

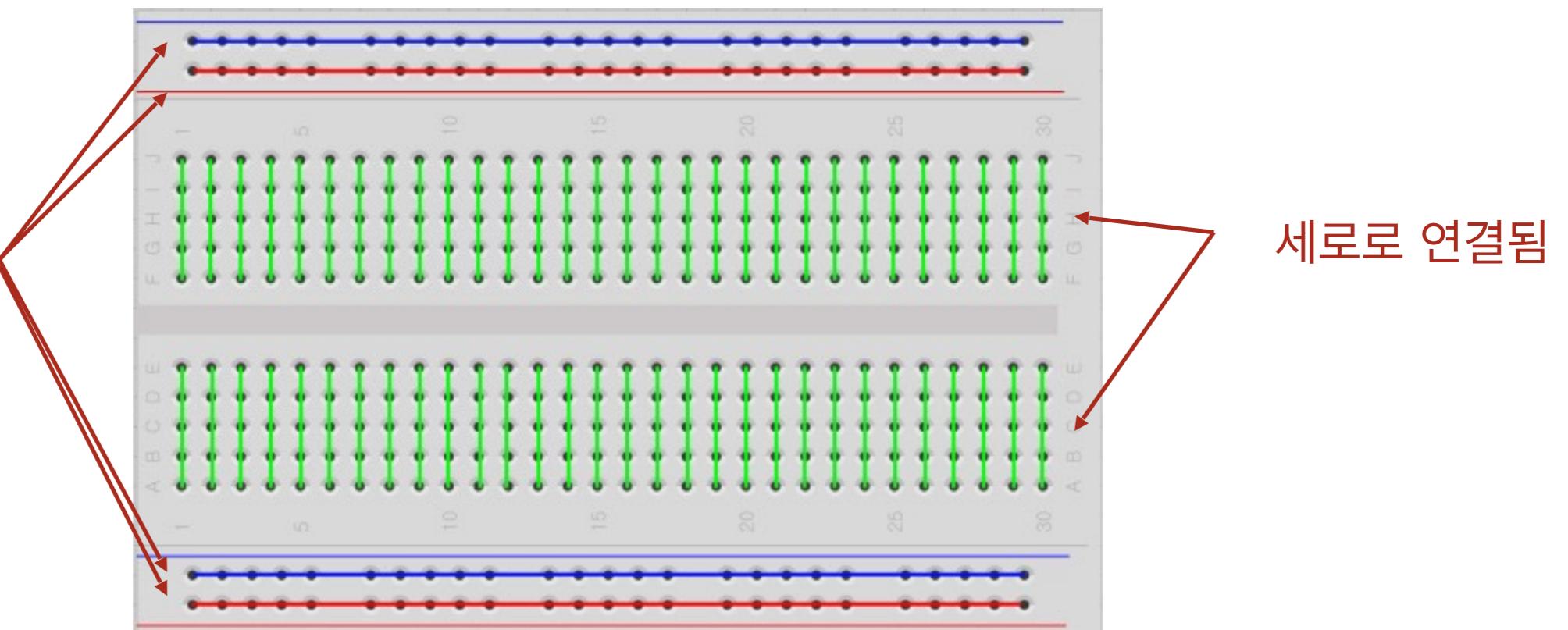
# **GPIO 입출력**

**Lesson 02**

# Bread board

## ❶ 소자들을 연결하기 위해 사용

가로로 연결됨  
(전원용)



세로로 연결됨

# 점프 와이어 (Jump Wire)

- 소자들을 연결하기 위한 전선



# 저항 (Register)

- 회로 내의 전류 흐름을 억제하는 소자

- 회로에 흐르는 전류의 양을 조절하는데 사용할 수 있음

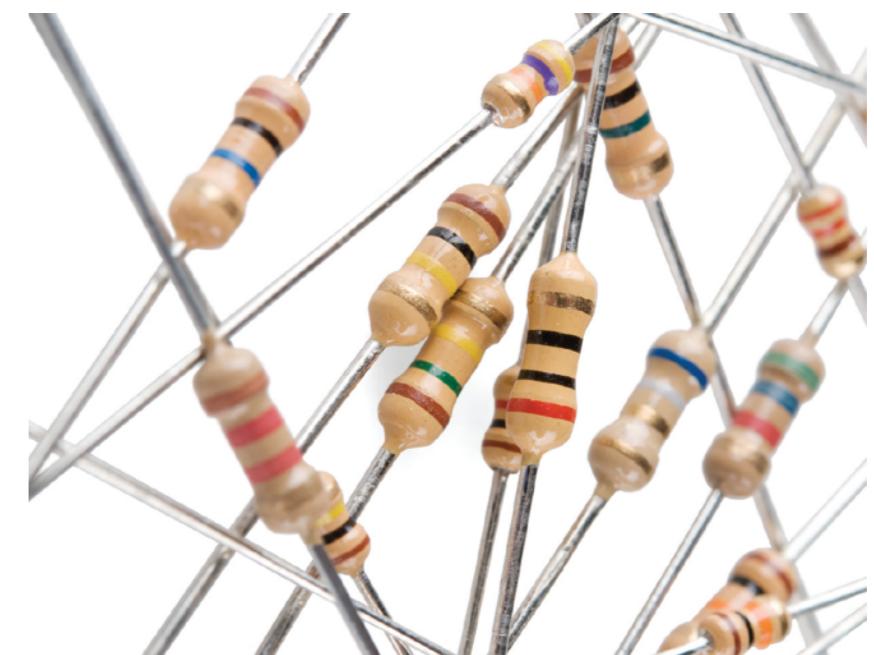
- 극성이 없음

- +/- 연결 방향에 무관

- 단위

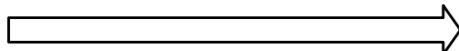
- 오옴 ( $\Omega$ )

- 회로 기호

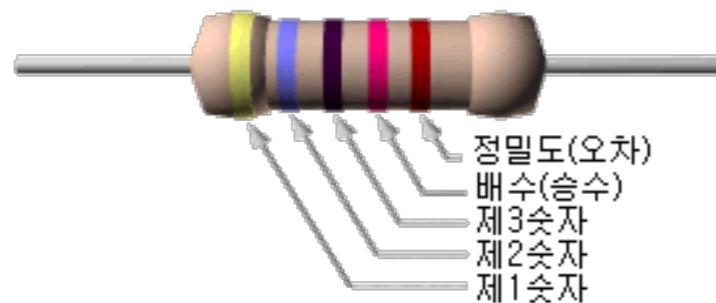


# 저항 읽는 법

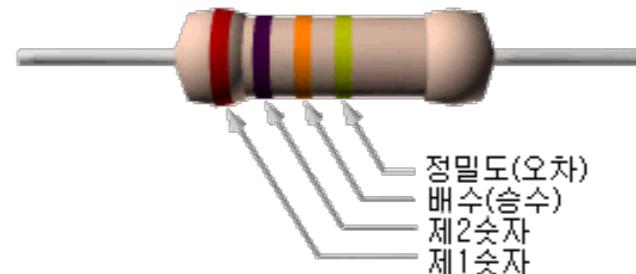
읽는 방향: 간격이 좀좀한 쪽부터 읽는다



5 Band 저항



4 Band 저항



이미지 출처: [hodorii.blogspot.com](http://hodorii.blogspot.com)

색상	첫째자리	둘째 자리	세째자리	10의 승수	오차
흑색(Black)	0	0	0	1	
갈색(Brown)	1	1	1	10	1%
적색(Red)	2	2	2	100	2%
오렌지색(Orange)	3	3	3	1000(1K)	
황색(Yellow)	4	4	4	10000(10K)	
녹색(Green)	5	5	5	100000(100K)	0.5%
청색(Blue)	6	6	6	1M	0.25%
자색(Violet)	7	7	7	10M	0.1%
회색(Gray)	8	8	8	100M	0.05%
백색(White)	9	9	9	1000M	
금색(Gold)	-	-	-	-	5%
은색(Silver)	-	-	-	-	10%

# 저항 읽기 예시

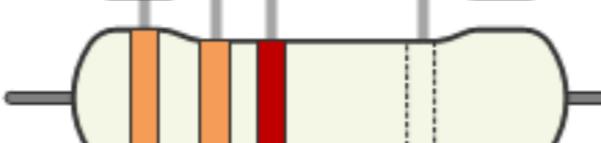
[www.resistorguide.com](http://www.resistorguide.com)

	Color	Significant figures			Multiply	Tolerance (%)	Temp. Coeff. (ppm/K)	Fail Rate (%)
Bad	black	0	0	0	$\times 1$		250 (U)	
Beer	brown	1	1	1	$\times 10$	1 (F)	100 (S)	1
Rots	red	2	2	2	$\times 100$	2 (G)	50 (R)	0.1
Our	orange	3	3	3	$\times 1K$		15 (P)	0.01
Young	yellow	4	4	4	$\times 10K$		25 (Q)	0.001
Guts	green	5	5	5	$\times 100K$	0.5 (D)	20 (Z)	
But	blue	6	6	6	$\times 1M$	0.25 (C)	10 (Z)	
Vodka	violet	7	7	7	$\times 10M$	0.1 (B)	5 (M)	
Goes	grey	8	8	8	$\times 100M$	0.05 (A)	1(K)	
Well	white	9	9	9	$\times 1G$			
Get	gold				$\times 0.1$	5 (J)		
Some	silver				$\times 0.01$	10 (K)		
Now!	none					20 (M)		

