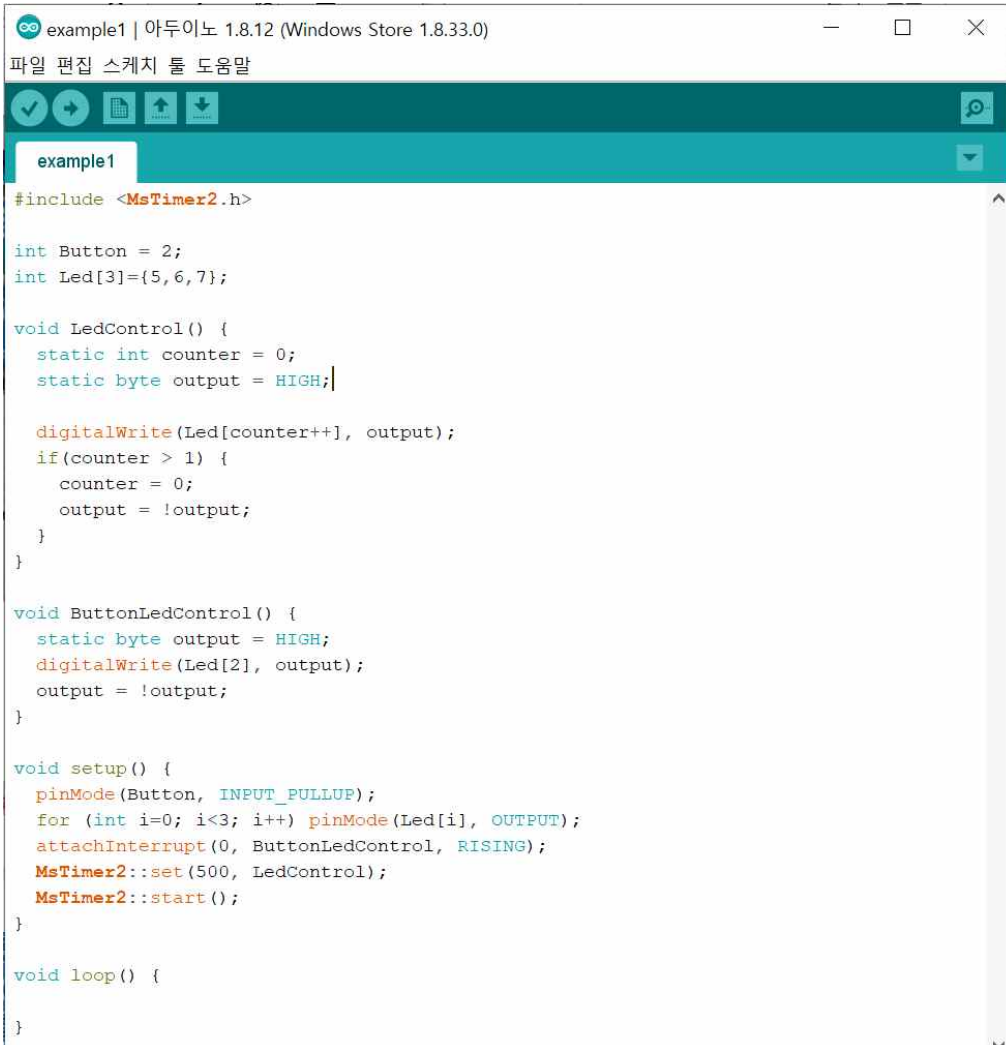


예제 1)



```
#include <MsTimer2.h>

int Button = 2;
int Led[3]={5, 6, 7};

void LedControl() {
    static int counter = 0;
    static byte output = HIGH;

    digitalWrite(Led[counter++], output);
    if(counter > 1) {
        counter = 0;
        output = !output;
    }
}

void ButtonLedControl() {
    static byte output = HIGH;
    digitalWrite(Led[2], output);
    output = !output;
}

void setup() {
    pinMode(Button, INPUT_PULLUP);
    for (int i=0; i<3; i++) pinMode(Led[i], OUTPUT);
    attachInterrupt(0, ButtonLedControl, RISING);
    MsTimer2::set(500, LedControl);
    MsTimer2::start();
}

void loop() {
}
```

