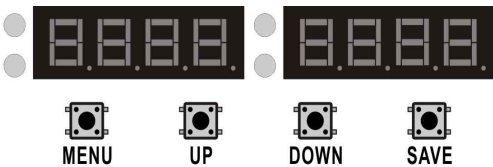
    DDRB = 0xff;


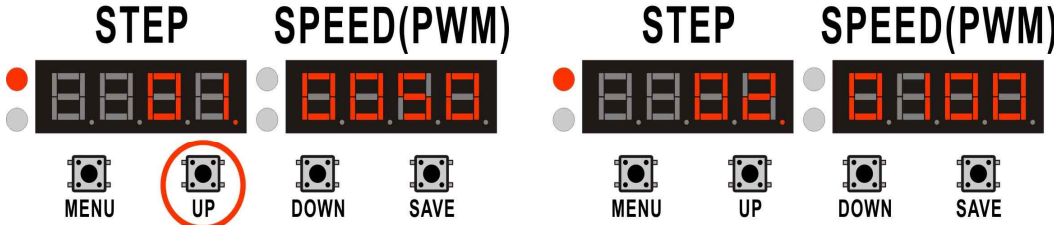
    while(1)
    {
        OC1A = 500;
    }
}
```


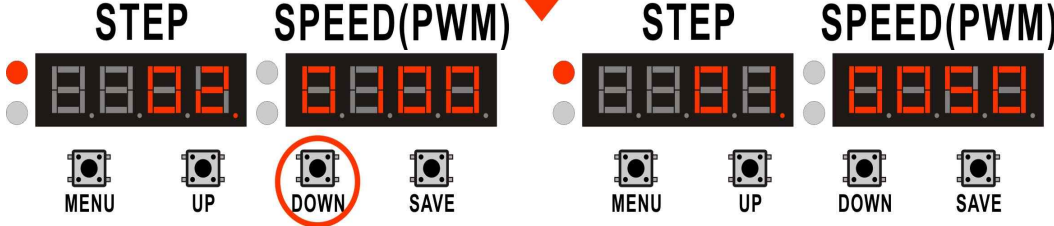
3.4 응용해보기

DC 모터가 무엇인지 알았고 PWM을 사용하는 방법까지 이해했다면 이제 이 두 개를 응용해서 모터제어기를 만들어보자!

응용1

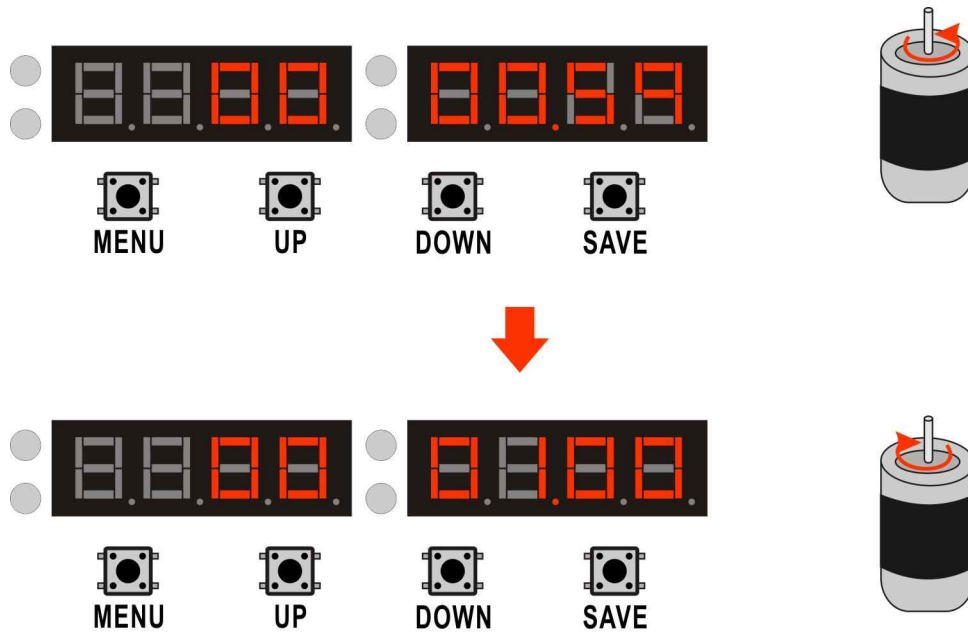


여기서 MENU는 쉽게 말해 설정이야. 그래서 MENU를 누르면 UP/DOWN 스위치로 단계를 바꿀 수 있는 거지. MENU를 눌렀다는 걸 표시하려면 LED를 켜든가 어떤 표시를 하는 게 좋겠지? 오른쪽 FND는 너희가 코드에서 설정한 각 단계별 PWM값을 띄우면 되는 거고. 설정을 다 한 뒤에 SAVE 를 하면 모터가 그 속도에 맞게 돌아가게 하면 돼.

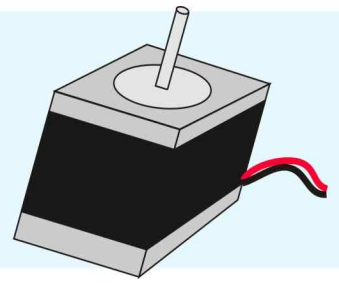
응용2



1분의 타이머를 만들어서 1분마다 모터의 방향을 바꾸면 돼. 대신 이때 다시 0초부터 세는 거고. 속도는 너희 마음대로 해도 좋아. 이 외의 여러 가지 기능을 넣고 싶다면 넣어도 좋아!

4

프로젝트 3 : 스텝모터제어기



4.1 스텝 모터

이번에는 스텝 모터로 전 단원과 비슷한 제어기를 만들 거야. 이번에도 만들기 전에 스텝 모터가 무엇인지 알고 가야겠지? DC모터는 전원을 넣어주면 그저 돌기만 했다면 스텝 모터는 펄스 신호에 의해 회전 각도와 속도를 쉽게 제어하는 모터야.



그림4-1.스텝모터 실제 사진

스텝 모터는 선이 4개인 바이폴라 방식, 선이 6개인 유니폴라 방식으로 나뉘어. 바이폴라는 전류가 양방향으로 흐르는 것이고 유니폴라는 한쪽 방향으로만 흐르는 방식이지 아래 그림을 봐봐.

