

7 장 : LED로 X-mas 트리 만들기



JCnet
제이씨넷

신 상 석

목차

1. LED
2. GPIO
3. ATmega128 입출력 포트
4. JKIT-128-1에서의 LED 연결 설계
5. 실습 LED-1 : GPIO로 LED 켜고 끄기
6. 실습 LED-2 : GPIO로 LED 움직임 표현하기
7. 실습 LED-3 : LED로 X-mas 트리 만들기

LED



□ LED (Light-Emitting Diode)

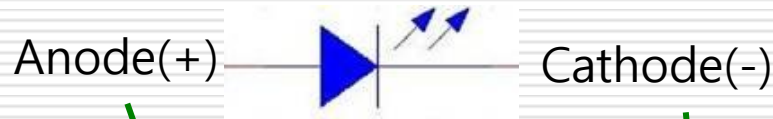
- 발광다이오드라 칭하며, 칼륨, 인, 비소 등을 재료로 한 다이오드(diode)로 순방향으로 전류를 흘리면 빛을 발함
- 빛의 색깔은 크리스털 도핑의 양과 종류에 따라 빨강, 노랑, 녹색, 파랑의 색을 나타냄
- 일반 전구나 네온램프 등 다른 발광 소자와 비교하여 전기→빛의 변환 효율이 높으며, 열이 나지 않고, 소형 경량이기 때문에 수명이 김
- 각종 숫자·문자표시기, 카메라의 자동초점용 광원, 광통신용, 광고판, TV 등에 사용되며 응용범위가 넓음
- 보통 10~20mA 전류에 1.5V~2.5V 전압 강하

LED가 가장 많이 쓰이는 곳은?

신호등, 전광판
조명, TV, 전원표시기...

LED

□ LED 심볼



□ LED 타입

- DIP
- SMD

Red, Green,
Orange, Yellow,
Blue, White
(2/3/5mm)
(2/3/5색)

다리가 긴 쪽이
양극(+, Anode)

1005, 1608,
2012, 3216
크기

녹색 점(Cathod Mark)이
있는 곳이 음극(-, Cathode)

LED

□ LED 규격

●COMMODITY : T-1 3/4 Standard 1.0"Lead, 5 ϕ

●DEVICE NUMBER : BL-B5134

PAGE: 2

●ELECTRICAL AND OPTICAL CHARACTERISTICS (Ta=25°C)

VERSION : 1.0

Chip		Lens Appearance	Absolute Maximum Rating				Electro-optical Data (At 20mA)			Viewing Angle 2 θ 1/2 (deg)
Emitted Color	Peak Wave Length λ P(nm)		Δ λ (nm)	Pd (mW)	If (mA)	Peak If(mA)	Vf(V)		Iv Typ. (mcd)	
							Typ.	Max.		
Bright Red	700	Red Diffused	90	40	15	50	2.2	2.6	12.0	35

소모전력

=

전류

x

전압

LED

□ LED 연결 방법

- Anode 쪽에 신호를 연결하고 Cathode 쪽에 시리얼로 저항을 연결한 후 GND에 연결하는 방법

1? 0? +5V? GND?



- Anode 쪽에 전원(+5V 등)을 연결하고 Cathode 쪽에 시리얼로 저항을 연결한 후 신호를 연결하는 방법

1? 0? +5V? GND?



LED와 저항의
순서는 바뀌어도
상관없음!

LED와 저항의 순서가 바뀌면 어떻게 될까?

Source 전류와 Sink 전류의 최대 크기는?

각각 80mA로 동일,

LED

□ LED 시리얼 저항값 구하는 방법



LED 연결시
대표 저항값!

$$R = \frac{\text{공급전압} - \text{LED전압}}{\text{LED전류}} = \frac{5V - 1.7V}{10 \text{ mA}} = 330 \Omega$$

LED 전압이 2.8V인 고휘도 청색 LED를 5V의 공급전압으로 제어하려고 할 때 함께 연결할 시리얼 저항의 저항값은 얼마로 하여야 할까? 단, LED의 전류는 11mA가 적정 전류라고 한다.

GPIO

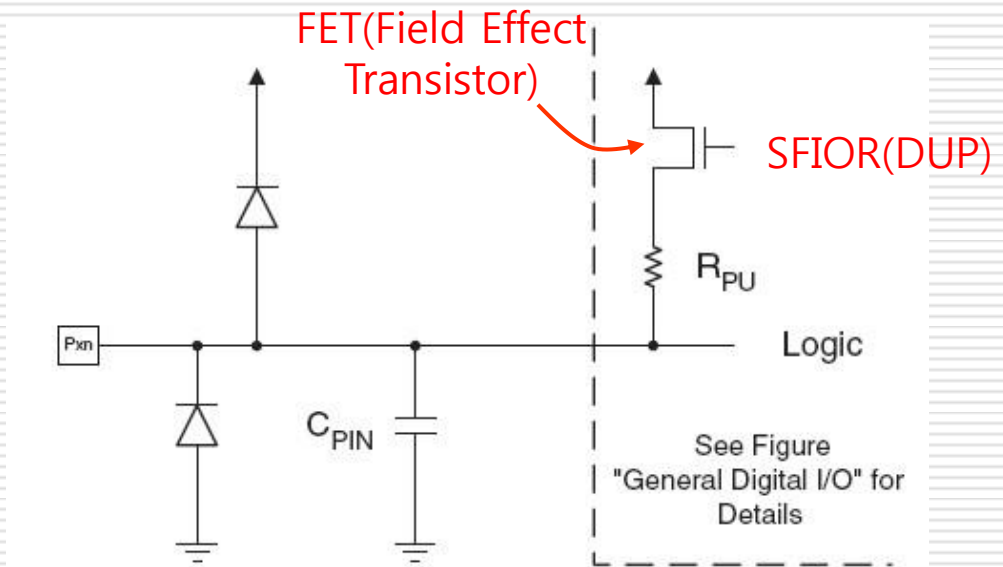
□ GPIO(General Purpose Input Output)

