AVR - FND

임베디드스쿨2기

Lv1과정

2021. 06. 25

박태인

FND
7-segment

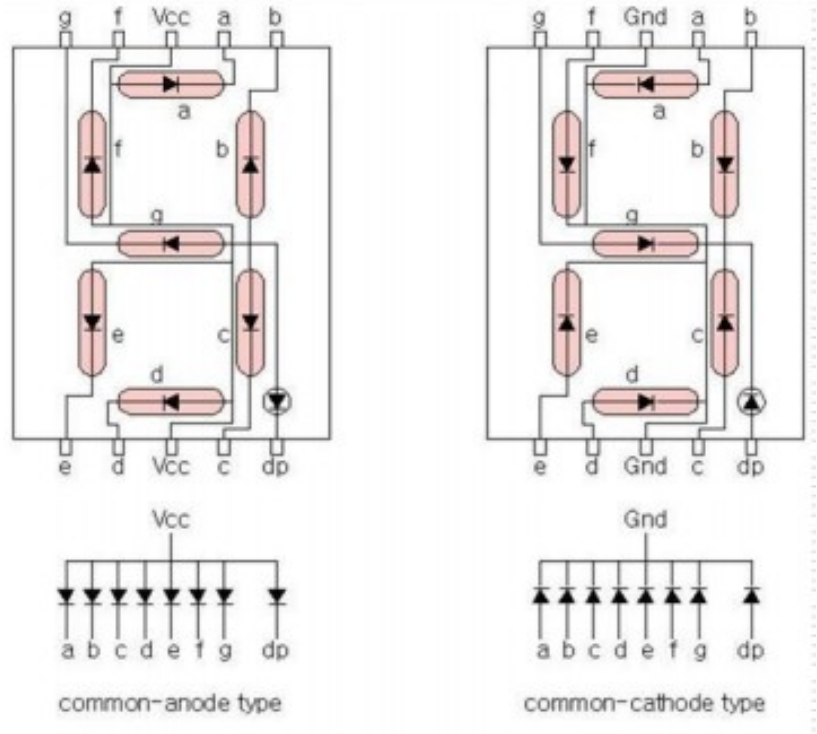
우선 FND에 대해 조사해 보자.

FND란?

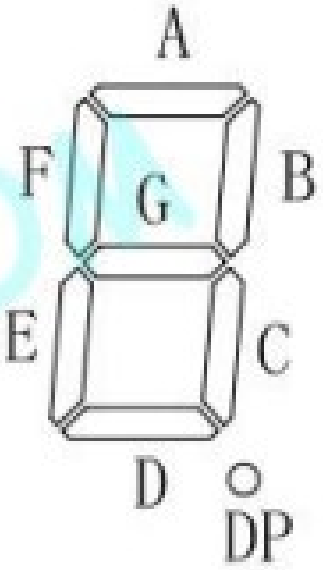
- Flexible Numeric Device의 약자
- 보통 7-Segment LED 라고 칭함
- LED 7개(점 포함 8개)로 숫자를 표시하기 쉽도록 배열한 제품
- 1개, 2개, 3개, 4개를 함께 디스플레이 하는 형태의 제품 판매
- 많이 사용하는 곳 : 엘리베이터 층 표시기, 임베디드 제품 상태 표시기