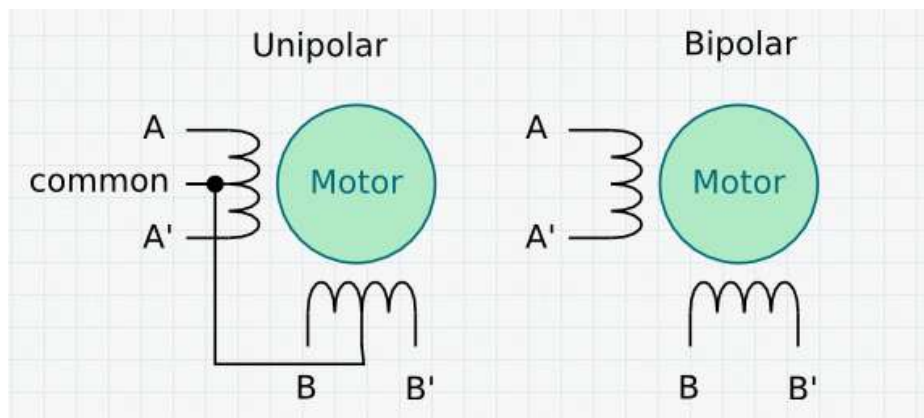
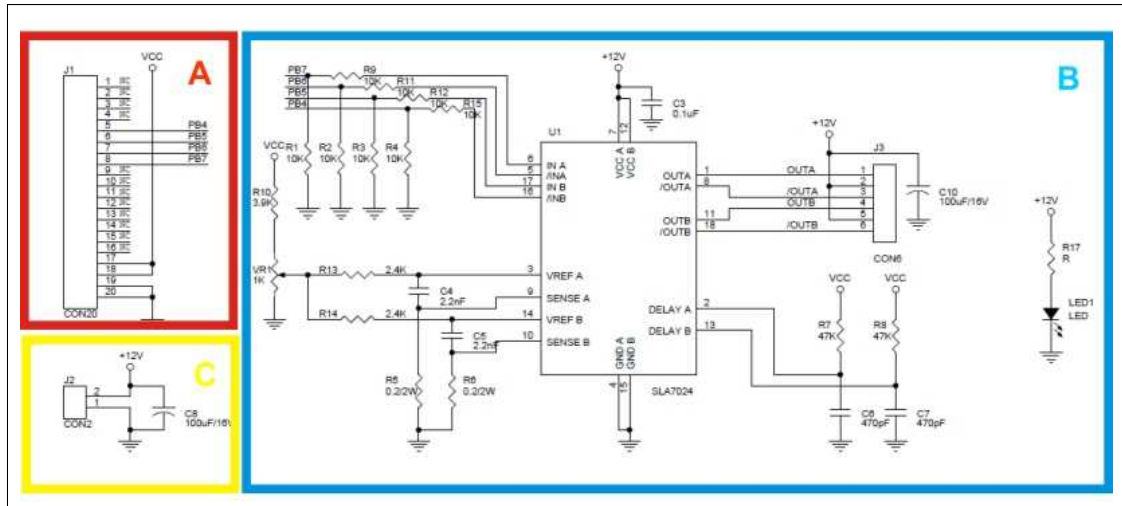


그림4-2. 바이폴라와 유니폴라

4.2 회로도



A. 박스헤더

1개의 스텝 모터를 구동할 수 있는 포트가 연결된 박스헤더야.

B. SLA7024

SLA7024은 전과 마찬가지로 데이터 시트를 읽어 보는 걸 추천할게.

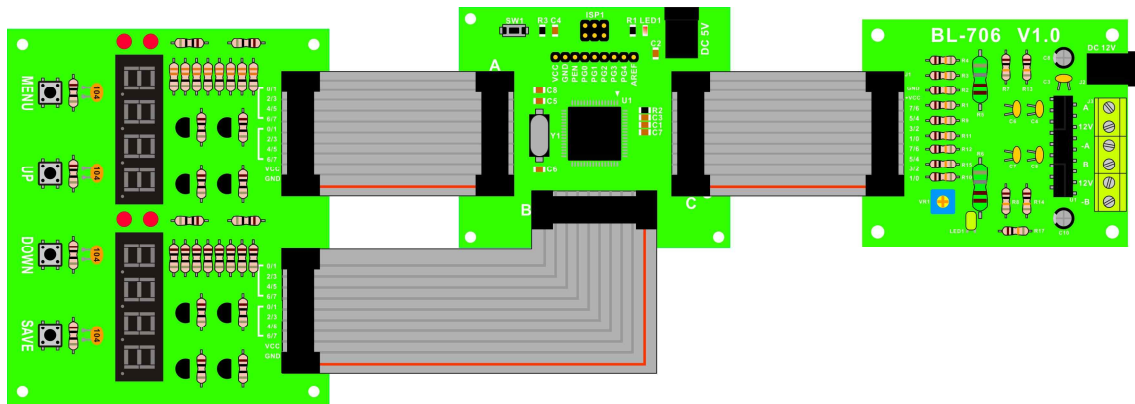
C. 커넥터

이 커넥터는 너희가 구동할 스텝 모터에 맞춰 전압을 넣으면 돼. 물론 그 전에 데이터 시트를 통해 미리 SLA7024의 전압 및 전류 등을 체크해야겠지?

4.3 구동해보기

다음은 구동방식을 알려줄 거야. 그전에 펄스를 넣어주는 순서는 A-B-/A-/B 순서야. 물론 B부터 시작하던 A-부터 시작하든 상관은 없는데 순서는 꼭 저렇게 해야 해. 이해했다고 생각하고 먼저 1상 여자방식을 알려줄게. 아래 그림들은 전부 SLA7024 데이터 시트를 보면 나와 있어.

보드 간의 연결은 다음에 그림과 같이 했어.



1상 여자 방식은 각 신호선에 하나의 펄스를 상의 순서에 맞게 넣어주어서 회전시키는 방법이야. 이 방식은 효율이 좋은 대신 진동이 발생해.

WAVE DRIVE (FULL STEP) for SLA7024M and SLA7026M

Sequence	0	1	2	3	0
Input A	H	L	L	L	H
Input \bar{A}	L	L	H	L	L
Input B	L	H	L	L	L
Input \bar{B}	L	L	L	H	L
Output ON	A	B	\bar{A}	\bar{B}	A

1상 여자방식 코드만 알려줄게. 이것만 이해하면 2상, 1-2상 여자방식도 쉽게 할 수 있을 거야.