- 범용으로 사용되는 입출력 포트로 설계자가 입력과 출력을 마음대로 선택할 수 있음
- 출력으로 사용시, 0과 1의 출력 신호를 임의로 만들어줄 수 있는 구조를 가짐
- 입력으로 사용할 때는 외부 인터럽트를 처리할 수 있도록 하는 기능이 있는 경우가 있음
- 관련 레지스터는 크게 **입출력 방향 전환 레지스터, 출력용 레지스터, 입력용 데이터 레지스터**의 3가지가 필요
- 내부적으로 pull-up 저항을 가지고 있는 경우가 많음
- 마이크로컨트롤러에서는 대부분의 핀들을 GPIO로 설정하는 경우가 많고, 보통 다른 신호와 중복(multiplexing)하여 사용

ATmega128 입출력 포트

□ ATmega128 입출력 포트

- 6개의 8비트 I/O포트(PA, PB, PC, PD, PE, PF)와 1개의 5비트 I/O포트(PG)로 구성 (총 53개 포트)
- 모든 포트 핀은 개별적으로 내부 풀업 저항을 사용할 수 있음
- 모든 I/O핀은 VCC와 GND사이에 각각 다이오드를 접속하여 포트를 보호
- Read-Modify-Write 기능으로 비트 단위의 포트 설정 가능



ATmega128 입출력 포트

□ ATmega128 GPIO 관련 레지스터

■ DDRx (Data Direction Register)

- 각 포트에 대한 데이터 입출력 방향 지정용 레지스터
- DDRA~DDRG 레지스터의 해당 비트에 '1'을 쓰면 출력(default), '0'을 쓰면 입력으로 설정

■ PORTx (Port Output Register)

- 데이터 출력 레지스터
- 출력을 원하는 데이터값을 PORTx 레지스터에 쓰면 됨

■ PINx (Port Input Register)

- 데이터 입력 레지스터
- PINx 레지스터의 값을 읽으면 그것이 입력된 값임

■ SFIOR(Special Function IO Register) 레지스터

- SFIOR의 비트2(PUD: Pull-Up Disable)를 '1'로 세트하면 풀업 저항이 비활성화되고(기본 상태) '0'으로 하면 활성화 됨

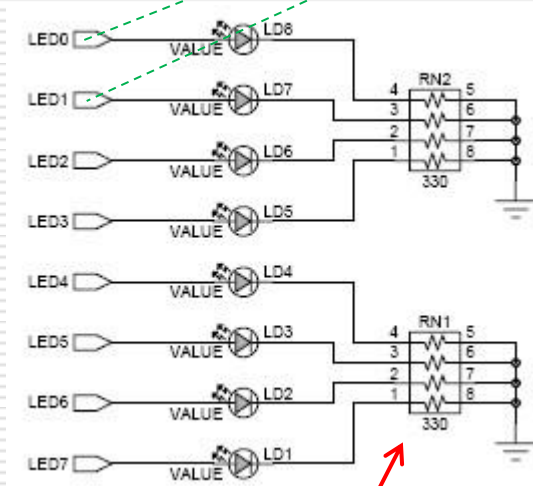
JKIT-128에서의 LED 연결 설계

□ JKIT-128-1에서의 LED 연결 설계 개념

- LED는 8개(1바이트)를 동일한 포트(PA)에 할당
- LED의 Anode 쪽에 신호를 연결하고 Cathode 쪽에 시리얼로 저항을 연결한 후 GND에 연결하는 방법 사용
- LED는 일반적인 특성(전압 1.7V, 10mA)을 갖는 LED로 함
- LED는 보드 크기를 고려하여 SMD LED로 하되, 크기는 조금 큰 3216 크기로 함
- 시리얼 저항은 8개 모두 동일한 저항값을 가지므로 동일한 저항 4개가 함께 묶여있는 어레이 저항을 사용
- 시리얼 저항값은 계산에 의하여 330 옴으로 함
- 추후 배치시에 가장 왼쪽편이 PA7, 가장 오른쪽편이 PA0 신호선에 연결되도록 주의하여야 함!