6 band   $3.21\text{k}\Omega$  1% 50ppm/K

5 band   $521\Omega$  1%

4 band   $82\text{k}\Omega$  5%

3 band   $330\Omega$  20%

gap between band 3 and 4  
indicates reading direction

# LED (Light Emitting Diode)

- 빛을 발산하는 반도체

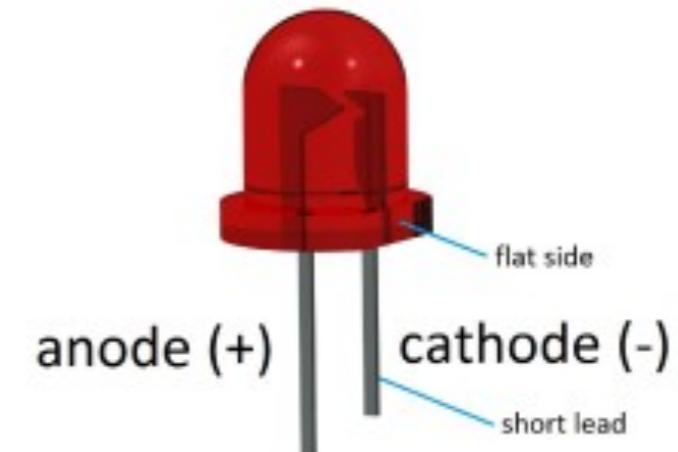
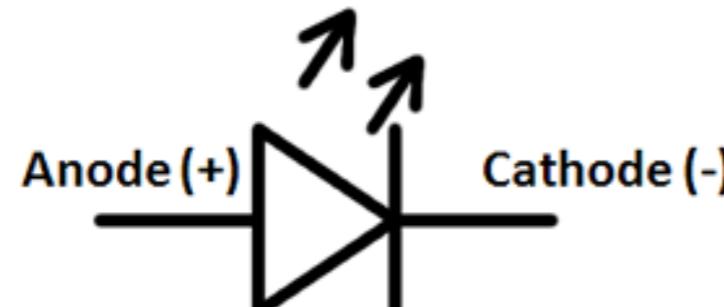
- 극성이 있음

- 전류가 한쪽 방향으로만 흐를 수 있음
  - (+)극과 (-)극을 정확히 연결해야 함

- 자체 저항이 매우 작으므로 전원과 연결할 때는 반드시 저항을 사용해야 한다

- 저항을 사용하지 않고 전원과 연결하면 과전류가 흘러 LED가 고장날 수 있음

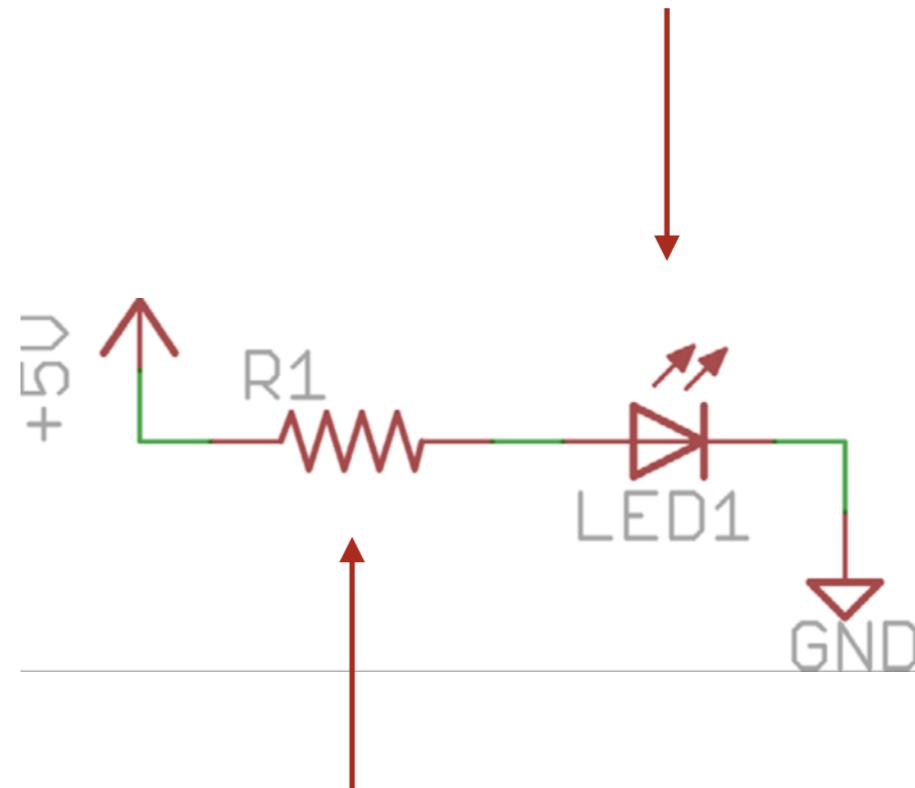
- 회로 기호



# Ohm's Law

$$V=IR$$

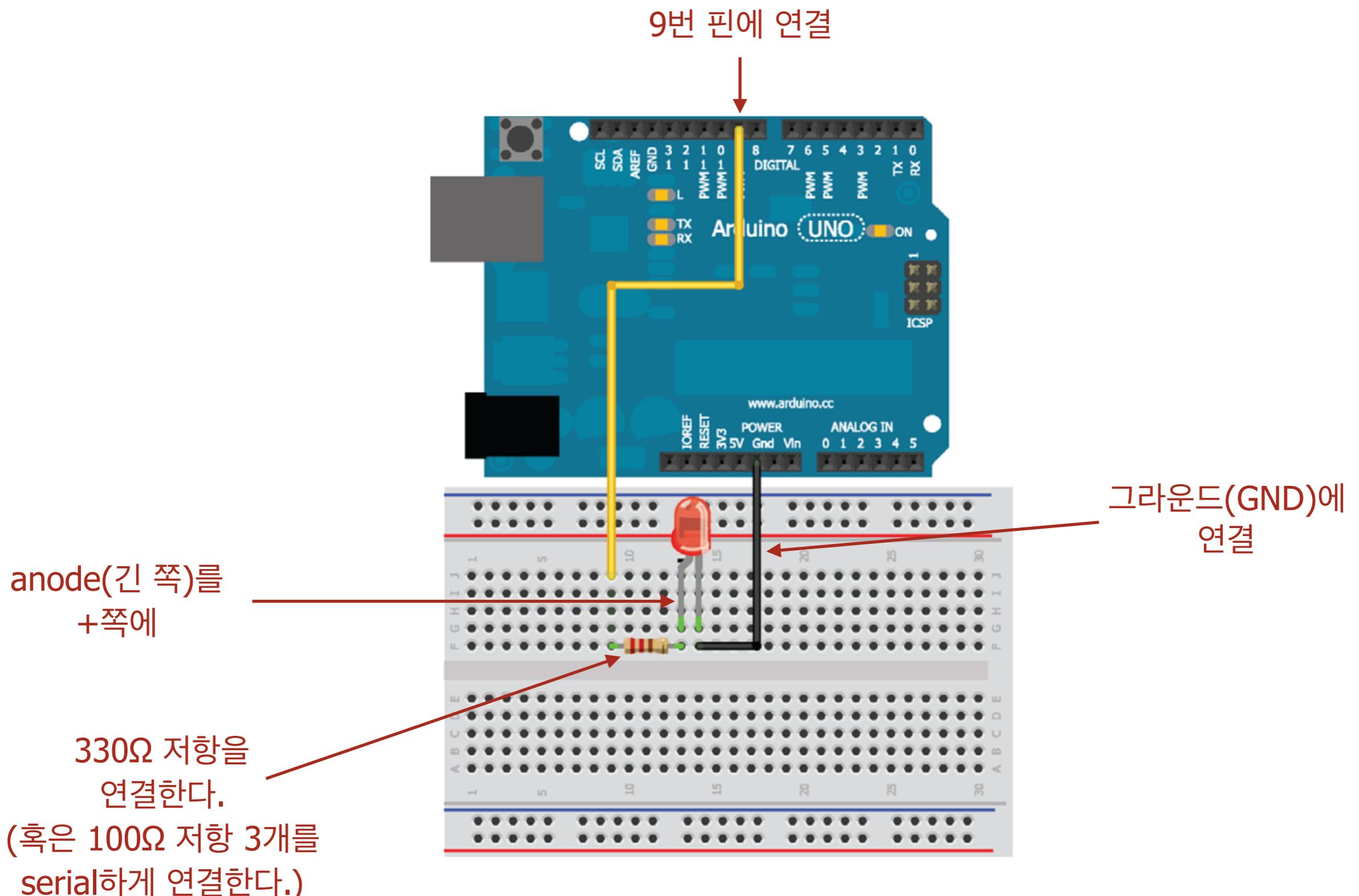
LED 자체의 저항은 매우 작다.



따라서 이렇게 저항을 추가하지 않고  
LED를 직접 5V와 GND에 연결하면  
LED가 감당할 수 없는 과전류가 흐르  
게된다. 330Ω 정도가 적당하다.

# 디지털 출력

# 회로구성



# Sketch 01: Blinking

9번 핀에 LED가 연결되어 있음



```
const int LED=9; // define LED for Pin 9

void setup()
{
    pinMode(LED, OUTPUT); // Set the LED pin as an output
}

void loop()
{
    digitalWrite(LED, HIGH);
    delay(1000);
    digitalWrite(LED, LOW);
    delay(1000);
}
```

# GPIO pin mode

- 아두이노의 디지털 핀들(0-13번)과 아날로그 핀들(A0-A5)은 입력 혹은 출력 용도로 사용 가능
  - default는 입력 모드
- 핀을 입력 혹은 출력 용도인지 지정해 주어야 한다.