```
#include <mega128.h>
#include <delay.h>

#define STEP_PORT    PORTB
unsigned char one_ms_count=0;
unsigned char step_count=0;
unsigned char step[4]={0x10,0x40,0x20,0x80};

void step_up(void);
void step_down(void);
void step_task(void);
void cpu_init(void);
```

```

interrupt [TIM0_COMP] void timer0_comp_isr(void)
{
    // Place your code here
    one_ms_count=one_ms_count+1;
    if(one_ms_count>10)
    {
        one_ms_count=0;
        step_up();
    }
}

void main(void)
{
    cpu_init();
    while (1)
    {
        step_task();
    }
}

void step_up()
{
    step_count=(step_count+1)%4;
}

void step_down()
{
    if(step_count==0)
    {
        step_count=3;
    }
    else
    {
        step_count=step_count-1;
    }
}

void step_task()
{

```



```

        STEP_PORT=step[step_count];
    }

void cpu_init(void)
{
    PORTB=0x00;
    DDRB=0xF0;
    ASSR=0x00;
    TCCR0=0x0C;
    TCNT0=0x00;
    OCR0=0xF9;
    TCCR2=0x00;
    TCNT2=0x00;
    OCR2=0x00;
    TIMSK=0x02;
    ETIMSK=0x00;
    #asm("sei")
}

```

다음은 2상 여자 방식이야. 2상 여자 방식은 2개의 펄스를 넣어주는 방식이야. 1상에 비해 효율이 떨어지지만, 진동은 1상에 비해 적어.

2-PHASE (FULL STEP) OPERATION for SLA7024M and SLA7026M

Sequence	0	1	2	3	0
Input A	H	L	L	H	H
Input \overline{A}	L	H	H	L	L
Input B	H	H	L	L	H
Input \overline{B}	L	L	H	H	L
Outputs ON	AB	$\overline{A} B$	$\overline{A} \overline{B}$	A \overline{B}	AB

마지막으로 1-2상 여자 방식이야. 이 1-2상 방식은 1상과 2상을 교대 해서 하는 거라 스텝 각이 반으로 줄어들어. 스텝 각이 반이 되었기 때문에 진동과 소음이 줄지만, 정확도가 떨어져.

HALF-STEP OPERATION (2-1-2 SEQUENCE) for SLA7024M, SLA7026M, and SMA7029M

Sequence	0	1	2	3	4	5	6	7	0
Input A	H	H	L	L	L	L	L	H	H