JKIT-128에서의 LED 연결 설계

□ JKIT-128-1 LED 회로도



Array 저항

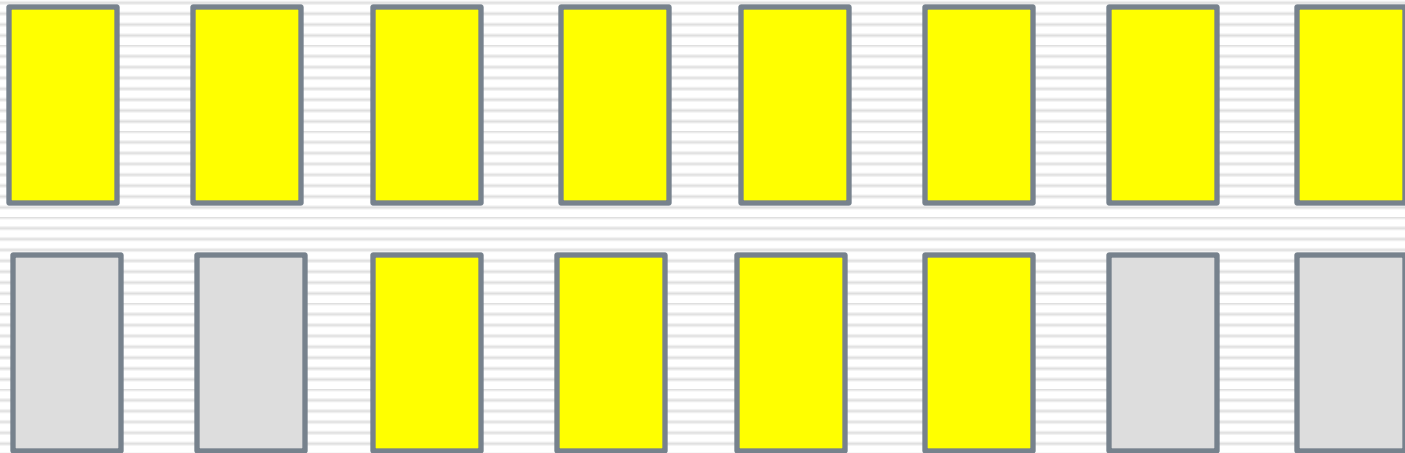
LED0	51	PA0_AD0	PE0_RXD0_FDI	2
LED1	50	PA1_AD1	PE1_TXD0_PDO	3
LED2	49	PA2_AD2	PE2_XCK0_AIN0	4
LED3	48	PA3_AD3	PE3_OC3A_AIN1	5
LED4	47	PA4_AD4	PE4_OC3B_INT4	6
LED5	46	PA5_AD5	PE5_OC3C_INT5	7
LED6	45	PA6_AD6	PE6_T3_INT6	8
LED7	44	PA7_AD7	PE7_ICP3_INT7	9
CDS	61	PF0_ADC0	PB0_SS	10
PF1	60	PF1_ADC1	PB1_SCK	11
PF2	59	PF2_ADC2	PB2_MOSI	12
PF3	58	PF3_ADC3	PB3_MISO	13
PF4	57	PF4_ADC4_TCK	PB4_OC0	14
PF5	56	PF5_ADC5_TMS	PB5_OC1A	15
PF6	55	PF6_ADC6_TDO	PB6_OC1B	16
PF7	54	PF7_ADC7_TDI	PB7_OC2_OC1C	17
FND0	35	PC0_A8	PEN*	1
FND1	36	PC1_A9	RESET*	20
FND2	37	PC2_A10	XTAL1	24
FND3	38	PC3_A11	XTAL2	23
FND4	39	PC4_A12	AVCC	64
FND5	40	PC5_A13	AREF	62
FND6	41	PC6_A14	VCC1	21
FND7	42	PC7_A15	VCC2	52
MP_SCL	25	PD0_SCL_INT0	GND1	22
MP_SDA	26	PD1_SDA_INT1	GND2	53
PD2	27	PD2_RXD_INT2	GND3	63
PD3	28	PD3_TXD1_INT3		
PD4	29	PD4_ICP1		
PD5	30	PD5_XCK1		
PD6	31	PD6_T1		
PD7	32	PD7_T2		
ID_SEL0	33	PG0_WR*		
ID_SEL1	34	PG1_RD*		
ID_SEL2	43	PG2_ALE		
ID_SEL3	18	PG3_TOSC2		
PG4	19	PG4_TOSC1		

ATMEGA128_16AU

실습 LED-1 : GPIO로 LED 켜고 끄기

□ 실습 내용

1. LED 모두 켜기
2. LED를 양쪽 가장자리 2개는 OFF, 가운데 4개는 ON 하기



실습 LED-1 : GPIO로 LED 켜고 끄기

- 구동 프로그램 설계 : LED 모두 ON (led_1_1.c)
 - LED를 켜기 위해서는 LED 신호에 '1'을 인가하여야 함 (끄기 위해서는 '0'을 인가하여야 함)
 - 즉, ATmega128 GPIO PA포트의 해당 비트에 '1'을 출력하여야 함
 - ATmega128 GPIO PA 포트의 특정 비트에 '1'을 출력하려면
 1. DDRA 레지스터의 해당 비트에 '1'을 write하여 방향을 '출력' 상태로 만들고,
 2. PORTA 레지스터의 해당 비트에 '1'을 write하면 됨

실습 LED-1 : GPIO로 LED 켜고 끄기

□ 구동 프로그램 설계 : LED ON (led_1_1.c)

1. DDRA를 모두 출력 방향으로 설정 (0xff write)



2. LED를 모두 켜야 하므로 0xff를 PORTA에 write
(모두 끄려면 0x00을 write)

실습 LED-1 : GPIO로 LED 켜고 끄기

□ 구동 프로그램 코딩 : LED ON (led_1_1.c)

/* GPIO로 LED 켜고 끄기 1번 예

LED 8개가 연결되어 있는 포트 : Port A(PA)

- 비트7 : LED7(LD1), 비트6 : LED6(LD2)

- ...

- 비트1 : LED1(LD7), 비트0 : LED0(LD8) */

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

// ATmega128 register 정의

```
int main()
```

```
{
```

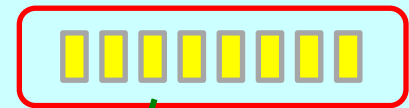
```
    DDRA = 0xff;
```

// 포트 A를 출력 포트로 사용

```
    PORTA = 0xff;
```

// 0xff = 0b11111111, LED 모두 ON

```
}
```



실습 LED-1 : GPIO로 LED 켜고 끄기

□ 구동 프로그램 코딩 : LED 가운데 4개 ON (led_1_2.c)

/* GPIO로 LED 켜고 끄기 1번 예

LED 8개가 연결되어 있는 포트 : Port A(PA)

- 비트7 : LED7(LD1), 비트6 : LED6(LD2)

- ...

