SWITCH 읽기

OneKick

- 기초 학습
  - 제어구조
- Switch 읽기
  - 개 요
  - 예 제(1)
  - 예 제(2)
- 응용 실습

Basic learning

기초 학습

제어 구조

- if
  - 조건부에 의해 참이면 동작
  - if(조건부)의 형태로 되어 있으며 조건부가 참이면 동작

```
if(x>50)
{
  // 조건부가 참일 때 실행
}
```

제어 구조

OneKick

- if - else
  - 조건부가 참이면 if 문을 실행
  - 거짓이면 else 문을 실행

```
if(x>50)
{
  // 조건부가 참일 때 실행
}
else
{
  // 조건부가 거짓일 때 실행
}
```

제어 구조

OneKick

- 조건부가 2개 이상인 경우
  - 조건A를 판단하여 참이면 action A를 실행
  - 거짓인 경우에는 다시 조건B를 판단
    - 참이면, action B를 실행
    - 거짓이면, action C를 실행
  - "else if" 문은 여러 개가 반복적 사용 가능

```
if(조건A)
{
  // action A
}
else if(조건B)
{
  // action B
}
else
{
  // action C
}
```

제어 구조

OneKick

- for
  - for문은 중괄호로 묶인 블록을 반복하는데 사용
  - 증분 카운터는 루프의 증가 및 종료하기 위한 조건에 사용
  - 횟수가 결정된 반복 작업에 유용
  - 종종 데이터/핀의 집합에서 작동하도록 배열과 함께 사용

```
for(initialization; condition; increment or decrement)
{
  // action
}
```

제어 구조

OneKick

- initialization은 처음 한 번만 실행
  - 반복할 때마다 condition으로 루프 실행 여부 판단
  - 판단된 조건이 참이면 increment or decrement에 의해 증가/감소 실행
  - 다음 조건을 다시 판단
  - 조건이 거짓이면 루프가 종료

- for문이 사용되는 예
  - i를 0으로 초기화하고 i가 255보다 작거나 같으면 계속 실행
  - 실행할 때마다 i값을 증가
  - i 값이 255보다 커지면 루프 종료

```
for(i=0; i<=255; i++)
{
  analogWrite(PWMpin, i);
}
```

제어 구조

OneKick

- switch-case
  - if문과 유사한 프로그램의 흐름을 제어문
  - case에 지정된 값을 switch 문의 변수 값과 비교
  - case의 값이 변수와 일치하면 해당 case 문에서 코드가 실행
  - break 키워드
    - switch 문을 종료하는 역할
    - 일반적으로 각각의 case 문의 끝부분에 사용
    - break 키워드가 없다면 switch 문이 중단 될 때까지 다음을 실행하거나 switch 문의 끝부분에 도달

```
switch(var)
{
  case label1:
    // statements
    break;
  case label2:
    // statements
    break;
  default:
    // statements
}
```

제어 구조

OneKick

- while
  - 괄호 안의 조건이 거짓이 될 때까지 무한히 계속 루프를 반복 실행
  - expression을 판단하여 거짓이 되기 전까지 루프 반복

```
while(expression)
{
    // statements
}
```

제어 구조

OneKick

- do - while
  - do-while은 while 문과 유사
  - 다른 점은 조건식 판단을 언제 하느냐의 차이
    - while 문은 먼저 조건식을 판단하여 루프가 실행
    - do-while 문은 처음 한번 루프 실행
  - 이후는 while 문과 동일

```
do
{
    // statements
}while(expression);
```

제어 구조

OneKick

- break
  - do-while, while, for문의 루프 조건 종료
  - switch문 종료

- continue
  - 반복문(do-while, while, for)의 현재 반복의 나머지 부분을 건너 뜀
  - 루프의 처음으로 이동
  - 루프의 조건식을 검사 후 반복 진행 계속

