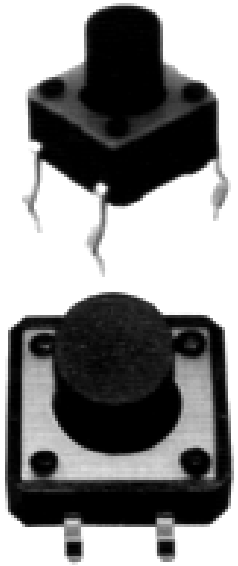
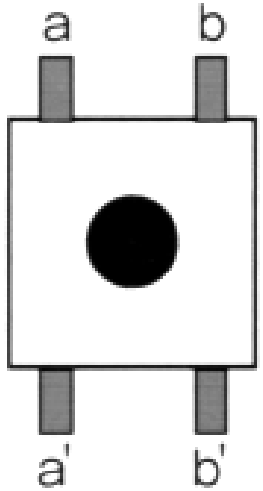
스위치 다루기

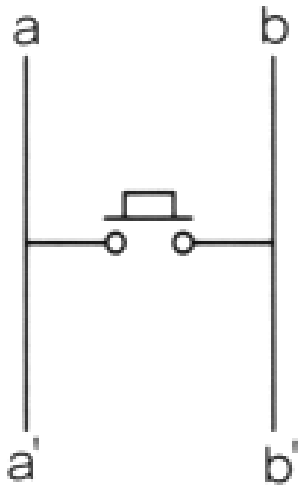
스위치 다루기

❖ 택트(Tact) 스위치

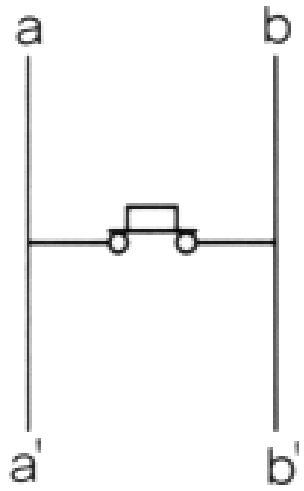


(a) 외관





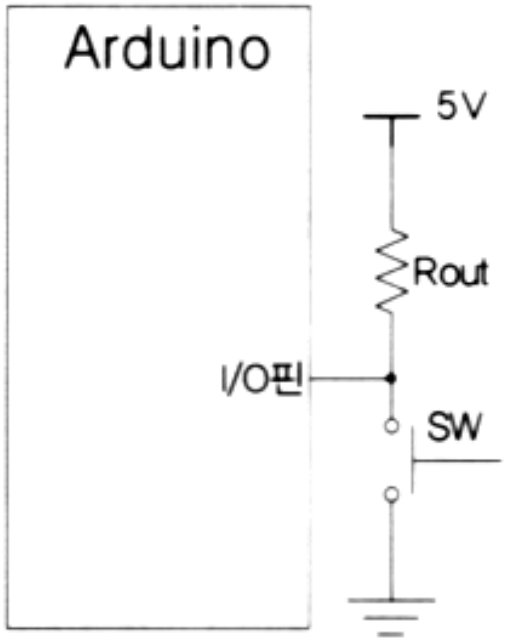
(b) 내부 연결도



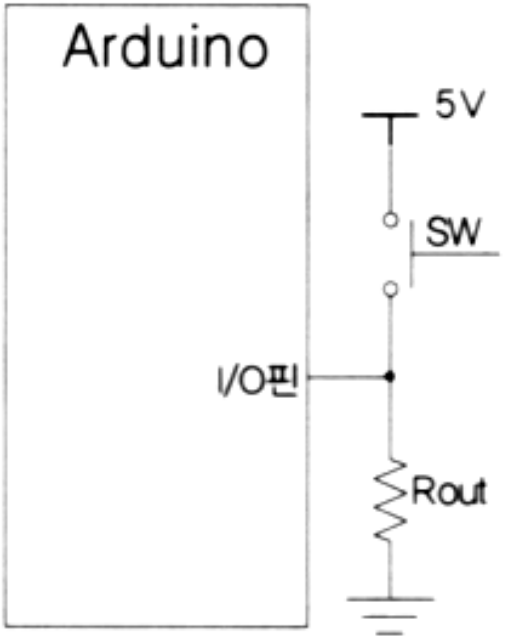
(c) on 상태

스위치 다루기

❖ 스위치 연결 방법



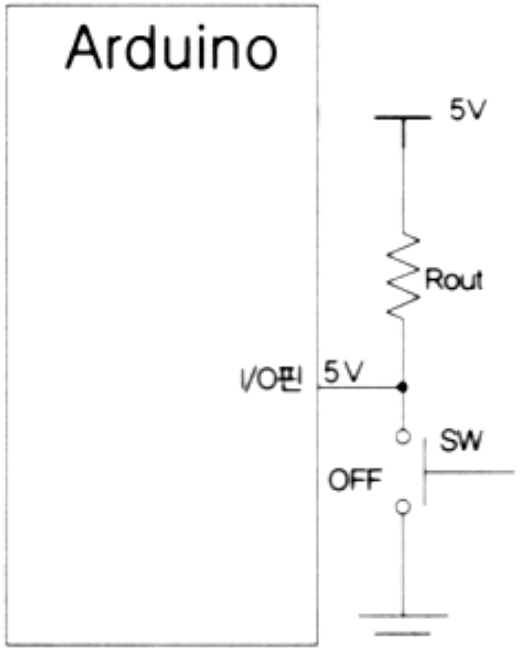