- 비트1 : LED1(LD7), 비트0 : LED0(LD8) */

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

// ATmega128 register 정의

```
int main()
```

```
{
```

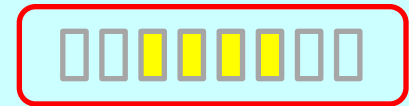
```
    DDRA = 0xff;
```

// 포트 A를 출력 포트로 사용

```
    PORTA = 0x3c;
```

// 0x3c = 0b00111100,

```
}
```



실습 LED-1 : GPIO로 LED 켜고 끄기

□ 더 해보기

1. LED를 왼쪽 4개는 켜고 오른쪽 4개는 끄기
2. LED를 1칸씩 건너가면서 켜기
3. LED를 내가 원하는 임의의 형태로 표현해 보기

실습 LED-2 : GPIO로 LED 움직임 표현하기

□ 실습 내용

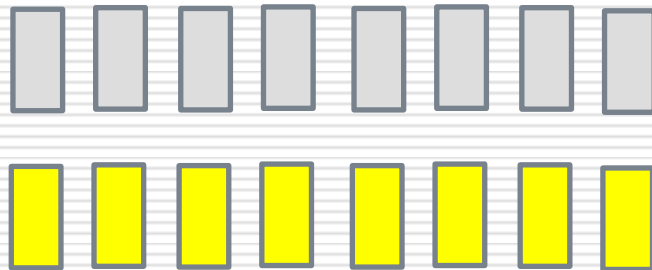
1. LED를 모두 약 1초 동안 ON, 약 1초 동안 OFF 반복하기
2. LED로 숫자 세기 (Binary Counter)

LED 구동 실례

- 소리에 반응하는 LED
- 8 x 8 LED 동작
- 3 x 3 x 3 LED 동작

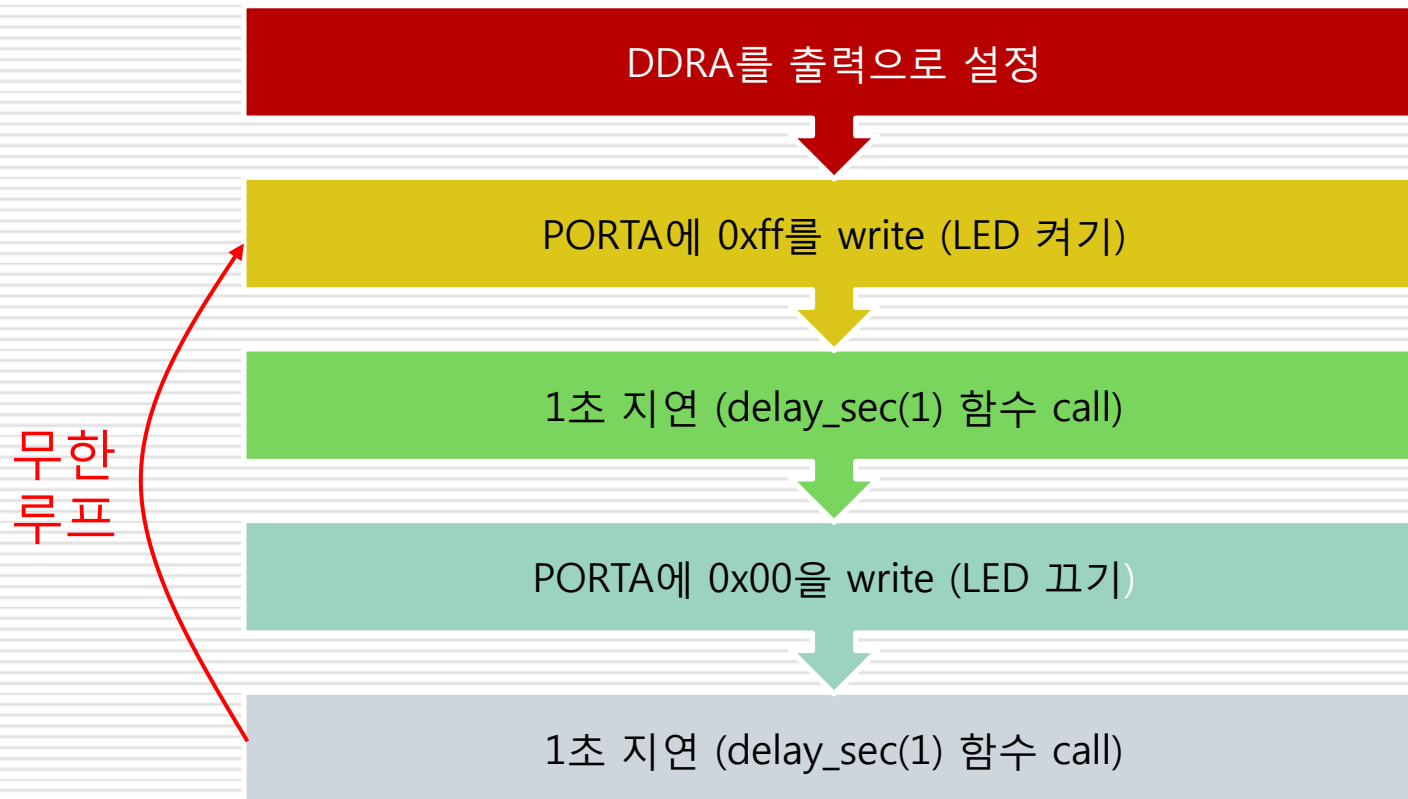
실습 LED-2 : GPIO로 LED 움직임 표현하기

- 구동 프로그램 설계 : LED ON/OFF (led_2_1.c)
 - LED를 1초 동안 켜기 위해서는 LED 신호에 '1'을 인가하고 1초 동안 기다리면 되고, 반대로 1초 동안 끄기 위하여는 LED 신호에 '0'을 인가하고 1초 동안 기다리면 됨
 - LED에 '1'이나 '0'을 인가하는 방법은 led_1_1.c에서 구현
 - 1초를 기다리는 것은 delay_sec() 함수를 call하여 구현 가능
 - ON-OFF를 반복하여야 하므로 위의 내용을 while(1) 무한루프로 수행하면 됨