- return
  - 함수에서 사용
  - 함수를 종료하고 결과 값 반환

```
return;
or
return value;
```

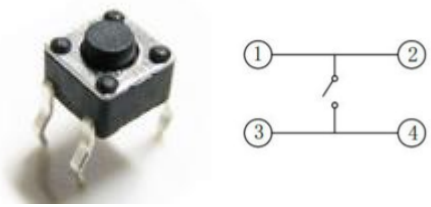
Read a switch

SWITCH 읽기

개요

- 목적
  - 디지털 입력(Digital In)의 개념 이해
  - 푸쉬버튼 스위치 동작 이해
  - Uno Board는 0.6xVcc(0.6x5=3V) ~ Vcc+0.5 까지 'High'로 인식
  - 0.5 ~ 0.3xVcc(0.3x5=1.5V) 까지 'Low'로 인식

- 관련이론
  - Tact switch : "버튼" 또는 "Key"라고 불리고, 1회로 1접점의 누르면 "ON", 손을 떼면 "OFF" 방식의 회로구성
  - Push button switch : 누름 스위치라고 불리고, PC의 전원, 자동차 등 누름방식의 스위치로써 Locking Type과 Nonlock Type으로 분류



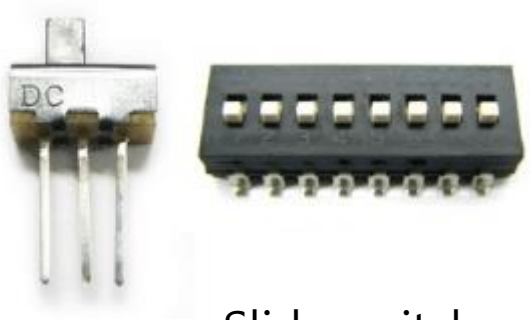
Tact switch

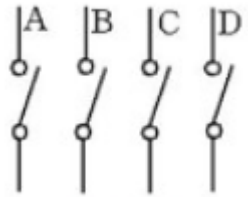
Push button switch

개요

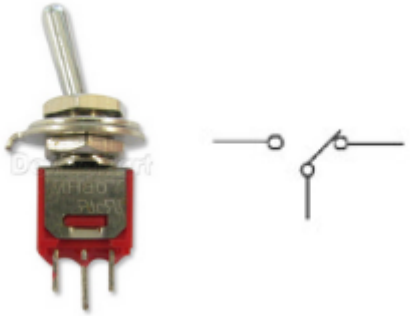
- Slide switch
  - 사이즈와 용량의 따라 작은사이즈는 MP3의 Hold용으로 사용되고, 큰사이즈는 자동, 수동 변환 또는 회로 변환 등으로 사용