`pinMode(pinNum, MODE)`



MODE는 INPUT, OUTPUT,  
혹은 INPUT\_PULLUP

## Sketch 02: PWM

```
const int LED=9;           //define LED for Pin 9

void setup()
{
    pinMode(LED, OUTPUT); //Set the LED pin as an output
}

void loop()
{
    for (int i=0; i<256; i++)
    {
        analogWrite(LED, i); ←
        delay(10);
    }

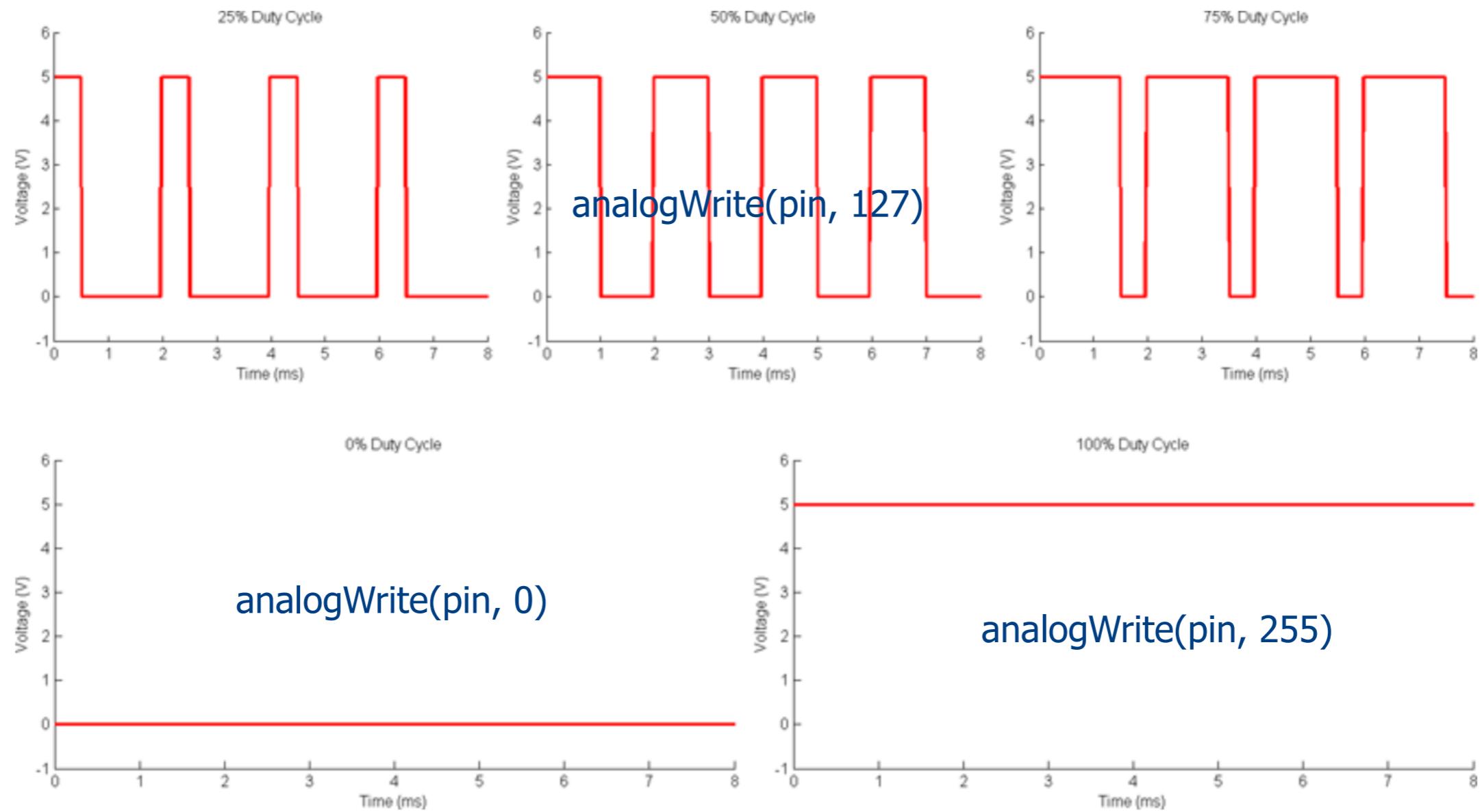
    for (int i=255; i>=0; i--)
    {
        analogWrite(LED, i);
        delay(10);
    }
}
```

9번 핀은 PWM을 사용하여 analog 출력을 지원한다  
이렇게 0에서 255사이의 값을 출력할 수 있다.

## PWM (Pulse Width Modulation)

- 아두이노는 **DAC**이 없으므로 아날로그 출력을 지원하지 않는다. 대신 **PWM**을 이용하여 유사한 효과를 얻을 수 있다.
- 아두이노 디지털 “~”로 표시된 포트(핀)들만 **PWM** 기능을 지원
  - 3, 5, 6, 9, 10, 11번 핀
- **Pulse Width**를 변경하여 0~255사이의 값을 표현
- 대표적인 응용: LED 전등의 밝기 조절 (**Dimming**)

# PWM with varying duty cycles



# Why millis() instead of delay()

- ☞ **delay() 함수는 busy waiting**

- ☞ delay()함수가 실행되는 동안 다른 기능은 중단됨
- ☞ 따라서 복잡한 일을 하는 sketch의 경우 바람직하지 않음
- ☞ **millis() 함수는 프로그램 시작 후 경과된 시간을 milisecond 단위로 반환**

## Sketch 3: Blank without delay()

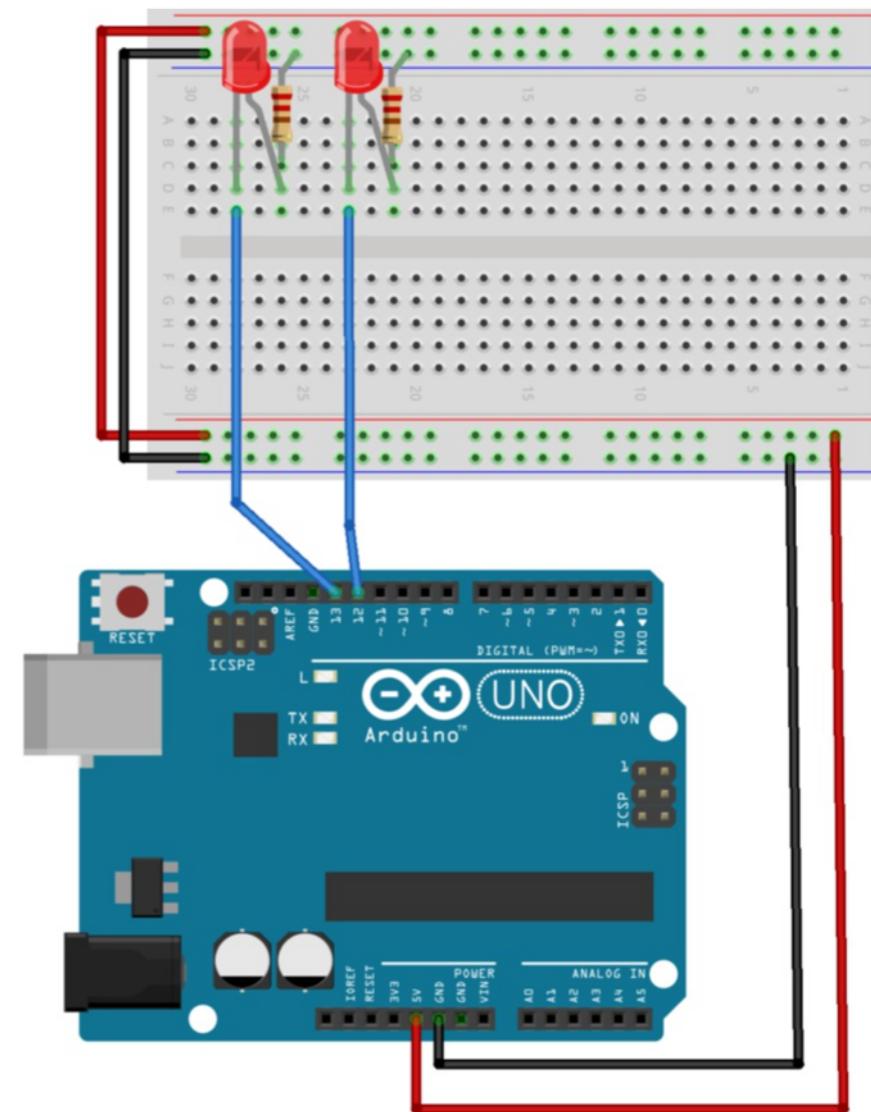
```
const int ledPin = 9;
int ledState = LOW;
long previousMillis = 0;
long interval = 1000;

void setup() {
    pinMode(ledPin, OUTPUT);
}

void loop() {
    unsigned long currentMillis = millis(); ← 현재 시각
    if(currentMillis - previousMillis > interval) {
        previousMillis = currentMillis;
        if (ledState == LOW)
            ledState = HIGH;
        else
            ledState = LOW;
        digitalWrite(ledPin, ledState);
    }
}
```

# Sketch 4: Object-Oriented Programming

```
class Flasher {  
  
    int ledPin;  
    long OnTime;  
    long OffTime;  
  
    int ledState;  
    unsigned long previousMillis;  
  
public:  
    Flasher(int pin, long on, long off)  
    {  
        ledPin = pin;  
        pinMode(ledPin, OUTPUT);  
        OnTime = on;  
        OffTime = off;  
        ledState = LOW;  
        previousMillis = 0;  
    }
```



2개의 LED를 12, 13번 핀에 연결

## Sketch 4: Object-Oriented Programming (계속)

```
void Update() {  
    unsigned long currentMillis = millis();  
    if((ledState==HIGH) && (currentMillis-previousMillis >= OnTime)) {  
        ledState = LOW;      // Turn it off  
        previousMillis = currentMillis; // Remember the time  
        digitalWrite(ledPin, ledState); // Update the actual LED  
    }  
    else if ((ledState == LOW) &&  
             (currentMillis - previousMillis >= OffTime)) {  
        ledState = HIGH;     // turn it on  
        previousMillis = currentMillis;  
        digitalWrite(ledPin, ledState);  
    }  
}
```

