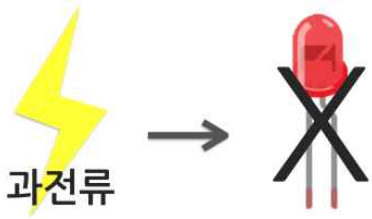
<table>
<tr><td>차 시</td><td>AS_02</td><td>대상</td><td></td></tr>
<tr><td>교육주제</td><td>아두이노 LED 및 버튼 제어 프로젝트</td><td>교육시간</td><td>120분</td></tr>
<tr><td>교육목표</td><td colspan="3">1. 아두이노를 이용하여 LED를 제어할 수 있다.
2. 아두이노를 이용하여 삼색 LED를 제어할 수 있다.
3. 아두이노에서 버튼 입력을 수행하여 프로그램에 연동할 수 있다.</td></tr>
<tr><td>교육자료
(준비물)</td><td colspan="3">※장비류는 미리 배분하고, 재료는 각 실험 단계에서 배분한다.
<table>
<tr><td>Arduino</td><td>1.0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>USB 케이블</td><td>1.0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>300Ω</td><td>5.0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>LED-RED(고휘도)</td><td>5.0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>브레드보드(소형)</td><td>1.0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>텍스위치</td><td>3.0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>10kΩ</td><td>3.0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>LED-RGB(고휘도)</td><td>1.0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>점퍼케이블(F/F)</td><td>12</td><td></td><td></td><td></td><td></td></tr>
</table></td></tr>
</table>

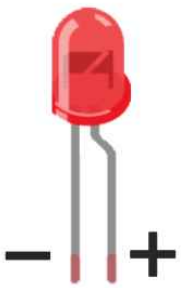
<부품 이해하기>

1. 저항

<table>
<tr>
<td> 
<그림 1> 저항</td>
<td> 
<그림 2>
과전류로 인한 부품 고장</td>
<td> 

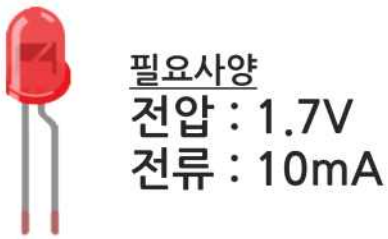
<그림 3> 저항의 역할</td>
</tr>
<tr>
<td colspan="3">LED나 전자부품을 사용하는 경우 부품을 보호하기 위해 저항과 함께 사용하는 경우가 많습니다. 만약 LED 등의 부품에 너무 많은 전류가 통과하면 망가질 확률이 높기 때문입니다. 전자부품을 저항과 함께 연결하면 저항은 전류의 일부를 열 에너지로 교체하여 전류의 양을 조절합니다. 이 때 저항이 전류를 열 에너지로 바꾸는 양은 ohm(저항의 단위)에 따라 다릅니다. 만약 저항 값이 필요한 값보다 작은 경우에도 전류가 많이 흘러 부품이 망가질 수 있으므로, 적당한 값의 저항을 사용해야 합니다.</td>
</tr>
</table>

2. LED

<table>
<tr>
<td> 
<그림 1> LED</td>
<td> 
<그림 5> LED의 극성</td>
<td> 

<그림 6> 전류의 흐름</td>
</tr>
<tr>
<td colspan="3">LED는 전류를 빛으로 변환하는 부품입니다. 다양한 모양, 크기, 색깔을 가지고 있습니다.
또한, LED는 극성을 갖습니다. 극성이란 한 방향으로만 전류가 흘러야 하는 특징을 뜻합니다.
LED의 다리를 살펴보면 긴 쪽이 양극(+)이고, 짧은 쪽이 음극(-)입니다. 양극을 다른 말로
애노드(anode), 음극을 캐소드(cathode)라고 부릅니다. 전류는 양극에서 음극으로 흘러야 하므로
브레드보드에 조립할 때 유의해야 합니다. LED는 회로도에서 <그림 3>과 같이 표시합니다.</td>
</tr>
</table>

3. 옴의 법칙 : 전압(V) = 전류(I) x 저항(R)



<그림 7> LED의 필요사양

LED와 같은 부품에는 사용할 때 확인할 수 있는 사양의 값과 옴의 법칙을 이용하여 필요한 저항의 값을 구할 수 있습니다.

(A) 전압(V) = 전류(I) x 저항(R)
(B) 저항(R) = 전압(V) / 전류(I)

먼저 옴의 법칙 (A)에서 양변을 전류로 나눠 (B)와 같이 만듭니다.

(C) 저항(R) = (5-1.7) / 전류(I)

아두이노 핀의 출력 전압이 5V이고 LED의 필요 전압이 1.7V이기 때문에 (B)의 전압 부분에 각 전압의 차를 넣습니다.