- Toggle switch
  - 레버를 움직임으로써 ON/OFF 상태를 변경할 수 있는 방식



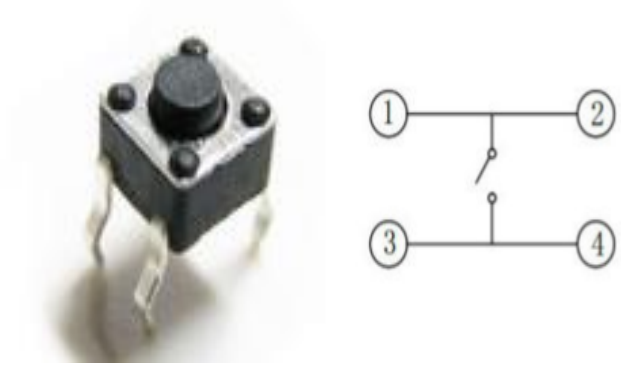


Slide switch

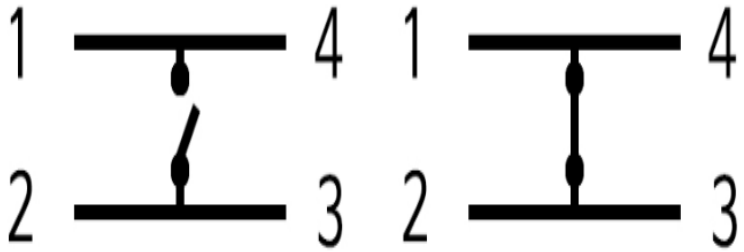


Toggle switch

스위치로 LED 켜기



- 1-2, 3-4, 1-3, 2-4 의 회로를 사용해야 연결 됨



스위치로 LED 켜기

OneKick

- 준비물
  - 아두이노 & 케이블
  - 스위치
  - LED
  - 220옴 저항
  - 점퍼케이블
  - 브레드보드



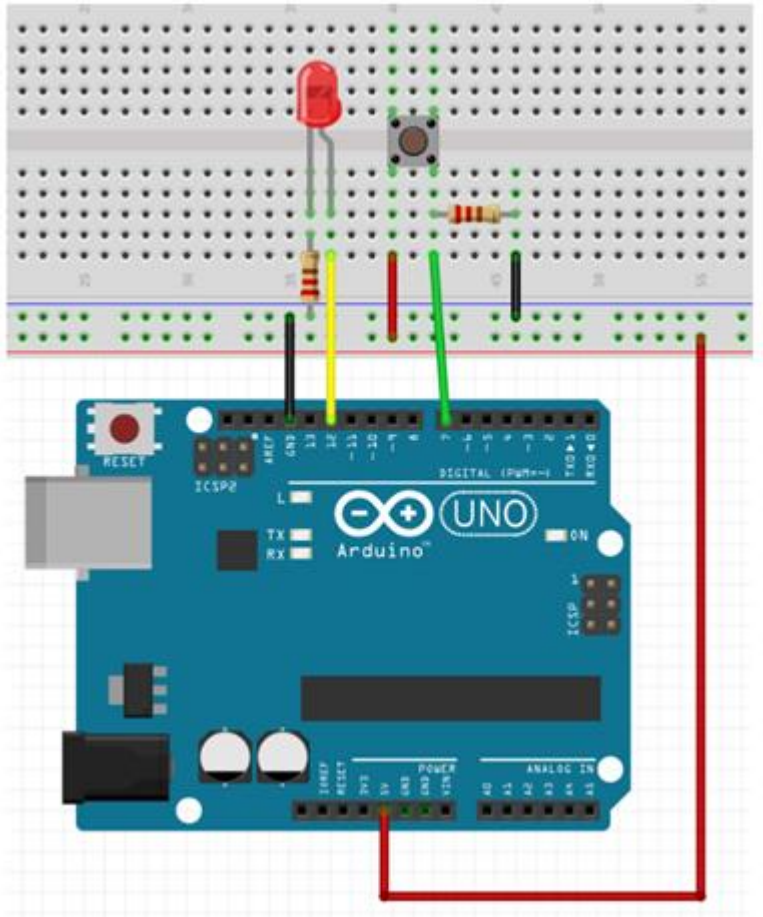





스위치로 LED 켜기(회로도)

- 스위치 한쪽 D7-220옴-GND연결
- 스위치 한쪽 VCC연결
- LED(+)-D11, LED(-)-220옴-GND 연결

- 스위치(D7) 값이 없을때는 LED(D10)는 LOW상태(LED꺼짐)