## Sketch 4: Object-Oriented Programming (계속)

```
Flasher led1(12, 100, 400);  
Flasher led2(13, 350, 350);
```

```
void setup() {  
}
```

```
void loop() {  
    led1.Update();  
    led2.Update();  
}
```

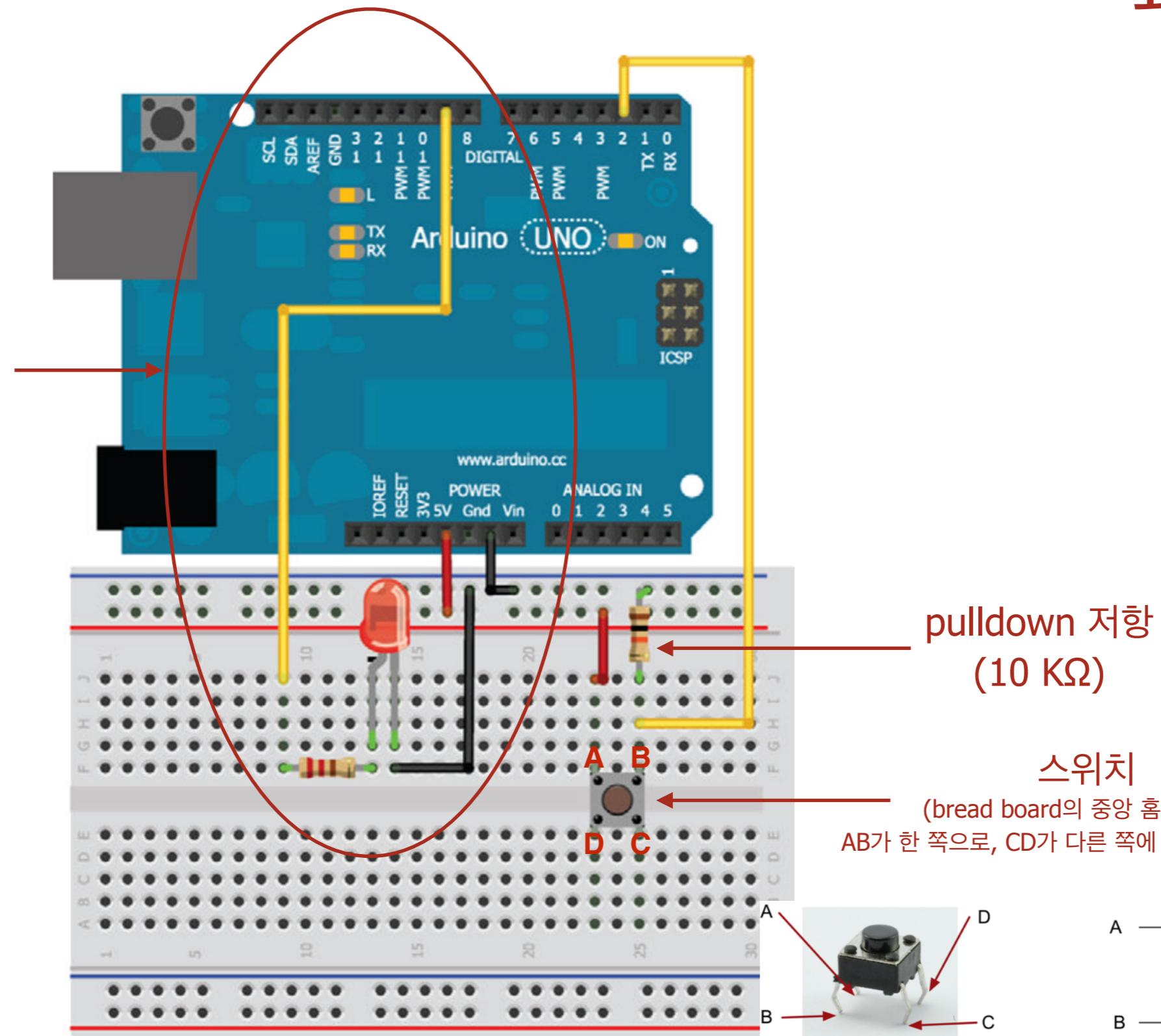
## 실습 과제 01

- ❶ 3개 이상의 LED를 연결 한 후 각각 서로 다른 주기로 점멸하도록 만들어라.
- ❷ 예를 들어서 3비트 이진 카운터를 만들어보자.

# **디지털 입력**

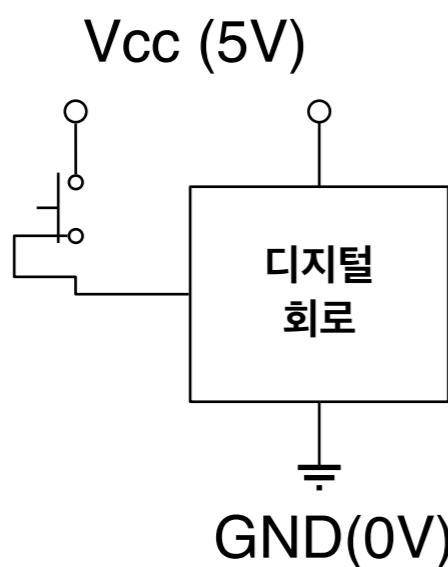
# 회로구성

LED쪽 연결은 실  
습01과 동일하다.

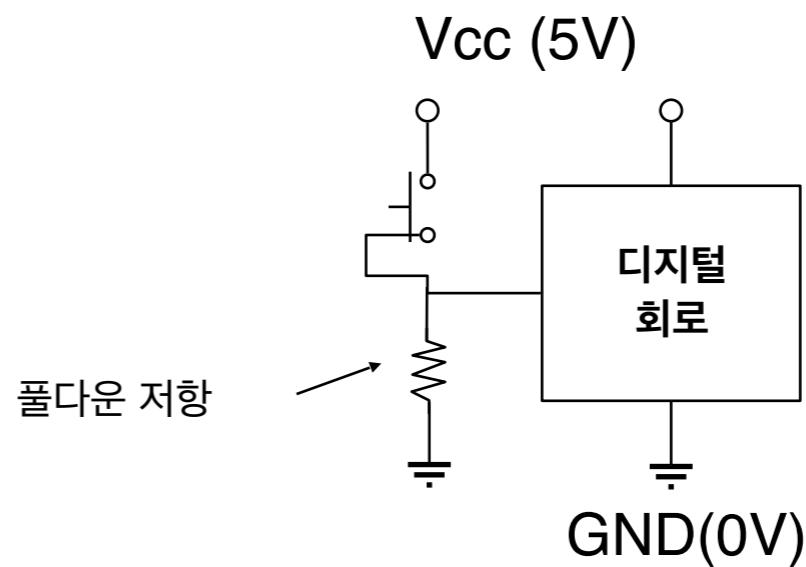


## 풀다운(pull-down) 저항

풀다운 저항을 사용하지 않은 경우



풀다운 저항을 사용한 경우

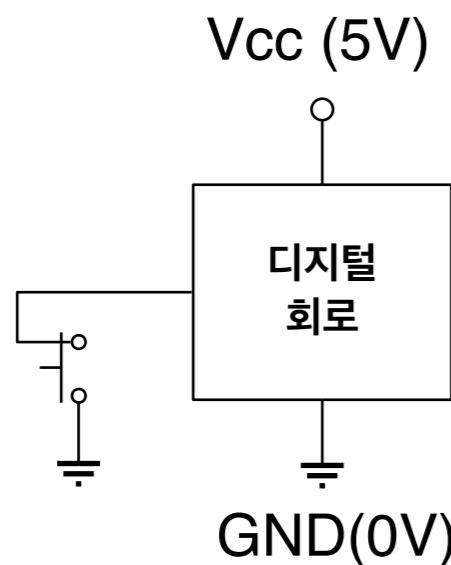


스위치 상태	스위치에서 회로에 전달되는 전압(전위차)
개방(open)	알 수 없음 (불안정)
폐쇄(close)	HIGH (5V)

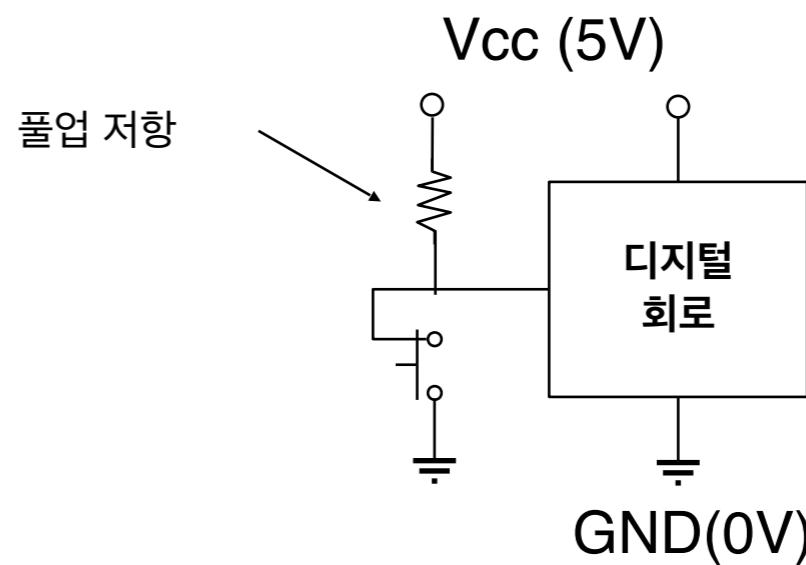
스위치 상태	스위치에서 회로에 전달되는 전압(전위차)
개방(open)	LOW (0V)
폐쇄(close)	HIGH (5V)