(a) 풀업 방식



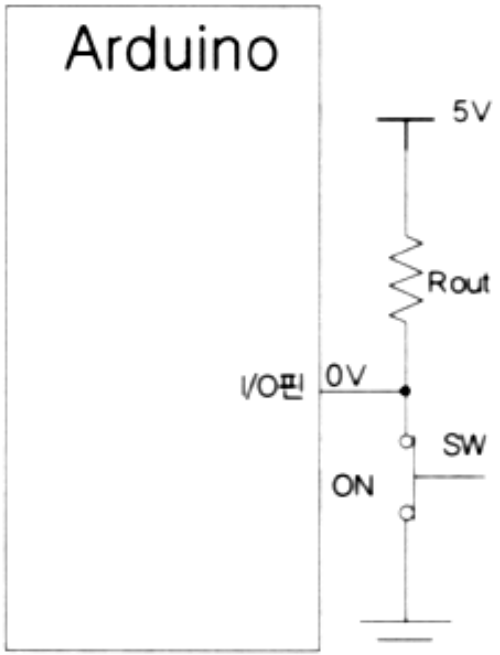
(b) 풀다운 방식

스위치 다루기

❖ 풀업 방식 스위치 on, off 상태



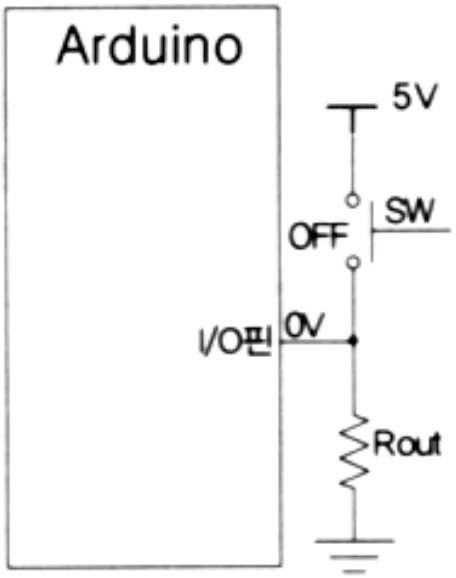
(a) 스위치 OFF



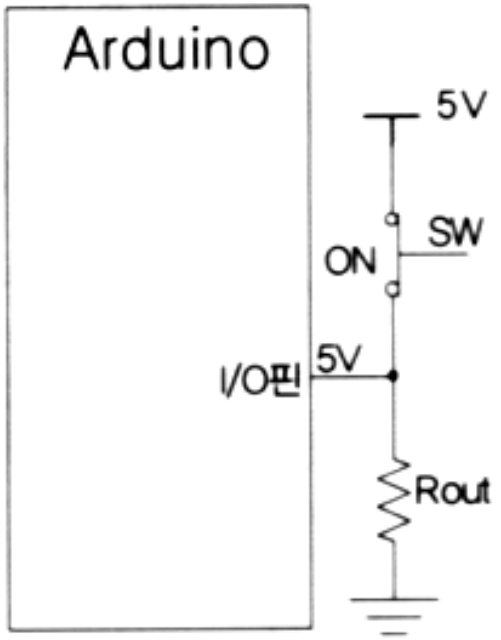
(b) 스위치 ON

스위치 다루기

❖ 풀다운 방식 스위치 on, off 상태



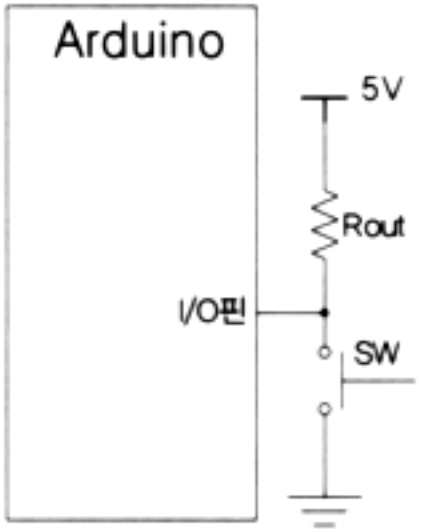
(a) 스위치 OFF



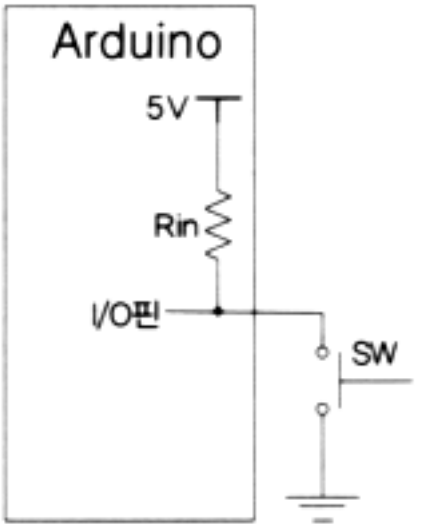
(b) 스위치 ON

스위치 다루기

❖ 내부 풀업 저항 사용 시 I/O



