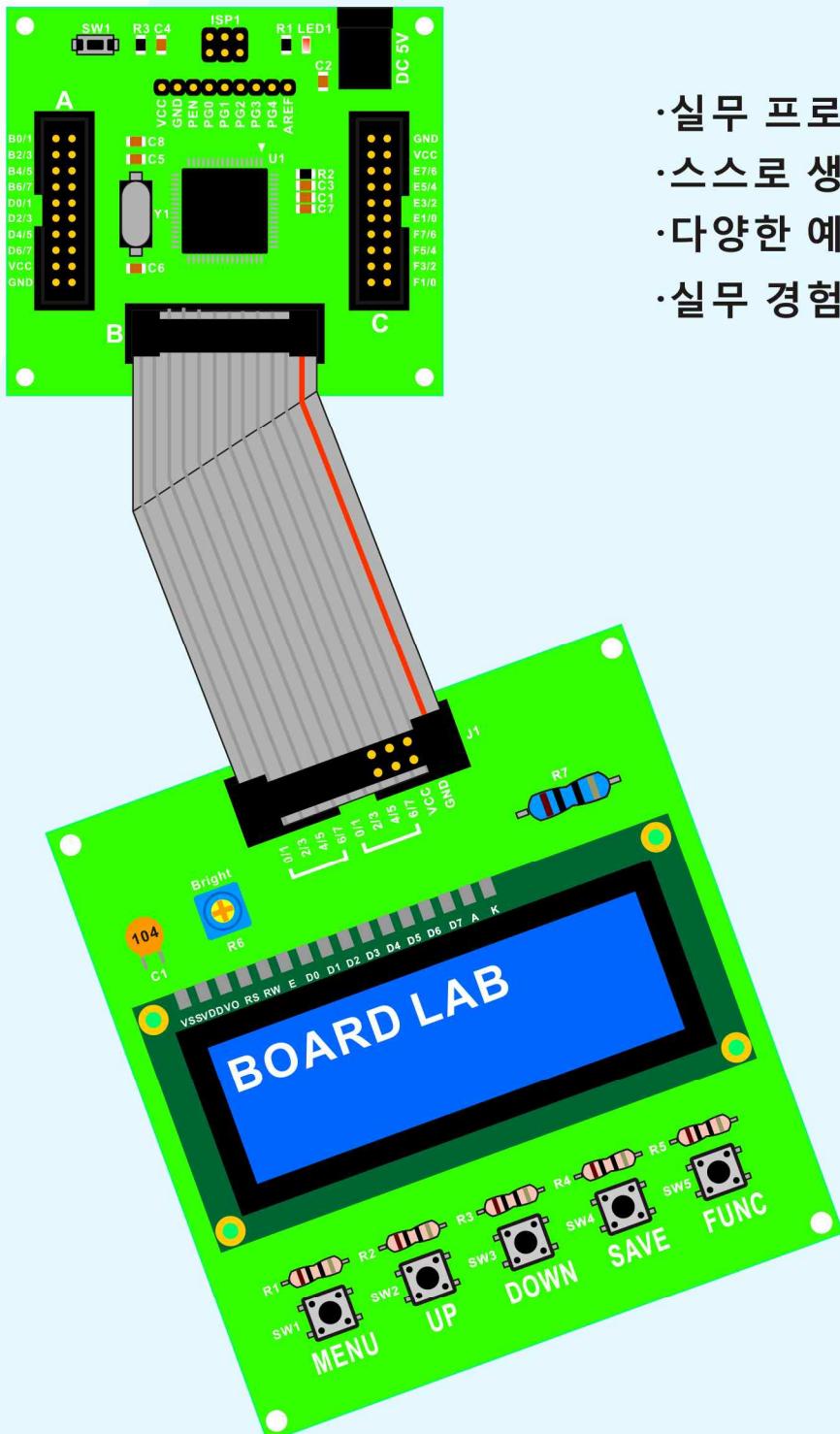


선배가 취업나와 보니까 꼭 필요하더라

# 전자분야 취준생을 위한 실무 프로젝트



- 실무 프로젝트를 활용한 교육
- 스스로 생각하는 힘 형성
- 다양한 예제를 통한 실력 향상
- 실무 경험 꿀팁 전달

[www.BoardFree.kr](http://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 벨브

## 04. LCD

기본설명, 예제

프로젝트 1 : 전자시계

프로젝트 2 : 모터제어기

프로젝트 3 : 스텝모터제어기

프로젝트 4 : 전자온도계

프로젝트 5 : 온도감지 후드

프로젝트 6 : 온도감지 벨브

## 05. 만능기판 사용법

기본설명, 예제

제어 루프 프로그램

Buck타입을 이용한 전압제어

Boost타입을 이용한 전압제어

## 06. 자료 정리

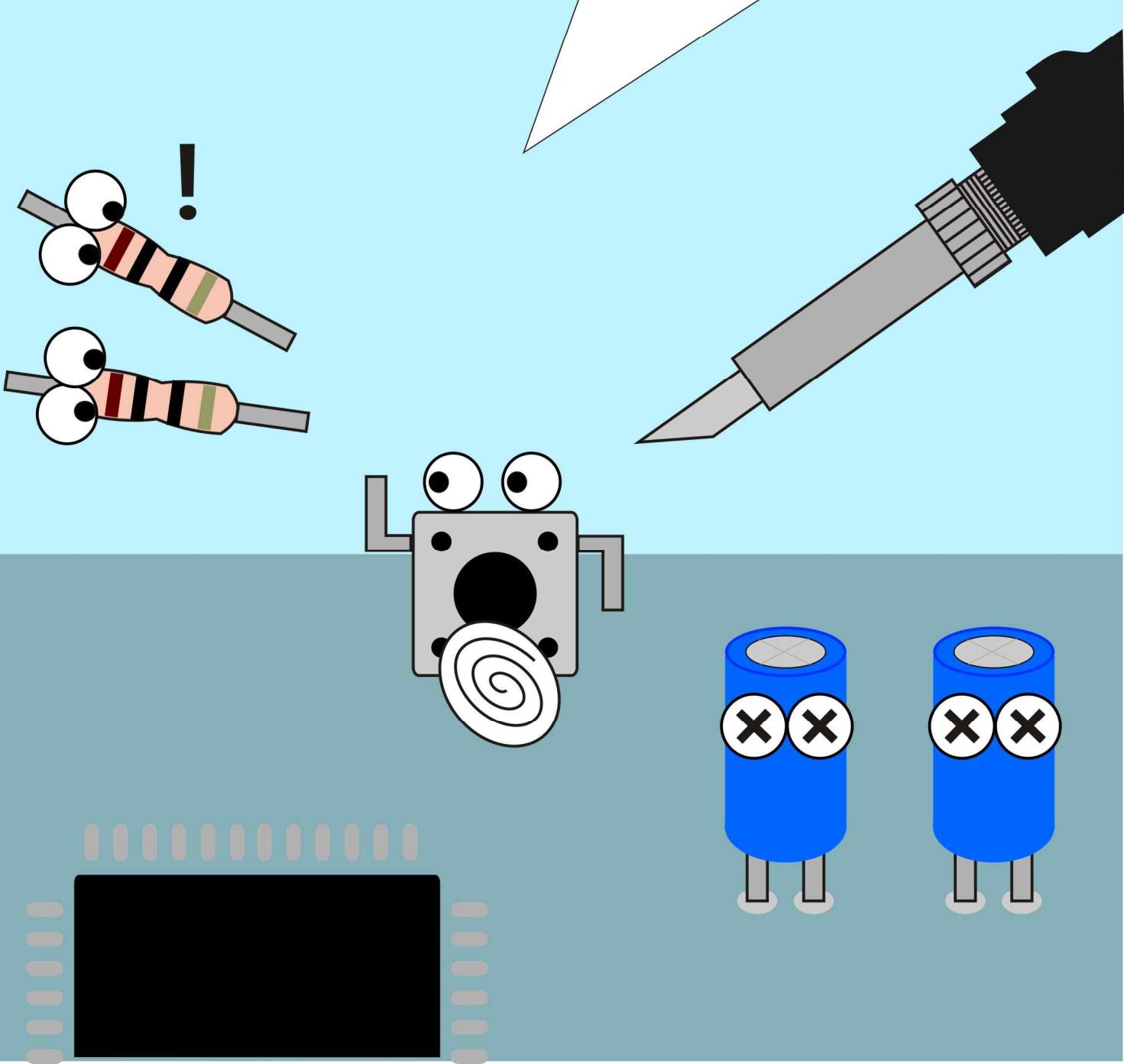
출처모음

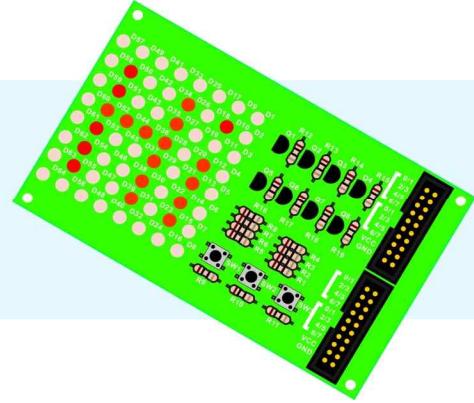
## 02. LED MATRIX

기본설명 및 예제

프로젝트 1: 미니 전광판

프로젝트 2: 통신 전광판





## 1.1 LED Matrix가 뭘까?

이름 그대로를 한국말로 하면 LED 행렬이야. 행렬이 뭔지는 알지? LED Matrix는 LED를 8X8, 16X16 등으로 배열하여 문자나 패턴을 표현할 수 있도록 하는 장치야. 상상하기 어렵다면 쉬운 예로 전광판을 생각하면 될 것 같아. 이런 LED Matrix는 다른 이름으로 DOT MATRIX라고 부르기도 해. 아래 사진과 같은 LED Matrix 빛이 나오는 곳이 점(DOT)과 같기 때문이지.

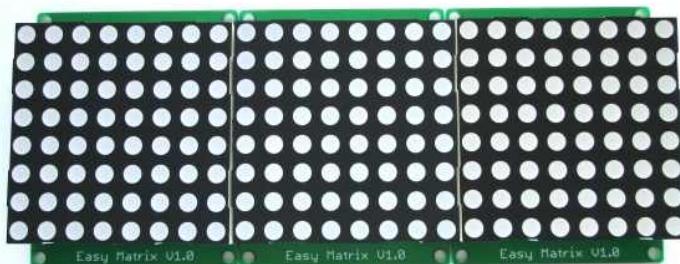


그림 1-1. 8 X 8 LED MATRIX 3개

MATRIX는 한국말로 행렬이야. 행렬이 뭔지 알지? 가로줄을 행이라고 하고 세로줄을 열이라고 해. 즉, LED 행렬인 거지. 행렬이 이해가 가지 않는다면 아래 그림을 보면 이해하기가 쉬울 거야.