# 풀업(pull-up) 저항

풀업 저항을 사용하지 않은 경우



풀업 저항을 사용한 경우



스위치 상태	스위치에서 회로에 전달되는 전압(전위차)
개방(open)	알 수 없음 (불안정)
폐쇄(close)	LOW (0V)

스위치 상태	스위치에서 회로에 전달되는 전압(전위차)
개방(open)	HIGH (5V)
폐쇄(close)	LOW (0V)

## Sketch 05: 스위치를 이용한 LED 깜빡이기

```
const int LED=9;      //The LED is connected to pin 9
const int BUTTON=2;   //The Button is connected to pin 2
void setup()
{
    pinMode (LED, OUTPUT);
    pinMode (BUTTON, INPUT);
}

void loop()  {
    if (digitalRead(BUTTON) == LOW)  {
        digitalWrite(LED, LOW);
    }
    else {
        digitalWrite(LED, HIGH);
    }
}
```

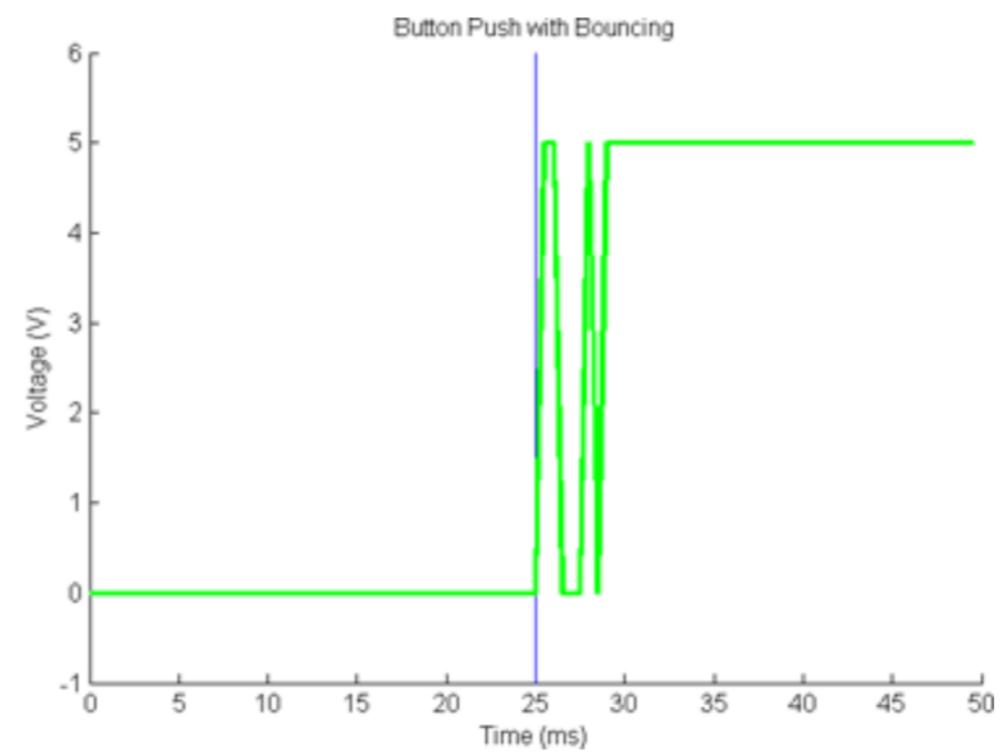
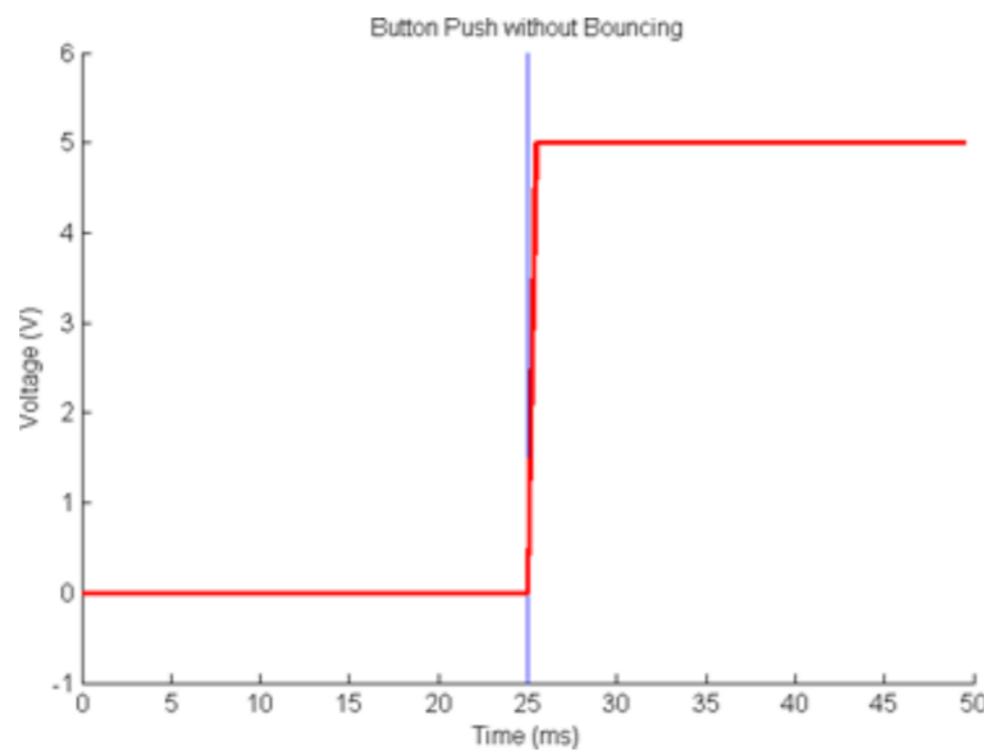
## Sketch 06: LED 토글(toggle)하기

```
const int LED=9;
const int BUTTON=2;
boolean lastButton = LOW;
boolean ledOn = false;

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

void loop() {
    boolean currentButton = digitalRead(BUTTON);
    if (currentButton == HIGH && lastButton == LOW) //if it was pressed...
    {
        ledOn = !ledOn;
    }
    lastButton = currentButton;
    digitalWrite(LED, ledOn);
}
```

# Bouncy Buttons



## Sketch 07: S/W Debounce

```
const int LED=9;                                //The LED is connected to pin 9
const int BUTTON=2;                             //The Button is connected to pin 2
boolean lastButton = LOW;                         // The previous button state
boolean currentButton = LOW;

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

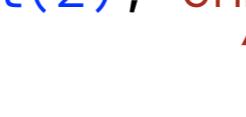
boolean debounce(boolean last)  {
  boolean current = digitalRead(BUTTON);
  if (last != current)  {
    delay(5);
    current = digitalRead(BUTTON);
  }
  return current;
}
```

## Sketch 07: S/W Debounce(계속)

```
void loop() {
    currentButton = debounce(lastButton);
    if (lastButton == LOW && currentButton == HIGH) //if it was pressed...
    {
        ledOn = !ledOn;
    }
    lastButton = currentButton;
    digitalWrite(LED, ledOn);
}
```

# Sketch 08: Using External Interrupt

```
volatile int state = LOW;  
const int LED = 9;      //The LED is connected to pin 9  
const int BUTTON = 2;   //The Button is connected to pin 2  
  
void setup(){  
    pinMode(LED, OUTPUT);  
    attachInterrupt(digitalPinToInterrupt(2), onPressed, RISING);  
}  
  
void loop() {  
    digitalWrite(LED, state);  
}  
  
void onPressed() {  
    static unsigned long lastMillis = 0; ←———— C 언어에서 static local 변수는  
    unsigned long newMillis = millis();    자신이 속한 함수가 처음 호출될 때  
    if (newMillis - lastMillis >= 50){        한 번 초기화된다. 그 다음부터는  
        state = !state;                      이전 호출에서의 값이 계속 보존된다.  
        lastMillis = newMillis;                (전역변수처럼 스택이 아닌 data section에 저장됨)  
    }  
}
```



INT0 (2번 핀에 고정)에 대해서  
onPressed 함수를 핸들러로 등록한다.

C 언어에서 static local 변수는  
자신이 속한 함수가 처음 호출될 때  
한 번 초기화된다. 그 다음부터는  
이전 호출에서의 값이 계속 보존된다.  
(전역변수처럼 스택이 아닌 data section에 저장됨)

# External Interrupt

- Arduino Uno의 경우 2개의 External interrupt를 제공:
- INT0 (2번 핀에 대응), INT1 (3번 핀에 대응)
- 인터럽트 Trigger 조건:
  - RISING: 0→1
  - FALLING: 1→0
  - CHANGE: 0→1 혹은 1→0
  - LOW: 0
- attachInterrupt 함수를 이용해서 인터럽트 서비스 루틴(ISR)을 등록

```
attachInterrupt(digitalPinToInterrupt(2), onPressed, RISING);
```



# Interrupt Service Routine (ISR)

- ⦿ 매개변수와 반환값을 가질 수 없음
- ⦿ 전역변수를 통해서 메인 프로그램과 데이터를 전달함
- ⦿ ISR에서 접근하는 전역변수는 **volatile**로 선언되어야 함
- ⦿ ISR이 실행중인 동안 다른 인터럽트는 무시됨.
  - ⦿ 따라서 ISR은 가능한 한 짧게
  - ⦿ delay()함수는 인터럽트에 의존하므로 ISR이 실행중인 동안에는 동작하지 않음
  - ⦿ millis() 함수는 ISR이 실행중인 동안에는 카운터가 증가하지 않음
  - ⦿ 반면 delayMicroseconds()는 인터럽트를 사용하지 않으므로 정상적으로 동작

(Currently, the largest value that will produce an accurate delay is 16383. This could change in future Arduino releases. For delays longer than a few thousand microseconds, you should use delay() instead.)