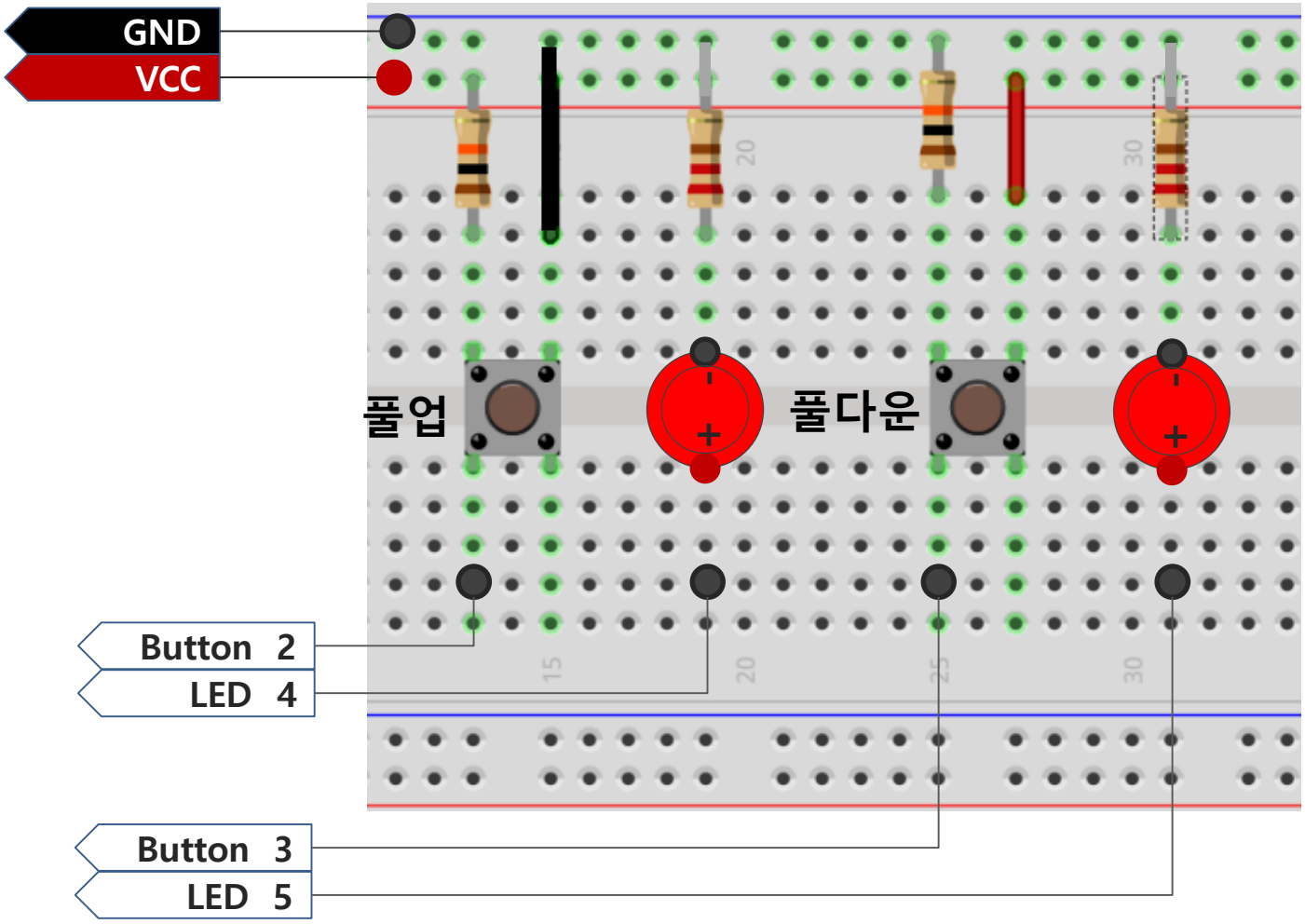
(a) 외부 풀업 연결



(b) 내부 풀업 연결

실습1: 스위치 풀업/풀다운 동작 화인

❖ 회로도



실습1: 스위치 풀업/풀다운 동작 화인

❖ ex01/app.ino

```
// 스위치 풀업/풀다운 동작 확인
#include <Led.h>

const int pd_sw_pin = 2;
Led led1(4);

const int pu_sw_pin = 3;
Led led2(5);

void setup()
{
  pinMode(pd_sw_pin, INPUT);  //  풀다운 스위치 연결핀 입력 설정
  pinMode(pu_sw_pin, INPUT);  //  풀업 스위치 연결핀 입력 설정
}
```