실습 LED-2 : GPIO로 LED 움직임 표현하기

□ 구동 프로그램 설계 : LED ON/OFF (led_2_1.c)



실습 LED-2 : GPIO로 LED 움직임 표현하기

□ 초 단위 시간 지연 함수 delay_sec() 대충 구현하기

/* 1초 delay 함수 구현 : 중복(nested) for 문을 이용하고
적당한 값을 대입하여 차음 */

```
void delay_sec(int sec)
{
    volatile int i, j, k;
    for (i=0; i<sec; i++)
        for (j=0; j<1000; j++)
            for (k=0; k<1000; k++)
                ;
}
```

16Mhz 클럭이므로
1600만개 정도의
Assembly 명령어 수
행이면 약 1초가 걸림

총 100만번의 loop를
수행하는데 1번의
loop에 대충 16개 정
도의 명령어가 수행
된다고 가정하여 수
행해 보고 차이가 많
이 나면 숫자 조정

실습 LED-2 : GPIO로 LED 움직임 표현하기

□ 구동 프로그램 설계 : LED ON/OFF (led_2_1.c)

```
#include <avr/io.h>                // ATmega128 register 정의
int main()
{
    DDRA = 0xff;                    // 포트 A를 출력 포트로 사용
    while (1)
    {
        PORTA = 0xff;              // LED 모두 ON
        delay_sec(1);              // 1초 기다림
        PORTA = 0x00;              // LED 모두 OFF
        delay_sec(1);              // 1초 기다림    }
    }
```


volatile 선언에 대하여

□ volatile의 의미

- 컴파일러는 volatile로 선언된 변수나 포인터와 관련된 코드를 임의로 최적화하지 말라는 의미
- 사용되는 곳
 - 인터럽트 루틴과 메인 루틴에서 같은 변수를 사용하는 경우
 - 자동변수의 경우 일정시간 CPU에서 operation을 수행해 주기를 원하는 경우
 - 외부 버스 마스터와 특정 메모리를 공유하는 경우
 - 코드의 순서를 지키고 싶은 경우
 - I/O 레지스터 등을 메모리맵에 매핑시켜서 사용하는 경우

volatile 선언에 대하여

□ volatile의 사용 예

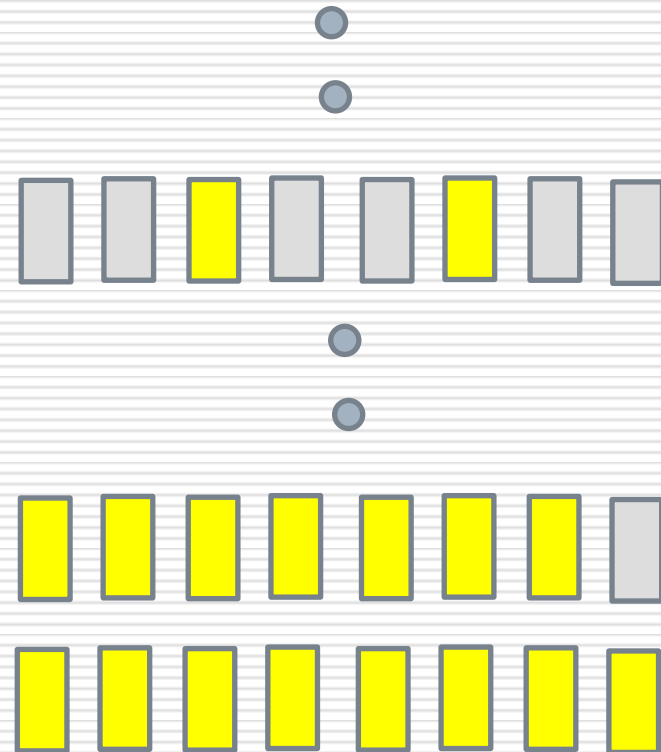
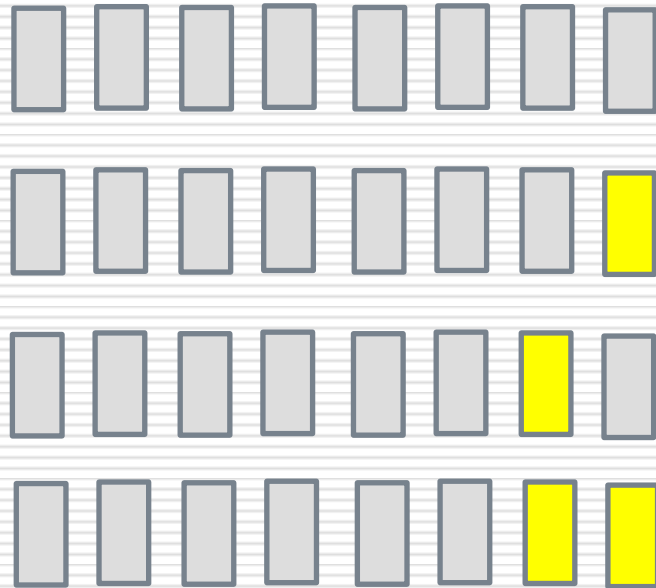
```
volatile int    i, a, b;  
a = 0;  
b = 0;  
for (i=0; i<10; i++)  
    b += a * 100    // volatile이 없다면 a = 0 이므로  
                   // b += 0 으로 최적화
```

```
volatile int    a;  
a = 10;          // volatile이 없다면 이 코드는  
                // 최적화로 인하여 삭제될 수 있음  
  
a = 20;
```

