

# 사물인터넷(IoT)

## 2. 옴의 법칙과 브레드 보드 활용 기본

# 목차

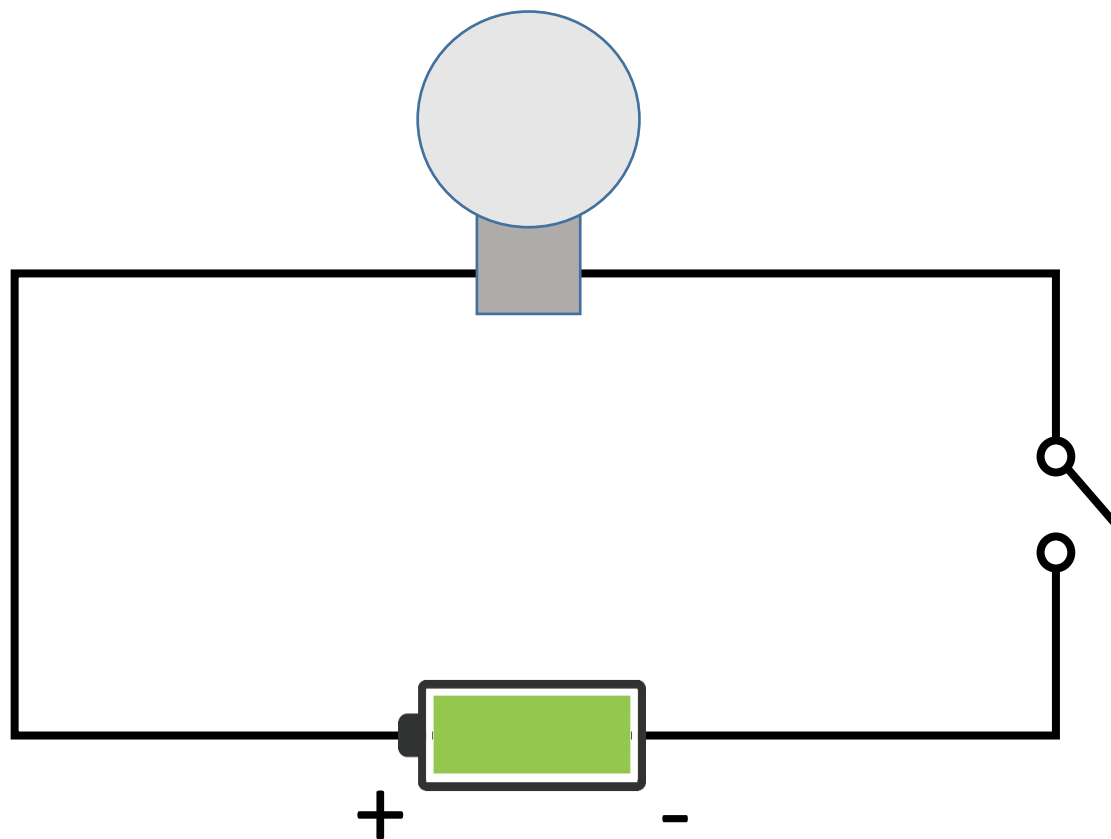
- 전자회로 용어
- 옴의 법칙( $V=IR$ )
- 저항읽기
- 브레드 보드 사용법 (fritzing 설치, 활용)
- 다이오드, LED 소개
- 기본함수: pinMode(), digitalWrite(), delay()
- 회로제작(LED 점멸 제어)
- 스케치 작성