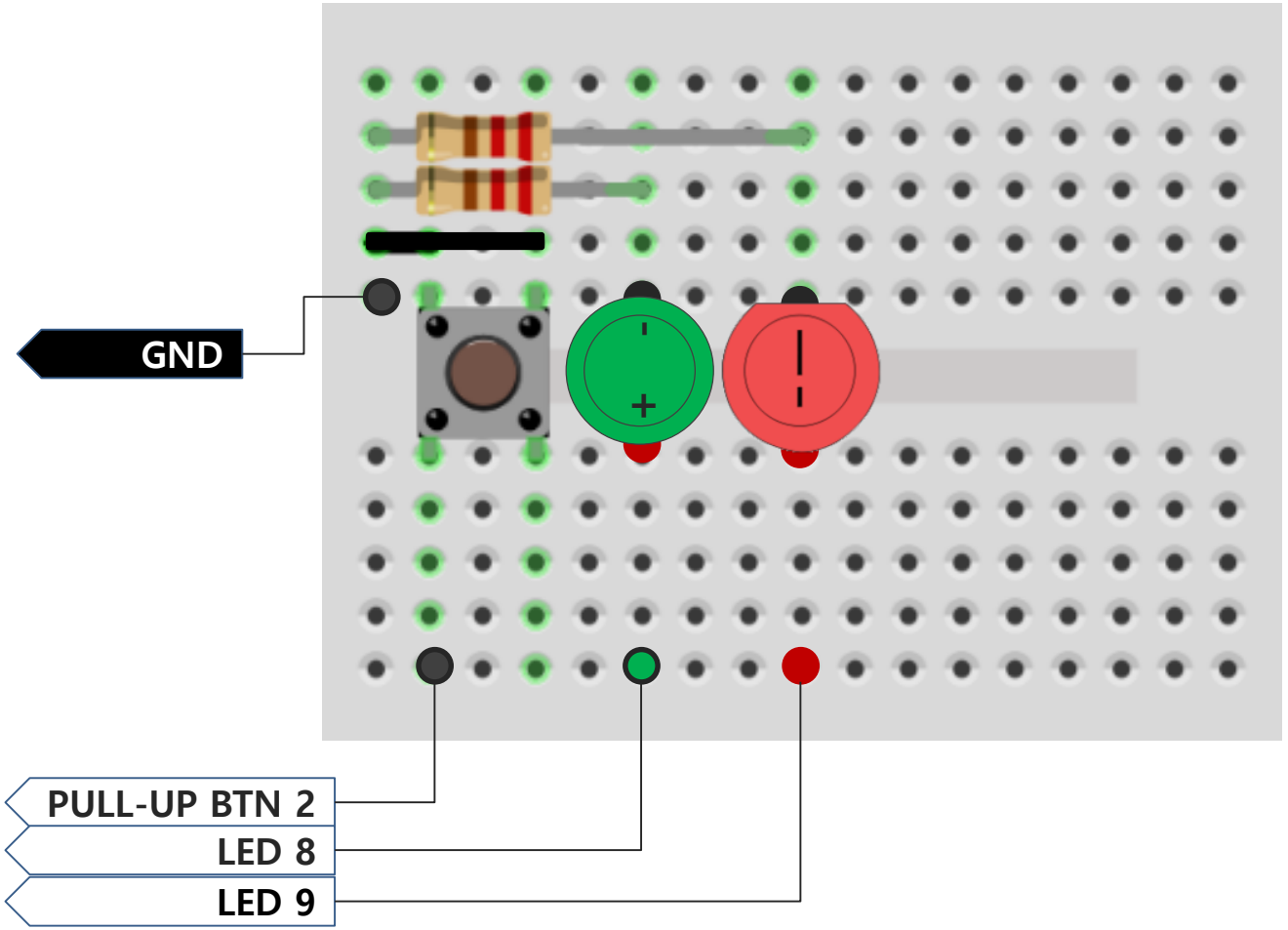
실습1: 스위치 풀업/풀다운 동작 화인

❖ ex01/app.ino

```
void loop()
{
    boolean pd_sw, pu_sw;

    pu_sw = digitalRead(pu_sw_pin);  // 풀업 스위치 상태 읽기
    led1.setValue(pu_sw);            // 풀업 스위치 상태 LED 출력

    pd_sw = digitalRead(pd_sw_pin);  // 풀다운 스위치 상태 읽기
    led2.setValue(pd_sw);            // 풀다운 스위치 상태 LED 출력
}
```

실습2: 내부 풀업 연결 스위치 동작 확인

❖ 회로도