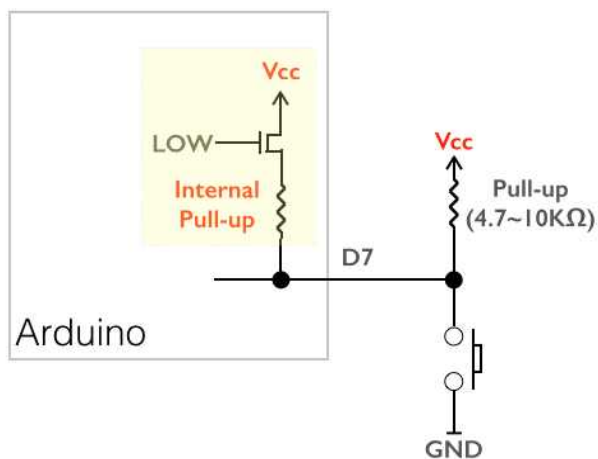
LED 2, 10은 저번 과제에서 했던 Timer 예제를 통해서 500m sec마다 점등이 되게끔 제어하고 LED 16은 인터럽트 핸들러를 통해 입력이 들어왔을 때 점등이 되게 하는 예제입니다. 타이머를 통해서 제어하는 방식은 Lab01에서 했던 방식과 같습니다. 다만 인터럽트를 통해서 독립적으로 LED를 제어하기 위해 attachInterrupt 함수를 사용하였습니다.

attachInterrupt에서는 3가지 변수가 사용됩니다. 첫 번째 변수는 인터럽트 번호입니다. Arduino MEGA ADK의 데이터 시트를 보면 인터럽트 번호마다 대응되는 핀 번호가 존재합니다. 0번 인터럽트 번호를 제어하기 위해서는 2번 핀번호를 입력으로 설정해놓아야 하기 때문에 Button 변수를 2로 한 것입니다.

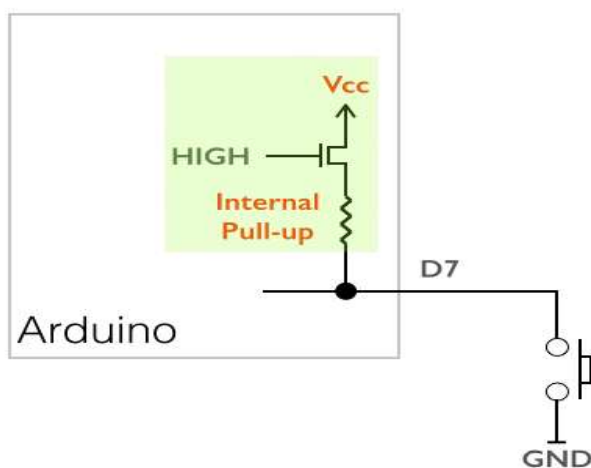
두 번째 변수는 인터럽트 함수로 인터럽트가 실행될 때 수행되어야 하는 함수 즉, 인터럽트 핸들러를 뜻합니다. 이 예제에서 인터럽트 핸들러는 ButtonLedControl 함수로 불이 켜져있을 때는 입력이 들어오면 꺼지고, 불이 꺼져있을 때는 켜지는 함수입니다.

마지막 변수는 mode인데 이 예제에서는 Rising edge에서 인터럽트가 발생합니다. 따라서 실제 동작을 해보면 버튼이 눌렀다가 떼어질 시에 인터럽트가 발생하는 것을 확인할 수 있었습니다.

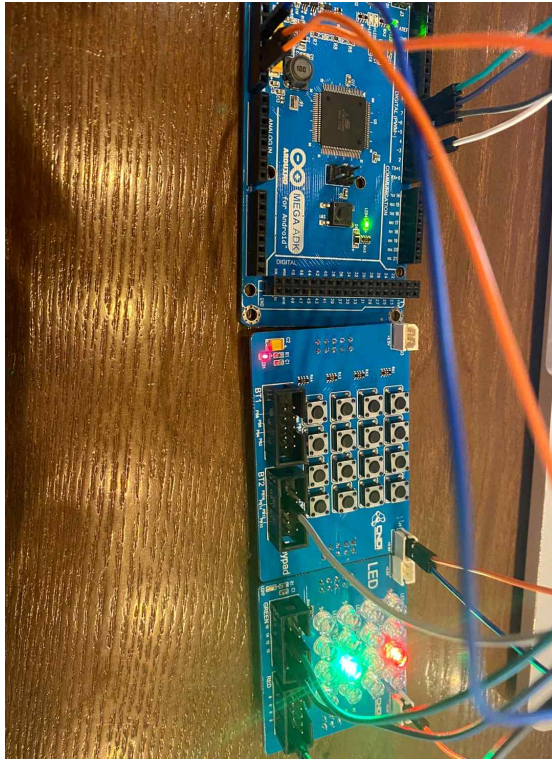
셋업에서 INPUT이 아니라 INPUT_PULLUP을 사용 시에는 아두이노 프로세서 내부에 있는 pull-up저항을 사용하겠다는 것입니다. INPUT_PULLUP을 사용 시에는 LED제어나 다른 OUTPUT제어 시에 아두이노 내부에 있는 별도의 pull_up 저항을 사용하게 됩니다.