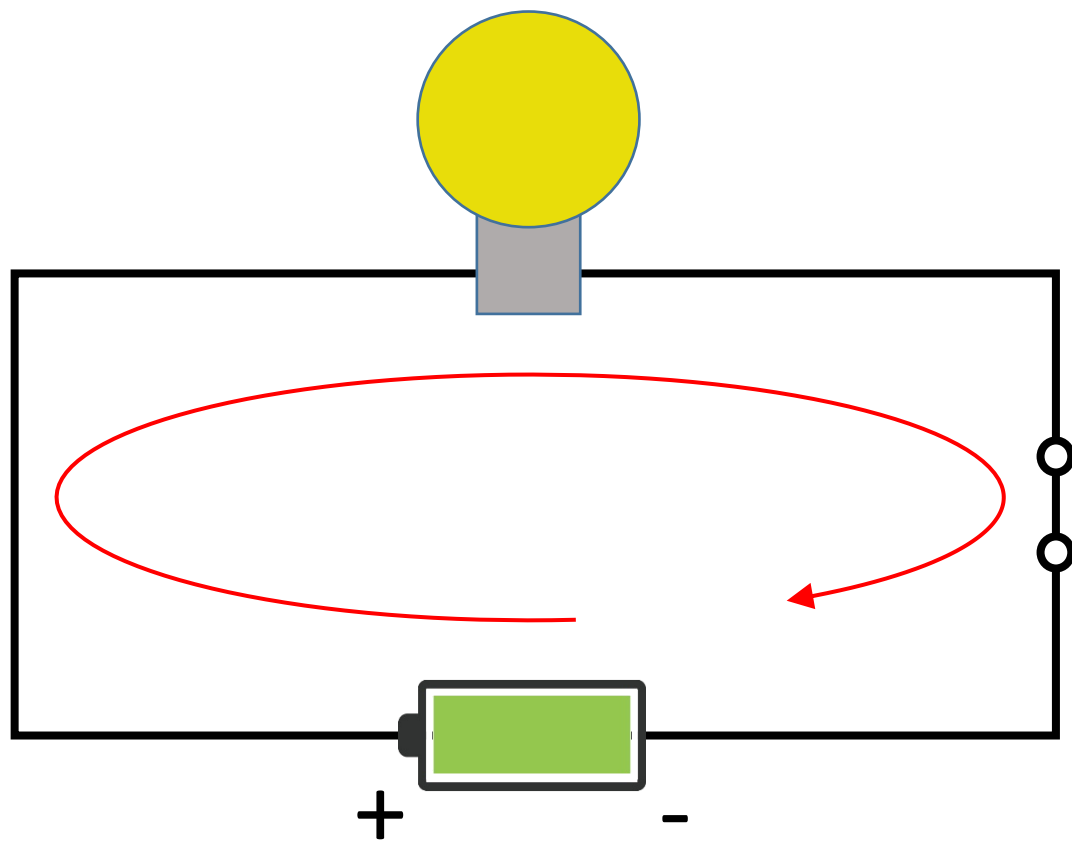
# 회로(Circuit)

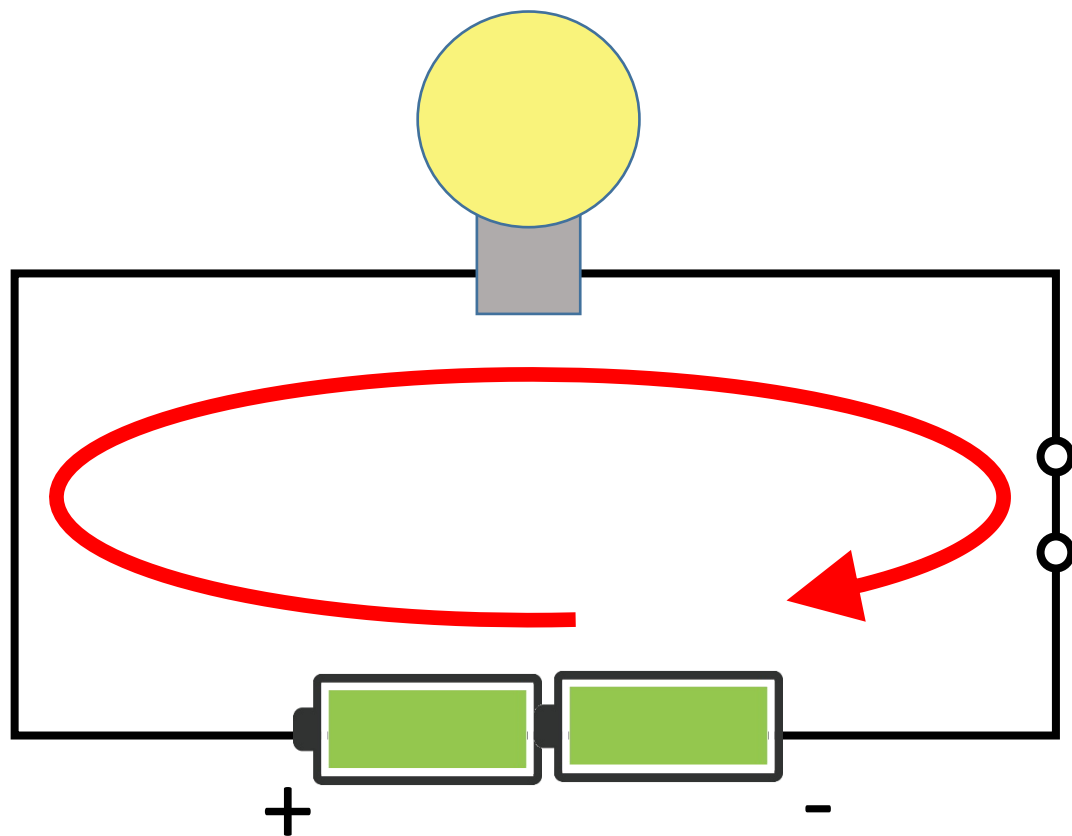
- 기능을 수행하기 위해 **전자 부품**들이 서로 **연결**되어 있는 것
- 전원(VCC)과 접지(GND)는 반드시 존재
- 중간 끊어짐 없이 전원에서 출발하여 접지까지 연결되어야 함
- 각 부품들의 정격에 맞는 전압과 전류가 연결되어야 함