실습2: Switch 외부/내부 풀업 연결 스위치 동작 확인

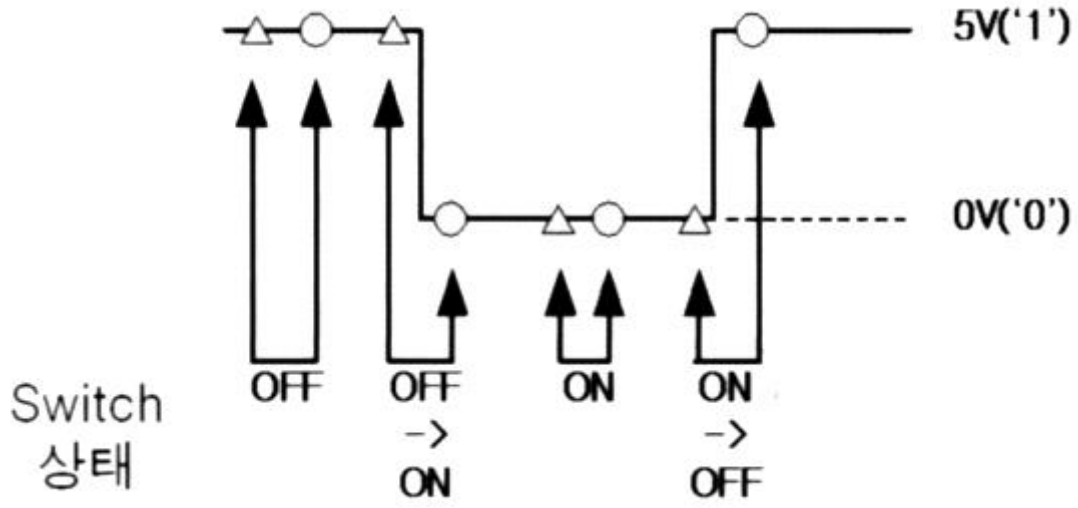
❖ ex02/app.ino

```
#include <Led.h>

const int in_pu_sw_pin = 2;    // 내부 풀업 스위치 연결핀
Led led(8);

void setup() {
  pinMode(in_pu_sw_pin, INPUT_PULLUP);// 내부 풀업 Switch 연결핀 입력 설정
}

void loop() {
  boolean in_pu_sw;
  in_pu_sw = !digitalRead(in_pu_sw_pin);    // 내부풀업 스위치 상태 읽기
  led.setValue(in_pu_sw);
}
```