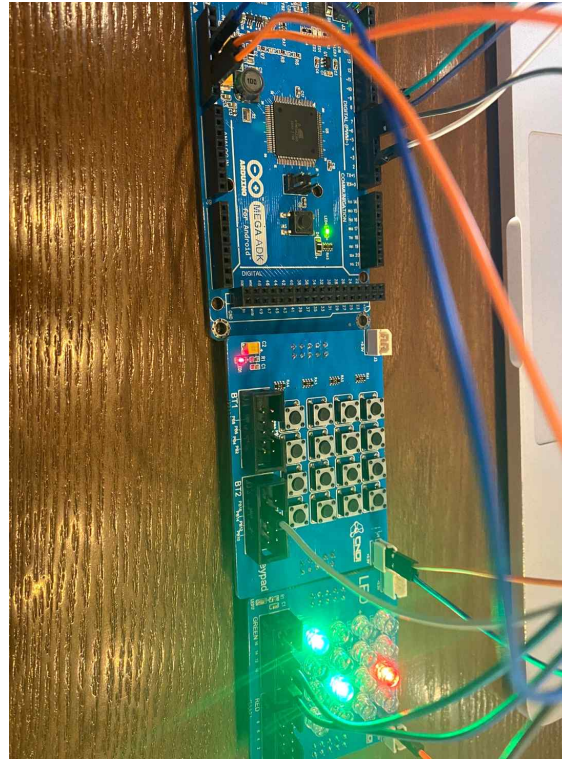
<INPUT>



<INPUT-PULLUP>



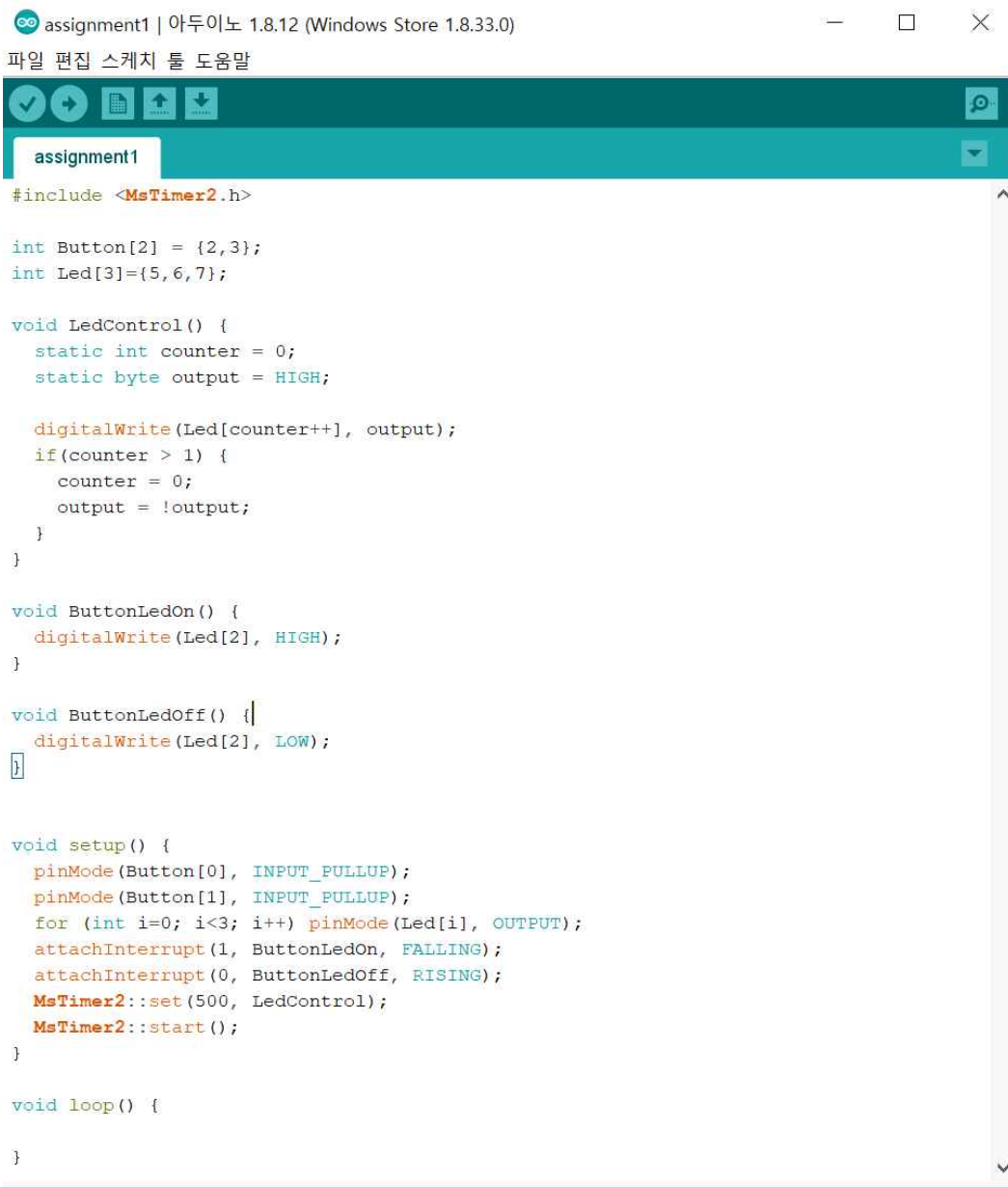
<인터럽트를 통해 LED16이 꺼져있을 때>



<LED16이 켜졌을 때>

실제 동작에서는 LED2, 10이 독립적으로 동작하는 것을 확인하였습니다. LED2, 10은 타이머를 통해 동작하기 때문입니다.

실습과제1)



```
assignment1 | 아두이노 1.8.12 (Windows Store 1.8.33.0)
파일 편집 스케치 툴 도움말

assignment1
#include <MsTimer2.h>

int Button[2] = {2,3};
int Led[3]={5,6,7};

void LedControl() {
    static int counter = 0;
    static byte output = HIGH;

    digitalWrite(Led[counter++], output);
    if(counter > 1) {
        counter = 0;