(D) 저항(R) = (5-1.7) / 0.01

LED의 필요 사양에 표시된 전류 값인 10mA를 전류에 넣습니다. mA는 기존 A의 1,000분의 1 단위이므로 10에서 1,000을 나눈 0.01을 전류 부분에 넣습니다.

(E) 330 = (5-1.7) / 0.01

계산하면 필요한 저항 값이 330으로 나옵니다. 이렇게 계산된 저항 값을 바로 사용해도 되지만 보다 안전한 사용을 위해 좀 더 높은 값의 저항을 사용하는 것이 추천합니다.

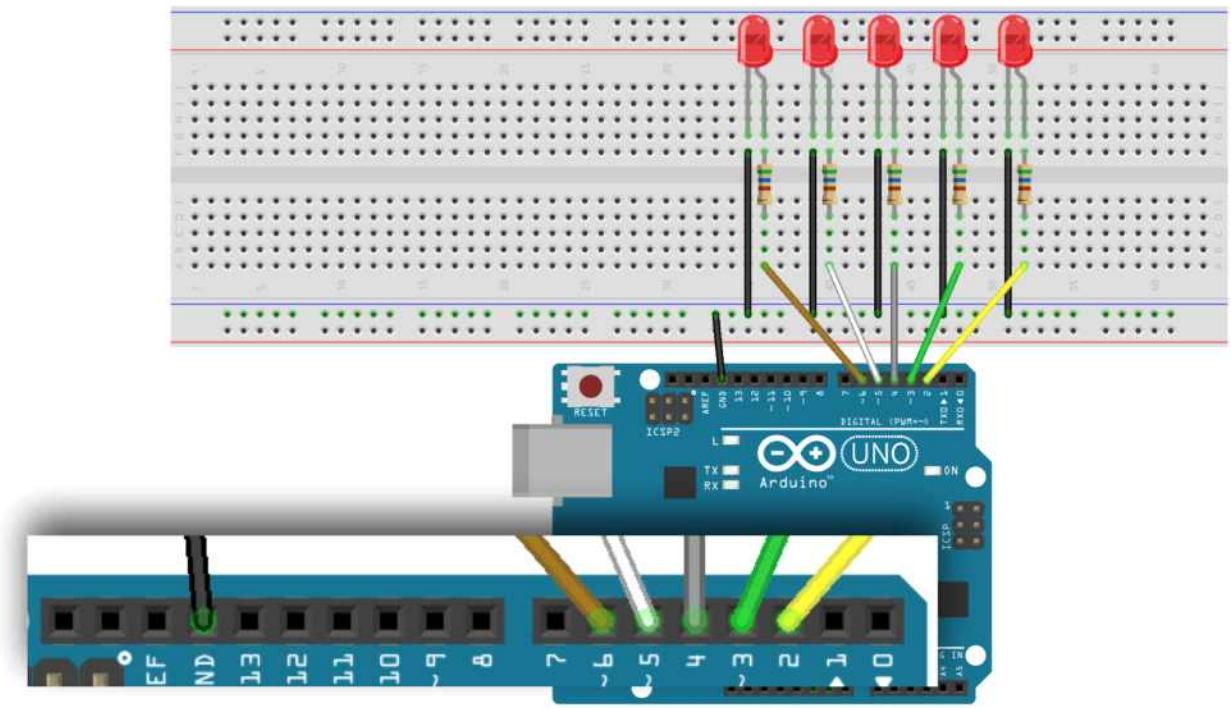
<실습>

1. 파도치는 LED 구현하기: LED 여러 개를 이용해 파도처럼 빛나도록 만들어 봅시다.

(1) 재료: LED 5개, 510 ohm 저항 5개

<그림 8> 재료

(2) 회로도



<그림 9> 회로도

회로는 <그림 9>와 같이 2~6번까지의 핀에 510 ohm 저항과 LED를 연결합니다.

(3) 아두이노 프로그래밍

<코드 1> 파도치는 LED

```
#define DELAY_TIME 100

void setup(){
   pinMode(2, OUTPUT);
   pinMode(3, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(5, OUTPUT);
   pinMode(6, OUTPUT);
}

void loop(){
   digitalWrite(2, HIGH);
   delay(DELAY_TIME);
   digitalWrite(2, LOW);
   digitalWrite(3, HIGH);
   delay(DELAY_TIME);
   digitalWrite(3, LOW);
   digitalWrite(4, HIGH);
   delay(DELAY_TIME);
   digitalWrite(4, LOW);
   digitalWrite(5, HIGH);
   delay(DELAY_TIME);
   digitalWrite(5, LOW);
   digitalWrite(6, HIGH);
   delay(DELAY_TIME);
   digitalWrite(6, LOW);
   digitalWrite(5, HIGH);
   delay(DELAY_TIME);
   digitalWrite(5, LOW);
   digitalWrite(4, HIGH);
   delay(DELAY_TIME);
   digitalWrite(4, LOW);
   digitalWrite(3, HIGH);
   delay(DELAY_TIME);
   digitalWrite(3, LOW);
}
```