실습 LED-2 : GPIO로 LED 움직임 표현하기

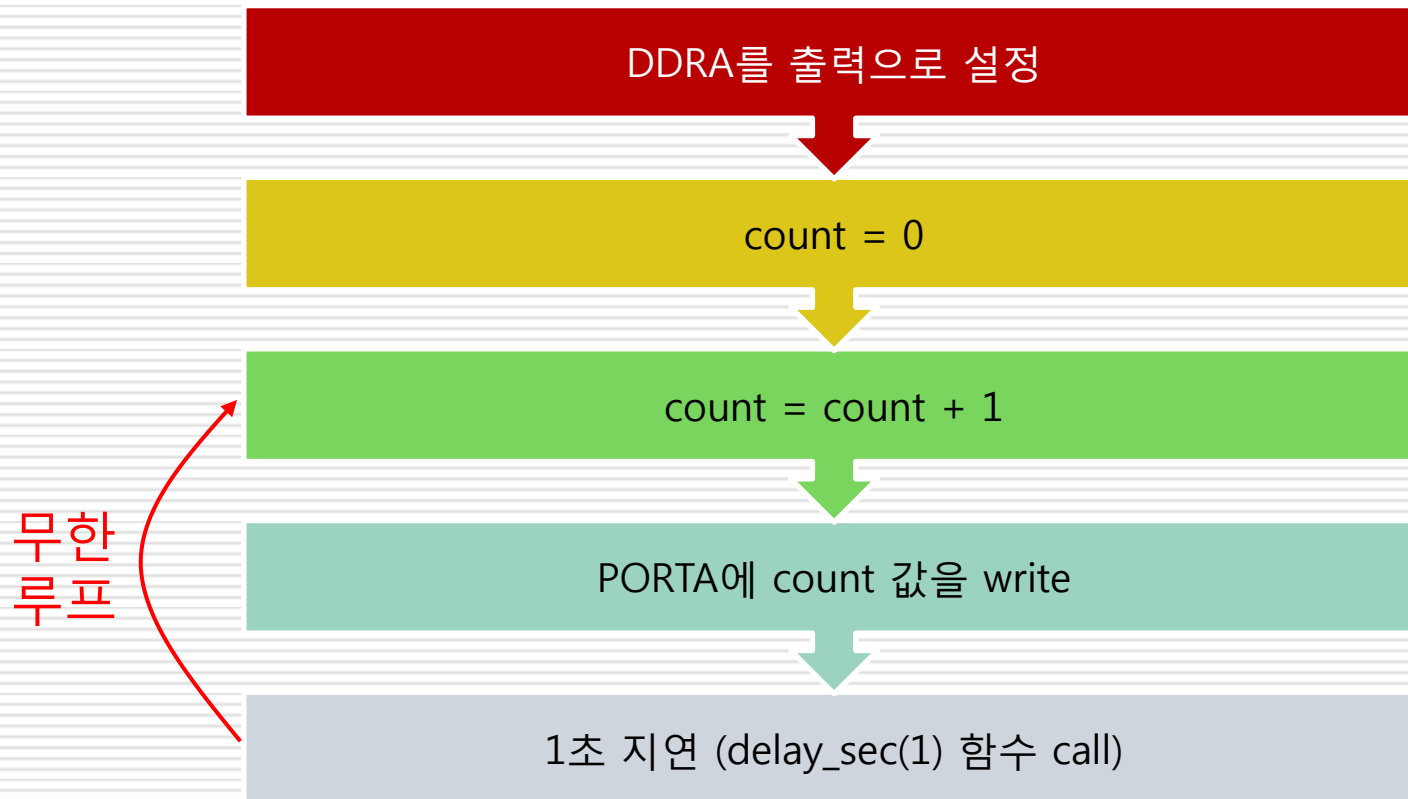
- 구동 프로그램 설계 : Binalry Counter (led_2_2.c)
 - LED로 숫자를 세려면 숫자를 0 부터 1씩 증가시키면 되므로 1초마다 1씩 증가된 값을 PORTA로 보내면 됨
 - 즉, count 값을 초기에 0으로 설정한 상태에서 $count = count + 1$ 로 하여 이를 PORTA로 보내고 일정시간 동안 기다린 다음 다시 $count = count + 1$ 하는 과정을 계속 무한 반복 수행하면 됨

Binary Counter



실습 LED-2 : GPIO로 LED 움직임 표현하기

□ 구동 프로그램 설계 : Binary Counter (led_2_2.c)



실습 LED-2 : GPIO로 LED 움직임 표현하기

□ 구동 프로그램 코딩 : Binary Counter (led_2_2.c)

```
/* GPIO로 LED 움직임 표현하기 : Binary Counter 구현 */

#include <avr/io.h>                                // ATmega128 register 정의
void delay_sec(int sec)
{
    volatile int i, j, k;
    for (i=0; i<sec; i++)
        for (j=0; j<1000; j++)
            for (k=0; k<1000; k++)
                ;
}
```