        output = !output;
    }
}

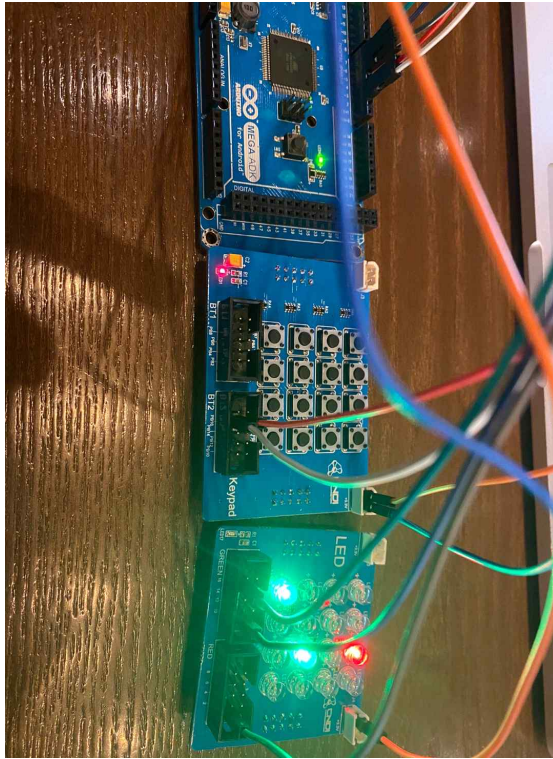
void ButtonLedOn() {
    digitalWrite(Led[2], HIGH);
}

void ButtonLedOff() {
    digitalWrite(Led[2], LOW);
}

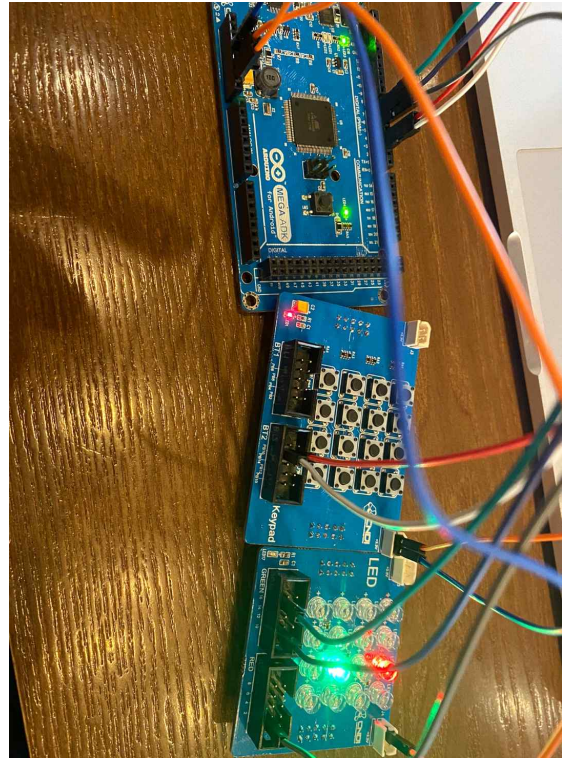
void setup() {
    pinMode(Button[0], INPUT_PULLUP);
    pinMode(Button[1], INPUT_PULLUP);
    for (int i=0; i<3; i++) pinMode(Led[i], OUTPUT);
    attachInterrupt(1, ButtonLedOn, FALLING);
    attachInterrupt(0, ButtonLedOff, RISING);
    MsTimer2::set(500, LedControl);
    MsTimer2::start();
}

void loop() {
}
```