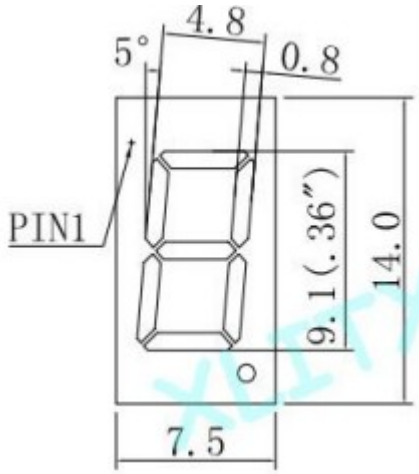
FND 종류 및 구조

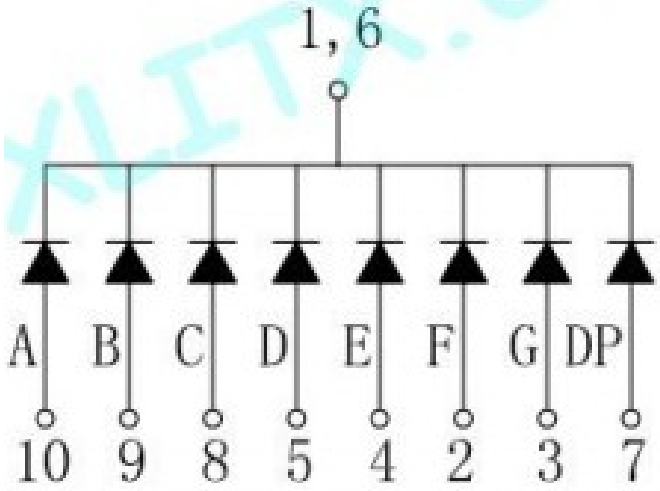
- **Common Cathode 타입**
- **Common Anode 타입**



FND 숫자 표현(Common-Cathode 경우), 아래 FND 모델명 3611AS







16 진수	7-세그먼트의 비트값								데이터 값 (HEX)
	DP	G	F	E	D	C	B	A	
0									
1									
2									
3	0	1	0	0	1	1	1	1	0X4F
4	0	1	1	0	0	1	1	0	0X66
5	0	1	1	0	1	1	0	1	0X6D
6	0	1	1	1	1	1	0	1	0X7D
7	0	0	1	0	0	1	1	1	0X27
8	0	1	1	1	1	1	1	1	0X7F
9	0	1	1	0	1	1	1	1	0X6F
A	0	1	1	1	0	1	1	1	0X77
B	0	1	1	1	1	1	0	0	0X7C
C	0	0	1	1	1	0	0	1	0X39
D	0	1	0	1	1	1	1	0	0X5E
E	0	1	1	1	1	0	0	1	0X79
F	0	1	1	1	0	0	0	1	0X71

FND

FND H/W 연결

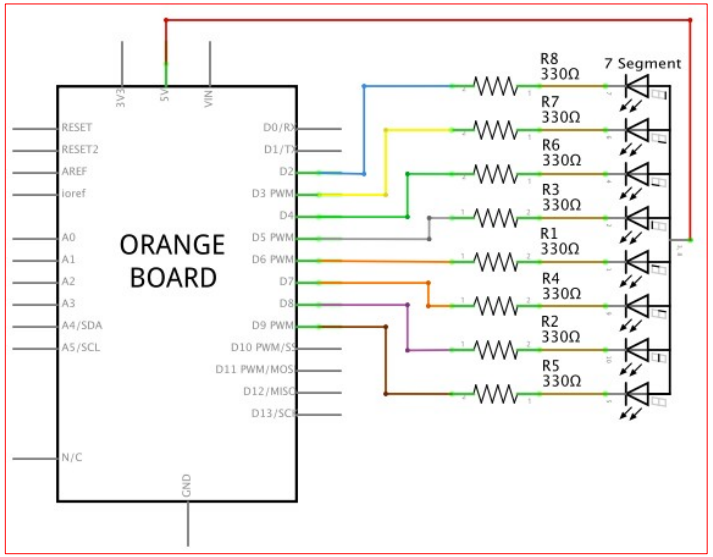
1. Electro-Optical Characteristics(Ta=25℃)

PARAMETER	SYMBOL	DEVICES (ULTRA-BRIGHT RED)		UNIT	TEST CONDIONS
		TYP	MAX		
Peak Emission Wavelengrth	λp	640		nm	IF=10mA
Forward Voltage	VF	1.8		V	IF=10mA
Reverse Current	IR		50	μA	VR=5V
Segment To Segment (Dot To Dot) Luminonous Intensity Ratio	IV-M	1.5:1			IF=20

위는 FND의 Datasheet상의 스펙
VF 전압 : LED On 시 걸리는 전압
IF 전류 : LED 소모 전류

위 값을 보고 저항값 R = (5-1.8) / 0.01A = 320 옴
ㄴ **따라서 FND 1 seg 당 (약 330 옴의 저항을 사용하면 된다.)**