(4) 코드 분석

① #define DELAY_TIME 100

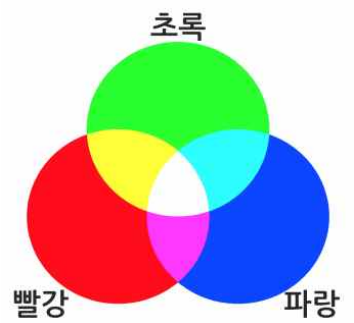
#define은 매크로 상수를 선언하는 명령어입니다. 상수란 변수와 비슷하지만 일단 한 번 값이 정해지면 다음에 바꿀 수 없습니다. 매크로 상수 선언은 변수 선언과 달리 마지막에 세미콜론을 입력하지 않습니다.

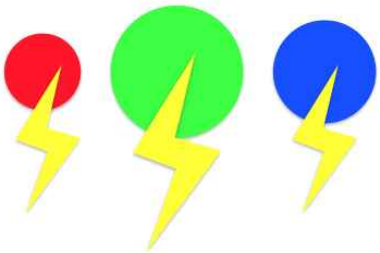
(5) 확인하기

완성되었다면 매 0.1초마다 LED가 좌우 순서대로 켜지는 것을 볼 수 있습니다.

2. 3색 LED 자동 색 변경: 아날로그 신호 출력을 이용해 삼색 LED를 제어해봅시다.

(1) 삼색 LED

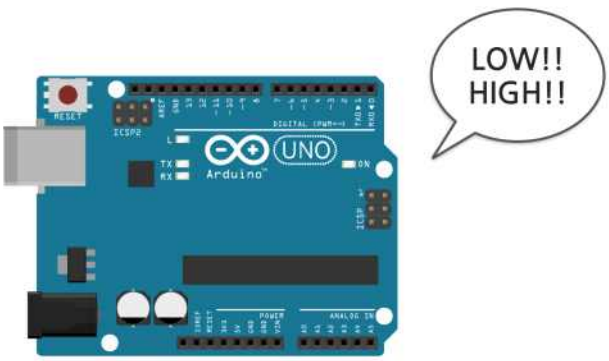




<그림 10> 삼색 LED <그림 11> 삼원색 <그림 12> 전압을 이용한 색의 양 조절

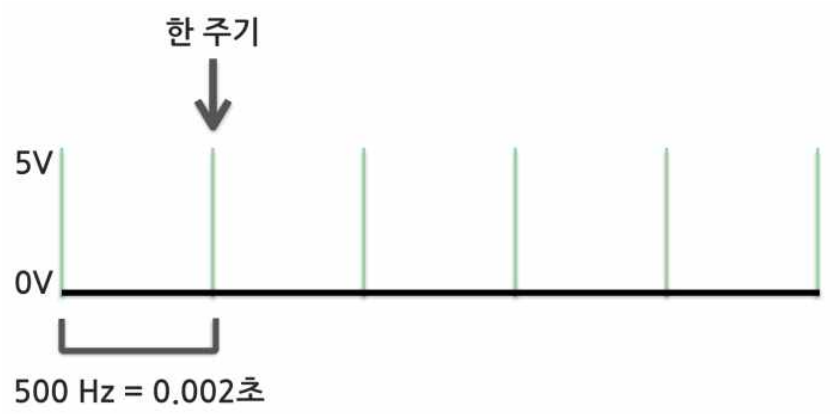
빛의 삼원색을 이용해 원하는 색을 표현할 수 있는 LED입니다. 삼원색은 빛을 구성할 때 핵심이 되는 세 가지 색을 말합니다. TV나 PC 모니터에서 삼색 LED를 사용해 삼원색으로 색을 표현합니다. 색의 양은 전압을 조절해 변경할 수 있으며 0V에서 5V사이 중 원하는 값으로 변경합니다.

(2) PWM



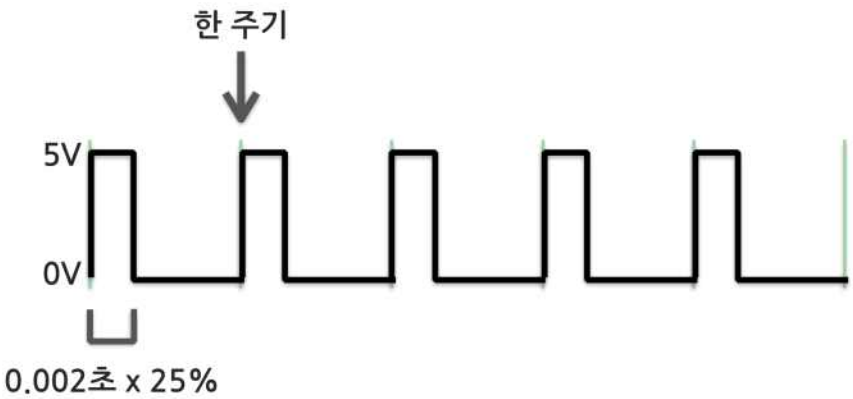
<그림 13> 하드웨어의 출력

아두이노와 같은 하드웨어는 전압 출력을 오로지 LOW 또는 HIGH로 설정할 수 있습니다. 따라서 LOW나 HIGH가 아닌 아날로그 값으로 변경하기 위해 펄스 폭 변조(PWM) 방식을 사용합니다.