## 실습 과제 02

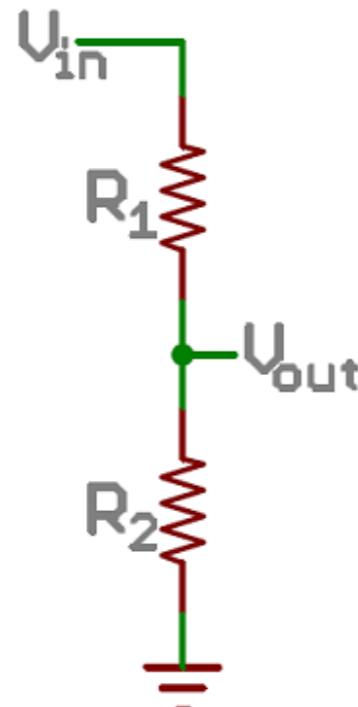
- **R, G, B 3개의 LED를 사용하여 버튼을 누를때 마다 번갈아가며 켜지도록 구현하라. delay 함수를 사용하지 않고 sketch를 작성하라.**

# 아날로그 입력

- ⦿ 가변저항 (Potentiometer)
- ⦿ 적외선 근접 센서 (Infrared Proximity Sensor)
- ⦿ 온도 센서 (Temperature Sensor)
- ⦿ 조도 센서 (Photocell)
- ⦿ 가속도 센서 (Accelerometer)
- ⦿ 자이로센서 (Gyroscope)

# 실습 01: Potentiometer (Pot)

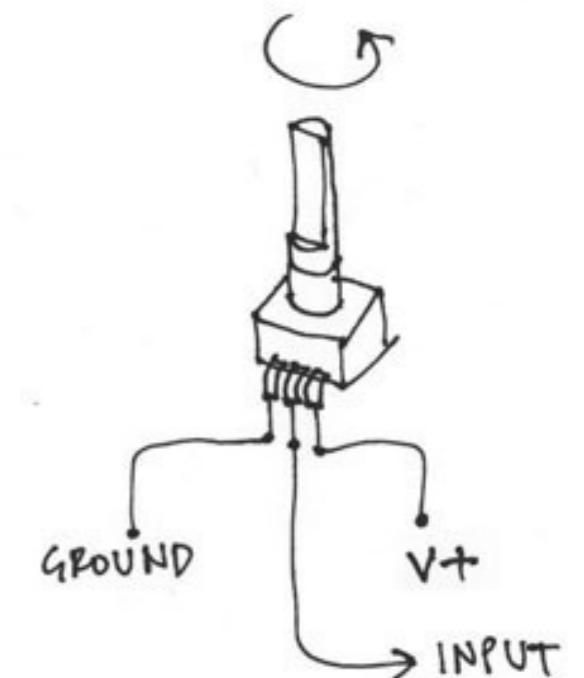
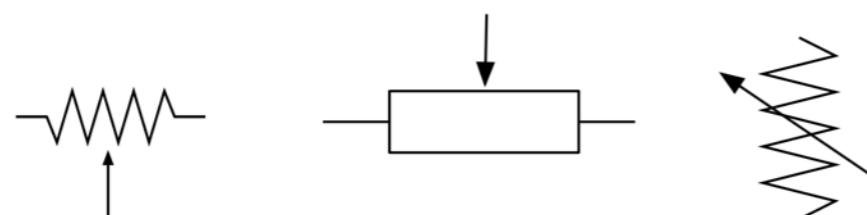
- 다이얼로 저항값의 크기를 조절할 수 있는 일종의 **voltage divider**
- **voltage divider**의 원리



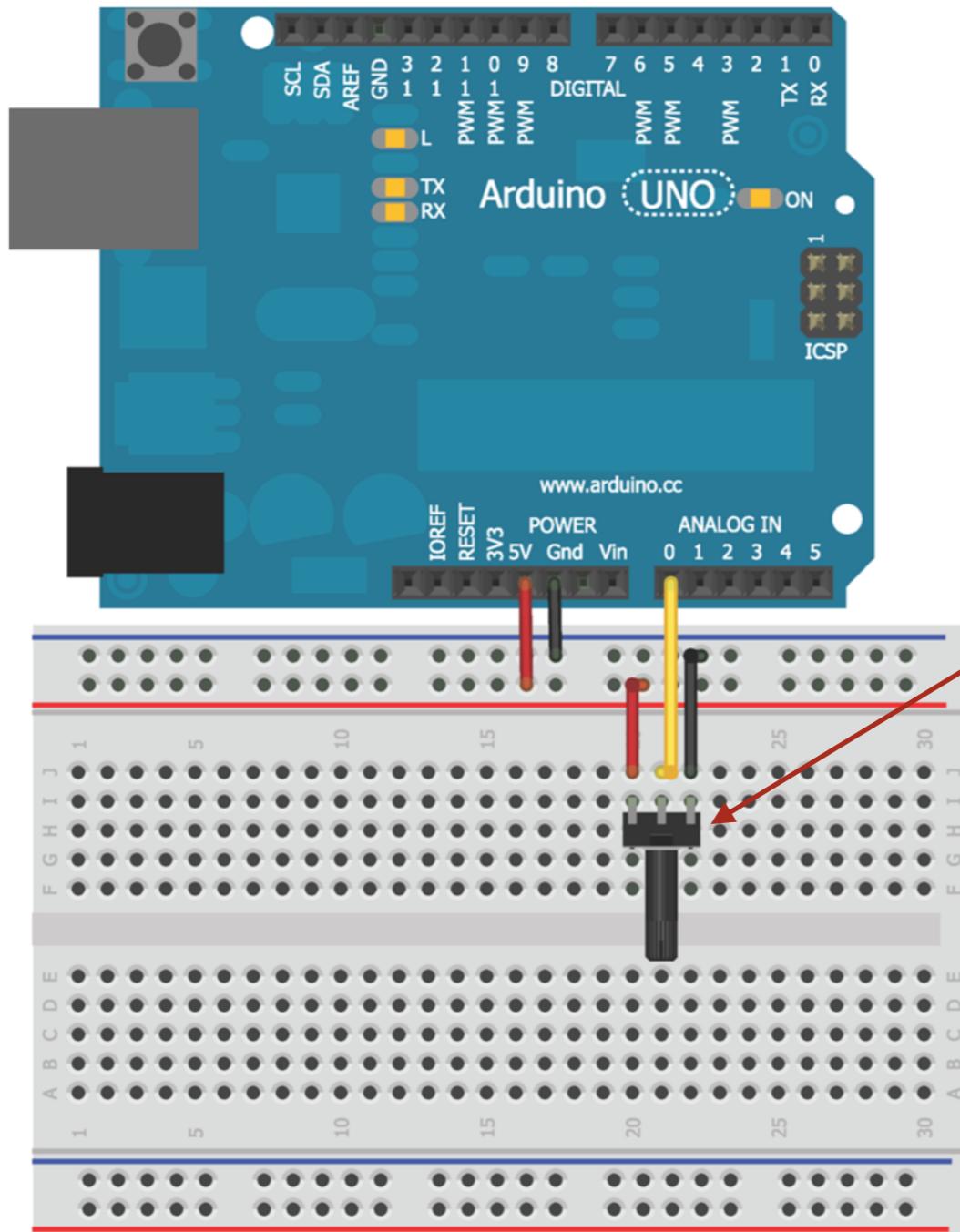
$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2}$$



- 기호:



# 실습 01: Potentiometer



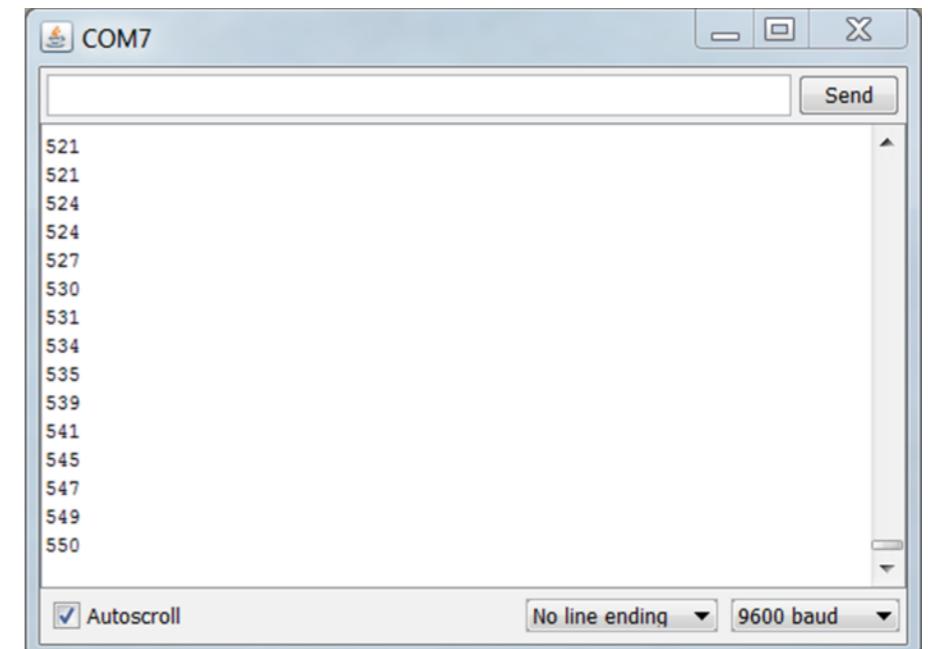
pot의 가운데 핀을 아날로그 입력핀 A0에 연결하고,  
좌우의 두 핀을 각각 5V와 GND에 연결한다.  
대칭적이므로 어느 쪽이 5V에 연결되어도 상관없다.