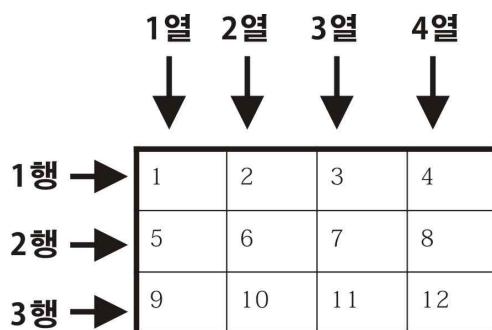


그림 1-2. 행렬

이런 LED MATRIX는 RGB LED로 해서 색깔을 바꾸기도 하고 8X8을 두 개를 나란히 두어서 8X16 으로 길게 하는 등 표현을 하는 방법이 다양해.

## 1.2 회로도

1단원에서 MAIN BOARD에 대해 설명했던 것과 마찬가지로 이번에도 회로도를 설명해줄 거야. 그 전에 너희가 먼저 이해를 해둬야 할 부분이 있어. 우리가 한 단원에 첫 부분에 앞으로 계속 회로도에 대해 설명을 먼저 할 거야. 근데 이런 의문을 가질 수도 있어. ‘왜 회로도를 알아야 해? 소프트웨어를 배우는 게 아닌가?’라고.

그런데 하드웨어 대한 이해도가 떨어진다면 1시간도 안 돼서 끝낼 코드를 아마 2시간, 3시간.. 더 오래 걸리게 될 수도 있어. 왜냐고? 코드를 어떻게 생각하는 도중에 하드웨어가 어땠더라? 1을 줘야 했던가? 0을 줘야 했던가? 등등. 쉬운 예시로 너희가 곱셈을 해야 해. 근데 곱하기가 뭔지 모르고 곱셈을 하고 있는 거야. 이해가 갔지? 그래서 너희에게 하드웨어에 대한 이해가 100%되었을 때 해야 한다고 하는 거야.

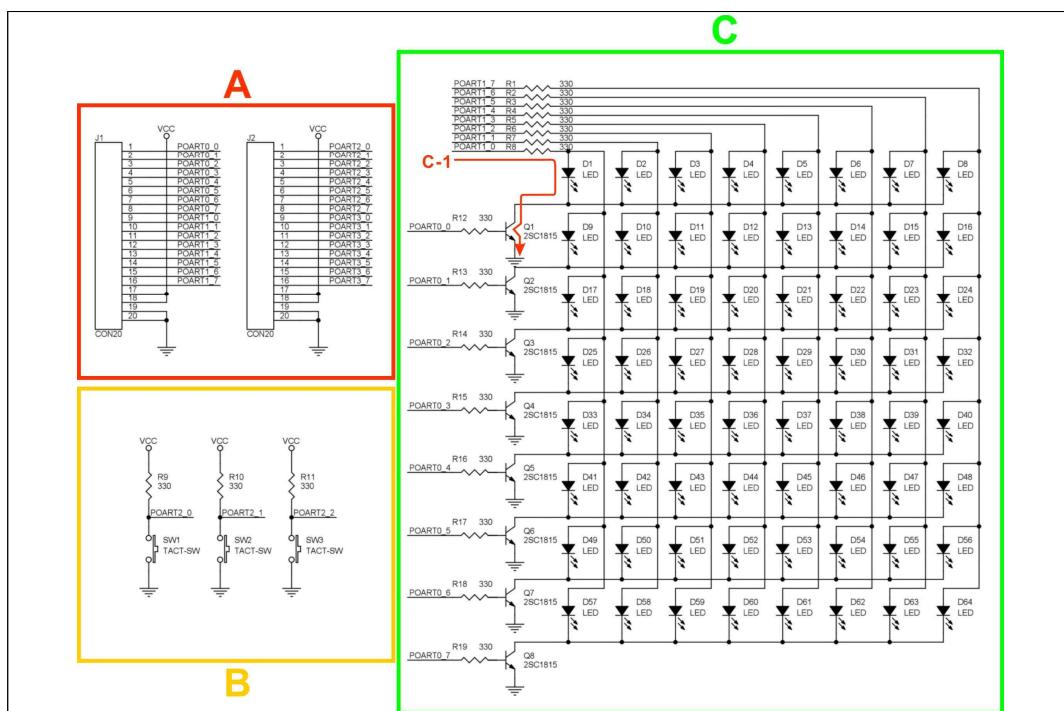


그림 1-3. LED MATRIX 회로도

## A. 박스헤더

박스 헤더는 전에도 설명해서 어떤 역할을 하는지는 너희가 충분히 알 거라고 생각해. 그런데 지금 박스 헤더 2개를 보면 하나는 PORT0,1이 연결되어 있고 또 하나는 PORT2,3이 연결되어 있어. 자세히 살펴보면 PORT0,1 박스헤더는 LED MATRIX와 연결되어있고 PORT2,3 박스헤더는 SW가 연결되어 있어. 정리하자면 LED MATRIX만 쓰고 싶다면 그에 해당하는 박스헤더 하나만 연결하면 되고 SW만 쓰고 싶다면 그 박스헤더 하나만 연결된다는 말이야.

## B. Switch

먼저, 너희 Pull-up, Pull-down이라는 말을 들어봤니? 배우지 않았을 수도 있고 혹은 제대로 이해하지 못하는 친구들을 위해 알려줄게.

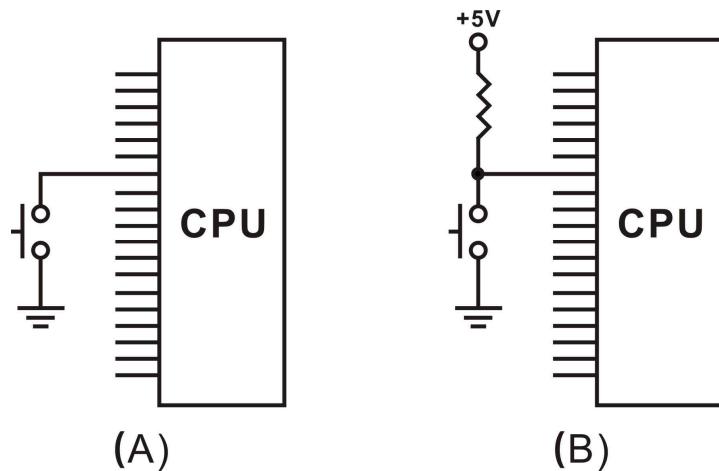


그림1-4. 풀업 저항의 유무

먼저 왼쪽 그림과 오른쪽 그림의 차이가 뭘까? 그냥 그림으로 차이를 생각해보면 +5V를 스위치에 연결해주느냐 안 해주느냐에 차이 정도로 생각이 들지? 그럼 (A),(B) 둘 다 스위치를 눌렀을 때의 상태를 생각해 보자. CPU로 들어가는 선이 GND와 도통 되지? 그럼 스위치를 누르지 않았을 때는 어떨까. (B)의 경우 CPU로 들어가는 값이 +5V야. 그럼 (A)는 어떨까. 이도 저도 아닌 상태지? 이걸 우리는 플로팅되었다고 해. 이런 경우 SW값을 CPU에서 읽어올 때 플로팅 상태이기 때문에 SW가 눌렸는지 안 눌렸는지를 구분하지 못해. 그래서 (A)보다 (B)를

많이 쓰는 거고.

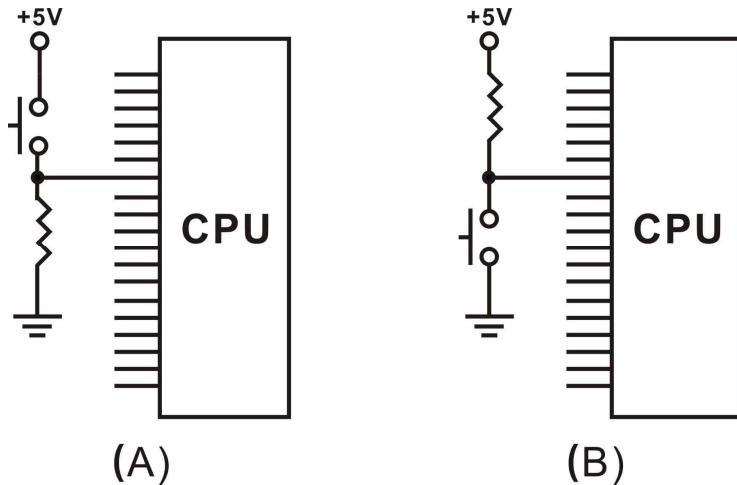


그림1-5. 풀다운과 풀업

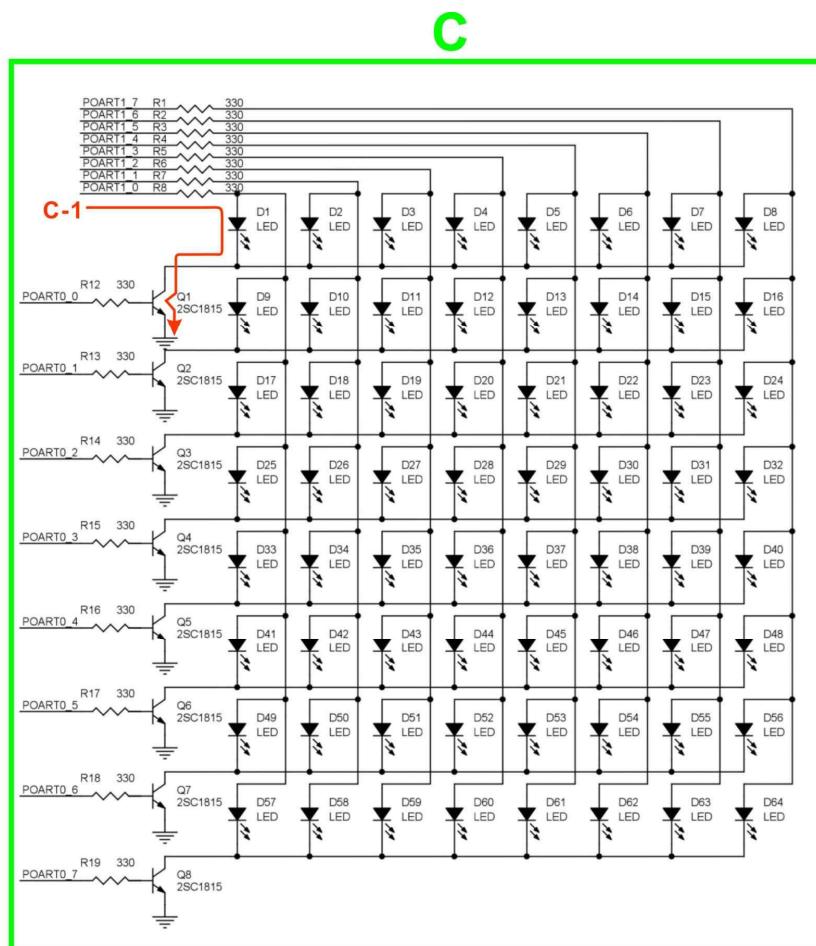
이번 그림에서 왼쪽 그림은 풀다운(pull\_down)이고 오른쪽은 풀업(pull-up)이야. 풀업과 풀다운의 차이는 저항과 SW의 위치 차이지? SW를 누르지 않았을 때 CPU로 읽어가는 값이 (A)는 0이고, (B)는 1이야. 반대로 SW를 눌렸을 때 CPU로 읽어가는 값은 (A)는 1이고, (B)는 0이지.