<그림 14> PWM의 기준 주기

PWM을 사용할 때 기준 주기를 고려해야 합니다. 아두이노에서 기준 주기는 대략 0.002초입니다. 기준 주기 동안의 전압과 출력 전압은 동일합니다.



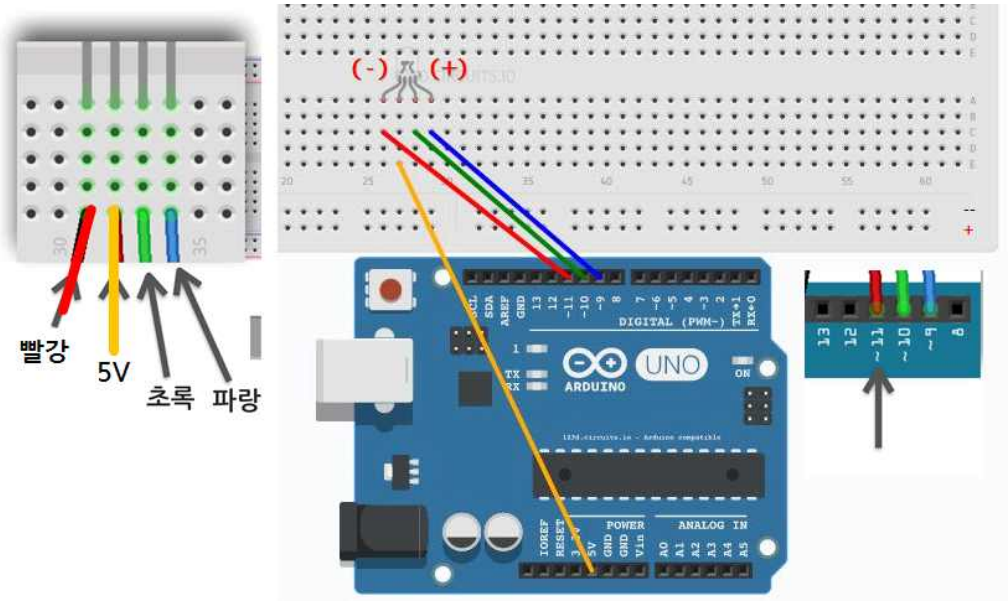
<그림 15> PWM의 이용해 전압을 25%로 조절하기

만약 주기의 특정 부분만 HIGH로 설정하고 나머지는 LOW로 설정하면 출력은 그 비율에 따라 변합니다. 예를 들어 5V를 25%로 하고 나머지 0V를 75%로 해서 5V의 25%인 1.25V가 됩니다. 이를 통해 전압 출력을 0V에서 5V 사이 원하는 값으로 바꿀 수 있습니다.

(3) 재료: 삼색 LED

<그림 16> 재료

(4) 회로도



<그림 17> 회로도

삼색 LED의 R, G, B를 각각 11, 10, 9에 연결하고 아두이노 5V를 삼색 LED의 가장 긴 핀과 연결합니다. 간혹 R과 B가 잘못 표시된 삼색 LED가 있는데 이런 경우 위치를 서로 바꾸어 연결합니다. 핀 번호 하단의 물결(~)표시는 PWM 기능을 사용할 수 있다는 뜻입니다.

(5) 아두이노 프로그래밍

<코드 2> 자동으로 색 바뀌게 하기

```
#define RED 11
#define GREEN 10
#define BLUE 9

void setup(){
  randomSeed(analogRead(0)); // 난수 생성기 초기화
}

void loop(){
  analogWrite(RED, random(255));          // 빨간색 조절
  analogWrite(GREEN, random(255));   // 초록색 조절
  analogWrite(BLUE, random(255));         // 파란색 조절
  delay(1000);
}
```

(6) 코드분석

① randomSeed(키 값);

난수 발생 함수는 프로그래밍에서 자주 사용합니다. 무작위 수(난수) 발생이 필요한 경우 randomSeed 함수로 초기화하고 random() 함수에서 무작위 수를 생성해 사용합니다. setup부분에 보면 randomSeed 명령어를 볼 수 있습니다. randomSeed는 난수 생성기를 초기화 시켜주는 명령어입니다. 여기서 random이라는 명령어를 통해 난수를 사용하는데, 사용 전에는 꼭 난수 생성기를 초기화 해줘야 합니다. 왜냐하면 난수 생성기를 초기화하지 않으면 동일한 순서로 난수가 만들어지기 때문입니다. 따라서 키 값을 사용해 난수 생성기를 초기화합니다.