# Sketch 01: analogRead()로 pot 읽기

```
const int POT=0; // Pot on analog pin 0
int val = 0; // Variable to hold the analog reading from the POT

void setup()
{
    Serial.begin(9600);
}

void loop() {
    val = analogRead(POT); // 0에서 1023사이의 정수로 반환됨
    // (0은 0V, 1023은 5V에 대응)
    Serial.println(val);
    delay(500);
}
```



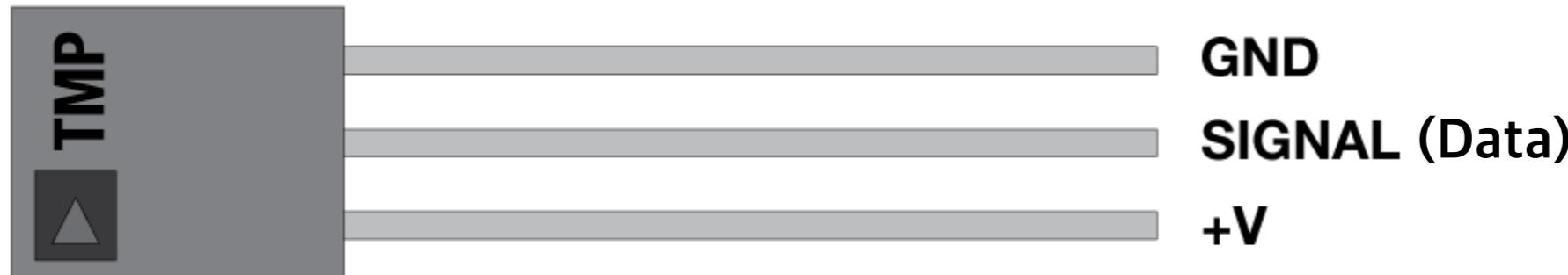
## 실습 과제 03

- Pot의 값(0~1023)을 읽고 그 값에 따라 LED의 밝기(0~255)를 조절하라.
- Arduino IDE는 2가지 유용한 utility 함수를 제공:

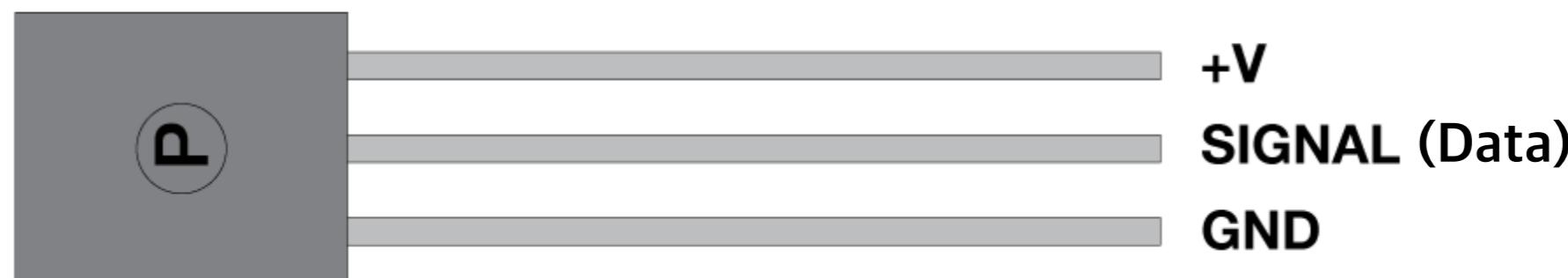
이 범위에 속한 값 value를  
output = map(value, fromLow, fromHigh, toLow, toHigh);  
이 구간으로 선형변환(linear transform)해준다.

output = constrain(value, min, max);  
value가 이 구간을 벗어날 경우 min혹은 max로 대체해준다.