실습3: Switch 눌려질 때 마다 LED on/off 점멸

❖ 스위치의 상태

스위치 상태	첫 번째 상태	두 번째 상태	비 고
OFF	OFF	OFF	
OFF → ON	OFF	ON	Falling Edge
ON	ON	ON	
ON → OFF	ON	OFF	Rising Edge



실습3: Switch 눌려질 때 마다 LED on/off 점멸

❖ ex03/app.ino

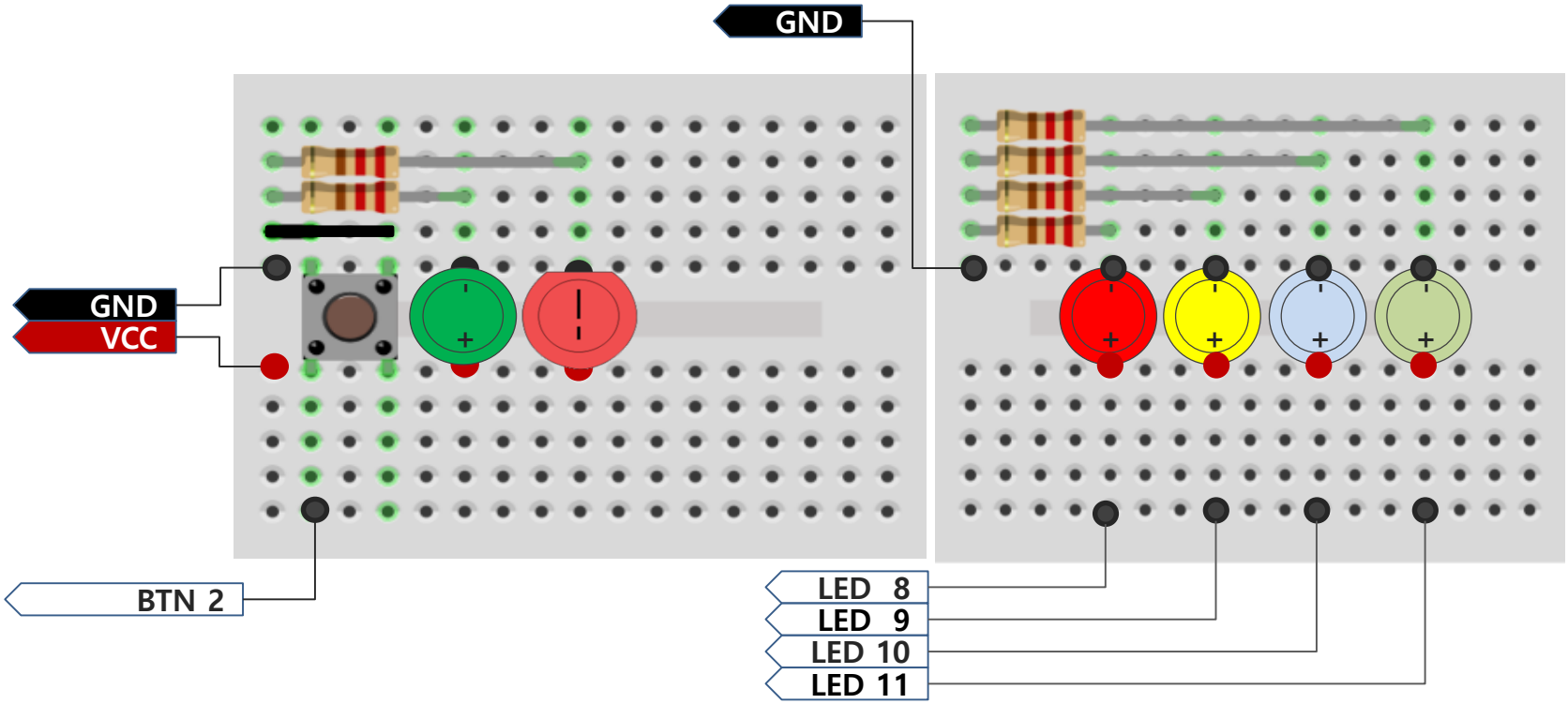
```
#include <Led.h>

#define OFF  0
#define ON   1

const int sw_pin = 2;   // 스위치 연결핀
Led led(8);
boolean  led_st = OFF;  // LED 초기 상태
int count = 0;          // 버튼 클릭 카운트

void setup()
{
  Serial.begin(115200);
  pinMode(sw_pin, INPUT_PULLUP);     // Switch 연결핀 입력 설정
  led.setValue(led_st);
}
```