둘의 동작은 반대될 뿐 별 차이가 없어 보이지? 그리고 배울 때에도 상황에 맞춰서 자기가 원하는 데로 사용하면 된다고 말하고. 그런데 실제로 일하는 사람들은 풀업(pull\_up)을 많이 사용해. +5V는 사실 +4...몇 V가 될 수도 있고 잡음이 섞여서 전압이 흔들릴 수도 있지만 GND는 항상 변하지 않아. 그래서 평소에는 흔들려도 상관없는 전원이 CPU로 읽히다가 SW가 눌렸을 때는 변하지 않는 GND를 읽는 풀업을 사용하는 거야.

### C. LED MATRIX

LED MATRIX는 드라이버를 사용해서 제어하기도 하고 아니면 위에 회로도 있지? 그런 식으로 해서 사용해. 드라이버는 많은 LED를 한꺼번에 제어하려면 CPU 사용량이 많아지다 보니까 사람들이 더 편하게 제어하기 위해 만들어진 거라서 여러모로 장점을 많이 갖추고 있어.

예를 들면 통신으로 신호만 보내면 밝기를 제어할 수 있고, 색깔도 바꿀 수 있고, 깜빡일 수도 있는 등 기능이 많아. 이렇게 설명하면 드라이버를 쓰면 좋은 게 아니냐는 생각이 들지? 대신에 드라이버는 비싸. 또, PCB의 부피가 늘어난다는 단점이 있지. 또 우리는 LED를 많이 사용하지 않을 뿐더러 저런 칩까지 너희가 공부해야 하는 상황이 온다면 어려울 테니 그냥 드라이버가 이런 기능이 있고 우리가 왜 안 사용하는 정도만 알아두면 될 것 같아.



자, 그러면 우리가 알아야 할 제어 방식은 어떨까? 정말 쉬워 C-1 그림 있지? 저 방향으로 전류가 흘러야 해. 그래서 POART1\_0에 1을 줘야하고 PORT0\_0을 ON 해야 GND로 연결돼서 그 방향으로 전류가 흐르겠지. 다른 LED들도 같은 방식으로 제어하면 돼.

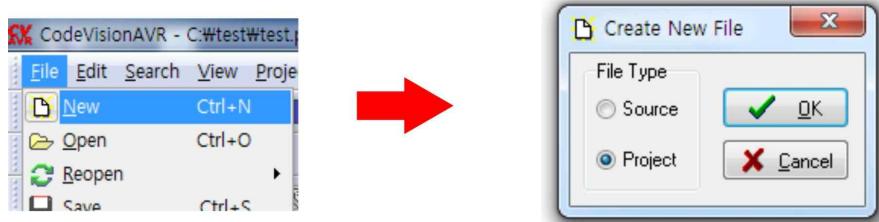
### 1.3 프로젝트를 만들기

하드웨어를 완벽하게 이해했다는 가정 하에 코딩을 해볼 거야. 그전에 먼저 관련 툴을 익혀야겠지?

1. CodeVisionAVR C Compiler Evaluation을 설치했다는 가정하에 설명해줄게.



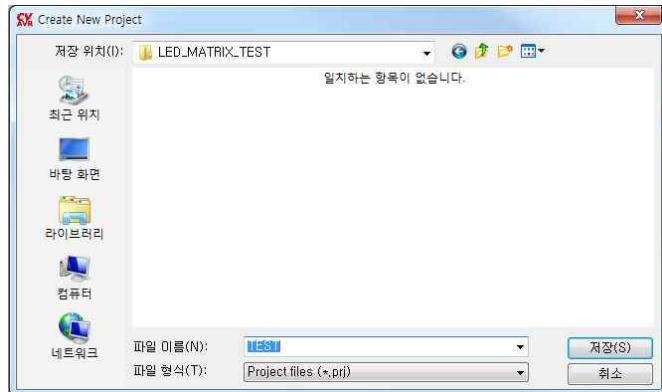
2. CodeVisionAVR C Compiler Evaluation에 들어가서 File-new를 눌러. 그럼 Create New File이라고 뜨지? 그중에 Project를 선택하고 OK.



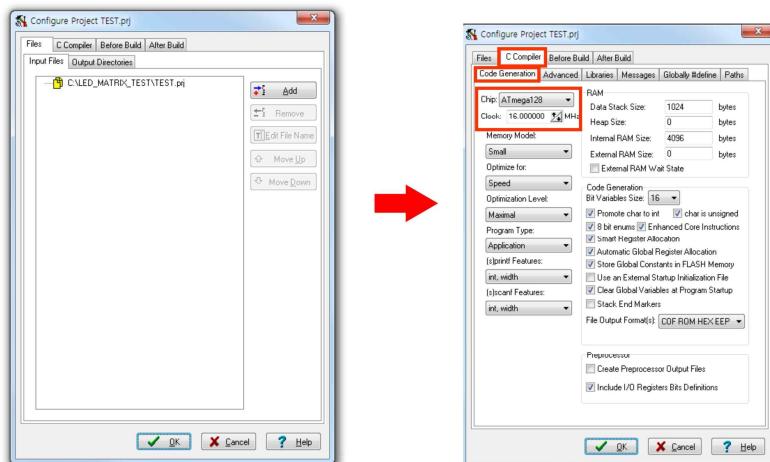
3. 그럼 아래와 같은 창이 뜨지? 아래에서 말하는 CodeWizardAVR은 쉽게 말해서 PORT 입, 출력 설정/PWM/ADC 등등 너희가 일일이 디코드를 작성하지 않고 설정만 해주면 코드가 자동으로 그에 맞춰서 코드를 만들어줘. 물론 섬세한 동작들은 너희가 작성해야 하고. 사실 그냥 데이터 시트 보고 설정하면 되는 거지만 너희에게 쉽고 빠르게 하는 방법을 알려주고자 하는 거니까. CodeWizardAVR은 뒤에 가서 알려줄 테니 No를 눌러.



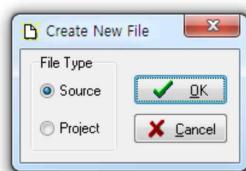
4. 그럼 아래와 같이 뜨지? 너희가 원하는 이름의 폴더와 파일 이름으로 하고 저장을 눌러.



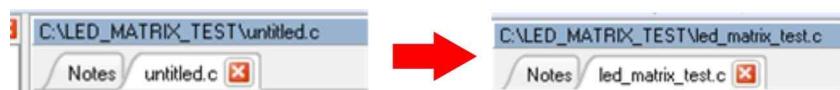
5. 그럼 왼쪽과 같은 창이 뜨지? 그럼 오른쪽 그림처럼 들어가서 Chip 이랑 Clock설정을 꼭 해줘야해.



6. 다했으면 OK누르고 다시 File-new-create New File에 들어가 아까는 Project를 선택했었잖아 이번엔 Source를 선택하고 OK를 눌러.

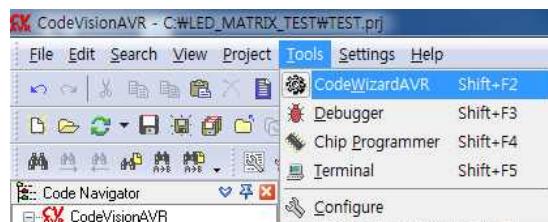


6. 그럼 아래와같이 파일 하나가 생긴게 보이지? 지금은 저장이 안되서 untitled.c라고 뜨는데 너희가 저장하면 이름을 바꿀 수 있어.



## 1.4 CodeWizardAVR 사용 방법

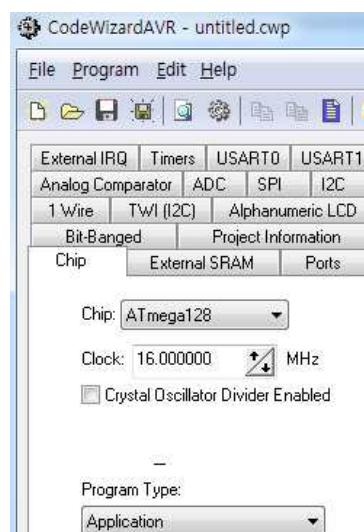
1. 이어서 알려줄게. Tools–CodeWizardAVR을 클릭.



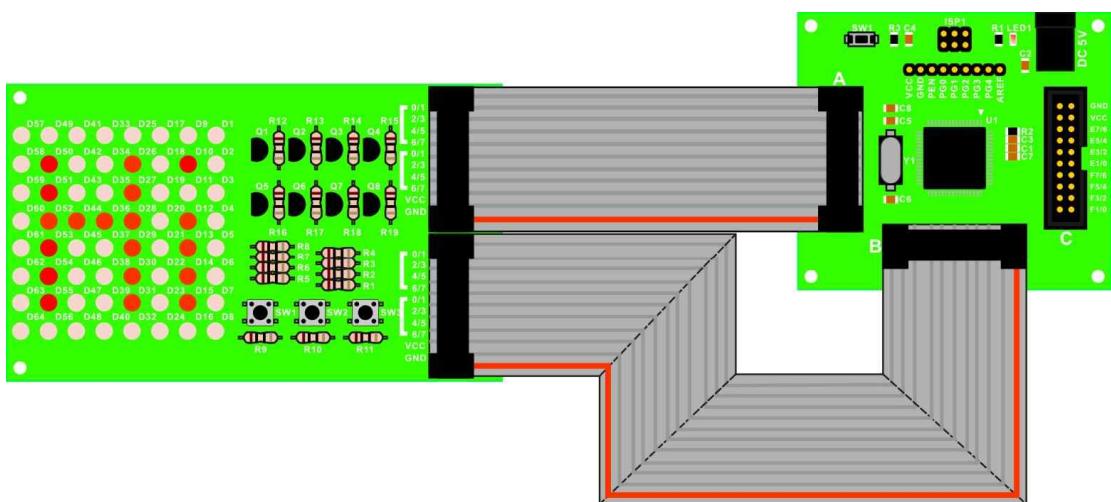
2. 그럼 이렇게 뜨지? 어떤 타입의 칩을 쓸 거냐인데 우리는 ATmega128을 하는 중이니까 AT90,ATtiny, ATmega, FPLSLIC를 선택하고 OK.



3. 그럼 아래와 같은 화면이 뜨지? Chip은 우리가 ATmega128을 사용하니까 그걸로 설정하면 되고 Clock는 상황에 따라 너희가 16Mhz를 쓰면 그걸로 맞춰주고 하면 되. 나는 16Mhz로 설정할거야.



6. 그다음은 가장 기본인 Ports 설정을 해볼 건데. 간단하게 말하자면 이 PORT1에 1번 핀이 사용되는 곳에 따라 입력이나 출력이냐를 결정 해주는 거야. 예를 들어 PORT1에 1번 핀이 LED를 ON/OFF하는 곳에 쓸 거야. 그럼 CPU 입장에서 출력을 내보내서 LED를 ON/OFF 하는 거지? 그럼 OUTPUT. 또는 PORT1에 1번 핀으로 SW 값을 읽어올 거야. CPU 입장에서 값을 읽어오는 거잖아? 그럼 입력, INPUT인 거지. 이것처럼 LED MATRIX의 PORT를 설정을 해볼 건데, 나는 아래 그림처럼 연결하였다는 가정하에 설명할 거야. 내 설명을 이해했다면 너희가 하고 싶은 곳에 연결해도 무관해.