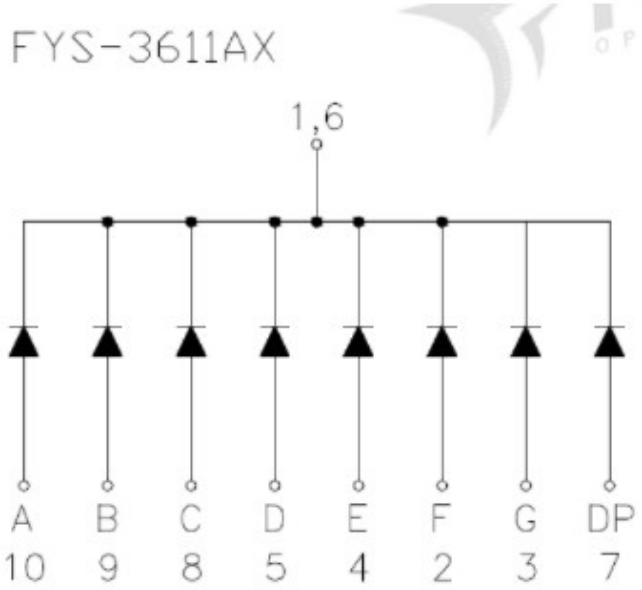
예시 회로

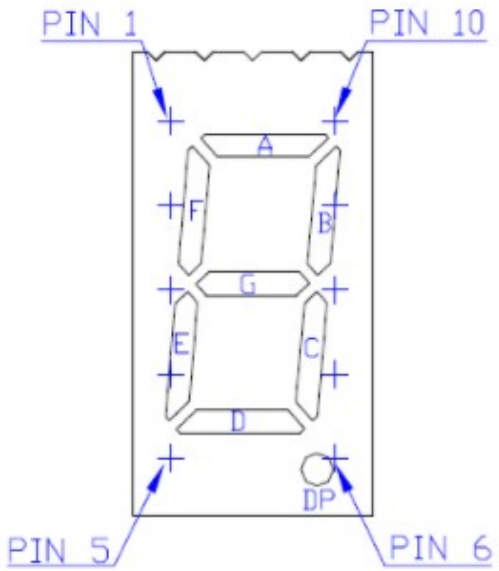


코딩

```
 // 3611AS 7segment, A-10, B-9, C-8 , D - 5, E -4, F -2, G -3, DP-7
#include <avr/io.h>

#define F_CPU 16000000UL
#include <util/delay.h>

int main(void)
{
    DDRD = 0xFF;

    // A
    PORTD = 0x01;
    _delay_ms(1000);

    // B
    PORTD = 0x02;
    _delay_ms(1000);

    // C
    PORTD = 0x04;
    _delay_ms(1000);

    // D
    PORTD = 0x08;
    _delay_ms(1000);

    // E
    PORTD = 0x10;
    _delay_ms(1000);
```

```
    // F
    PORTD = 0x20;
    _delay_ms(1000);

    // G
    PORTD = 0x40;
    _delay_ms(1000);

    // DP
    PORTD = 0x80;
    _delay_ms(1000);

    // A, B, C, D, E, F ===> 숫자 0
    // 0x01
    // 0x02
    // 0x04
    // 0x08
    // 0x10
    // 0x20
    // -------------
    // 0x3F 위를 전부 OR 하면 나오는 값

    // A, B, C, D, E, F, DP ===> 숫자 0
    // 0x01
    // 0x02
    // 0x04
    // 0x08
    // 0x10
    // 0x20
    // 0x80
    // -------------
    // 0xBF 위를 전부 OR 하면 나오는 값
```





코딩의 내용은 각 포트 별 High 신호를
1초씩 주어 LED가 반시계 방향으로 하나씩
돌다가 마지막에 FND가 0 을 표시하게 됩니다.
(마지막 0 값 관련 코드는 뒷 페이지)