예제 1과 달리 LED16을 켜는 함수와 끄는 함수를 따로 만들어 주었습니다. 두 함수는 각각의 다른 인터럽트 발생 시 동작하기 때문입니다. 인터럽트 번호 0 외에 또 다른 인터럽트 번호 1을 사용하기 위해 번호 1과 대응되는 핀 3번을 입력으로 설정해 주었습니다. 인터럽트 발생 시 LED가 켜질 때는 누르는 타이밍에 켜질 수 있도록 FALLING으로 하였고, 꺼질 때는 누르고 올라가는 타이밍에 켜지도록 RISING으로 하였습니다. 아두이노 핀번호 2는 스위치 모듈의 16, 핀번호 3은 스위치 모듈의 15번에 연결하였습니다.



<Falling Edge: LED ON>



<Rising Edge: LED OFF>

실습과제2)

assignment2 | 아두이노 1.8.12 (Windows Store 1.8.33.0)
파일 편집 스케치 툴 도움말

```

#include <MsTimer2.h>

int Button[2] = {2,3};
int Led[3]={5,6,7};
int ledState = HIGH;
int buttonState1;
int buttonState2;
int lastButtonState1= LOW;
int lastButtonState2= HIGH;

void LedControl() {
    static int counter = 0;
    static byte output = HIGH;

    digitalWrite(Led[counter++], output);
    if(counter > 1) {
        counter = 0;
        output = !output;
    }
}

void setup() {
    pinMode(Button[0], INPUT_PULLUP);
    pinMode(Button[1], INPUT_PULLUP);
    for (int i=0; i<3; i++) pinMode(Led[i], OUTPUT);
    MsTimer2::set(500, LedControl);
    MsTimer2::start();
}

```

```

void loop() {
    int reading1 = digitalRead(Button[0]);
    int reading2 = digitalRead(Button[1]);

    if(reading1 != buttonState1) {
        buttonState1 = reading1;
        if(buttonState1 == HIGH) {
            ledState = !ledState;
        }

        digitalWrite(Led[2], ledState);
        lastButtonState1 = reading1;
    }

    else if(reading2 != buttonState2) {
        buttonState2 = reading2;
        if(buttonState2 == LOW) {
            ledState = !ledState;
        }

        digitalWrite(Led[2], ledState);
        lastButtonState2 = reading2;
    }
}