실습3: Switch 눌려질 때 마다 LED on/off 점멸

❖ ex03/app.ino

```
void loop()
{
  boolean o_sw, n_sw;

  o_sw = !digitalRead(sw_pin);       // 스위치 첫 번째 상태 읽기
  delay(10);                         // 10ms 지연
  n_sw = !digitalRead(sw_pin);       // 스위치 두 번째 상태 읽기

  if(o_sw == OFF && n_sw == ON){     // 앞 상태 OFF and 뒤 상태 ON
    count++;
    Serial.println(count);
    led_st = !led_st;                // LED 상태 반전
    led.setValue(led_st);            // LED 상태 반전 출력
  }
}
```

실습4: Switch 눌려질 때마다 4개의 LED 순차 점멸



실습4: Switch 눌려질 때마다 4개의 LED 순차 점멸

❖ ex04/app.ino

```
// Switch 눌려질 때마다 4개의 LED  순차 점멸(1개 LED만 on)
#include <Led.h>

#define OFF 0
#define ON 1

const int sw_pin = 2; // 스위치 연결핀
Led leds[4] = {
    Led(8), Led(9), Led(10), Led(11)
};

int out_no = -1; // 출력 패턴 번호(0-3)

void setup() {
    Serial.begin(115200);
    pinMode(sw_pin, INPUT_PULLUP); // Switch 연결핀 입력 설정
}
```