이렇게 연결하였다고 했을 때 LED MATRIX 쪽은 PORTB와 PORTD에 연결되고, SW쪽은 PORTA와 PORTC에 연결돼. LED MATRIX 쪽은 CPU에서 출력을 내보내서 LED를 ON/OFF 하는 거니까 OUTPUT이겠지. SW쪽은 입력을 받아오는 거니까 INPUT 일 거고. 그거에 맞춰서 PORT 설정을 해주면 돼.

Port A	Port B	Port C	Port D	Port E
Data Direction	Pullup/Output Value			
Bit 0 Out	0 Bit 0			
Bit 1 Out	0 Bit 1			
Bit 2 Out	0 Bit 2			
Bit 3 Out	0 Bit 3			
Bit 4 Out	0 Bit 4			
Bit 5 Out	0 Bit 5			
Bit 6 Out	0 Bit 6			
Bit 7 Out	0 Bit 7			

Port A	Port B	Port C	Port D	Port E
Data Direction	Pullup/Output Value			
Bit 0 Out	0 Bit 0			
Bit 1 Out	0 Bit 1			
Bit 2 Out	0 Bit 2			
Bit 3 Out	0 Bit 3			
Bit 4 Out	0 Bit 4			
Bit 5 Out	0 Bit 5			
Bit 6 Out	0 Bit 6			
Bit 7 Out	0 Bit 7			

Port A	Port B	Port C	Port D	Pc	◀	▶
Data Direction		Pullup/Output Value				
Bit 0	In	T	Bit 0			
Bit 1	In	T	Bit 1			
Bit 2	In	T	Bit 2			
Bit 3	In	T	Bit 3			
Bit 4	In	T	Bit 4			
Bit 5	In	T	Bit 5			
Bit 6	In	T	Bit 6			
Bit 7	In	T	Bit 7			

Port A	Port B	Port C	Port D	Pc	◀	▶
Data Direction		Pullup/Output Value				
Bit 0	In	T	Bit 0			
Bit 1	In	T	Bit 1			
Bit 2	In	T	Bit 2			
Bit 3	In	T	Bit 3			
Bit 4	In	T	Bit 4			
Bit 5	In	T	Bit 5			
Bit 6	In	T	Bit 6			
Bit 7	In	T	Bit 7			

7. Ports 설정은 다 했지? 원래 기능에 따라 아래 그림에 나와 있는 것처럼 USART, Timer, ADC 등 더 설정하기도 해. 지금 우리는 아직 사용하지는 않으니까 Ports 설정만 하는 거야.

External IRQ	Timers	USART0	USART1
Analog Comparator	ADC	SPI	I2C
1 Wire	TwI (I2C)	Alphanumeric LCD	
Bit-Banged		Project Information	
Chip		Ports	

8. 상단에 보면  이런 모양이 있을 거야. 이 모양을 누르면 너희가 설정한 거대로의 코드가 보일 거야.

```

Program Preview
1 #include <mega128.h>
2
3 // Declare your global variables here
4
5 void main(void)
6 {
7     // Declare your local variables here
8
9     // Input/Output Ports initialization
10    // Port A initialization
11    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
12    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
13    PORTA=0x00;
14    DDRA=0x00;
15
16    // Port B initialization
17    // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
18    // State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
19    PORTB=0x00;
20    DDRB=0xFF;
21
22    // Port C initialization
23    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
24    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
25    PORTC=0x00;
26    DDRC=0x00;
27
28    // Port D initialization
29    // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
30    // State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
31    PORTD=0x00;
32    DDRD=0xFF;
33
34    // Port E initialization
35    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
36    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T

```

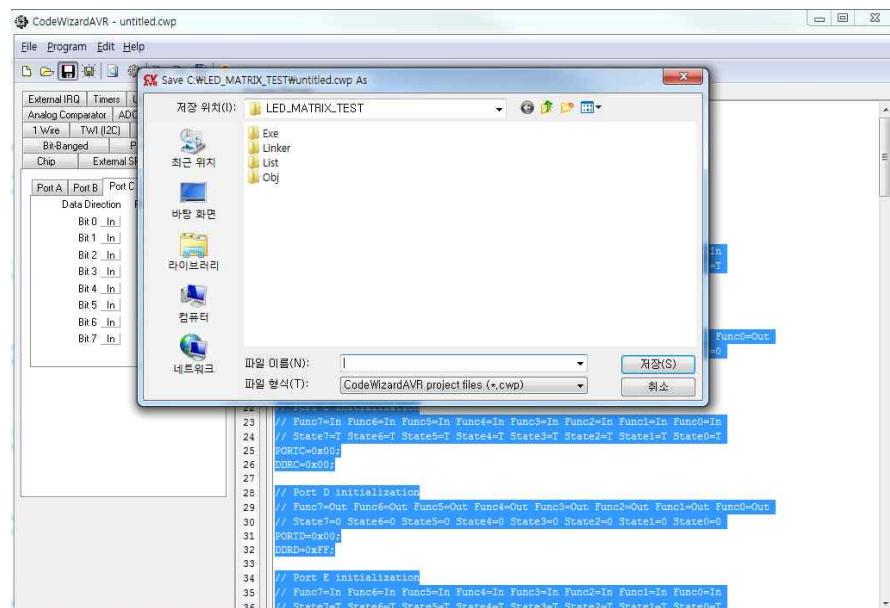
9. 위에 코드를 다 복사한 후 아까 만들었던 ‘너희가 설정한 이름.c’ 파일에 붙여넣어.

```

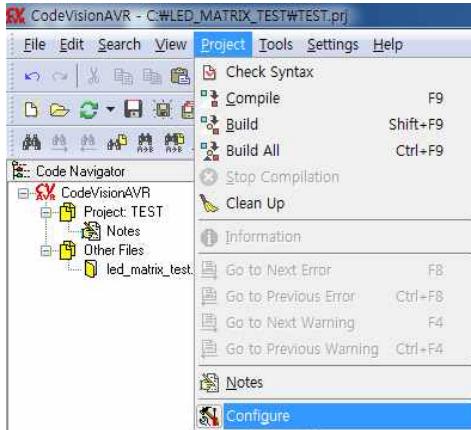
C:\LED_MATRIX_TEST\led_matrix_test.c
Notes led_matrix_test.c
1 #include <mega128.h>
2
3 // Declare your global variables here
4
5 void main(void)
6 {
7     // Declare your local variables here
8
9     // Input/Output Ports initialization
10    // Port A initialization
11    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
12    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
13    PORTA=0x00;
14    DDRA=0x00;
15
16    // Port B initialization
17    // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
18    // State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
19    PORTB=0x00;
20    DDRB=0xFF;
21
22    // Port C initialization
23    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
24    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
25    PORTC=0x00;
26    DDRC=0x00;
27
28    // Port D initialization
29    // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
30    // State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
31    PORTD=0x00;
32    DDRD=0xFF;
33
34    // Port E initialization
35    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
36    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
37    PORTE=0x00;
38    DDRE=0x00;
39
40    // Port F initialization
41    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
42    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
43    PORTF=0x00;
...

```

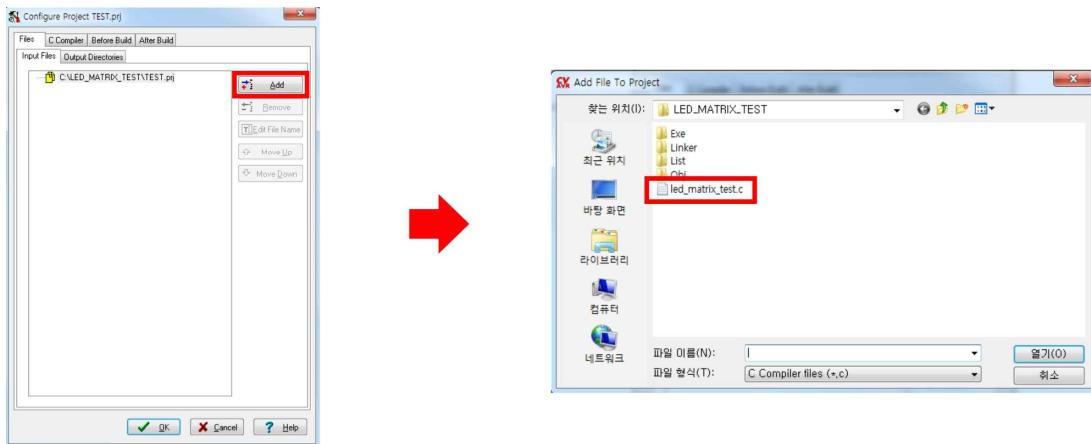
10. 여기까지 따라 했다면 일단 기본 설정은 다 한 거야. 그리고 아까 CodeWizardAVR있지? 그건 저장해두면 나중에 설정 수정해야 할 때 사용하면 처음부터 다시 설정하지 않아도 되니까 저장해둬.



11. 파일을 저장했으면 Project-configure에 들어가.



12. ADD를 누르면 너희가 저장해서 만들어진 .c 파일이 있을 거야.  
이걸 열어주면 돼.

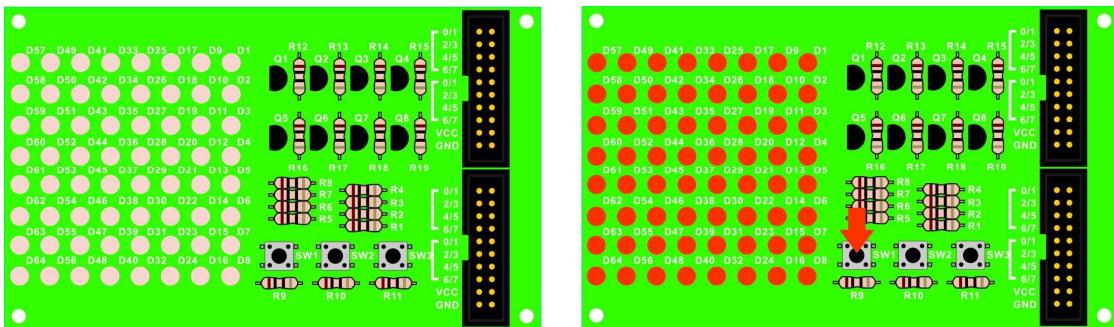


## 1.5 동작 해보기

1.3~1.4의 프로젝트를 만들고 Wizard를 사용해서 기본 설정을 다 했다면 이제 보드가 정말 동작하는지 테스트를 해볼 거야. 물론 다 확인하고 받은 보드인데 오동작을 하겠어? 라는 생각을 가질 수도 있지만 실제로 잘못된 보드일 수도 있고 또 납땜을 잘못했을 수도 있고 변수가 많아서 확실하게 테스트를 한 번 해보는 게 좋아. 원가 불확실한 상태로 진행되면 내가 코드를 잘못한 것인지 다른 게 잘못된 것인지 모르잖아? 예시코드로 어떻게 테스트했는지 알려줄게. 꼭 이렇게 하라는 게 아니라 다른 방법으로 너희가 원하는 대로 상관없다는 걸 알아둬.