코드에서는 키 값으로 analogRead(0)이 사용되었습니다.

② analogRead(0)

analogRead는 아날로그 입력 값을 읽는 명령어입니다. 아날로그 입력 핀은 A0와 같이 표시하는데, 여기서 A는 Analog의 약자이고 0은 핀 번호를 의미합니다. 즉, analogRead(0)는 A0번 핀의 값을 읽으라고 명령하는 것입니다. 아날로그 입력 핀은 처음부터 입력모드로 설정되며 플로팅 상태입니다. 따라서 전압이 LOW와 HIGH를 사이를 이동하면서 analogRead라는 명령어가 실행됐을 때 무작위의 값을 읽습니다. 무작위로 읽은 값은 다시 난수 생성기를 초기화할 때 키 값을 사용합니다. 이러한 방식으로 똑같은 값이 키 값으로 사용되어 난수 생성기를 초기화하는 것을 방지할 수 있습니다.

③ random(최대값);

random은 실제 난수를 만드는 명령어입니다. 0에서 최대값 사이에서 무작위로 값을 뽑습니다. 예를 들어 random(255)는 0에서 254 중에 무작위 수를 고릅니다. random 명령어는 random(최소값, 최대값)과 같이 최소 범위 또한 줄 수 있습니다.

(7) 확인하기

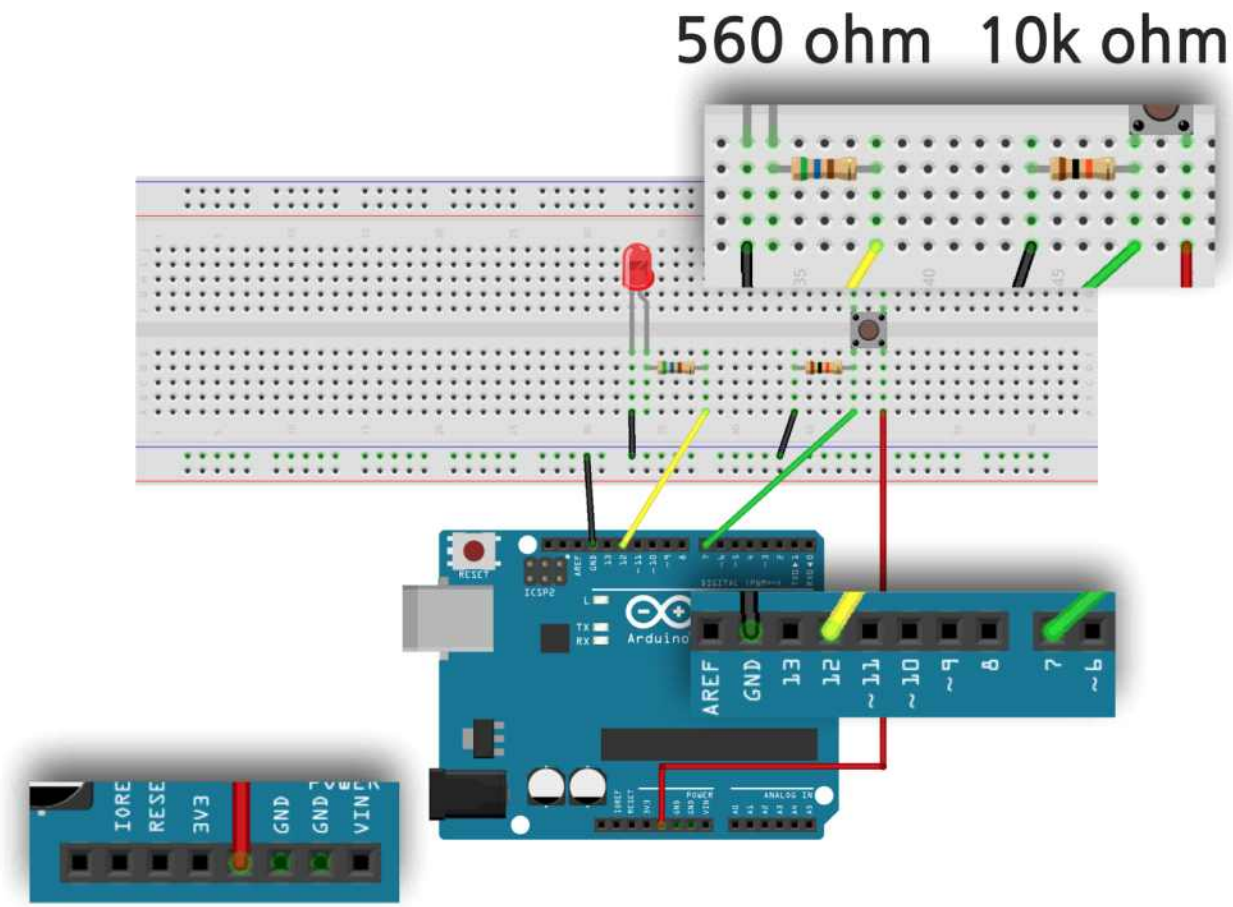
정상적으로 만들었다면 매 1초마다 무작위로 색이 변하면서 빛나는 것을 볼 수 있습니다.