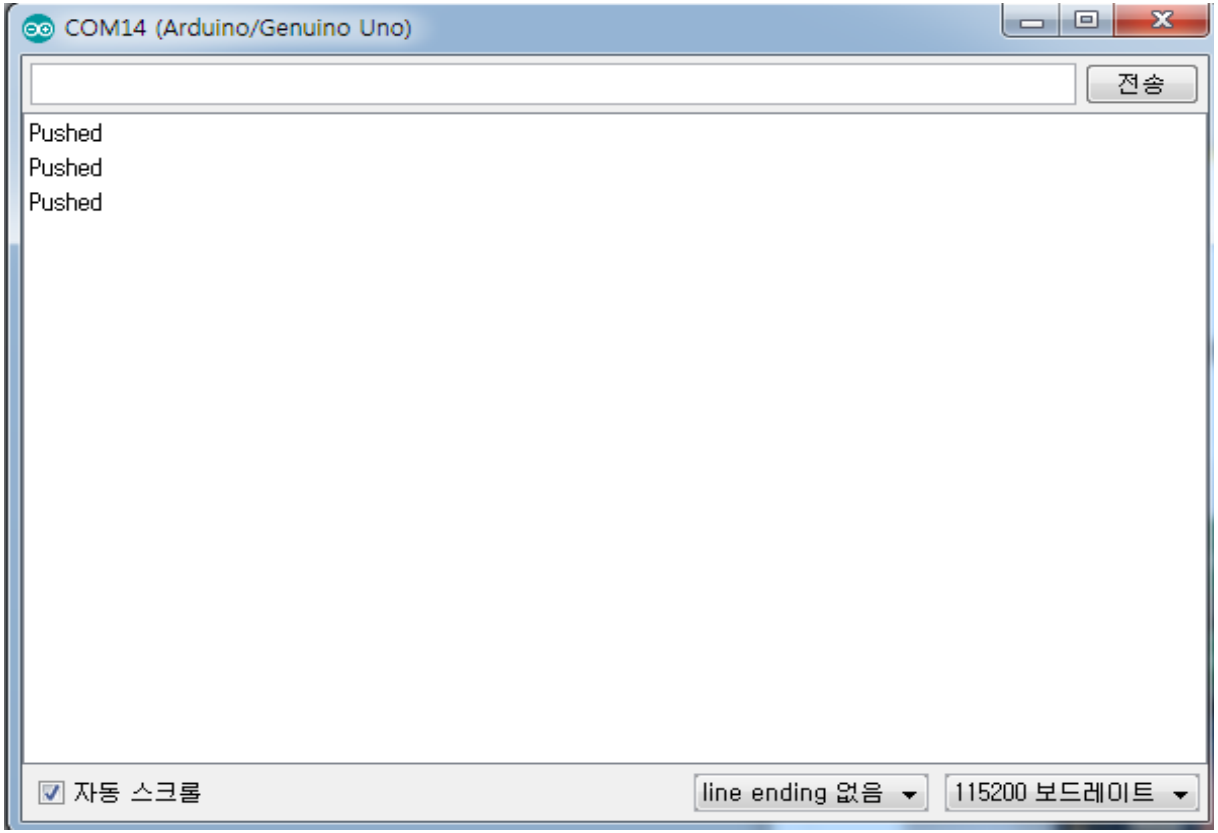
스위치로 LED 켜기(코드)

OneKick

```
int ledpin = 11;      //디지털11 LED
int inpin = 7;        //디지털7 버튼
int val;
void setup() {

  pinMode( ledpin, OUTPUT); //LED출력
  pinMode( inpin, INPUT);     //버튼 입력
}

void loop() {
  val = digitalRead( inpin );    //val 버튼 입력 정의

  if( val == LOW )                  //버튼 입력이 LOW이면
          digitalWrite(ledpin, LOW);      //LED 꺼짐
  else                                  //그렇지 않으면
          digitalWrite(ledpin, HIGH);     //LED 켜짐

}
```

예제(1)

- 예제
  - 버튼을 누르면 시리얼 모니터에 "Pushed"를 출력하시오.



예제(1)

- 프로그램 설명
  - 사용할 핀 정의

```
int pinSwitch = 13;
```

  - 초기화 구문

Serial port(UART 0)를 전송속도 115200, 데이터 비트 8, 패리티 없음, 스톱비트 1로 설정
Switch 핀을 입력으로 설정

```
Serial.begin(115200);

pinMode(pinSwitch, INPUT);
```

예제(1)

- 프로그램 설명

  - loop 구문

Switch 값을 읽어 1이면 PUSH 출력하고 0.5초 지연

```
if(digitalRead(pinSwitch))
{
   Serial.println(“Pushed”);
   delay(500);
}
```

예제(1)