실습 LED-2 : GPIO로 LED 움직임 표현하기

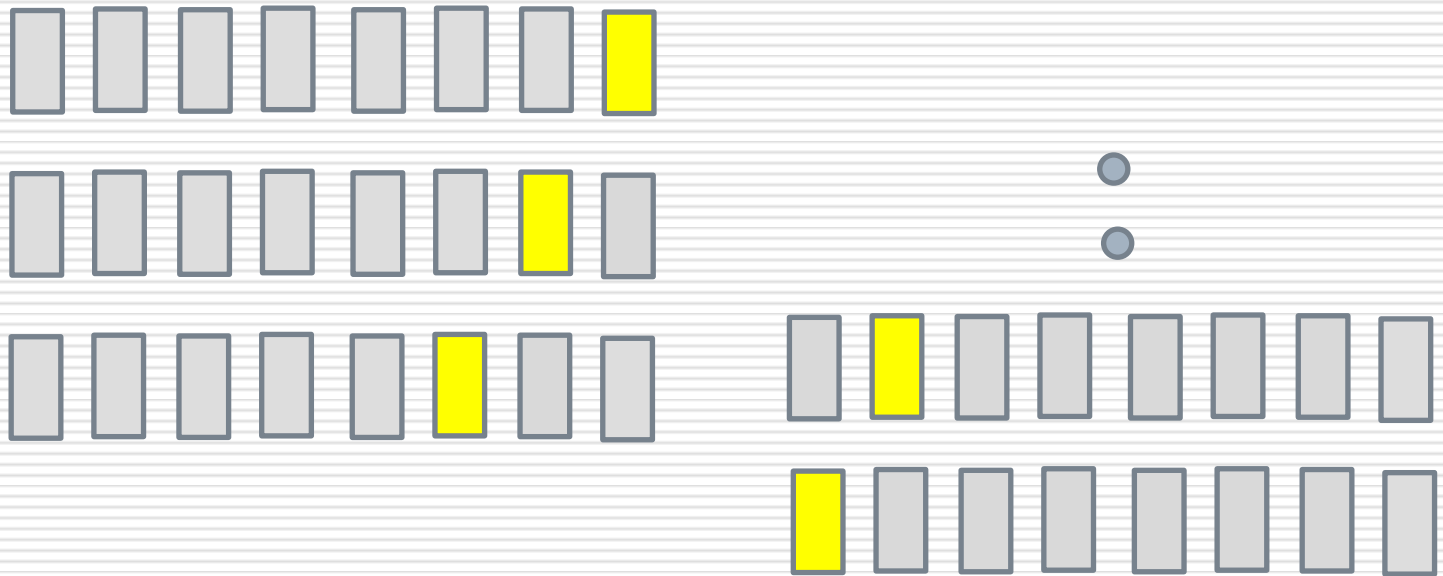
□ 구동 프로그램 코딩 : Binalry Counter (led_2_2.c)

```
int main( )
{
    unsigned char count=0; // LED용 count는 양수 1 바이트
                           // count 초기값 = 0
    DDRA = 0xff;           // 포트 A를 출력 포트로 사용
    while(1)               // 무한루프 실행
    {
        count = count + 1; // count 1 증가
        PORTA = count;     // LED에 숫자 표시
        delay_sec(1);      // 1초 기다림
    }
}
```

실습 LED-2 : GPIO로 LED 움직임 표현하기

□ 더 해보기

1. LED를 맨 오른쪽 1개만 켜고 이를 1초에 한칸씩 왼쪽으로 이동시키기 (Ring Counter)



실습 LED-2 : GPIO로 LED 움직임 표현하기

- 구동 프로그램 설계 : Ring Counter (led_2_3.c)
 - LED의 초기값은 1이고 $1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 16 \rightarrow 32 \rightarrow 64 \rightarrow 128 \rightarrow 1$ 의 순으로 1초마다 변하면 됨
 - 즉, LED의 값을 갖는 변수를 data라고 하면 data 값이 1초마다 2배씩 증가하도록 하고 ($\text{data} = \text{data} * 2$) 이를 PORTA에 write하면 됨
 - 다만, data 값이 128인 경우를 검사하여 이 때는 $128 \rightarrow 256$ 으로 되는 것이 아니라 $128 \rightarrow 1$ 이 되도록 data 값을 조정하여야 함 (왜냐하면 128이 마지막 위치)
 - 이러한 과정을 계속 무한 반복 수행하면 됨