3. 버튼 기초 실습: 버튼을 사용해 LED를 제어해봅시다.

(1) 재료: 버튼, LED, 560ohm 저항, 10k ohm 저항

<그림 18> 재료

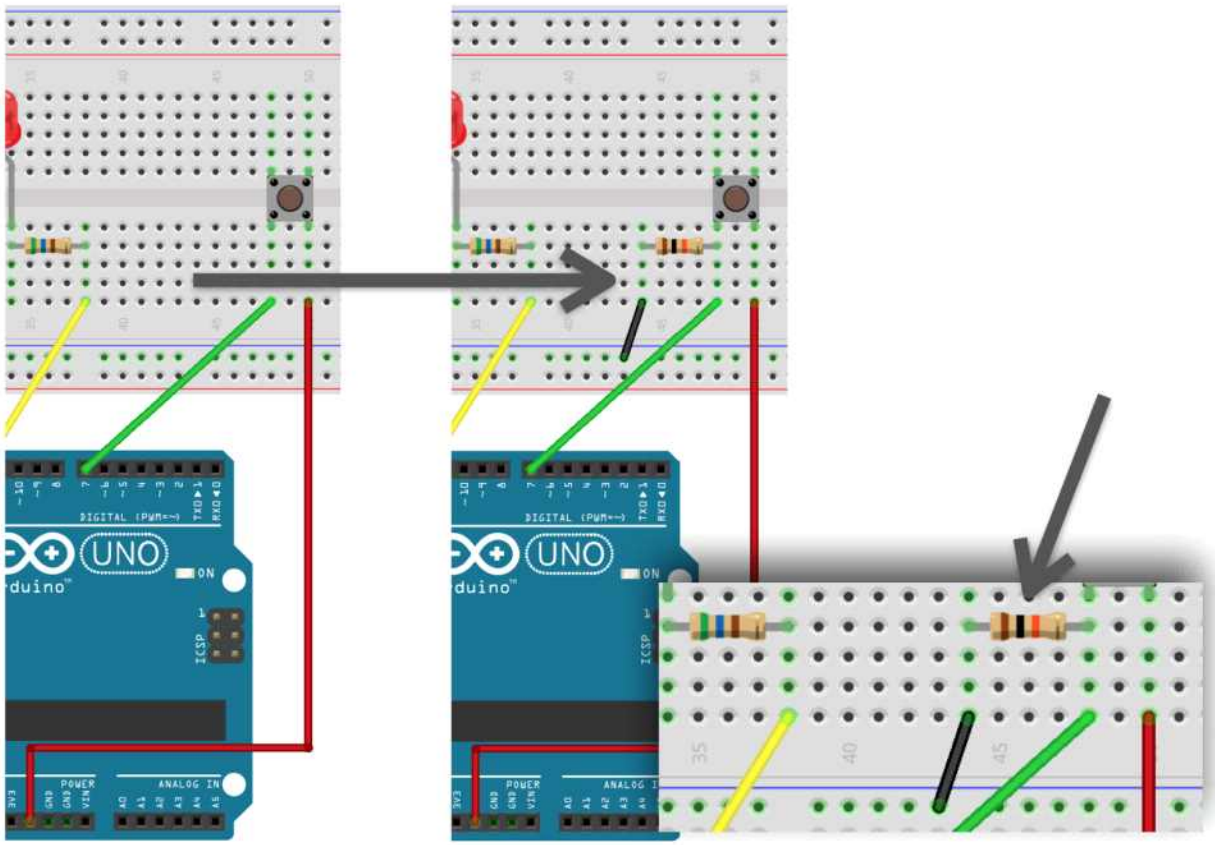
(2) 회로도



<그림 19> 회로도

12번 핀에 LED를, 7번 핀에 버튼을 연결합니다. 버튼에는 10k ohm 저항을 사용합니다.