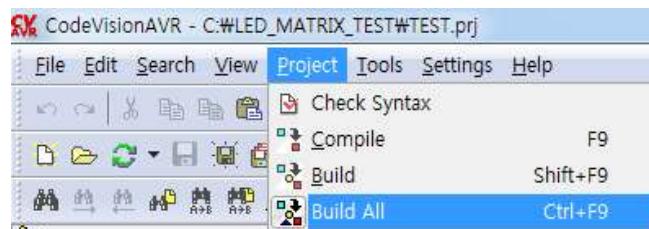
<pre> ///////////생략/////////// while (1) {     if(!PIN.C.0)     {         PORTD=0xff;         PORTB=0xff;     }     else     {         PORTD=0x00;         PORTB=0x00;     } } </pre>	<pre> #define SW0 PINC.0 #define COLUMN PORTB #define ROW PORTD ///////////생략/////////// while (1) {     if(!SW0)     {         ROW=0xff;         COLUMN=0xff;     }     else     {         ROW=0x00;         COLUMN=0x00;     } } </pre>
---	---

나는 while문 안에 이렇게 코드를 작성했어. 근데 동작이 어떤지 생각하기 전에 PORTB가 어떤 건지 싶어서 회로도를 보게 되지 않아? 그래서 말인데 너희 #define 사용해본 적 있지? 회사에 나와서 코딩을 하다 보면 다음 사람이 인수인계해야 하는 상황을 생각해서 쉽고 알아보기 편하게 코딩하는 걸 우선시해. 그래서 위에 왼쪽 코드를 오른쪽 코드처럼 알아보기 쉽게 해서 사용해.

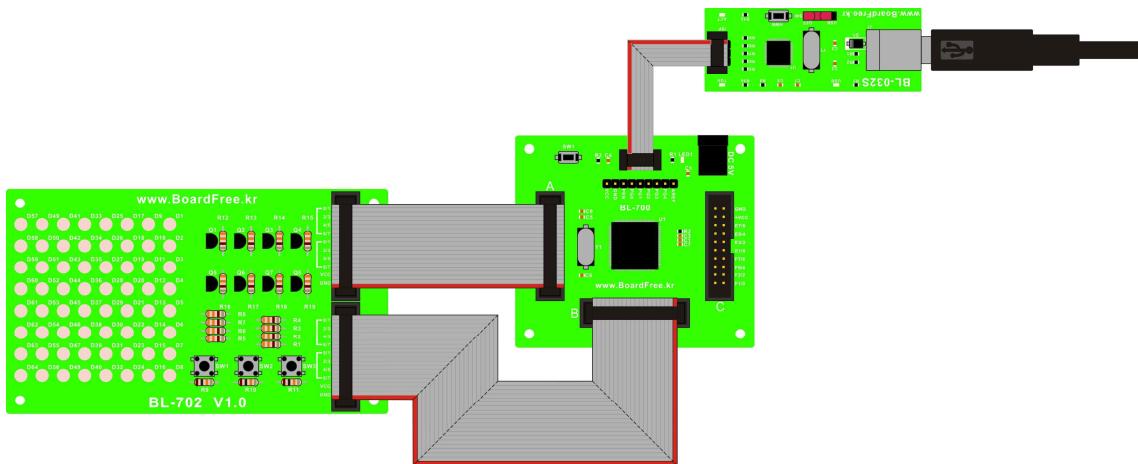


위에 코드의 동작은 SW를 누르면 LED가 다 켜지고 SW를 누르지 않으면 LED가 다 꺼지는 거야. 이렇게 하면 SW가 잘 동작하는지, LED가 동작하는지 다 알 수 있잖아?

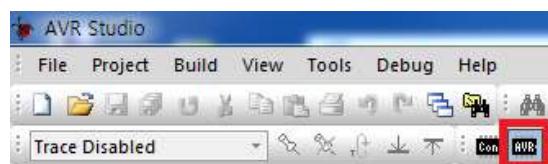
여기서 잠깐, 너희 프로그램 넣는 방법 잘 모르지? 일단 위에처럼 코드를 작성했다면 Project-Build All을 해. compile과 build, build all의 차이는 다들 알지?



1. 위에처럼 했다면 아래 그림과 같이 CPU 보드와 ISP를 연결해. 혹시나 해서 얘기하는 건데 CPU Board에 전원을 넣은 상태이면 ISP에 스위치를 OFF로 해줘야 해!!

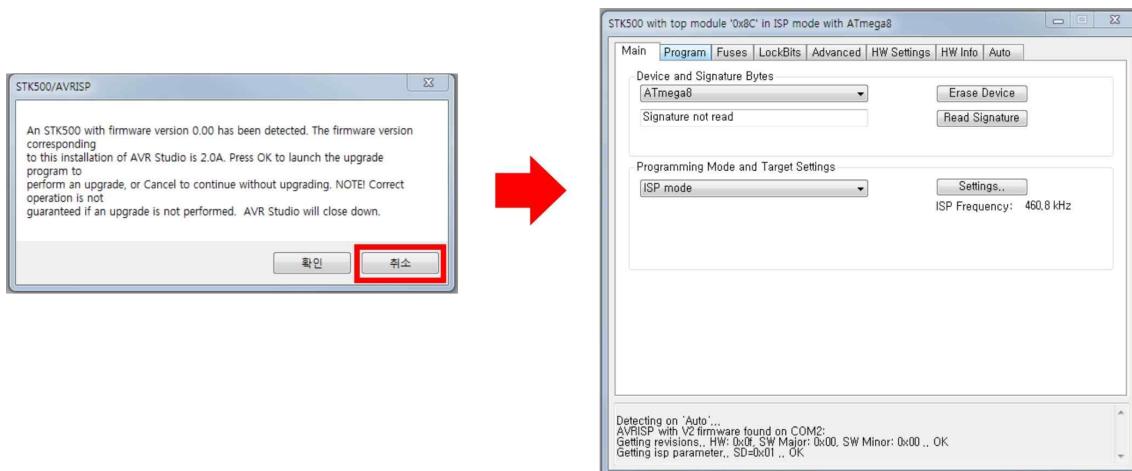


2. 다했어? 그럼 AVR Studio를 열어서 아래 AVR을 눌러.

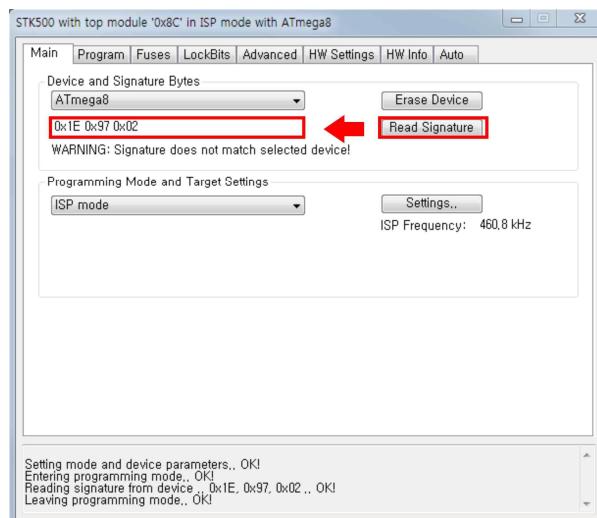


3. STK500을 사용하는 사람은 왼쪽처럼 뜰 텐데 그럼 취소를 눌러줘.

그럼 오른쪽 사진처럼 뜨지?

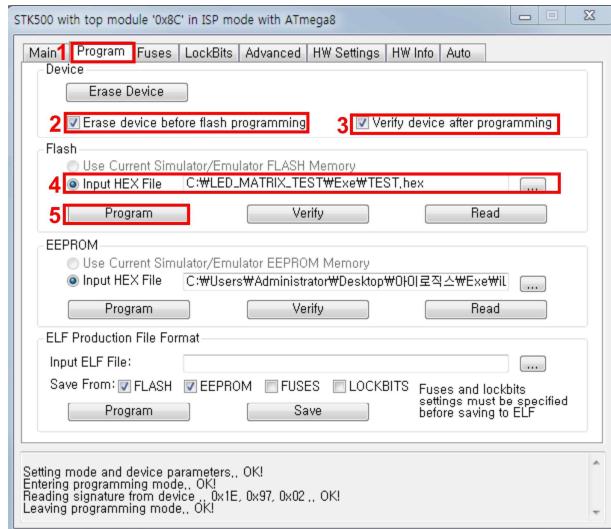


4. 다 했으면 ISP가 CHIP을 잘 읽어올 수 있는 상태인지 확인하기 위해 Read Signature를 눌러 그럼 왼쪽과 같이 칩에 대한 정보 값이 뜰텐데 만약 저렇게 뜨지 않았다면 ISP의 문제 혹은 Chip이 납땜이 제대로 되지 않은 상태 등등의 문제이니까 잘 살펴봐.



5. 아래 사진에 1번처럼 Program에 일단 들어가 봐. 2, 3번이 뭔가 싶지? 2번은 체크를 하면 매번 다운로딩 할 때마다 기존에 넣어둔 프로그램을 지운 후 다운로딩 하겠다는 거야. 3번은 체크를 하면 매번 다운로딩 할 때마다 chip이 제대로 읽히는 상태인지 확인한 후 다운로

딩 하겠다는 거고 이건 어디까지나 너희 선택이야. 4번은 이제 너희가 codevision에서 build all 했던 거 기억하지? 그 과정을 통해 생긴 hex 파일을 불러오면 돼. 그리고 5번을 눌러서 다운로딩을 하면 되는 거고. 쉽지?



위쪽에 동작을 해본 사람이면 알겠지만, LED가 어두워 보이지 않아? 이 차이는 아래처럼 하면 알 수 있어.

코드	<pre> while (1) {     if(SW0) {ROW=0x80; COLUMN=0x80;}     else {ROW=0x80; COLUMN=0xff;} } </pre>
동작	