FND

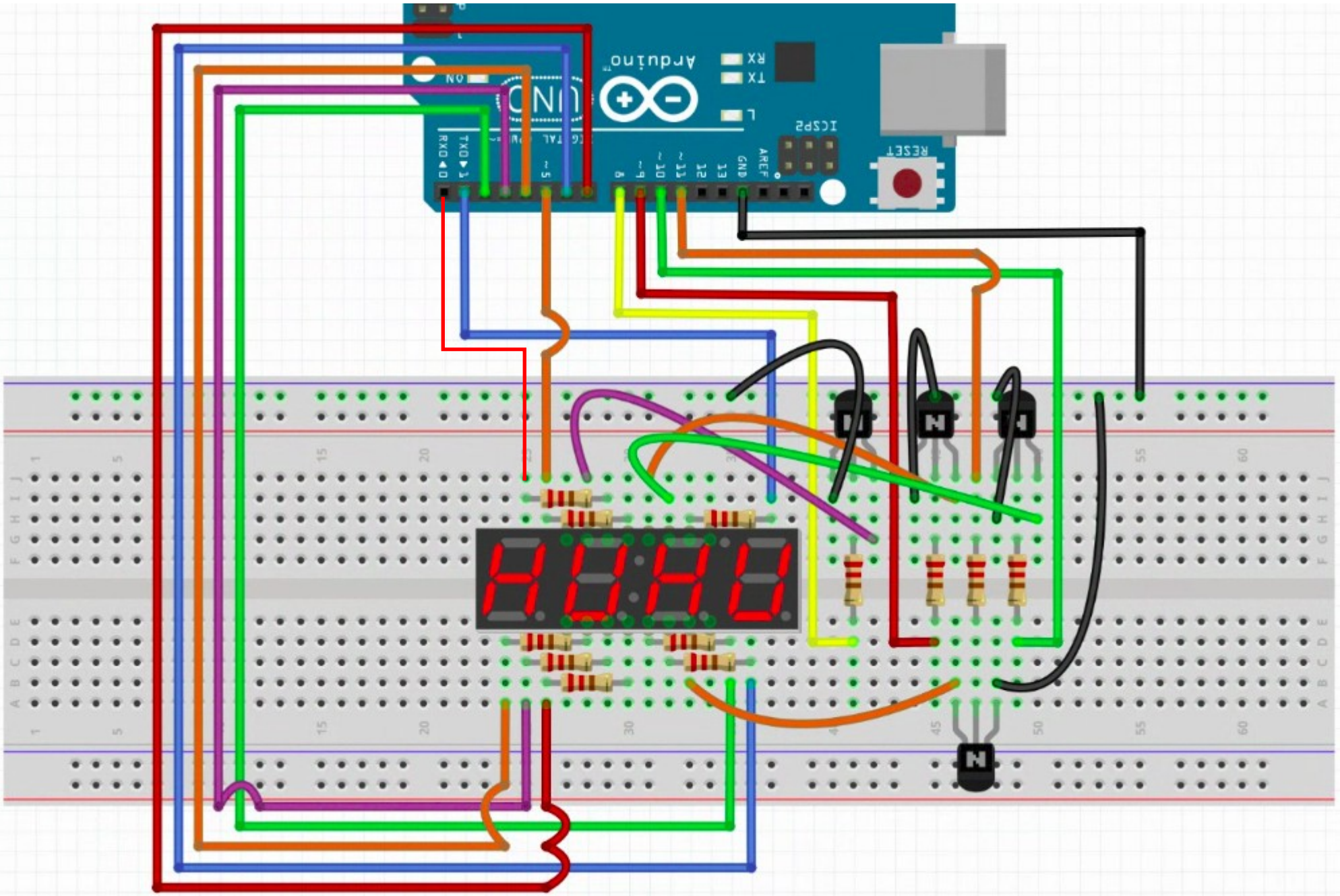
```
// F
PORTD = 0x20;
_delay_ms(1000);

// G
PORTD = 0x40;
_delay_ms(1000);

// DP
PORTD = 0x80;
_delay_ms(1000);

// A, B, C, D, E, F ===> 숫자 0
// 0x01
// 0x02
// 0x04
// 0x08
// 0x10
// 0x20
// -------------
// 0x3F 위를 전부 OR 하면 나오는 값

// A, B, C, D, E, F, DP ===> 숫자 0
// 0x01
// 0x02
// 0x04
// 0x08
// 0x10
// 0x20
// 0x80
// -------------
// 0xBF 위를 전부 OR 하면 나오는 값
```

```
    // 16진수에서 1자리는 2진수로 몇 자리 ? 4자리
    // x ------------------------------> xxxx
    // 16^0                              2^3 2^2 2^1 2^0
    //                                   각 자리 수를 모두 더하면 15(0xf)
    // 0,  1,  2,  3,  4,  5,  6, 7, 8, 9
    // 10, 11, 12, 13, 14, 15

    // F ====> 1111
    // E ====> 1110
    // D ====> 1101
    // C ====> 1100
    // B ====> 1011
    // A ====> 1010 ====> 110 ??? 결국 표현의 이중성 때문에 일관화 시키기 위해 기호를 도입
    // 9 ====> 1001
    // 8 ====> 1000
    // 7 ====> 0111
    // 6 ====> 0110
    // 5 ====> 0101
    // 4 ====> 0100
    // 3 ====> 0011
    // 2 ====> 0010
    // 1 ====> 0001
    // 0 ====> 0000
    PORTD = 0xBF;
    _delay_ms(1000);

    // Insert code

    while(1)
    ;

    return 0;
}
```

여기서 잠깐, 16진수 계산 할 때는
한 자리당 왼쪽과 같이 4 비트로 나타내어 지고
한번에 많은 LED를 ON
시킬 때는 OR 연산을 한 16진수 값을 적용한다.