Input \overline{A} or t_{dA}^*	L	L	L	H	H	H	L	L	L
Input B	L	H	H	H	L	L	L	L	L
Input \overline{B} or t_{dB}^*	L	L	L	L	L	H	H	H	L
Output(s) ON	A	AB	B	$\overline{A} B$	\overline{A}	$\overline{A} \overline{B}$	\overline{B}	A \overline{B}	A

*Logic signals to external open-collector inverter connected to t_{dA} and t_{dB} .

4.4 프로젝트를 해보자 : 스텝모터제어기

이제 스텝모터가 무엇인지 알았고 구동 방식도 알았으니 응용해보자.

응용1

MENU UP DOWN SAVE

→

MENU UP DOWN SAVE

MENU UP DOWN SAVE

→

MENU UP DOWN SAVE

FORWARD REVERSE

MENU UP DOWN SAVE

MENU UP DOWN SAVE

Motor rotating clockwise

FORWARD REVERSE

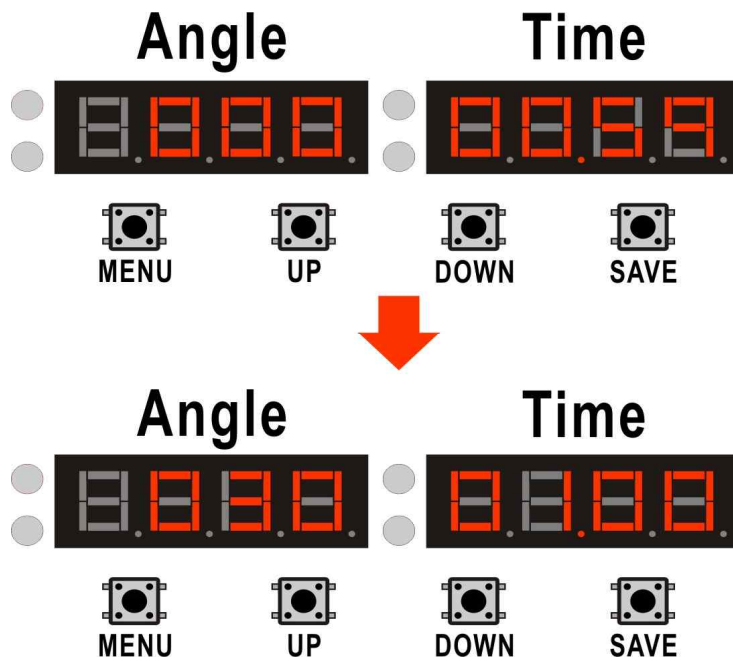
MENU UP DOWN SAVE

MENU UP DOWN SAVE

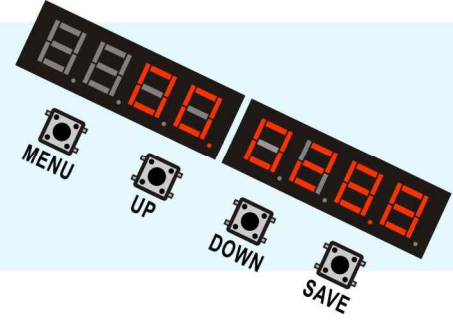
Motor rotating counter-clockwise

UP/DOWN을 통해 각도를 바꿀 수 있고 메뉴를 눌러 FORWARD/REVERSE를 설정할 수 있어. 그리고 SAVE를 누르면 설정한 각과 방향으로 모터가 움직이게 하면 돼.

응용2



이건 아날로그 시계의 동작을 따온 거야. 1시간 마다 30도씩 시계 방향으로 움직일 건데, 12시간을 기다려서 볼 순 없잖아? 그러니까 1분마다로 해도 좋고 너희가 원하는 시간으로 설정해서 360도가 돌아가게 해봐.

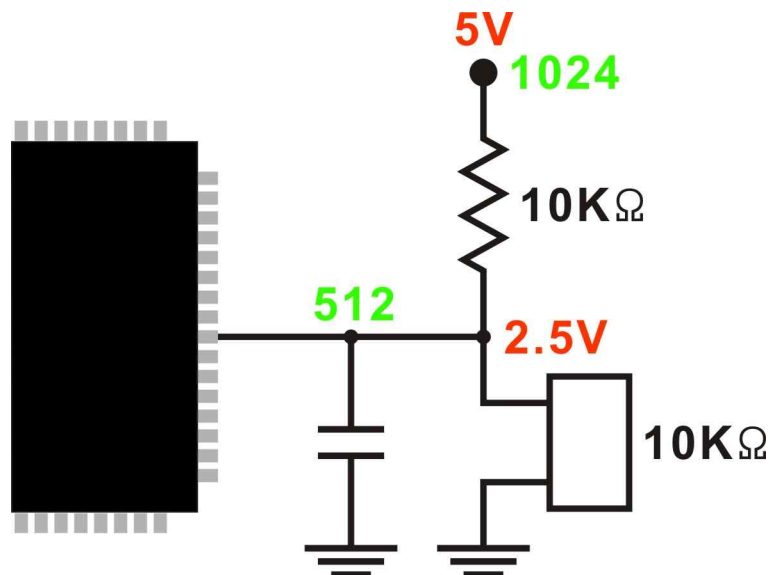


5.1 전자온도계

온도계가 뭔지 알지? 이번 단원에선 온도감지 센서를 이용해서 FND에 온도를 나타낼 수 있는 전자 온도계를 만들 거야. 우선 온도감지 센서에 대해 알아봐야겠지? 그건 아래쪽에서 나오니까 같이 보도록 하고 일단 ADC에 대해 공부하자.

5.2 ADC

써미스터는 기본적으로 온도가 올라가면 저항이 줄어들거나 늘어나는 소자야. 저항이 바뀌는데 온도가 어떻게 알 수 있지 싶지? 이걸 설명하기 위해선 ADC를 알아야 하니까 이 부분 먼저 설명할게. ADC는 아날로그 신호를 10비트 디지털 신호를 바꿔주는 기능이야. 예를 들면 5V를 기준일 때, 2.5V가 들어온다면 변수에 512라는 숫자가 들어가는 거야. 5V가 들어오면 1024라는 숫자가 변수에 들어가는 거지.



여기서 기준이 되는 전압은 내부 레퍼런스 값, AVCC 값, VREF 값으로 설정할 수 있어. 내부 레퍼런스는 내부에서 생성되는 2.54V를 기준

전압으로 삼는 거야. AVCC, VREF는 그 핀에 들어가는 전압을 ADC의 기준전압으로 만드는 거야.

이제부터 사용법에 대해 알려줄게. ADC는 ADMUX, ADCSRA의 값을 통해 사용할 수 있어.

A D M U X	Bit	7	6	5	4	3	2	1	0	ADMUX
		REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	
	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	Initial Value	0	0	0	0	0	0	0	0	

그 중 ADMUX는 어떤 채널을 사용할 건지 결정할 수 있어. 위에 붉은 네모 칸이 채널을 정하는 부분이야. 0이면 ADC0채널을 읽고 1이면 1번 채널을 읽는 식이지. 파란색 네모 칸은 기준 전압을 설정하는 칸이야. 이전 설명처럼 어떤 전압을 기준전압으로 가질지를 결정하는 거지.

ADCL AND ADCH	ADLAR = 0:									
	Bit	15	14	13	12	11	10	9	8	ADCH ADCL
		ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	
	Read/Write	R	R	R	R	R	R	R	R	
	Initial Value	0	0	0	0	0	0	0	0	
		0	0	0	0	0	0	0	0	
	ADLAR = 1:									
	Bit	15	14	13	12	11	10	9	8	ADCH ADCL
		ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	
	Read/Write	R	R	R	R	R	R	R	R	
	Initial Value	0	0	0	0	0	0	0	0	
		0	0	0	0	0	0	0	0	

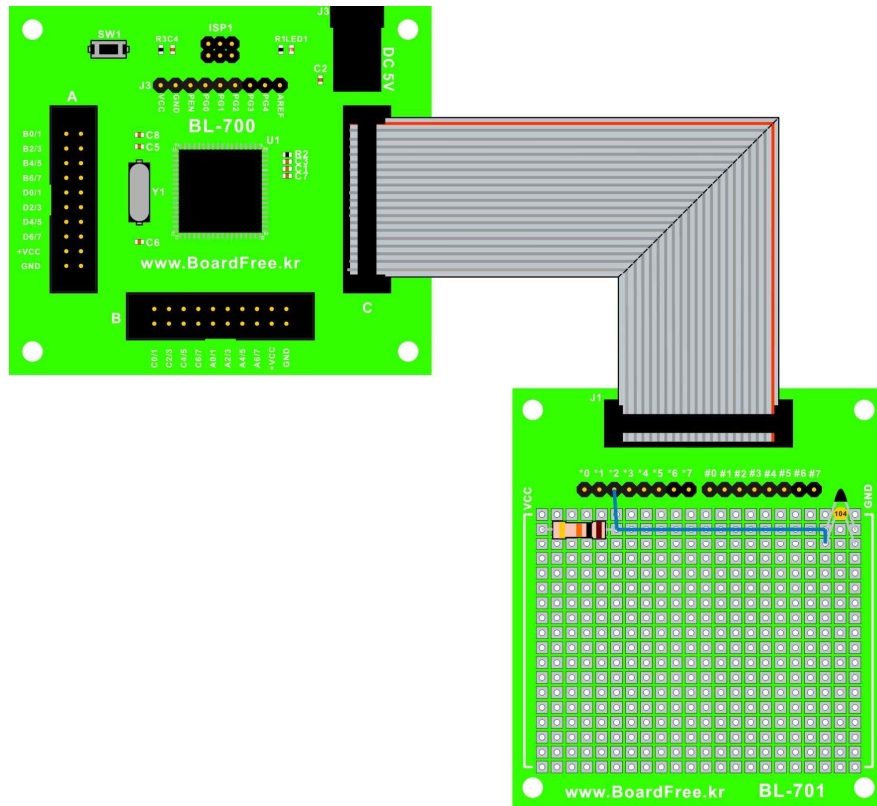
ADLAR은 10bit의 디지털 신호의 배치를 바꾼다고 생각하면 돼. 10bit는 ADCH와 ADCL로 나누어져. ADLAR의 값에 따라 ADCH의 8번 비트가 디지털 신호의 10bit가 되거나 ADCH의 2번 비트가 디지털 신호

의 10bit가 되는 것을 결정하는 비트야. 잘 이해가 안 가면 3번째 그림을 보면 좋을 거야.

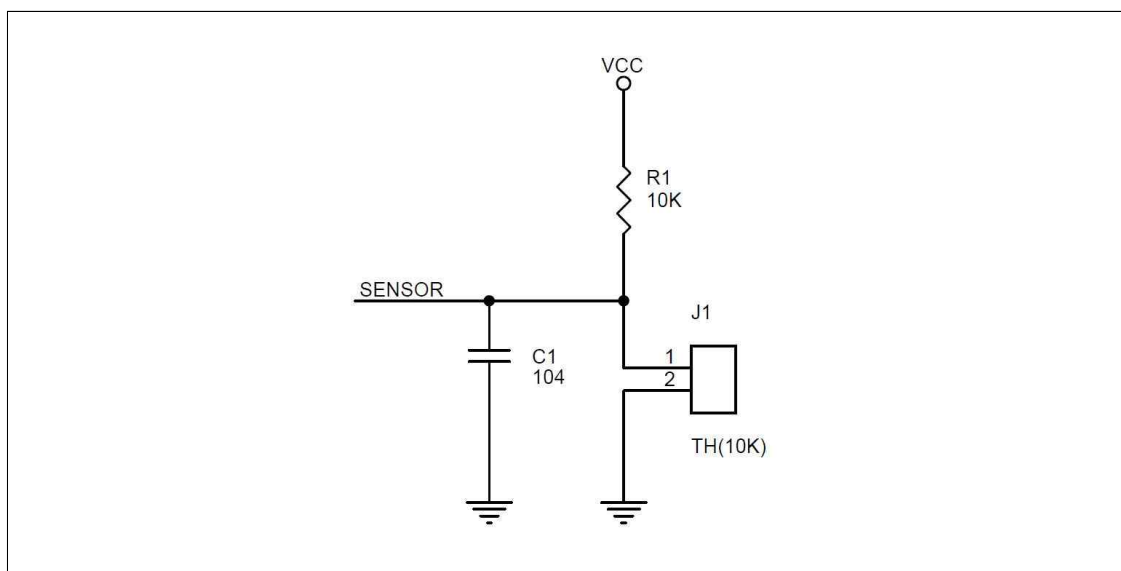
ADCSRA								
Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
7번 Bit	1이라면 ADC가 활성화 되는거고 0일때는 비활성화 되는 거야							
6번 Bit	변환 완료 비트로 싱글 컨버전 모드 시작 시점에 1로 설정하면 끝나고 자동으로 0으로 전환되는 거야.							
5번 Bit	프리 런닝 모드로 1로 설정하면 연속으로 변환을 시작하는 거야.							
4번 Bit	변환이 끝날 때마다 1로 설정되니까 이 비트로 변환 완료를 체크할 수 있어							
3번 Bit	ADC인터럽트로 ADC변환이 끝나면 인터럽트가 한번 들어오는 비트야							
2,1,0번 Bit	ADC의 분주비를 결정해주는 비트야. 분주비에 관한 설명은 위에서 봤기 때문에 알고 있지? 분주비로 나뉜 시간만큼 변환을 실행하는 거야.							

이제 각 비트의 용도는 알겠지? 이제 실제로 써미스터를 사용해서 ADC를 해볼 거야.

나는 아래 그림과 같이 보드를 연결했어.



5.3 써미스터



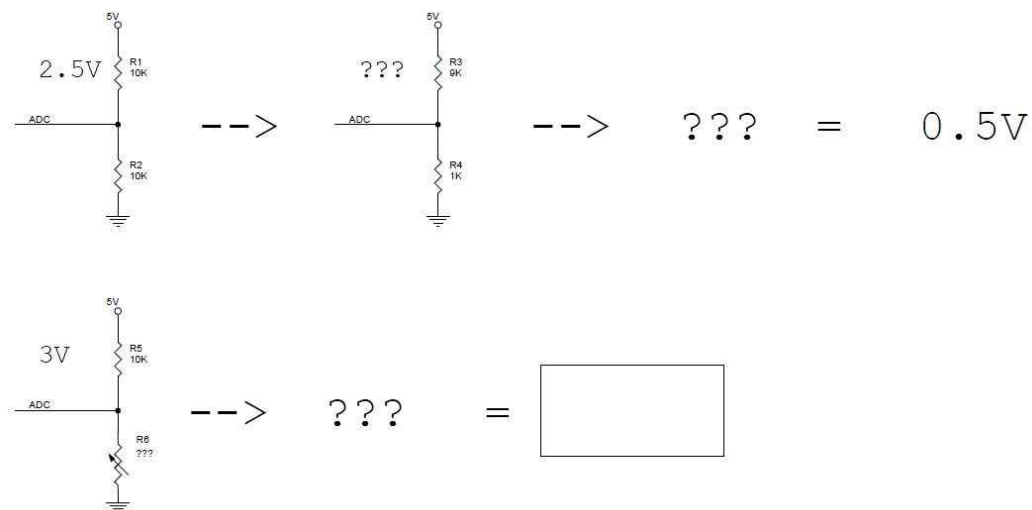
위에 회로의 써미스터는 뜨거워질수록 저항이 떨어지는 센서라고 가정하고 설명할게. 그럼 위에 회로도 써미스터 저항이 떨어질수록