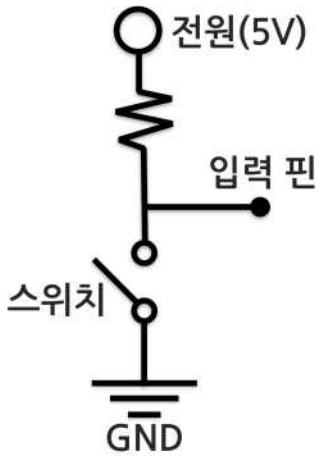
(3) 풀업 & 풀다운 Pull up & Down

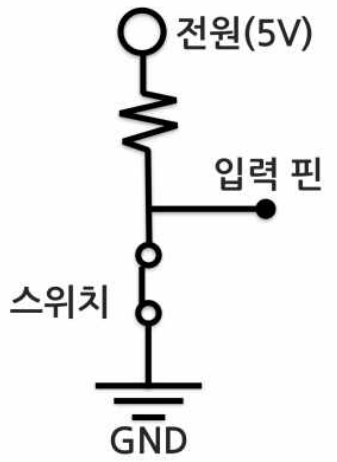


<그림 20> 풀다운(Pull-Down) 저항

회로 연결을 보면 버튼을 선 2개로 연결하지 않고 저항을 추가해 선 3개로 연결한 것을 볼 수 있습니다. 이와 같이 연결하는 이유는 핀 모드를 입력으로 설정하면 해당 핀이 플로팅 상태가 되기 때문입니다. 플로팅 상태란 해당 핀에 항시 소량의 전류가 흘러 핀의 전압이 LOW와 HIGH 사이를 계속 움직이는 현상을 말합니다. 이로 인해 올바른 신호 값을 인식할 수 없다는 문제점을 갖고 있습니다. 이러한 문제점을 해결하기 위해 저항을 버튼에 연결하여 핀의 전압을 LOW 또는 HIGH에 고정합니다. LOW에 고정시키는 것을 풀다운(Full-Down), HIGH에 고정시키는 것을 풀업(Pull-Up)이라고 합니다.

① 풀업

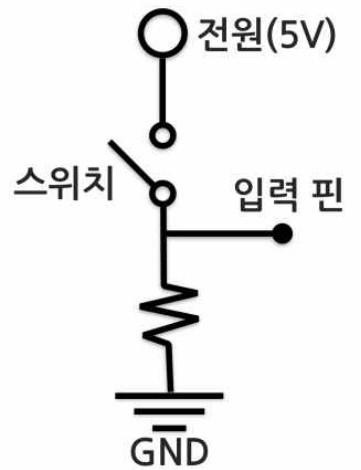


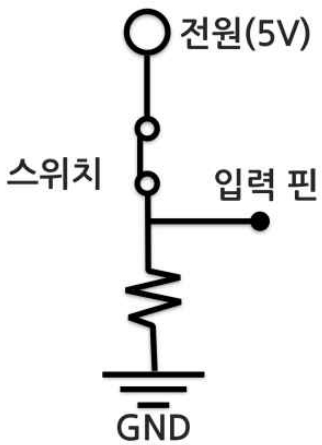


<그림 21> 풀업, 스위치가 열려진 상태 <그림 22> 풀업, 스위치가 닫힌 상태

입력 핀과 전원 사이에 저항을 연결합니다. 스위치가 열린 상태인 경우 입력 핀이 전원과 연결이 되어 있으므로 전압이 5V가 됩니다.(Pull-Up). 스위치가 닫힌 상태이면 입력 핀의 전압은 그라운드와 동일한 0V가 됩니다.

② 풀다운





<그림 23> 풀다운, 스위치가 열려진 상태 <그림 24> 풀다운, 스위치가 닫힌 상태

입력 핀과 그라운드 사이에 저항을 연결합니다. 스위치가 열린 상태인 경우 입력 핀이 그라운드와 바로 연결되므로 전압이 0V가 됩니다.(Pull-Down).

(4) 아두이노 프로그래밍

<코드 3> 버튼 사용하기

```
#define LED        12
#define BUTTON   7

void setup(){
   pinMode(LED, OUTPUT);
   pinMode(BUTTON, INPUT);
}

void loop(){
   if (digitalRead(BUTTON) == HIGH){ // 버튼 상태가 HIGH인지 확인한다.
      // 비교해서 참이라면 아래 코드가 실행된다.
      digitalWrite(LED, HIGH);
      delay(500);
      digitalWrite(LED, LOW);
   }
}
```

(5) 코드분석

① if (조건) { … }

loop부분에 보면 위와 같이 if문을 볼 수 있습니다. if문은 소괄호**()** 안의 조건이 참인 경우 중괄호**{}** 안의 코드를 실행합니다 예를 들어.

if (digitalRead(BUTTON) == HIGH) { … }

위 코드를 살펴봅시다. C언어에서 ==기호는 좌우가 동일한지 비교합니다. 만약 동일하면 참이고 다르면 거짓이며, 프로그래밍에서는 참을 true, 거짓을 false라고 부릅니다. 따라서 위 코드는 버튼의 상태가 HIGH인지 비교한다는 뜻입니다.

if (조건) { … } else { … }

if문 밑에 else가 붙는 경우가 있습니다. 조건이 거짓인 경우 else 밑 중괄호의 코드가 실행됩니다.

(6) 확인하기

정상적으로 연결했다면 버튼이 눌렸을 때 LED가 켜졌다가 0.5초 뒤에 꺼지게 됩니다.