실습 LED-3 : LED로 X-mas 트리 만들기

□ 실습 내용

1. LED가 X-mas 트리의 전구 역할을 수행하도록 프로그램 (X-mas Tree)

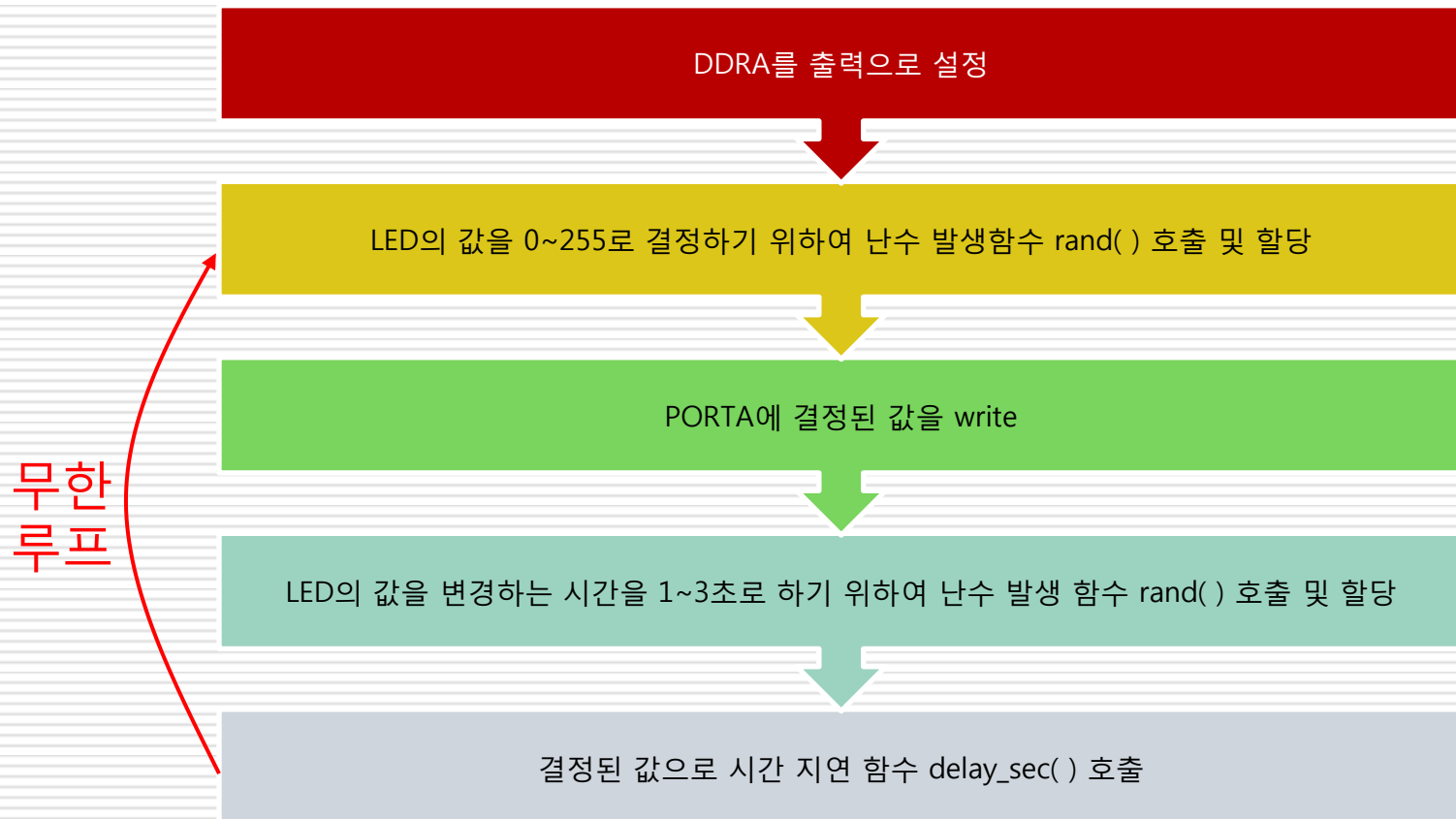
- X-mas 트리처럼 다양한 형태의 ON/OFF 제공
- 가능한 Random한 형태의 LED 로 표현되면 좋음
(난수 발생 함수인 srand() 와 rand() 를 이용하는 것도 좋은 방법임)
- 점등 속도까지 조절되면 더욱 좋음 (1초, 2초, 3초)

실습 LED-3 : LED로 X-mas 트리 만들기

- 구동 프로그램 설계 : X-mas Tree (led_3.c)
 - LED를 형태를 결정하는 값을 0~255의 Random한 값을 갖도록 하기 위하여 C에서 제공하는 라이브러리인 rand() 함수를 사용하여 결정
 - LED의 값을 변경하는 시간을 1~3초로 Random한 값을 갖도록 하기 위하여 이것도 C에서 제공하는 라이브러리인 rand() 함수를 사용하여 결정
 - ON-OFF를 반복하여야 하므로 위의 내용을 while(1) 무한루프로 수행하면 됨

실습 LED-3 : LED로 X-mas 트리 만들기

□ 구동 프로그램 설계 : X-mas Tree (led_3.c)



실습 LED-3 : LED로 X-mas 트리 만들기

□ 구동 프로그램 코딩 : X-mas Tree (led_3.c)

```
/* LED로 X-mas Tree 만들기 */

#include <avr/io.h>                // ATmega128 register 정의
#include <stdlib.h>                // rand(), srand() 함수
void delay_sec(int sec)
{
    volatile int i, j, k;
    for (i=0; i<sec; i++)
        for (j=0; j<1000; j++)
            for (k=0; k<1000; k++)
                ;
}
```

delay_ms()
함수 만들기

실습 LED-3 : LED로 X-mas 트리 만들기

□ 구동 프로그램 코딩 : X-mas Tree (led_3.c)

```
int main( )
{
    DDRA = 0xff;           // 포트 A를 출력 포트로 사용
    while(1)               // 무한루프 실행
    {
        PORTA = rand( )%256; // 0~255 난수 발생 및
        LED 표시
        delay_sec(rand()%3);  // 0~2 난수 발생 및 시
        간 지연
    }
}
```

숙제

1. 제출 내용

- a. LED를 맨 오른쪽 1개만 켜고 이를 1초에 한칸씩 왼쪽으로 이동시키고 왼쪽 끝에 도달하면 다시 오른쪽으로 한칸씩 이동시키는 프로그램 작성 (Boundary Detector) (힌트 : 왼쪽으로 이동은 <<, 오른쪽으로 이동은 >> 사용)
- b. LED로 스피커 볼륨처럼 1개 ON → 2개 ON → ... → 8개 ON → 7개 ON → ... → 2개 ON → 1개 ON → 2개 ON → ... 이렇게 반복하여 수행하는 프로그램 작성 (힌트 : 점점 커질 때는 << 실행 후 +1 하고, 점점 작아질 때는 >> 실행)
- c. 자기만의 창작 LED 움직임을 보이는 프로그램 작성
- d. 자신이 만든 프로그램은 동작 설명과 사진 및 소감 첨부

2. 제출 기한 : 다음주 수업시간 종료전까지

3. 제출 방법 : e-Class에 "학번-이름-LED.zip" 파일로 제출

문고 답하기

Q & A