4Array FND 7-segment

4FND (HW 연결)



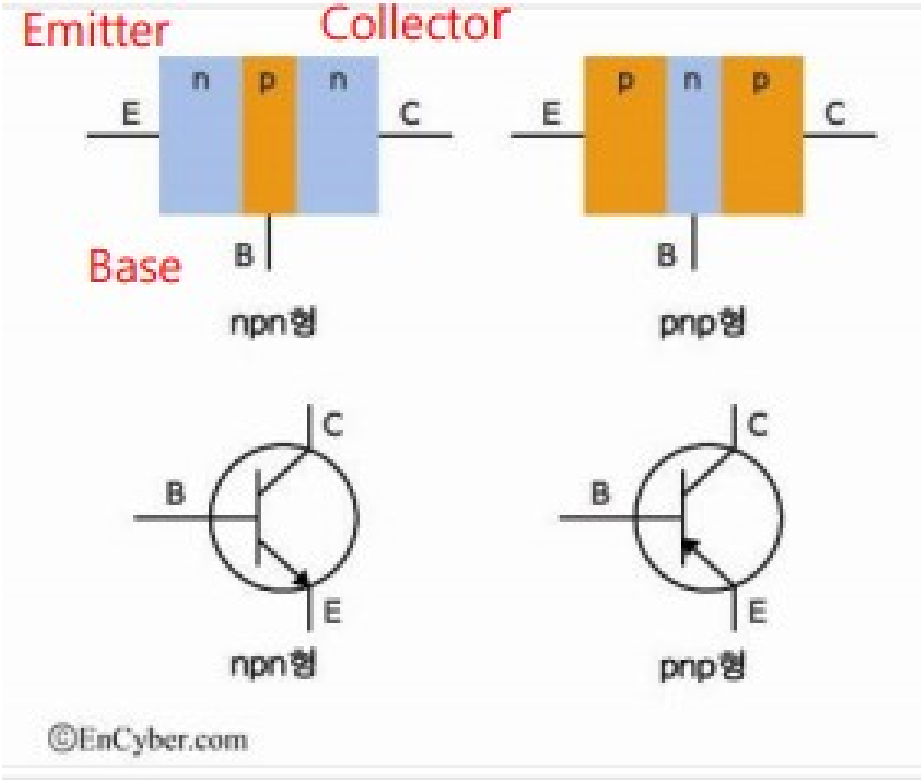
여기서 잠깐, 회로 연결 시 각각 FND의 동작 실행시에 TR(Transistor)를 사용하게 되는데 이것에 대해 알아 보자.

- 규소나 게르마늄으로 만들어진 반도체를 세 겹 으로 접합하여 만든 전자회로 구성요소이며, 전류나 전압흐름을 조절 하여 **증폭, 스위치 역할**을 하는 전자부품

- 대부분의 전자회로에 사용되며 이를 고밀도 집적하여 IC를 제작

- BJT(Bipolar Junction Transistor)와 FET(Field Effect Transistor)로 구분하는데 보통은 BJT를 의미함

- BJT는 다시 NPN형과 PNP형으로 구분되며 NPN은 Source 회로에, PNP은 Sink 회로에 사용된다.