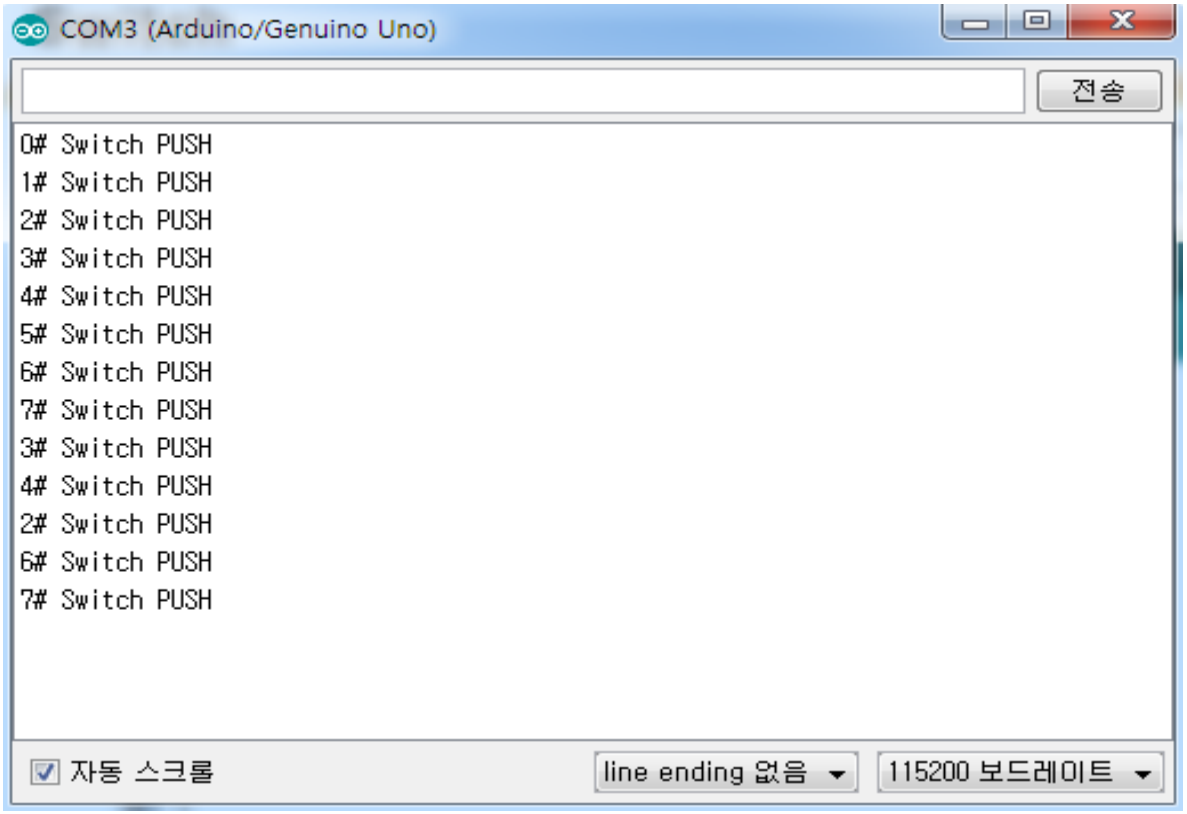
OneKick

- 전체 소스코드

```
1. int pinSwitch = 13;
2.
3. void setup() {
4.   Serial.begin(115200);
5.   pinMode(pinSwitch, INPUT);
6. }
7.
8. void loop() {
9.   if(digitalRead(pinSwitch)) {
10.    Serial.println("Pushed");
11.    delay(500);
12.  }
13.}
```