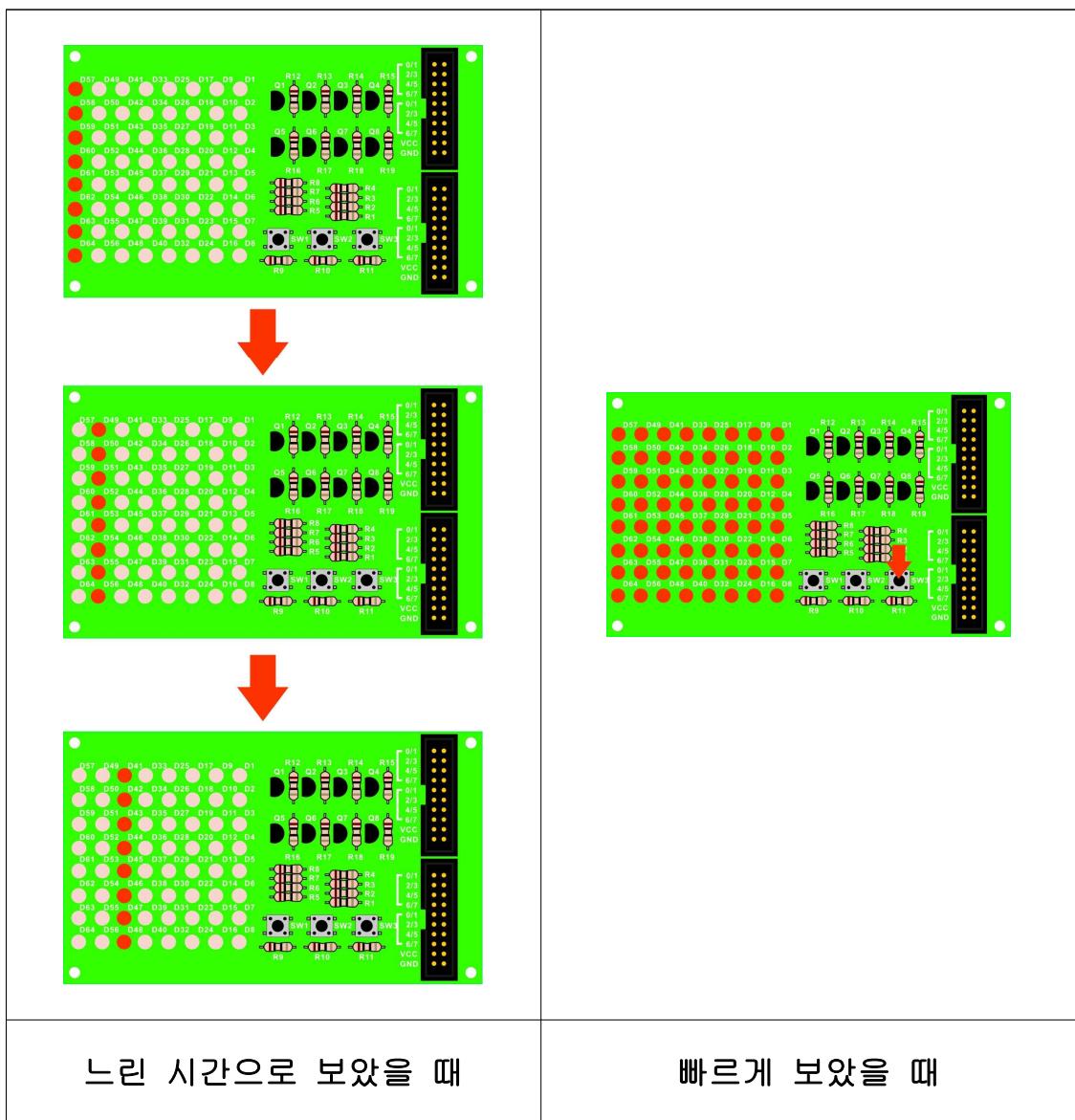
동작을 해보면 알겠지만 왼쪽은 밝고 오른쪽은 어두워. 이유가 뭘까? 아래 그림처럼 한번 해봐. 그럼 바로 알 수 있을 거야.

<p><b>코드</b></p> <pre> while (1) {     if(SW0) {ROW=0x80; COLUMN=0xff;}     else {ROW=0xff; COLUMN=0x80;} } </pre>	<p><b>동작</b></p>
--	------------------

무슨 차이인지 느꼈어?? 한 행의 LED를 다 킬 때 즉, 한 신호선에 둑여있는 LED들을 켜려고 하면 그렇게 되지 않아? CPU에서 나오는 전류에 비해 LED가 많아서 전류가 부족한 거지. 그래서 밝기가 어두운 거야.

반면 한 열에 1개는 LED마다 신호선이 연결되어 있어서 전류가 부족하지 않은 거지. 이해했지? 물론 그쪽에도 똑같이 트랜지스터를 단다 든가 하면 전류가 부족하지 않아! 실제로 그렇게 만들어서 파는 보드도 있어.

아무튼 이런 이유로 사용하는 제어방법이 바로 다이나믹 디스플레이야. 다이나믹 디스플레이는 쉽게 말해 우리의 눈을 속이는 거야. 느린 시간으로 보면 한 줄씩 켜지는 건데 빠른 시간으로 보면 전체가 ON인 걸로 보이는 거지. 다음 그림을 봐봐.



느린 시간으로 보았을 때

빠르게 보았을 때

그럼 다이나믹 디스플레이에 예를 하나 알려줄게.

### 코드

```

while (1)
{
    if(SW0) {ROW=0xff; COLUMN=0xff;}
    else
    {
        ROW=0xff; COLUMN=0x80;
        delay_ms(1);
        ROW=0xff; COLUMN=0x40;
        delay_ms(1);
    }
}

```

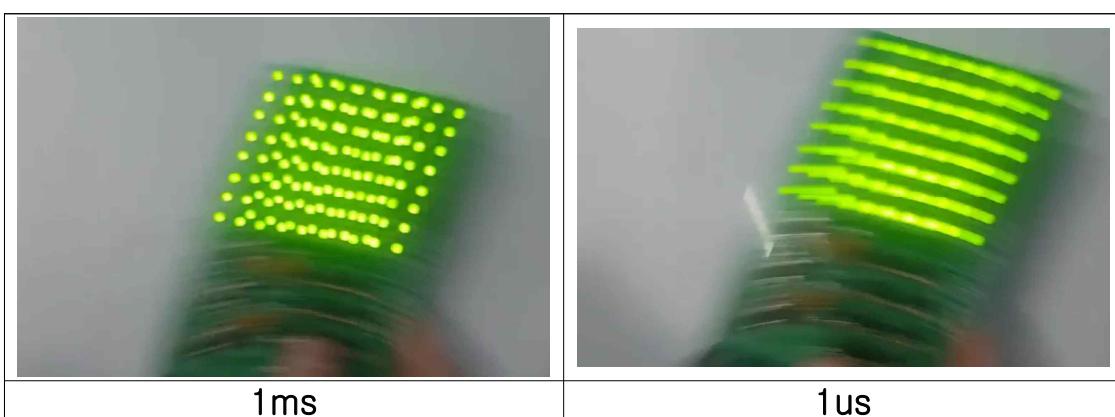
```

ROW=0xff; COLUMN=0x20;
delay_ms(1);
ROW=0xff; COLUMN=0x10;
delay_ms(1);
ROW=0xff; COLUMN=0x08;
delay_ms(1);
ROW=0xff; COLUMN=0x04;
delay_ms(1);
ROW=0xff; COLUMN=0x02;
delay_ms(1);
ROW=0xff; COLUMN=0x01;
delay_ms(1);
}
}

```

위에 코드는 다이나믹 디스플레이를 하느냐 안 하느냐의 차이야. 스위치를 누르지 않으면 어둡고 스위치를 누르면 밝지? 그럼 저 상태에서 delay 시간을 delay\_ms(10)로 해봐. 그럼 LED가 어디가 켜지는지 다 보이지? delay시간을 빠르게 할수록 우리 눈을 속여서 자연스럽게 다 켜진 걸로 보여서 그래.

그럼 delay\_us(1)로 해봐. delay\_ms(1)이랑 차이를 모르겠지? 그럼 1ms일 때랑 1 us 때랑 좌우로 막 흔들어봐. 그럼 확실히 차이를 알 수 있을 거야. 1ms일 때는 좌우로 흔들면 LED가 여기 있다 저기 있다 하는 거 같지 않아? 1 us 일 때는 선처럼 보이고. 사진을 보면 이해가 될 거야.



실제로 해보면 1ms처럼 제어해서 LED가 여기저기 있어 보이는 경우는 어지러울 거야. 그렇다고 1us 혹은 더 빠르게 무조건 제어하라는 게 아니라 너희가 알아두고 이런 문제가 있으니 필요한 경우에는 노력 을 해야 한다는 말이지.

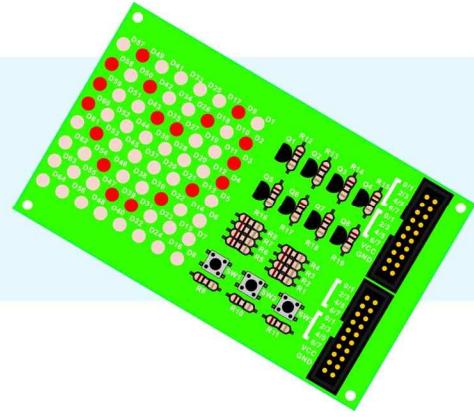
<pre>if(time_cnt&gt;=255) {     time_cnt=0;     if(state==7) state=0;     else state=state+1; } else time_cnt=time_cnt+1; if(state==0){ROW=0xff; COLUMN=0x80;} else if(state==1){ROW=0xff; COLUMN=0x40;} else if(state==2){ROW=0xff; COLUMN=0x20;} else if(state==3){ROW=0xff; COLUMN=0x10;} else if(state==4){ROW=0xff; COLUMN=0x08;} else if(state==5){ROW=0xff; COLUMN=0x04;} else if(state==6){ROW=0xff; COLUMN=0x02;} else if(state==7){ROW=0xff; COLUMN=0x01;}</pre>	<pre>ROW=0xff; COLUMN=0x80; delay_ms(1); ROW=0xff; COLUMN=0x40; delay_ms(1); ROW=0xff; COLUMN=0x20; delay_ms(1); ROW=0xff; COLUMN=0x10; delay_ms(1); ROW=0xff; COLUMN=0x08; delay_ms(1); ROW=0xff; COLUMN=0x04; delay_ms(1); ROW=0xff; COLUMN=0x02; delay_ms(1); ROW=0xff; COLUMN=0x01; delay_ms(1);</pre>
--	--

위에 코드는 둘 다 다이나믹 디스플레이 방식으로 동작하는 코드야. 물론 시간의 차이는 있지. 그렇지만 실제로 회사에선 오른쪽보다는 왼쪽의 방법을 사용해. 왜 그럴까? delay 함수는 그 자리에 정말 그 시간 동안 머물러있어. 그래서 delay\_ms(100)이라고 하면 100ms 동안 다음 줄로 안 넘어가고 기다리고 있는 거야.

반면 왼쪽 같은 경우에는 웃줄부터 맨 아랫줄까지 내려가는데 delay가 없어. 지금 우리는 led\_matrix만 사용하고 있어서 그렇지 다른 기능을 사용하게 된다면 한곳에 머무른 채로 있다는 건 다른 기능을 사용할 타이밍에 사용하지 않을 수도 있다는 거잖아? 물론 delay를 쓰지 말라는 게 아니라. 잘 생각해보고 사용해야 한다는 거지.

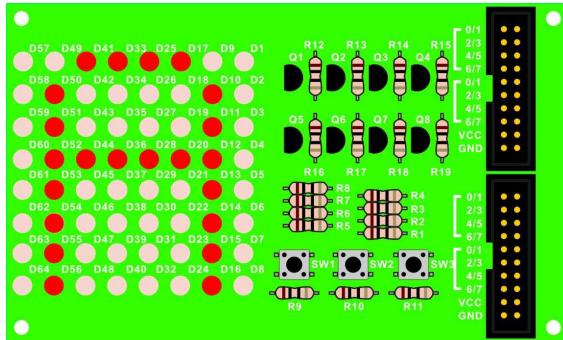
# 2

## 프로젝트 1: 미니 전광판



### 2.1 스위치에 따라 다른 글자 띄우기

지금까지는 그냥 LED를 다 켜보기만 했잖아? 스위치에 따라 다른 글자를 띄워볼 거야. 이전 소단원에서 스위치에 따라 LED 수를 다르게 키는 거랑 다이나믹 디스플레이를 해보았지? 이제 그 두 개를 합치면 되는 거야. 아래 그림처럼 먼저 해보자.