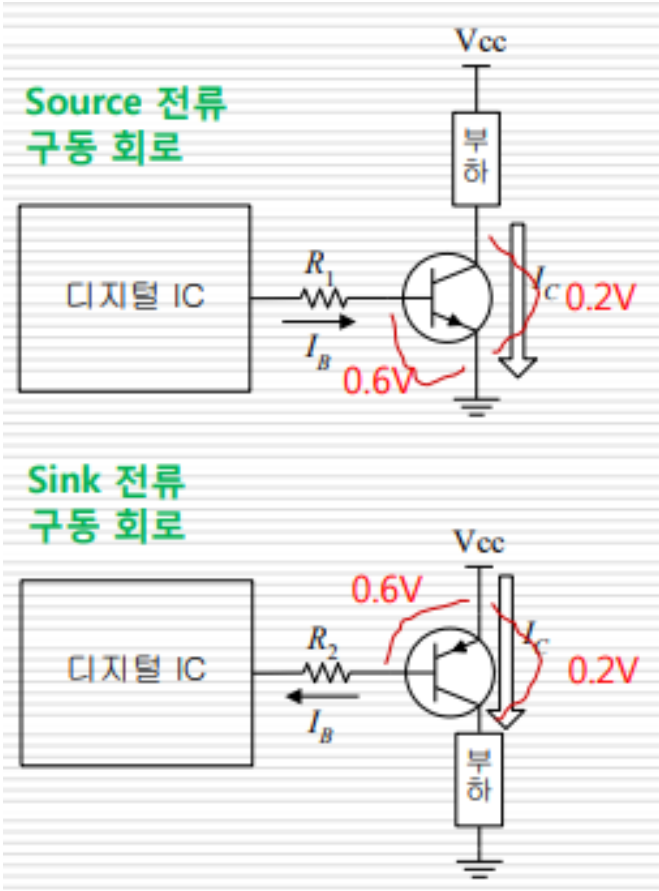
TR(Transistor)을 이용한 부하 구동 회로

- FND나 모터 등의 부하 구동은 전류 소모가 많기 때문에 일반적인 TTL의 출력만으로 직접 구동하기 어렵다.

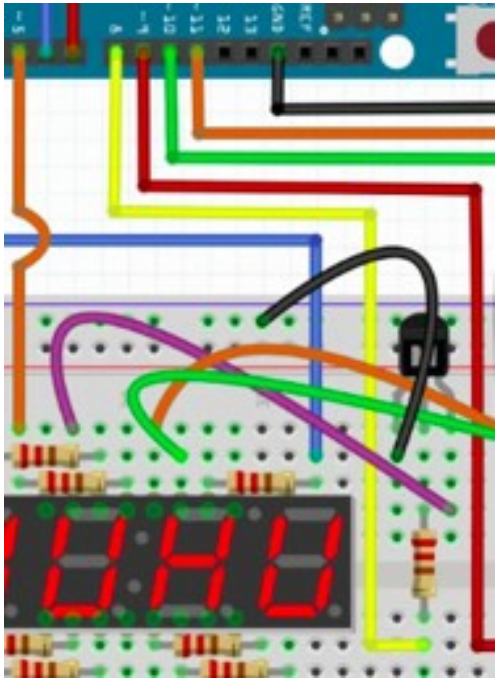
- 이런 경우 작은 전류로 큰 전류를 제어 할 수 있는 TR을 오른쪽과 같이 연결하여 이용하면 이를 해결 할 수 있다.

- Source 전류 구동 회로(신호를 출력하는 회로)를 예로 들면, 입출력신호를 B(Base)에 연결하고 C(Collecter)에 연결

- R1, R2는 보통 약 1K ~ 10K 정도

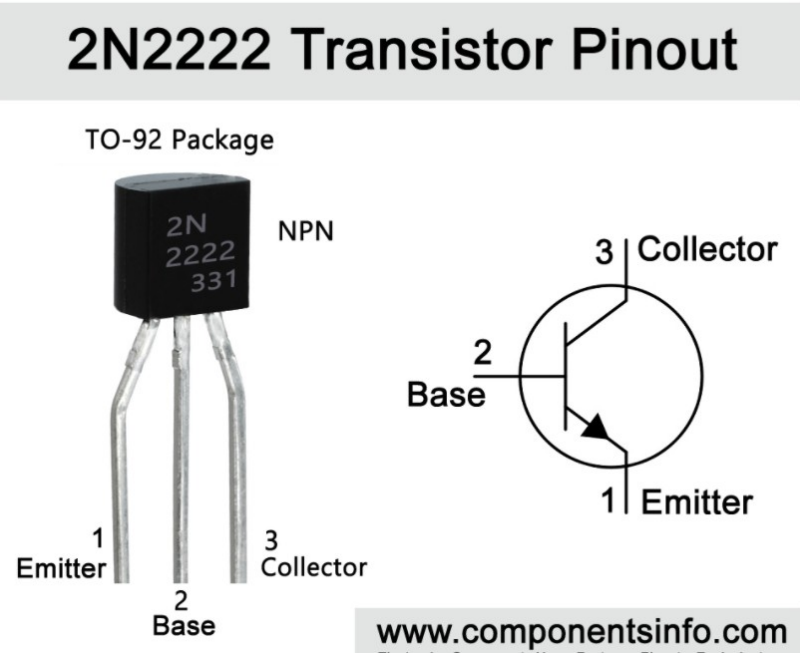


그렇다면 현재 회로 상에서 TR은 어떻게 연결 되어 있는 것 일까?