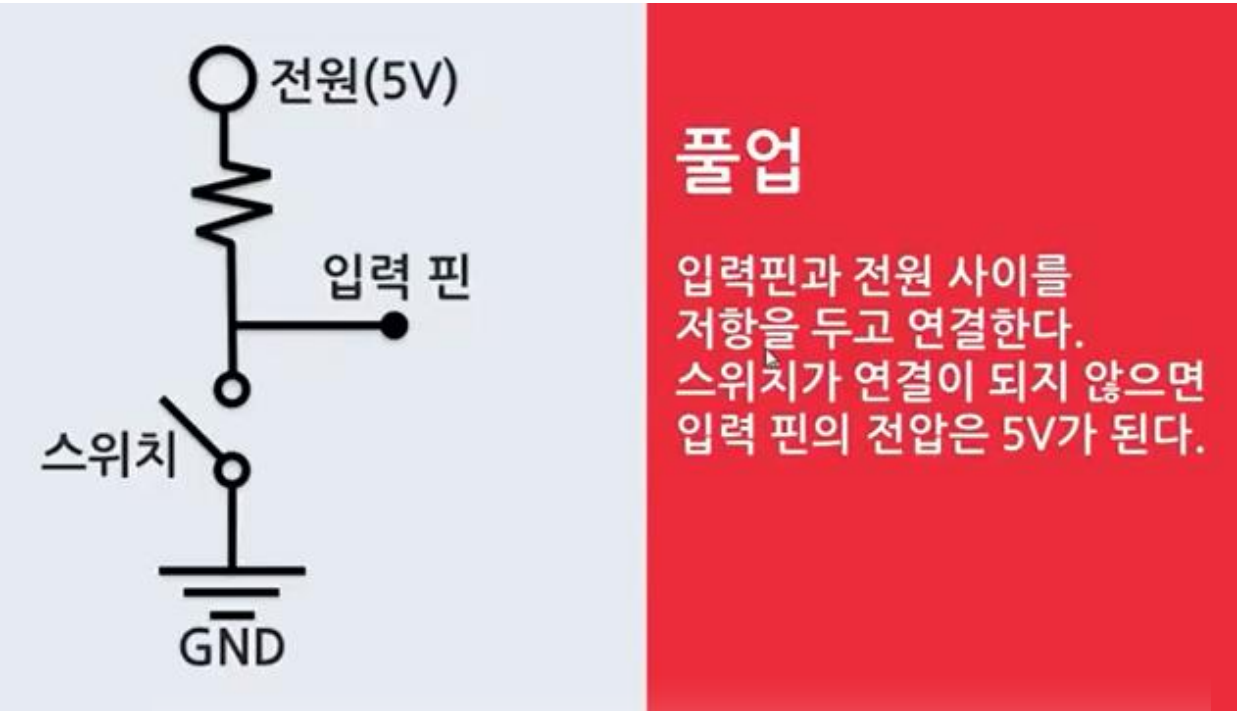
예제(2)

- 예제
  - 임의의 버튼을 누르면 시리얼 모니터에 해당 switch의 번호와 “Switch PUSH”를 출력하시오.



플로팅 상태(풀업)