다음 코드는 위에 그림처럼 동작한 예시 코드야. 꼭 이렇게 할 필요 없이 너희가 좋을 대로 하면 돼.

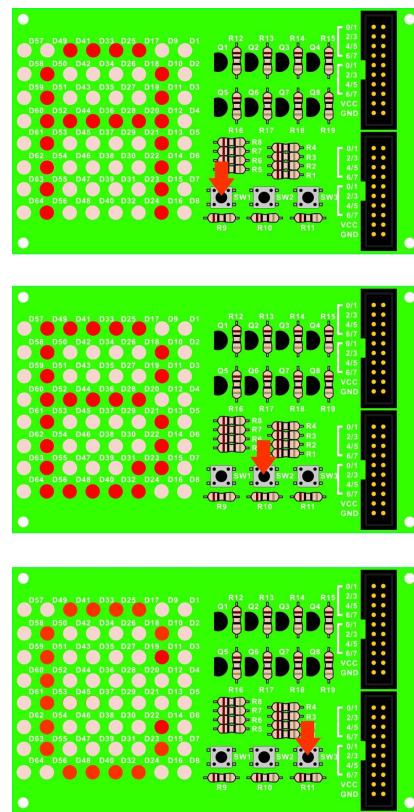
```
while (1)
{
    if(time_cnt>=255)
    {
        time_cnt=0;
        if(state==7) state=0;
        else state=state+1;
    }
    else time_cnt=time_cnt+1;
    if(state==0){ROW=0x00; COLUMN=0x80;}
```

```

else if(state==1){ROW=0xfe; COLUMN=0x40;}
else if(state==2){ROW=0x09; COLUMN=0x20;}
else if(state==3){ROW=0x09; COLUMN=0x10;}
else if(state==4){ROW=0x09; COLUMN=0x08;}
else if(state==5){ROW=0x09; COLUMN=0x04;}
else if(state==6){ROW=0xfe; COLUMN=0x02;}
else if(state==7){ROW=0x00; COLUMN=0x01;}
}

```

여기까지 했다면 스위치에 따라 다른 글자를 띄워보자.

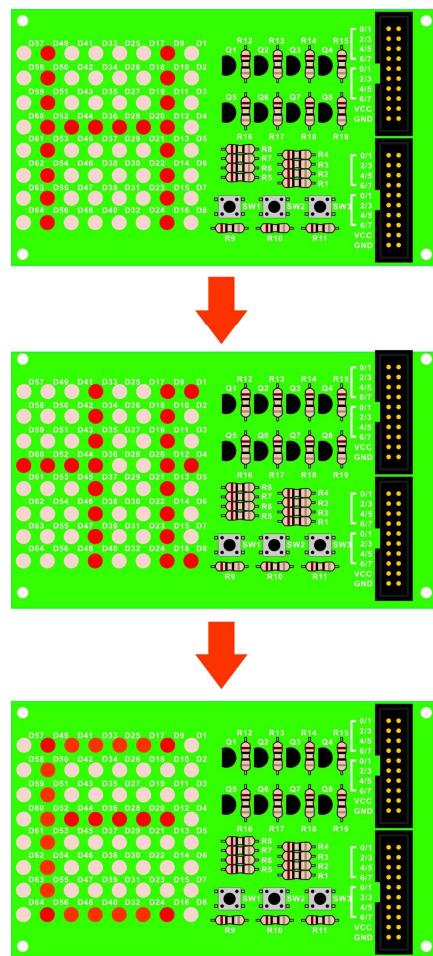


위의 그림과 같은 상태의 코드는 안줘도 되지? 충분히 할 수 있을 거라고 생각해.

## 2.2 프로젝트를 해보자 : 미니 전광판 만들기

전광판 하면 점점 옆으로 밀리면서 새로운 글자가 보이는 게 생각나지 않아? 2.1에서 너희가 했던 걸 응용해서 만들어보자. 맨 처음 간단한 코드는 알려줄 테니까. 대신 응용은 너희끼리 알아서 해봐!

아래 그림은 ‘HELLO’라는 문장이 H부터 시작해서 한 줄씩 왼쪽으로 밀리면서 O까지 보여주고 다시 H부터 보여주는 식이야. 보통 전광판의 동작이지. 너희가 원하는 문장으로 하면 돼.



```

unsigned int text[40]={0x00,0xff,0x08,0x08,0x08,0x08,0x0ff,0x00
    ,0x00,0xff,0x89,0x89,0x89,0x89,0x89,0x00
    ,0x00,0xff,0x80,0x80,0x80,0x80,0x80,0x00
    ,0x00,0xff,0x80,0x80,0x80,0x80,0x80,0x00
    ,0x00,0x7e,0x81,0x81,0x81,0x81,0x7e,0x00
}; //HELLO

//생략

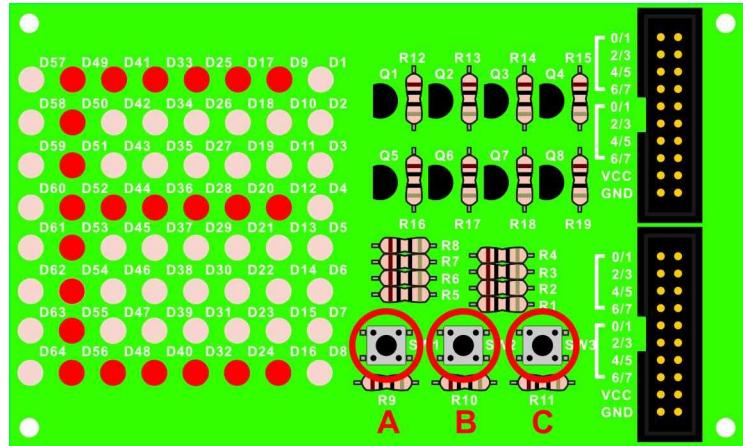
while (1)
{
    if(time_cnt>=255)
    {
        time_cnt=0;
        if(state==7) state=0;
        else state=state+1;
    }
    else time_cnt=time_cnt+1;

    if(time_text_cnt>=60000)
    {
        time_text_cnt=0;
        if(txt_cnt==39) txt_cnt=0;
        else txt_cnt=txt_cnt+1;
    }
    else time_text_cnt=time_text_cnt+1;

    if(state==0){ROW=text[0+txt_cnt]; COLUMN=0x80;}
    else if(state==1){ROW=text[1+txt_cnt]; COLUMN=0x40;}
    else if(state==2){ROW=text[2+txt_cnt]; COLUMN=0x20;}
    else if(state==3){ROW=text[3+txt_cnt]; COLUMN=0x10;}
    else if(state==4){ROW=text[4+txt_cnt]; COLUMN=0x08;}
    else if(state==5){ROW=text[5+txt_cnt]; COLUMN=0x04;}
    else if(state==6){ROW=text[6+txt_cnt]; COLUMN=0x02;}
    else if(state==7){ROW=text[7+txt_cnt]; COLUMN=0x01;}
}

```

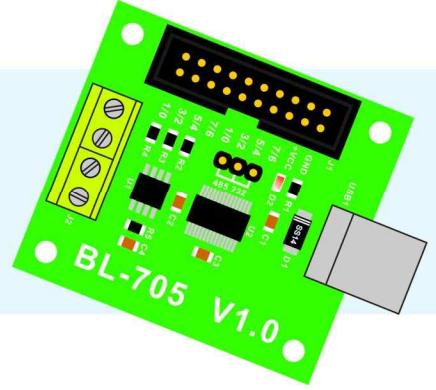
이제 이걸 응용한 걸 내볼테니까 한번 해봐. 물론 이번엔 코드를 알려주지 않을 테니까 스스로 한 번 해봐!



응용1	HELLO라는 글자를 띄울 거야. 근데 A 스위치를 눌렀어. 그럼 왼쪽으로 밀려서 HELLO라는 순서로 보일 거야. C 스위치는 반대로 오른쪽으로 밀면 돼. B 스위치는 정지! 다시 A나, C를 눌러야지 동작을 하게끔. 할 수 있지? 참고로 글자 수는 3글자 이상이어야 한다?
응용2	스위치마다 다른 글자를 띄우는 거야. A-HOW / B-ARE / C-YOU 이런 식으로. 기본은 오른쪽으로 밀고 다른 스위치도 마찬가지야.

# 3

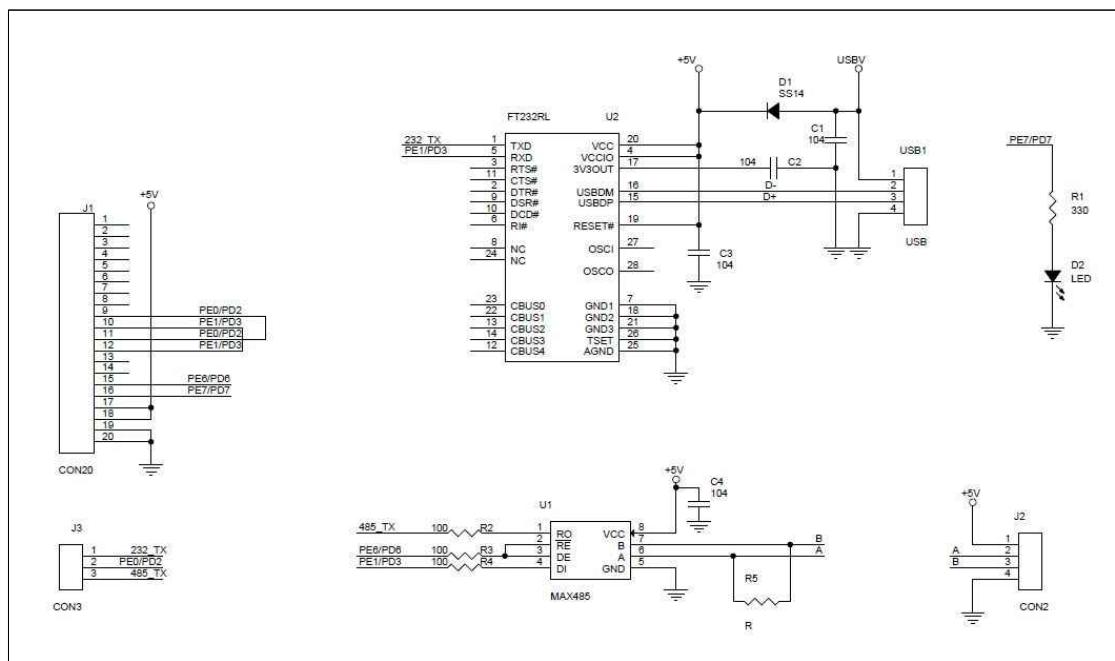
## 프로젝트 2: 통신 전광판



### 3.1 통신 해보기

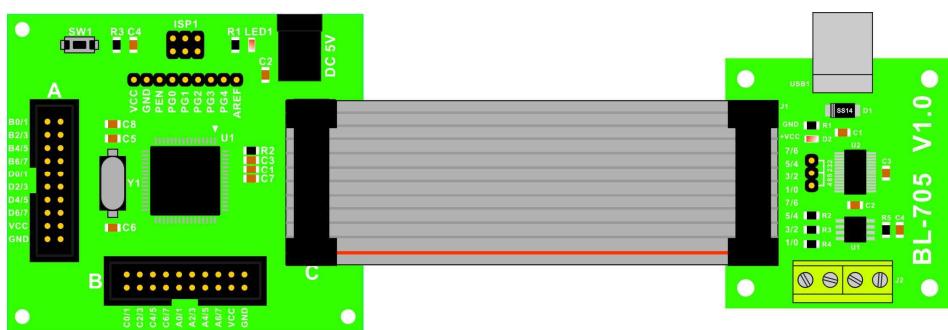
통신을 하려면 그 보드에 대한 회로 이해가 필요하겠지? 이 회로도에서 FT232라는 칩은 RX, TX로 들어온 통신 데이터를 232데이터로 USB포트 신호에 맞게 변환시켜서 통신하는 역할을 해. 신호를 변환시켜야 하는 이유는 여러 가지가 있지만 가장 큰 이유는 서로 사용하는 전압이나 통신 방식이 다르다는 차이 때문이야. 얘기가 잘 와 달지 않지? 예를 한 번 들어볼게.