```

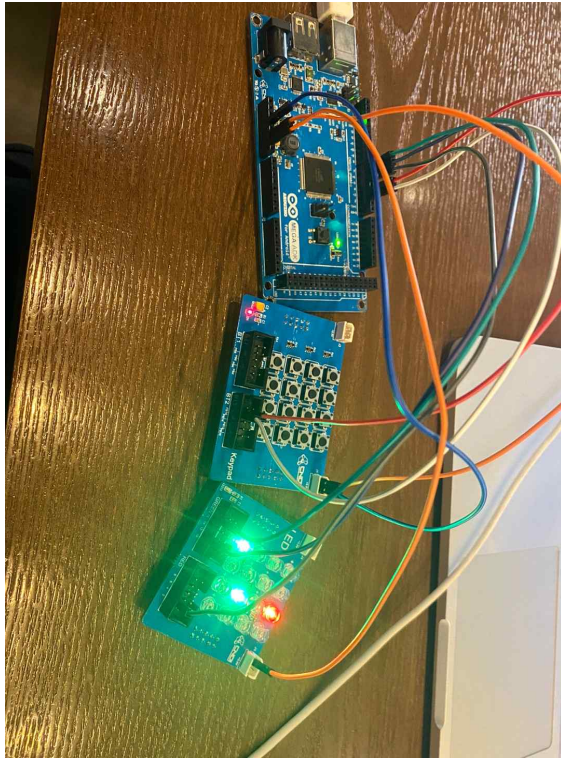
예제1, 및 과제2처럼 인터럽트를 사용하는 것이 아닌 loop문이 돌아갈 때마다 Input 입력신호가 들어왔는지 판단하는 Polling 방식을 통해 구현하였습니다. 따라서 loop문에서의 시작부분에서는 입력으로 설정해놓은 2번 핀과 3번 핀의 state를 확인하게 됩니다.

참고용 자료인 debounce에서는 버튼 하나로 rising edge 마다 LED를 점등하는 내용이 있었습니다. debouncedelay로 설정해 놓은 시간보다 더 큰 지연시간이 발생 시 함수가 동작하는 것입니다. 저는 debouncedelay부분은 빼고 Polling 방식을 구현 하였습니다.

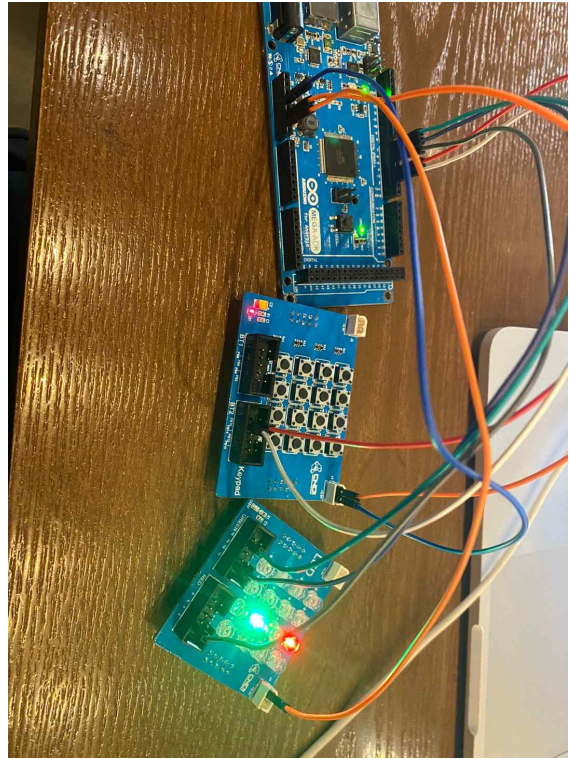
처음에는 ButtonState와 LastButtonState를 각각 두 개로 나누지 않고 하나 씩 제어하려 하였으나, Falling 일때와 Rising 일 때의 조건이 각각 달라지므로 실제 점등이 이상하게 작동하는 것을 확인 하였습니다. 따라서 변수를 Rising 일 때 Off되기 위한 ButtonState1, LastButtonState1 과 Falling 일 때 ON되기 위한 ButtonState2, LastButtonState2로 나누었습니다. 변수를 통일하기 위해서 여러 가지 방법을 생각 하였으나, 아직 명확한 답은 찾지 못하였고, 코드를 더 간편화하기 위한 방법은 질문게시판을 통해 해결해 보겠습니다.

먼저 Rising Edge일 때 LED가 OFF될 때입니다. reading1은 2번 핀의 입력이 들어왔을 때 LOW가 됩니다. 그렇다면 이후의 If문의 조건에 들어갈 것이고, Rising Edge에서 발생해야 하므로 reading1이 HIGH가 될 때 즉, buttonstate가 그대로 복사한 reading1의 값이 HIGH라는 뜻은 입력이 LOW -> HIGH라는 뜻이므로 Rising Edge에서 발생 했다는 것을 뜻한다. 결국 이 때 led제어가 발생하게 되고 LED State는 반대로 변하게 된다.

Falling 일 때는 반대로 작동하게 되는 것이다.



<15번 스위치를 통해 Falling에서 ON>



<16번 스위치를 통해 Rising에서 OFF>