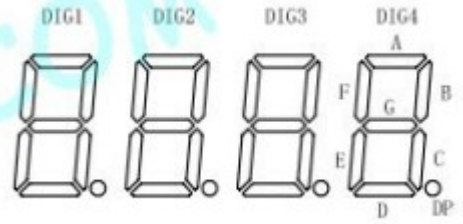


- 왼쪽 그림은 회로상에서
  2n2222 TR(NPN)이
  Eimmiter는 GND로(검정선)
  Base는 아두이노 출력(노란선)
  Collector은 FND로 (보라색 선)
  들어 가는 것을 볼 수 있다.

- 이런식의 구조는 선택 된
  세그먼트에서
  신호가 출력 되는 구조이기
  때문이다.



TR 방향성



부품 번호 : 3641AS

```
// H/W 연결시에 7segment 11번 핀이 PORTD 0번 (아두이노 0번 핀과 연결 되어 있어야 한다.)
// TR로 4개의 segment중 출력하게 될 Array의 segment를 선택하게 되고
// 선택된 segment 에서 신호가 출력 되는 것 이므로 TR의 base/emmitter/collector 연결을 주의한다.
#include <avr/io.h>

#define F_CPU 16000000UL
#include <util/delay.h>

// A-11, B-7, C-4, D-2, E-1, F-10, G-5, DP-3

// 12, 9, 8, 6 : selector

// 숫자 0 표현 : A,B,C,D,E,F -> 0,1,2,3,4,5 (PORTD 번호)
// 1 : B,C                   -> 1,2
// 2 : A, B, G, E, D         -> 0, 1, 3, 4, 6
// 3 : A, B, C, D, G         -> 0, 1, 2, 3, 6
// 4 : B, C, F, G            -> 1, 2, 5, 6
// 5 : A, C, D, F, G         -> 0, 2, 3, 5, 6
// 6 : C, D, E, F, G         -> 2, 3, 4, 5, 6
// 7 : A, B, C, F            -> 0, 1, 2, 5
// 8 : A, B, C, D, E, F, G   -> 0, 1, 2, 3, 4, 5, 6
// 9 : A, B, C, F, G         -> 0, 1, 2, 5, 6

unsigned char digit[10] = {0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7c, 0x27, 0x7f, 0x67};
// 두개를 선택 했을 때 비트식으로 선택하기 위해 아래 처럼 선언
unsigned char digit_select[4] = {0x01, 0x02, 0x04, 0x08};
```

- 4 Array FND 연결 시 숫자 표현
  16 진수 값 배열

- 각 FND Select 값 배열

4FND

```c
int main(void)
{
    int i,j;


    DDRD = 0xff; // 숫자표현
    DDRB = 0x0f; // 셀렉터

    PORTB = 0x08;
    for(i=0; i<10; i++)
    {
        PORTD = digit[i];
        _delay_ms(500);
    }

    // Insert code

    while(1)
    {
        for(i=0; i<4; i++)
        {
            PORTB = digit_select[i];    // 특정 자리수 선택

            for(j=0; j<10; j++)
            {
                PORTD = digit[j];       // 값 증가
                _delay_ms(100);
            }
        }

    }
    ;

    return 0;
}
```

- 숫자 LED 표현 PORTD
  selector PORTB 각각 출력 설정

- 초기 한번 동작
  for문을 10번 동작 시켜서 0.5초 단위로 숫자 값 증가.

위의 초기 한번 동작이 완료 된 후
중복 for문을 사용하여

Select이 하나 되면 그 select 된 자리에
0~9까지 값을 표시하고

그 다음 part의 select으로 넘어가서

다시 이어 가는 식으로 동작하는 코드이다.