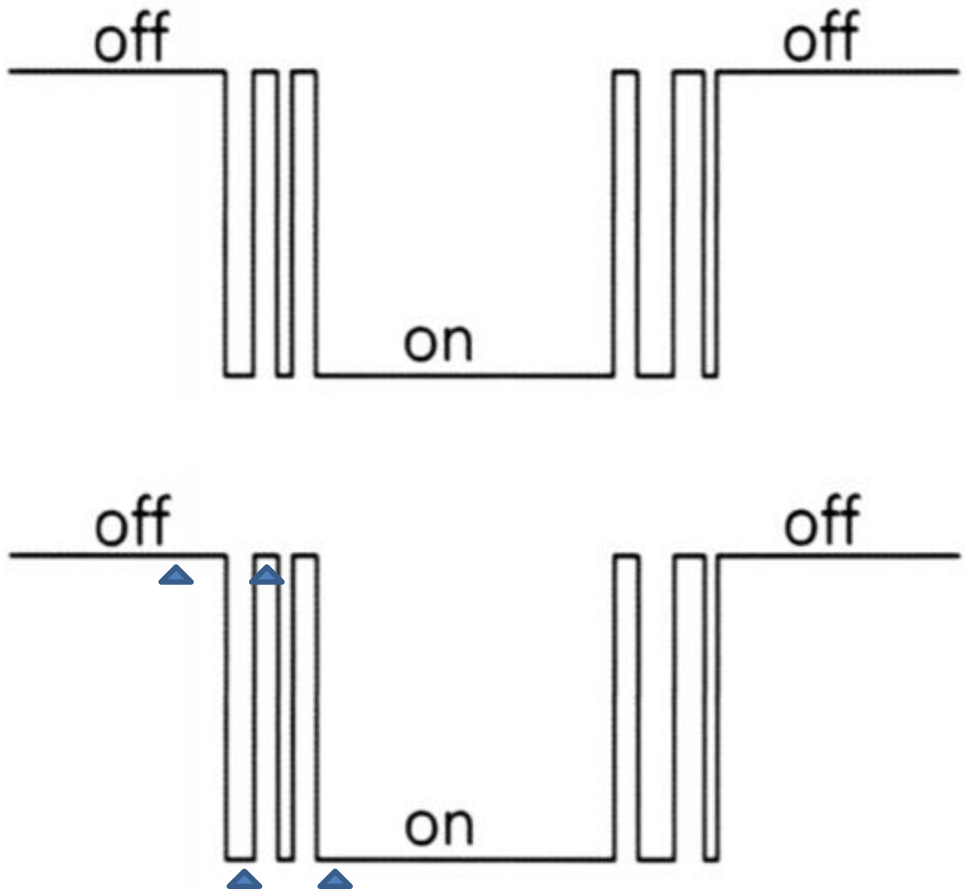
실습4: Switch 눌려질 때마다 4개의 LED 순차 점멸

❖ ex04/app.ino

```
void loop()
{
    boolean o_sw, n_sw;

    o_sw = !digitalRead(sw_pin);      // 스위치 첫 번째 상태 읽기
    delay(10);                        // 10ms 지연
    n_sw = !digitalRead(sw_pin);      // 스위치 두 번째 상태 읽기

    if(o_sw == OFF && n_sw == ON){    // 앞 상태 OFF and 뒤 상태 ON
        out_no = (++out_no)%4;         // 다음 출력 패턴 번호 설정
        Serial.println(out_no);
        for(int n = 0;n < 4;n++){
            leds[n].setValue(n==out_no);
        }
    }
}
```

실습 5: 채터링 방지를 위한 디바운싱 처리 추가

❖ 채터링 방지(디바운싱)



❖ Button 클래스 구현

Button 클래스

❖ ex05/Button.h

```
#pragma once

#include <Arduino.h>

// 매개변수 없는 void 함수에 대한 포인터를 button_callback_t로 정의
typedef void (*button_callback_t)();

class Button {
protected:
    int pin;
    button_callback_t callback; // callback 함수에 대한 포인터

public:
    Button(int pin);
    void setCallback(button_callback_t callback);
    int read();
    void check();
};
```

Button 클래스

❖ ex05/Button.cpp

```
#include "Button.h"

Button::Button(int pin): pin(pin) {
    pinMode(pin, INPUT_PULLUP);
    callback = NULL;
}

void Button::setCallback(button_callback_t callback) {
    this->callback = callback;
}


// 누른 경우에 H, 뗀 경우에 L을 리턴
int Button::read() {
    return !digitalRead(pin);
}
```

Button 클래스

❖ ex05/Button.cpp

```
// polling 방식으로 버튼이 눌러졌는지 체크
void Button::check() {
    bool o_sw, n_sw;

    o_sw = read();
    delay(10);  // 디바운스를 위한 지연시간
    n_sw = read();

    if(o_sw == 0 && n_sw == 1) {    // 버튼을 누른 시점
        if(callback != NULL) {
            callback();
        }
    }
}
```

실습5: Switch 눌려질 때마다 4개의 LED 순차 점멸

❖ ex05/app.ino

```
// Switch 눌려질 때마다 4개의 LED  순차 점멸(1개 LED만 on)
#include <Led.h>
#include "Button.h"

Button btn(2);
Led leds[4] = {
    Led(8), Led(9), Led(10), Led(11)
};

int out_no = -1; // 출력 패턴 번호(0-3)

void move_led() {
    out_no = (++out_no)%4;          // 다음 출력 패턴 번호 설정
    Serial.println(out_no);
    for(int n = 0;n < 4;n++){
        leds[n].setValue(n==out_no);
    }
}
```

실습5: Switch 눌려질 때마다 4개의 LED 순차 점멸

❖ switch4.ino

```
void setup() {
    btn.setCallback(move_led);
}

void loop() {
    btn.check();
}
```