ADC핀으로 들어가는 전압이 낮아지지? 아까전에 저항이 달라지는데 어떻게 온도가 달라지느냐는 얘기 기억해? 온도 센서는 저항마다 온도 값이 정해져 있어. 예를 들면 아래 예시처럼. 물론 센서마다 이 값들은 달라져.

Temp (°C)	Temp (°F)	Resistance Ratio Rt/R@+25°C	TC (%/°C)	10,000 Ω
				Ω
-80	-112.0	738.900000		7,389,000.00
-79	-110.2	675.753000	-8.91	6,757,530.00
-78	-108.4	618.446300	-8.84	6,184,463.00
-77	-106.6	566.401300	-8.77	5,664,013.00
-76	-104.8	519.099000	-8.70	5,190,990.00
-75	-103.0	476.076300	-8.63	4,760,763.00
-74	-101.2	436.917300	-8.56	4,369,173.00
-73	-99.4	401.250000	-8.50	4,012,500.00
-72	-97.6	368.739700	-8.43	3,687,397.00
-71	-95.8	339.086300	-8.36	3,390,863.00
-70	-94.0	312.020700	-8.30	3,120,207.00
-69	-92.2	287.299700	-8.23	2,872,997.00
-68	-90.4	264.705300	-8.17	2,647,053.00
-67	-88.6	244.040700	-8.11	2,440,407.00
-66	-86.8	225.129000	-8.05	2,251,290.00
-65	-85.0	207.810000	-7.99	2,078,100.00
-64	-83.2	191.939700	-7.93	1,919,397.00
-63	-81.4	177.387300	-7.87	1,773,873.00
-62	-79.6	164.035700	-7.81	1,640,357.00
-61	-77.8	151.777700	-7.75	1,517,777.00
-60	-76.0	140.517000	-7.69	1,405,170.00
-59	-74.2	130.166300	-7.63	1,301,663.00
-58	-72.4	120.646300	-7.58	1,206,463.00

그림5-1.저항 값마다의 온도가 정리된 표

저항마다 온도가 정해져 있다는 게 이해가 됐지?



위 그림처럼 저항값을 알면 전압을 알 수 있듯이 전압을 알고 있으니 저항값을 구할 수 있고 그 저항값이 곧 온도 센서의 온도가 되는 거야.

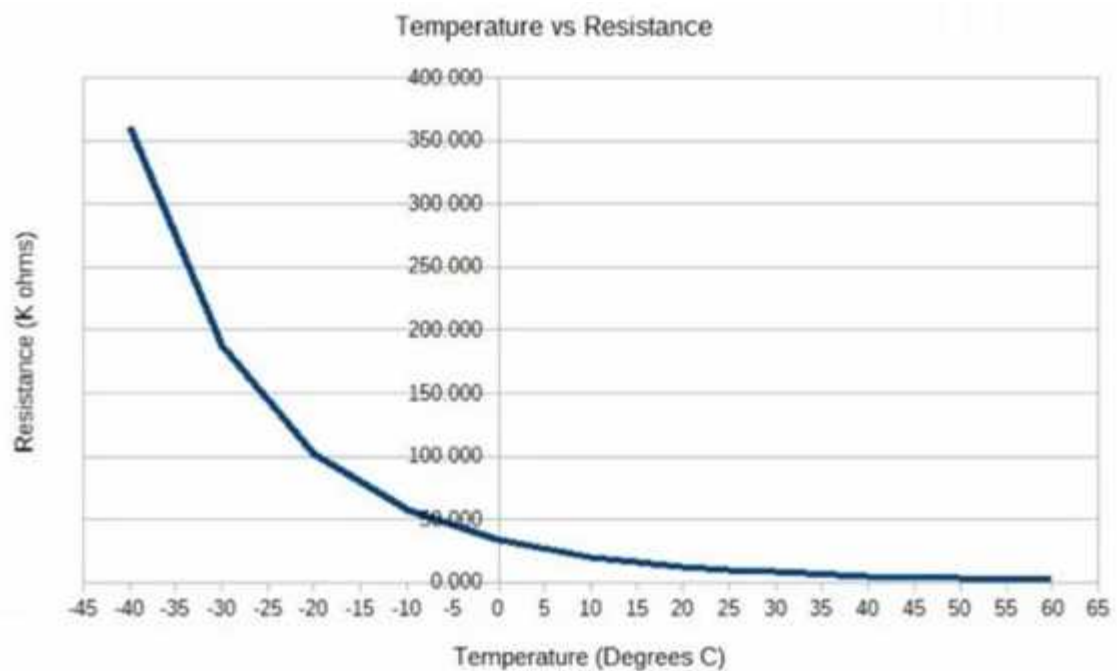


그림5-2.저항 값에 따른 온도

자, 이제 실제로 값을 구해볼 건데, 계산해야 한다고 생각하니 머리 아프지? 걱정하지 마. 실제로 변화율 계산은 계산기가 따로 있기도 하고 DataSheet에 변화율이 쓰인 소자도 있으니까 그렇게 어렵지는 않을 거야. 이 변화율을 구하는 공식은 슈타인 하트와 베타값으로 나누어져.

슈타인 하트는 NTC의 온도 저항 그래프에서 3개의 점을 찍고 그 안에 있는 곡선을 직선으로 펴주는 공식이야. 베타 공식은 두 개의 점을 찍고 그사이의 곡선을 직선으로 바꿔 주는 거야.