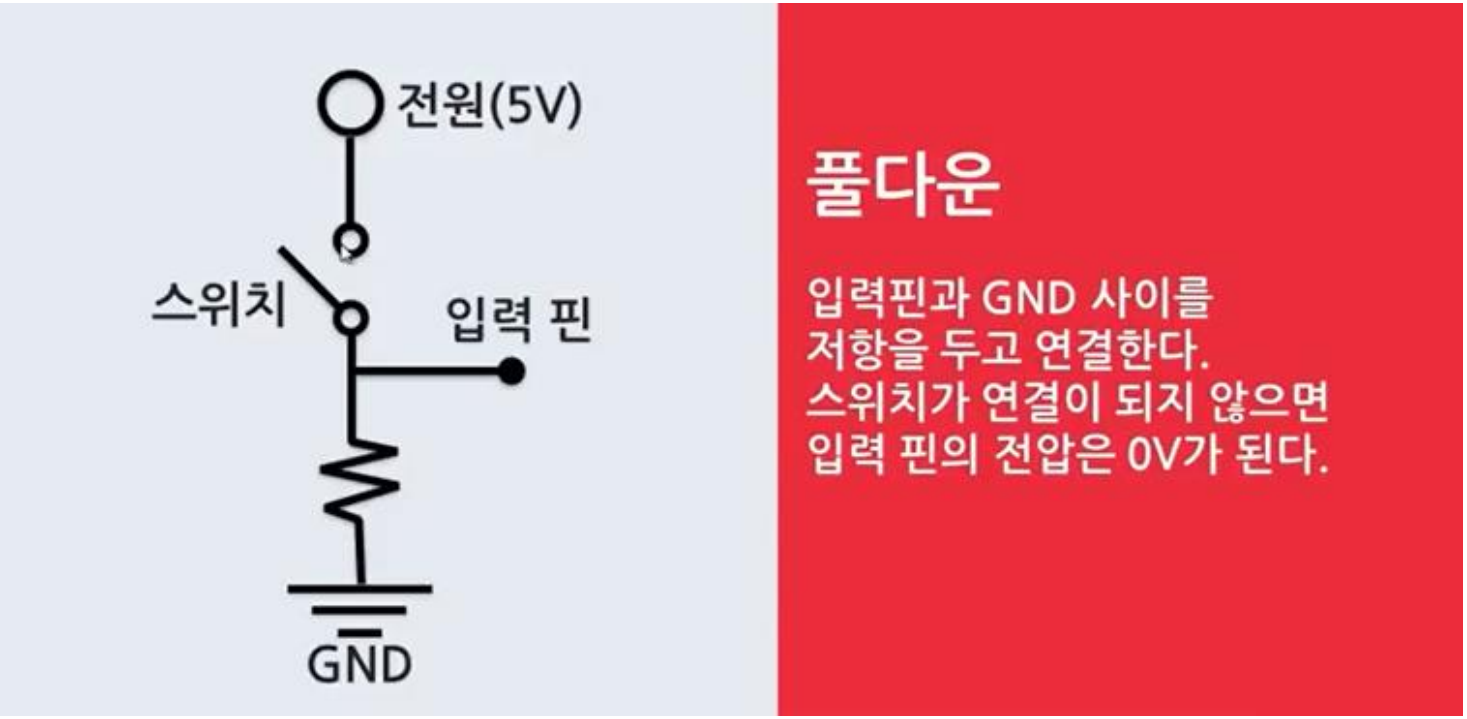
- 디지털 핀을 입력으로 설정하면 전압이 LOW와 HIGH를 왔다갔다하는 플로팅상태가 됨
- 플로팅 상태를 방지하기 위해 입력 핀의 전압을 고정해야 함
- 버튼을 눌렀을때 전력제어가 제대로 되지 않는 상태



스위치를 연결하면 입력 핀의 전압은 0v가 됨

플로팅 상태(풀다운)

- 저항을 앞이나 뒤에 붙여서 전기의 상태를 체크할 수 있음
- 버튼 사용 시 풀업, 풀 다운처럼 저항을 앞 뒤로 붙여줘야 버튼을 정확하게 제어 가능