## 실습 02: 온도 센서



FRONT



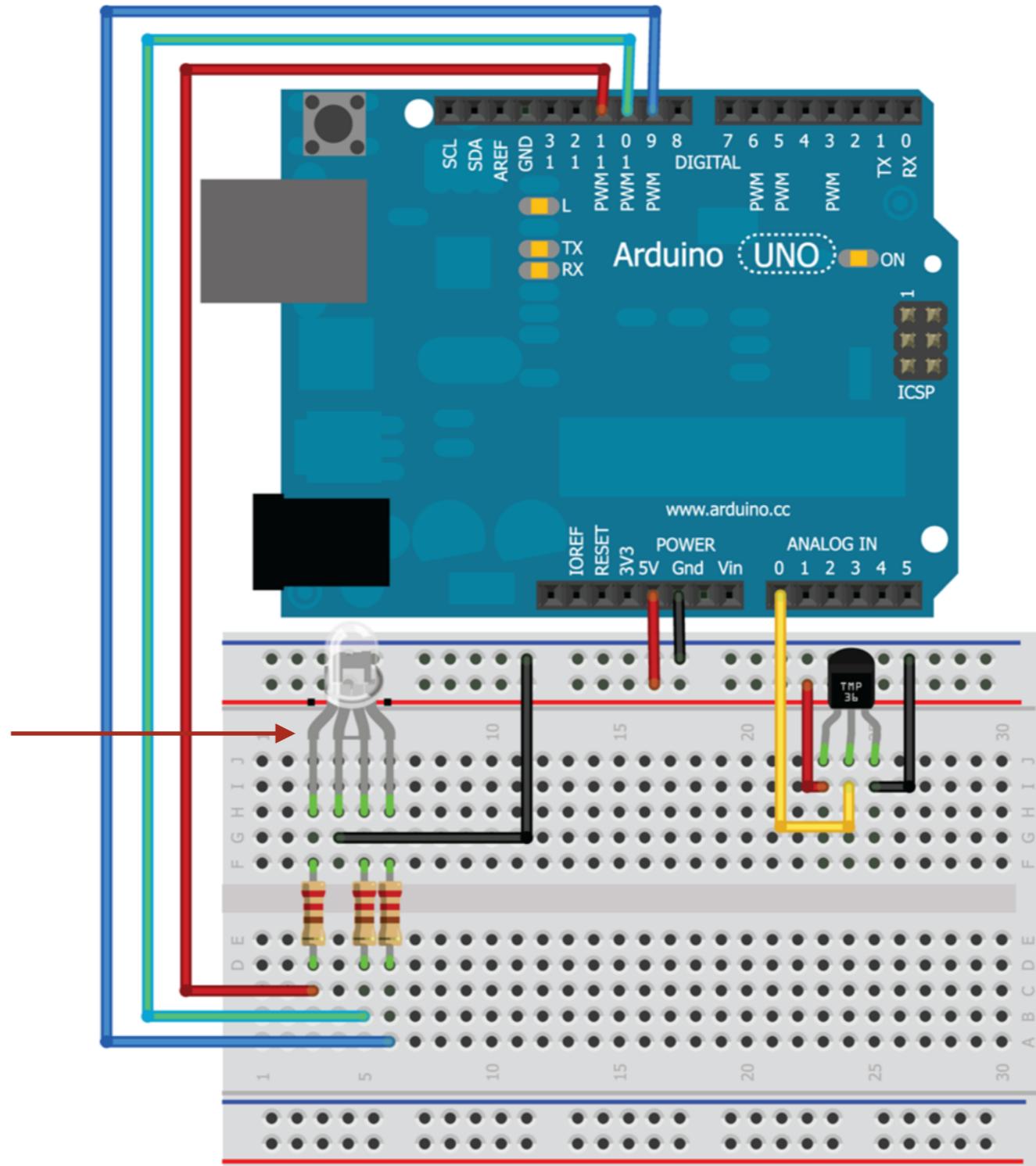
BACK



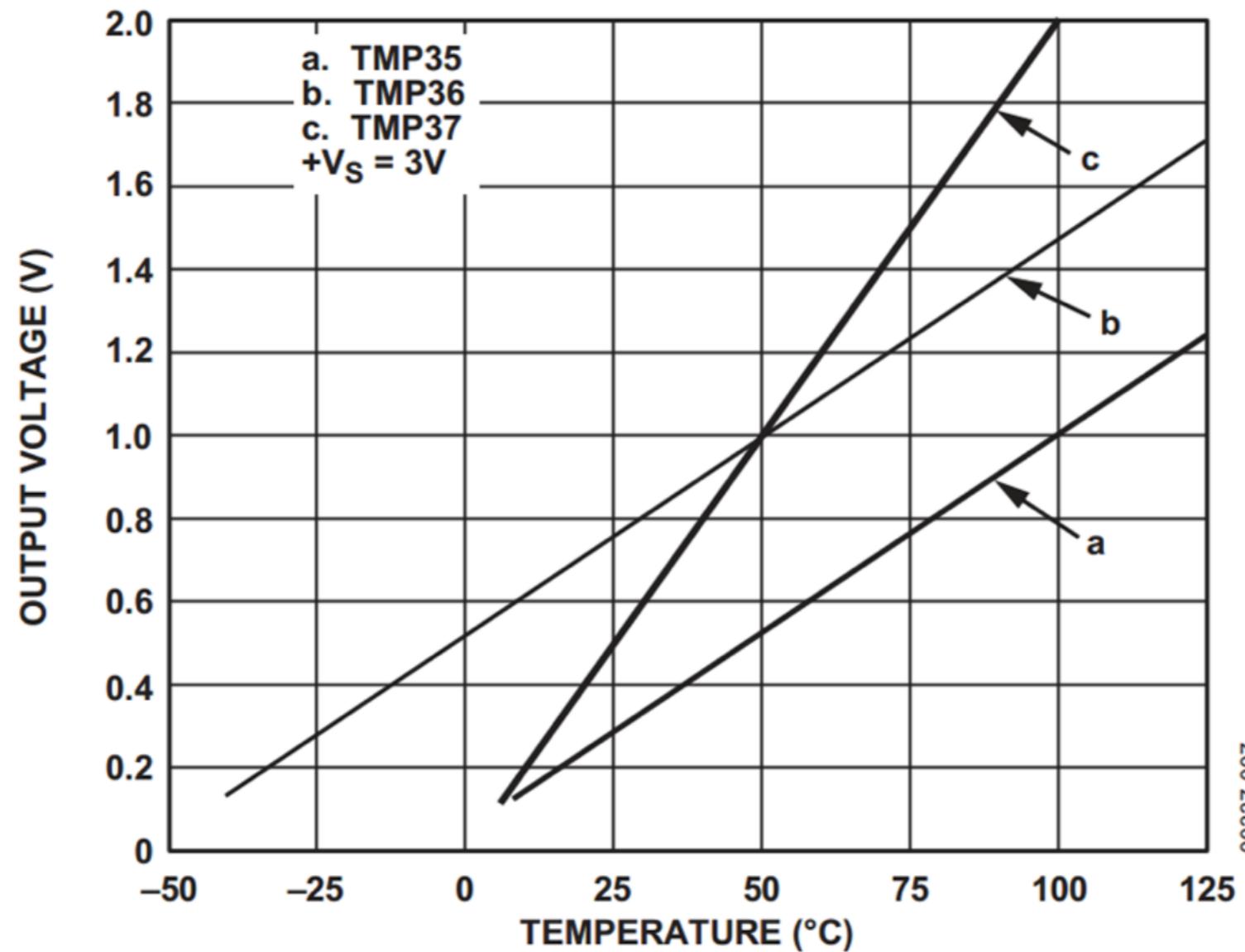
## 실습 02: 온도센서

온도가 일정값 이하이면  
파란 LED, 일정값 이상이  
면 빨간 LED, 중간이면 녹  
색 LED를 점등한다.

우리는 3색 LED가  
없으므로 3개의 LED  
로 대체한다.



## 실습 02: 온도값 매핑



센서마다 특성이 다르므로 Data Sheet를 참고하거나  
실험적으로 알아내야 한다.

## Sketch 02: 온도 경고등

```
const int BLED=9;      //Blue LED on pin 9
const int GLED=10;     //Green LED on pin 10
const int RLED=11;     //Red LED on pin 11
const int TEMP=A0;     //Temp Sensor is on pin A0

const int LOWER_BOUND=139; //Lower Threshold
const int UPPER_BOUND=147; //Upper Threshold
int val = 0;            //Variable to hold analog reading

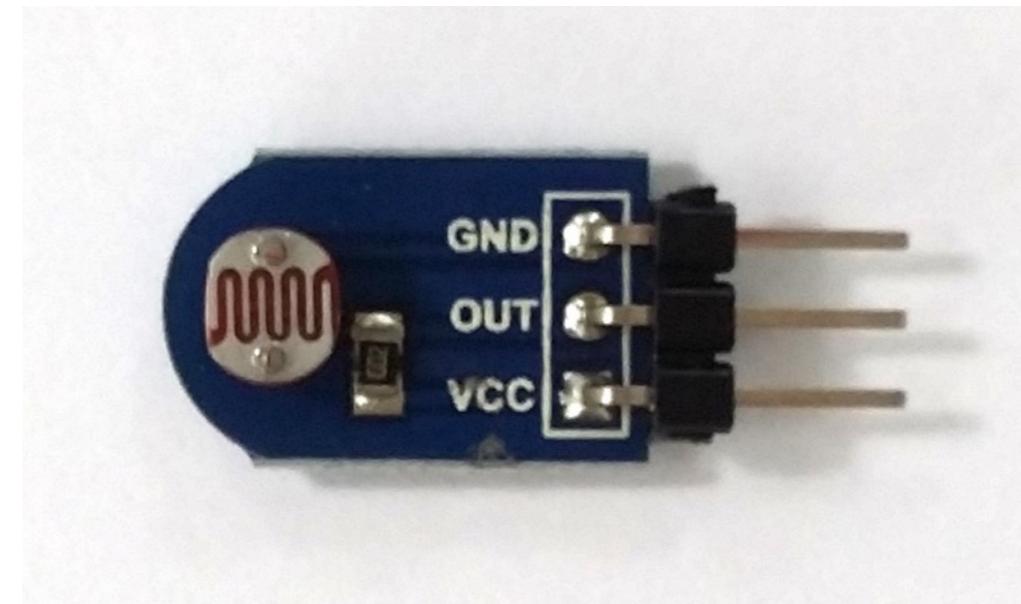
void setup()
{
    pinMode (BLED, OUTPUT); //Set Blue LED as Output
    pinMode (GLED, OUTPUT); //Set Green LED as Output
    pinMode (RLED, OUTPUT); //Set Red LED as Output
}
```

## Sketch 02: 온도 경고등

```
void loop() {
    val = analogRead(TEMP);
    if (val < LOWER_BOUND)
    {
        digitalWrite(RLED, LOW);
        digitalWrite(GLED, LOW);
        digitalWrite(BLED, HIGH);
    }
    else if (val > UPPER_BOUND)
    {
        digitalWrite(RLED, HIGH);
        digitalWrite(GLED, LOW);
        digitalWrite(BLED, LOW);
    }
    else {
        digitalWrite(RLED, LOW);
        digitalWrite(GLED, HIGH);
        digitalWrite(BLED, LOW);
    }
}
```

## 실습과제 04: 조도센서

- 조도센서의 값을 읽고 밝을 수록 LED를 어둡게, 어두울 수록 LED를 밝게 만드는 회로와 sketch를 작성하라.



# **Arduino 프로그래밍 언어 개요**

The screenshot shows the Arduino website's "Language Reference" page. At the top, there's a navigation bar with links for Home, Buy, Download, Products, Learning (with a dropdown menu), Forum, Support, Blog, LOG IN, and SIGN UP. A search bar is also at the top. Below the navigation, there are links for Reference, Language, Libraries, and Company. The main content area is titled "Language Reference". On the left, there's a sidebar with a red arrow pointing to the "Learning" tab, which is highlighted in blue. The main content is divided into three columns: "Structure", "Variables", and "Functions".

### Learning/Reference탭

Structure	Variables	Functions
- setup() - loop()	<b>Constants</b> - HIGH   LOW - INPUT   OUTPUT   INPUT_PULLUP - LED_BUILTIN - true   false - integer constants - floating point constants	Digital I/O - pinMode() - digitalWrite() - digitalRead()
<b>Control Structures</b> - if - if...else - for - switch case - while - do... while - break - continue - return - goto	<b>Data Types</b> - void - boolean - char - unsigned char - byte - int - unsigned int - word - long - unsigned long - short - float - double - string - char array - String - object - array	Analog I/O - analogReference() - analogRead() - analogWrite() - PWM
<b>Further Syntax</b> - ; (semicolon) - {} (curly braces) - // (single line comment) - /* */ (multi-line comment) - #define - #include	<b>Conversion</b> - char() - byte() - int()	Due & Zero only - analogReadResolution() - analogWriteResolution()
<b>Arithmetic Operators</b> - = (assignment operator) - + (addition) - - (subtraction) - * (multiplication) - / (division) - % (modulo)		Advanced I/O - tone() - noTone() - shiftOut() - shiftIn() - pulseIn()
		Time - millis() - micros() - delay() - delayMicroseconds()
		Math - min() - max()

# 상수 (Constants)

## ➊ Digital

- ➏ HIGH, LOW
- ➏ true, false

## ➋ GPIO 설정

- ➏ INPUT, OUTPUT, INPUT\_PULLUP

## ➌ Numeric

- ➏ Integer: B11010101, -123, 123uL, 0x3C, 0123
- ➏ Float: -1.2, 1.7e5, -62E-12

# Data Types

<b>int</b> 16b	<b>long</b> 32b	<b>char</b> 8b	<b>float</b> 32b
<b>unsigned int</b> 16b	<b>unsigned long</b> 32b	<b>boolean</b> 1b/8b	<b>double</b> 32b
<b>word</b> 16b	<b>byte</b> 8b	<b>string</b>	<b>array</b>
<b>short</b> 16b	<b>unsigned char</b> 8b	<b>String</b>	<b>void</b>

# I/O Functions

Digital	Digital	Analog
<b>pinMode</b> 0,...,13, A0,...,A5	<b>tone</b> 0,...,13, A0,...,A5	<b>analogReference</b>
<b>digitalWrite</b> 0,...,13, A0,...,A5	<b>noTone</b> 0,...,13, A0,...,A5	<b>analogRead</b> A0,...,A5
<b>digitalRead</b> 0,...,13, A0,...,A5	<b>pulseIn</b> 0,...,13, A0,...,A5	<b>analogWrite (PWM)</b> 3,5,6,7,10,11

# Interrupts

- ⌚ **attachInterrupt**
- ⌚ **detachInterrupt**
- ⌚ **interrupts()**
- ⌚ **noInterrupts()**

# Bits and Bytes

- ⦿ **lowByte**
- ⦿ **highByte**
- ⦿ **bitRead**
- ⦿ **bitWrite**
- ⦿ **bitSet**
- ⦿ **bitClear**
- ⦿ **bit**

# Serial

<b>available()</b>	<b>parseFloat()</b>	<b>println()</b>
<b>begin()</b>	<b>parseInt()</b>	<b>write()</b>
<b>end()</b>	<b>peek()</b>	<b>read()</b>
<b>find()</b>	<b>setTimeout()</b>	<b>readBytes()</b>
<b>findUntil()</b>	<b>serialEvent()</b>	<b>readBytesUntil()</b>
<b>flush()</b>	<b>print()</b>	

# Math and Timing

MATH			TIMING
<b>min()</b>	<b>constrain()</b>	<b>sqrt()</b>	<b>millis()</b>
<b>max()</b>	<b>map()</b>	<b>random()</b>	<b>micros()</b>
<b>abs()</b>	<b>pow()</b>	<b>randomSeed()</b>	<b>delay()</b>
<b>sin()</b>	<b>cos()</b>	<b>tan()</b>	<b>delayMicroseconds()</b>