4Array FND 7-segment timer

4FND Timer

```c
#include <avr/io.h>


#define F_CPU 16000000UL
#include <util/delay.h>

unsigned char digit[10] = {0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7c, 0x27, 0x7f, 0x67};
// 두개를 선택 했을 때 비트식으로 선택하기 위해 아래 처럼 선언
unsigned char digit_select[4] = {0x01, 0x02, 0x04, 0x08};
unsigned char fnd[4];

int main(void)
{
    int i, count = 0;

    DDRD = 0xFF;
    DDRB = 0x0F;

    // Insert code

    while(1)
    {
        count++;

        if(count == 10000)  // sin 파 만들 때 오버플로우 에러를 방지 하고자 한계치 정함
        {
            count =0;
        }
        fnd[0] = (count / 1000) % 10;   // 0.01초 자리
        fnd[1] = (count / 100) % 10;
        fnd[2] = (count / 10) % 10;
        fnd[3] = count % 10;
```

윗 부분 코드는 앞서 있었던 FND 숫자 표시 및 셀렉터 배열 이라고 생각 하면 된다.

Counter 숫자가 현재 가동되고 있는 bit 수의 숫자 보다 크게 되면 overflow 현상이 일어나게 되고 이에 따라 오동작을 일으킬 수 있으므로 최대 카운터 숫자 값을 이처럼 정해 두는 것이 좋다.
(sin파 만드 때라는 표현이 이해가 잘 안 갈 경우 10~11번째 숙제 ppt 파일을 참고 할 것.)

Fnd 선택 된 자리마다 어떤 의미가 숫자가 나타날지를 정하는 코드이다.

%10을 함으로써 0~9의 숫자만 나타나게 하고,

나누기는 각각의 자릿수 의미를 부여하기 위함이다.

4FND Timer

```
    while(1)
    {
        count++;

        if(count == 10000)  // sin 파 만들 때 오버플로우 에러를 방지 하고자 한계치 정함
        {
            count =0;
        }
        fnd[0] = (count / 1000) % 10;   // 0.01초 자리
        fnd[1] = (count / 100) % 10;
        fnd[2] = (count / 10) % 10;
        fnd[3] = count % 10;

        for(i=0; i<4; i++)
        {
            PORTD = digit[fnd[i]];
            PORTB = digit_select[i];

            //_delay_ms(2); // 계산상 4번 for문 돌아서 8ms고 아래 2ms 추가 되어 10 ms 로 만들기 위함
                        // 왜냐면 한번 4자리 숫자가 다 뜨는 시간은 0.01초 즉 10ms 가 되어야 하기 때문.
                        // but 실제로 나타나는 것에 대한 오차는 있으므로 노가다 성이 생길수 있다.

            /*if(i % 2)      // TR select와의 H/W적 오차 때문
            {
                _delay_ms(1);
            }*/

            // 따라서 다음과 같은 방법도 가능
            _delay_us(2500); // 2.5 * 4 = 10 ms
                           // 더 디테일 하게 가려면 코드가 실행되는 시간 까지 따질수도 있긴 하다.
                           //
        }
    }
    ;
    return 0;
}
```

For문 해석

한번에 4개의 LED를 for문을 통해서 선택 출력하게 됩니다.

그런데 여기서 4 FND는 각각 0~99초를 0.01초 자리 까지 나타내어 타이머를 만들어 내게 됩니다.

따라서 4개의 숫자가 변동하는 타이밍은 10ms 수준으로 되어야 할 것 입니다.

이를 위해 delay를 사용하게 됩니다.

Delay 표현 방법을 왼쪽의 주석을 보면 2가지로 해 본 것 입니다.

첫번째,

Delay_ms(2)를 통해 for문을 4번 돌려 8ms로를 만들고 아래 if문을 통해 2ms가 추가 되어 10ms를 만든 방법. [이렇게 나눈 이유는 TR selec시에 H/W 적 오차가 있을 수 있기 때문]

두번째,

Delay_us를 통해 한번에 10ms 값을 만들어 내는 경우

✓ 위의 방법으로 delay를 주어도 디테일 하게 들어가면 시간적 오차가 생길 순 있으나, 너무 큰 노가다 성이 될 수 있으므로 이정도로 하고 넘어가도록 한다.

프로젝트

프로젝트 주제는 **BLDC 모터 제어기** 주제이지만, 우선 제어기에 사용될
IC인 Atmega128 칩을 브레드 보드에 적용하여

모터의 상태를 나타내기 위한 캐릭터 LCD 제어를 미리 해보는 것으로 하였습니다.

아래는 시험 하기 위해 사용된 Atmega128 보드와, LCD 정보 입니다.

상품명 : 캐릭터 LCD
품 번 : OTM651 Y-YG-1