```
#define BETA_A 0.0011275 //HART CALCULATOR A
#define BETA_B 0.00023441 //HART CALCULATOR B
#define BETA_C 0.000000086482 //HART CALCULATOR C
#define ADC_REG 10 //ARD REG VALUE(KOHM)

//생략

THERMISTOR_R=(ADC_REG*1000000)/(((10230000/temp_adc)-10000)*10;
TEMP = ((1/(BETA_A + (BETA_B*log(THERMISTOR_R)) + BETA_C *
(log(THERMISTOR_R) * log (THERMISTOR_R) * log
```

(THERMISTOR_R))))-273.15)*10;

NTC103F(슈타인 하트 공식 이용)

위에는 저항값과 온도를 구했던 계산법이야. 써미스터에 따라 공식은 달라질 수도 있으니 잘 이해해야 해.

여기까지 이해가 충분히 되었다면 위에 회로를 만능기판에 구성해서 ADC 값의 변화를 FND를 통해 볼 거야. ADC 값을 읽는 코드를 활용해서 FND에 ADC값의 변화를 확인해봐. 다음으로는 저항값을 계산해서 동작시켰을 때 차트와 비교해보고 맞는다면 마지막으로 온도를 계산하면 돼. 주의해야 할 점은 저항값하고 온도 값을 한 번에 계산하면 안 돼! 안 그러면 온도를 잘못 계산했는지 저항을 잘못 계