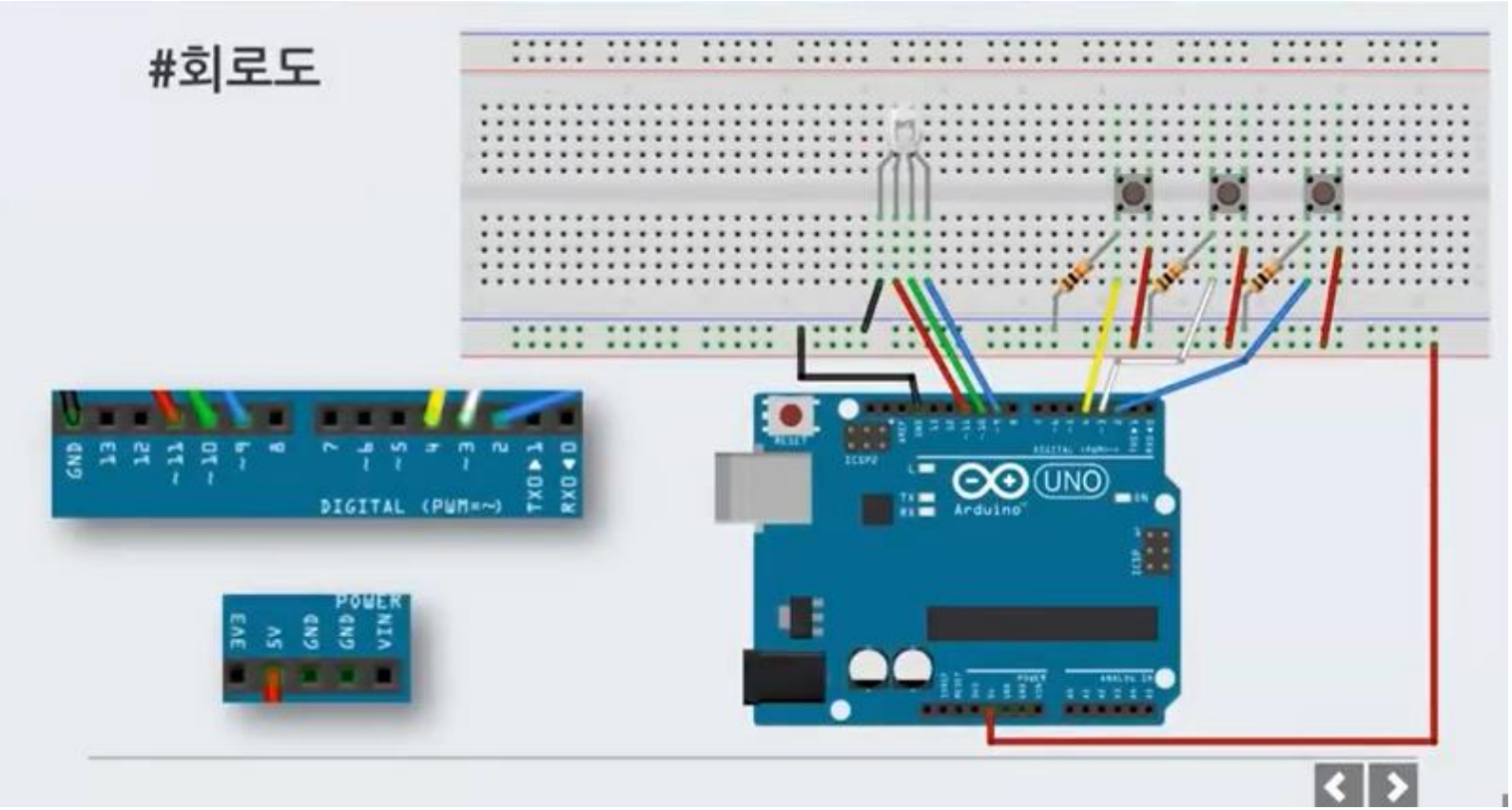
스위치를 연결하면 입력 핀의 전압은 5v가 됨

도전 과제

OneKick

- 버튼방식을 풀업, 풀 다운으로 구현
- 버튼 여러 개, led 여러 개 연결해보자. 해당 버튼을 누르면 해당 led 켜지게 실습

버튼사용하여 삼색led 사용



버튼사용하여 삼색led (코드) – 1/2

OneKick

```
#define RED 11
#define GREEN 10
#define BLUE 9
#define RED_BUTTON  4
#define GREEN_BUTTON 3
#define BLUE_BUTTON 2

int r = 0, g = 0, b = 0;

void setup(){
        pinMode(RED_BUTTON, INPUT);
        pinMode(GREEN_BUTTON, INPUT);
        pinMode(BLUE_BUTTON, INPUT);
}
void loop(){
        if( digitalRead(RED_BUTTON) == HIGH){
                ++r;
                if(r > 255){
                        r=0;
                }
        }
```

버튼사용하여 삼색led (코드) – 2/2

OneKick

```
        if( digitalRead(GREEN_BUTTON) == HIGH){
                ++g;
                if(g > 255){
                        g = 0;
                }
        }
        if( digitalRead(BLUE_BUTTON) == HIGH){
                ++b;
                if(b > 255){
                        b = 0;
                }
        }
        analogWrite( RED, r );
        analogWrite( GREEN, g );
        analogWrite( BLUE, b);
        delay( 10 );
]
```

응용실습

응용실습

- 인터넷에서 원하는 색의 rgb값을 찾아 색 바꾸기
- 신호등 만들기
- 버튼을 추가하여 색의 값을 감소시키기
- (rgb에서 동일하게 증가하면 회색이 됨)