서로 통신을 해야 하는 두 개의 기계가 있어 근데 두 기계의 거리가 멀어. 그런데 거리가 멀면 통신할 때 잡음이 생겨서 통신이 어려워. 그래서 이 경우엔 보통 높은 전압에서 통신을 하는 거야. 어때 이해가 됐지? 상황에 맞는 통신 방식이 다른 이유는 이런 이유들 때문인 거야.

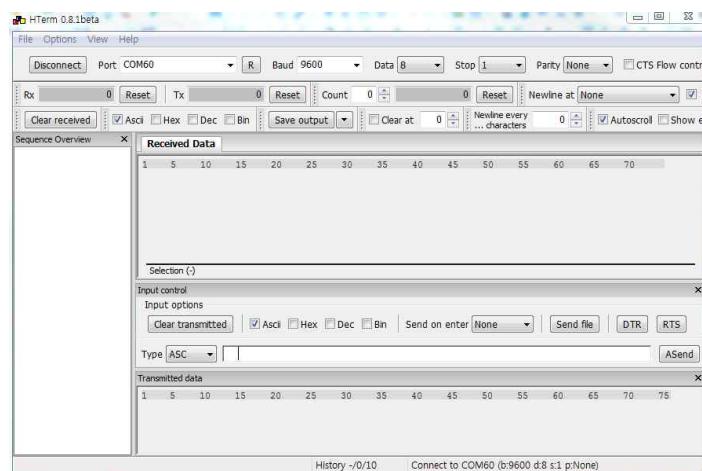


위에 있는 FT232는 아까 설명한 것처럼 기본적으로 컴퓨터와 통신을 위해 사용된 거고 아래에 있는 칩은 MAX485라는 칩인데 한 번에 여러 기기와 통신을 할 수 있는 역할을 해. 하지만 이 칩을 이용해서 바로 컴퓨터와 통신할 수는 없어. 반대쪽에서 MAX485가 다시 변환해준 다음 컴퓨터와 통신을 할 수 있어.

나는 아래와 같이 보드 간의 연결을 했어.



먼저 통신을 해보기 전에 너희 통신 보드가 동작하는지 확인을 해야겠지? 그러기 위해선 터미널 창 하나가 필요해. 나는 아래와 같은 툴 (HTerm)을 쓰고 있어.



꼭 이 툴이 아니고 다른 툴이여도 괜찮아. 그럼 툴을 마련했다면 보드가 동작하는지 꼭 확인해봐.

```

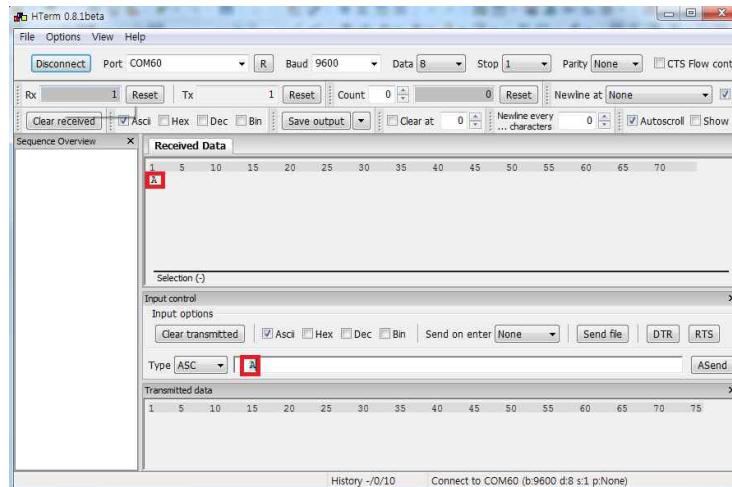
UCSR0A=0x00;
UCSR0B=0x98;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;
//생략

void putuchar0(unsigned char c)
{
    while ((UCSR0A & (1<<5)) == 0);
    UDR0=c;
}

interrupt [USART0_RXC] void usart1_rx_isr(void)
{
    data=UDR0;
    putuchar0(data);
    delay_ms(100);
}

```

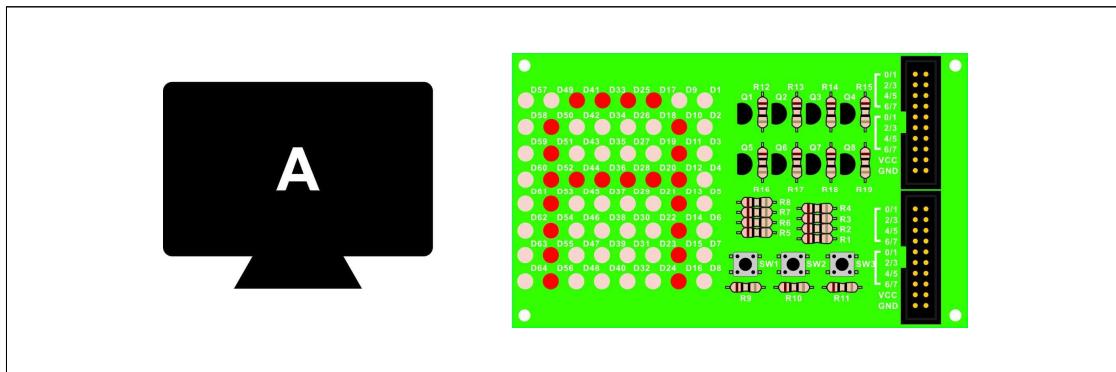
위에 코드는 터미널 창에서 어떠한 데이터를 보내면 다시 터미널 창으로 돌아오는 코드야 결과는 아래와 같아. A를 보냈더니 A가 왔지? 그럼 보드가 송신, 수신이 제대로 된다는 게 확인된 거야.



위에 있는 코드는 데이터 시트를 보면 나와 있어. 물론 조금 변형되긴 했지만 같은 동작을 하는 코드야. 설정 등은 꼭 이렇게 하라는 게 아니라 데이터 시트를 보고 너희가 원하는 데로 수정해도 괜찮아.

### 3.2 프로젝트를 해보자 : 통신 전광판 만들기

소제목 2단원에서 했었던 거 있지? 그걸 통신으로 하는 거라고 생각하면 돼. 아래 그림과 코드는 통신으로 보낸 문자를 LED MATRIX에 띄우는 거야. 물론 너희가 띄울 문자는 알아서 코드로 짜야 하고. 나는 컴퓨터에서 A를 보냈을 때 LED MATRIX에 A가 띄워지고 그 외에 문자를 보내면 다 꺼지도록 했어.



The diagram consists of two parts. On the left, there is a black computer monitor icon with the letter 'A' displayed on its screen. On the right, there is a green breadboard with various electronic components. The breadboard has several rows of pins labeled D01 through D18, Q1 through Q5, R11 through R19, and SW1 through SW4. There are also resistors (R1-R10), capacitors (C1-C4), and a 74HC595 integrated circuit. Power supply pins VCC and GND are also present.

```
//생략
while(1)
{
    if(data=='A')data_sig=1;
    else data_sig=0;

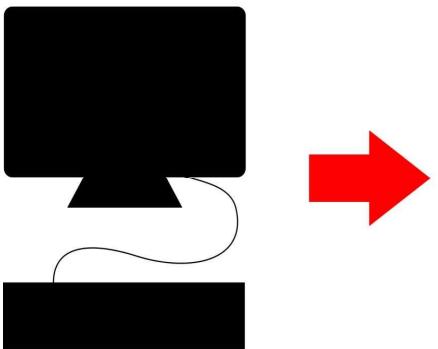
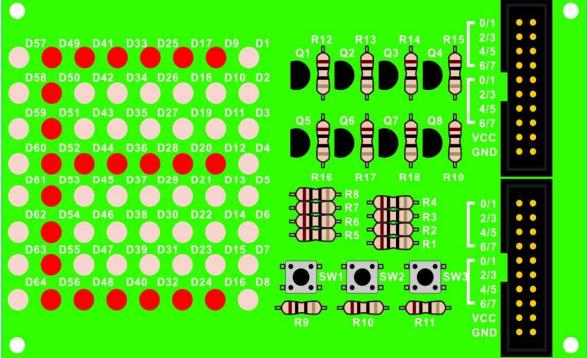
    if(data_sig==0)
    {
        ROW=0xfe;
        COLUMN=0X40;
    }
    else
    {
        if(time_cnt>=255)
        {
            time_cnt=0;
            if(state==7) state=0;
            else state=state+1;
        }
        else time_cnt=time_cnt+1;
        if(state==0){ROW=0x00; COLUMN=0x80;}
    }
}
```

```

        else if(state==1){ROW=0xfe; COLUMN=0x40;}
        else if(state==2){ROW=0x09; COLUMN=0x20;}
        else if(state==3){ROW=0x09; COLUMN=0x10;}
        else if(state==4){ROW=0x09; COLUMN=0x08;}
        else if(state==5){ROW=0x09; COLUMN=0x04;}
        else if(state==6){ROW=0xfe; COLUMN=0x02;}
        else if(state==7){ROW=0x00; COLUMN=0x01;}
    }
}

```

다했어? 그럼 이제 응용해서 해보자.

		
00 01 1	터미널로 1을 보내면 HOW,2를 보내면 ARE,3을 보내면 YOU라고 띄울 거야. 물론 문자는 너희가 맘대로 해도 좋아. '<'를 보내면 왼쪽으로 밀고, '>'를 보내면 오른쪽으로 밀면 돼. 'S'를 보내면 멈추는 기능을 넣으면 되고. 왼쪽으로 밀리는 상태에서 '<'를 누르면 1-2-3단계로 속도가 빨라져야 해. 오른쪽도 마찬가지이고. 해본 건데 통신만 추가된 거라 별로 안 어렵지?	
응 용	'P'를 보내면 전원 ON/OFF기능이야. OFF 상태에서는 터미널에 뭘 보내도 응답하면 안 돼. ON 상태에서는 미리 만들어놓은 문	

2

장이 끄기 전 상태의 설정으로 나오면 되. '<'를 누르면 왼쪽으로 밀리고 '>'를 누르면 오른쪽으로 밀리면 돼. 왼쪽으로 밀리는 상태에서 '<'를 누르면 1-2-3단계로 속도가 빨라져야 해. 오른쪽도 마찬가지이고. 'S'를 누르면 멈추면 돼.