5.4 프로젝트를 해보자 : 전자온도계

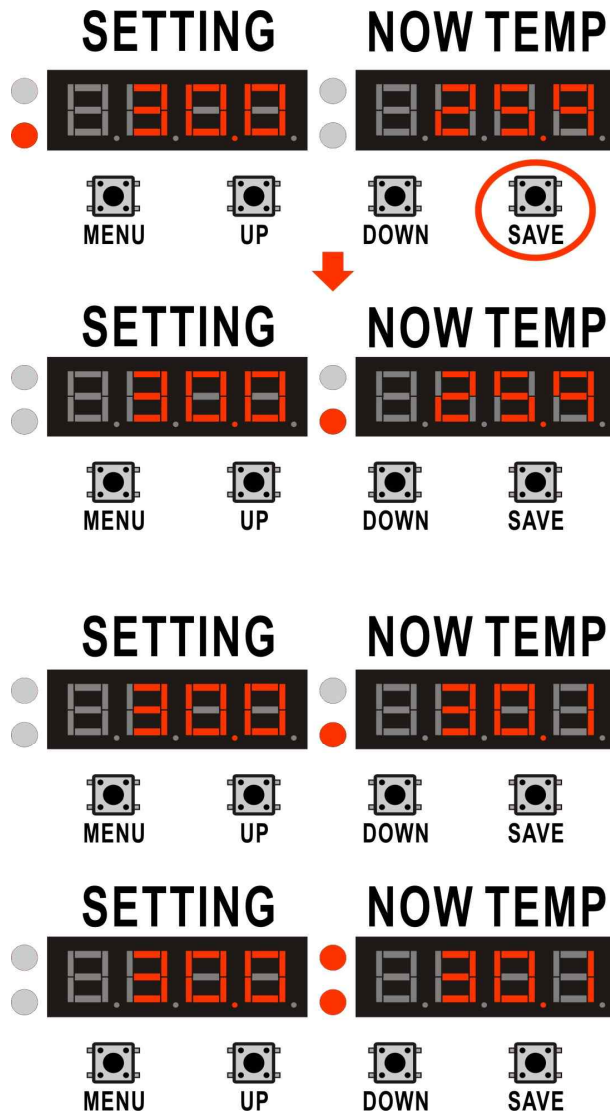
위여까지 완성했다면 이제 응용해보자!

응용1

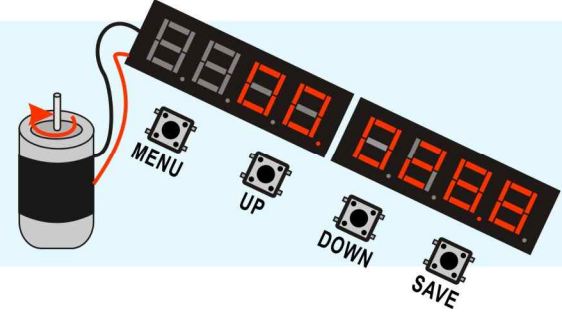


왼쪽 FND에는 현재 온도를 표시하고 1분마다 그 값들의 평균을 구해서 오른쪽 FND에 평균 온도를 표시해줘.

응용2



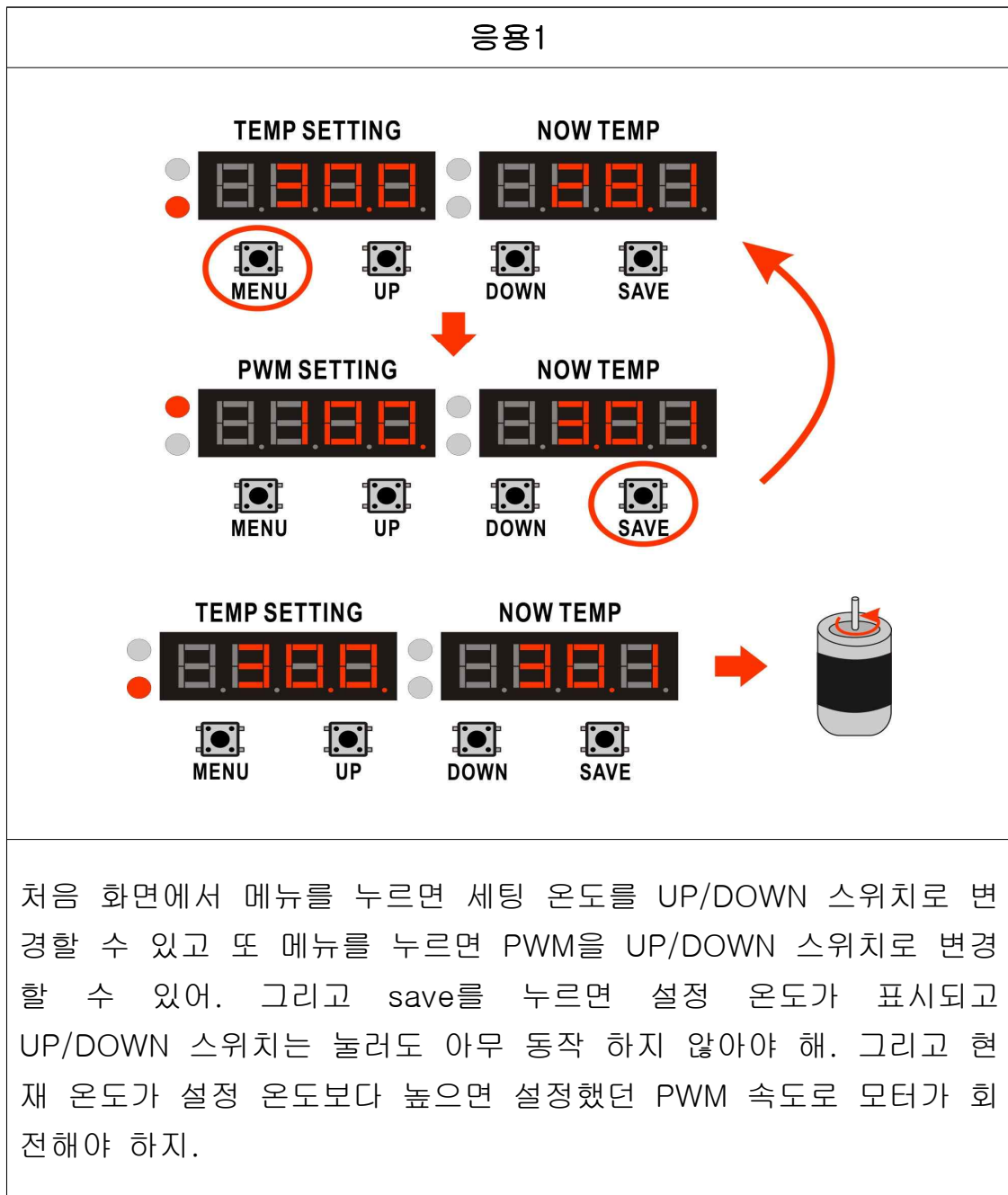
메뉴를 누르면 설정 온도를 정할 수 있어. 설정 온도는 UP/DOWN 스위치를 이용하면 되는거고 다 했다면 SAVE를 눌러 셋팅값을 저장하면 되. 이때 메뉴를 누른건지 저장이 되어서 동작중인지 LED로 표시해줘. 또, 현재 온도와 설정 온도를 비교해서 설정온도보다 높다면 LED가 켜줘.

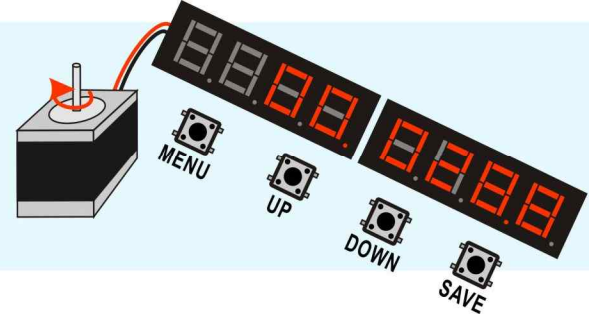


6.1 온도감지 후드

온도감지 후드는 위에서 만든 온도계와 DC모터 제어기를 이용해 만들 거야. 즉, ADC와 PWM을 사용하는 거지.

6.2 프로젝트를 해보자 : 온도감지 후드





7.1 온도감지 밸브

온도감지 후드는 위에서 만든 온도계와 스텝 모터 제어기를 이용해 만들 거야. 스텝모터의 각을 잘 조절하면 밸브처럼 사용할 수 있겠지?

7.2 프로젝트를 해보자 : 온도감지 밸브