(7) 도전하기

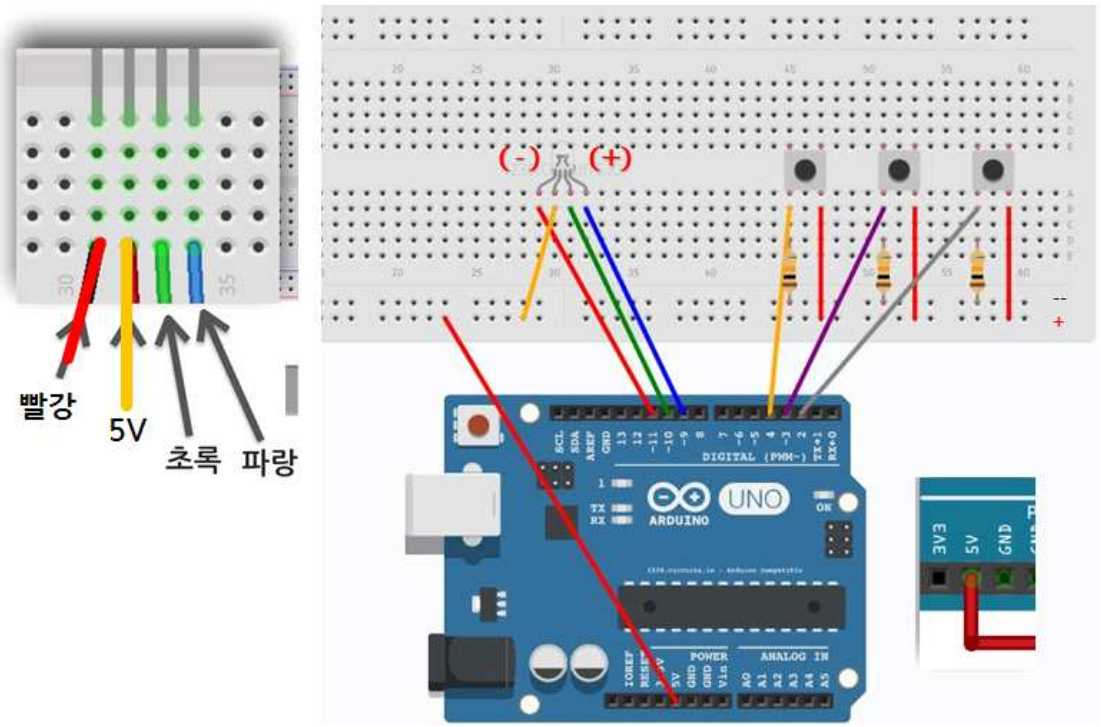
- 파도처럼 반짝이는 LED의 순서를 바꿔봅시다.
- 버튼 방식을 풀업으로 바꿔 구현해봅시다.

4. 버튼으로 LED 제어

(1) 재료: 삼색 LED, 버튼, 10k ohm 저항

<그림 25> 재료

(2) 회로도



<그림 27> 회로도

2~4번까지 버튼을 연결하고, 9~11번까지 삼색 LED에 연결합니다.

(3) 아두이노 프로그래밍

<코드 4> 버튼을 이용해 제어하기

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

int r=255, g=255, b=255;

void setup(){
  pinMode(RED_BUTTON, INPUT);
  pinMode(GREEN_BUTTON, INPUT);
  pinMode(BLUE_BUTTON, INPUT);
}

void loop(){
  if(digitalRead(RED_BUTTON) == HIGH){
    --r; // r 값을 감소시킨다.
    if(r<0){
      r=255;
    }
  }

  if(digitalRead(GREEN_BUTTON) == HIGH){
    --g; // g 값을 감소시킨다.
    if(g<0){
      g=255;
    }
  }

  if(digitalRead(BLUE_BUTTON) == HIGH){
    --b; // b 값을 감소시킨다.
    if(b<0){
      b=255;
    }
  }

  analogWrite(RED, r);
  analogWrite(GREEN, g);
  analogWrite(BLUE, b);
  delay(10);
}
```

(4) 코드분석

① ++a (증가연산자)

++기호는 해당 변수에 1을 더한다는 뜻입니다.

② --a (감소연산자)

--기호는 해당 변수에 1을 뺀다는 뜻입니다.

③ 증감연산자는 변수 앞이나 뒤에 모두 사용 가능합니다.

if(++a == HIGH){ **vs** if(a++ == HIGH){

++기호의 위치 차이는 if문을 통해 비교할 수 있습니다. 전자 같은 경우 a의 값을 먼저 증가하고 HIGH를 비교합니다. 후자 같은 경우 HIGH와 먼저 비교를 한 뒤에 a의 값을 증가합니다.

(5) 확인하기

정상적으로 만들었다면 각 버튼을 누를 때마다 빨강, 파랑, 초록의 색 양이 변경됩니다.

(6) 도전해보기

- 인터넷에서 원하는 색의 RGB값을 구해 색을 바꿔 봅시다.

- 색의 값을 감소시키도록 변경해봅시다.