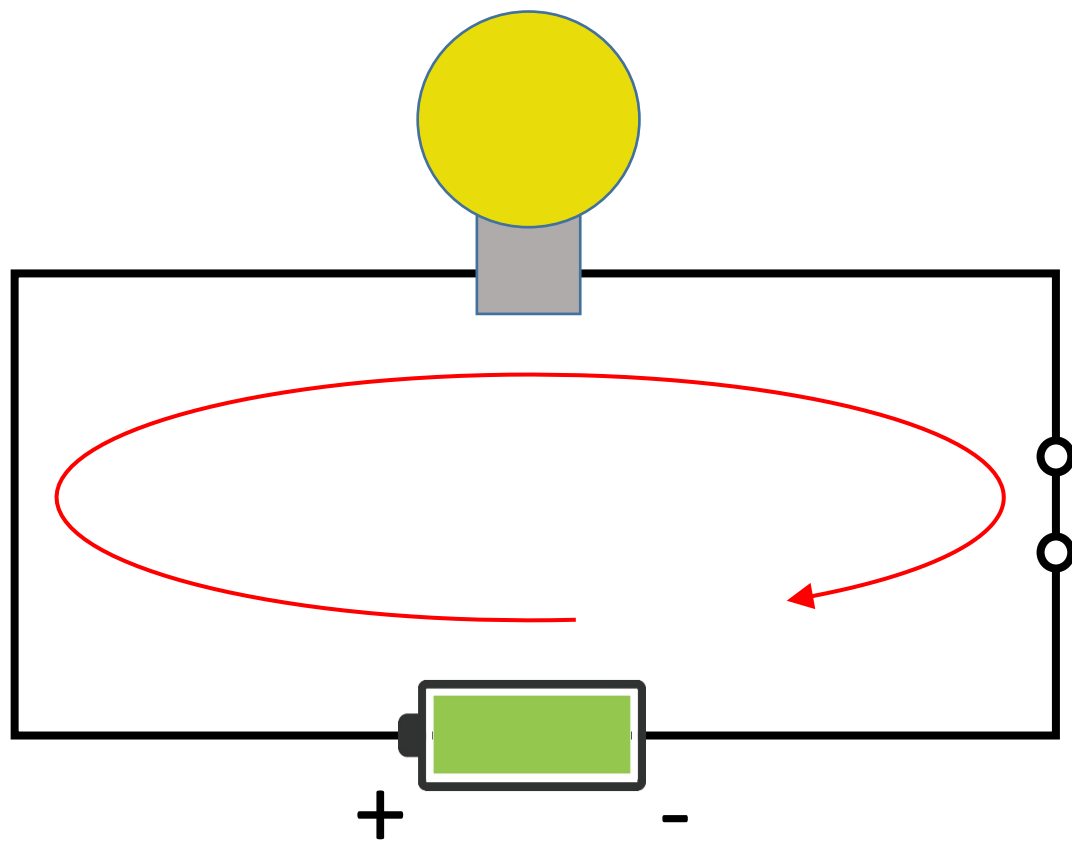
## 정격(定格)

- 전기 기기 또는 그 밖의 기기의 정격은 지정된 조건하에서의 사용 한도 의미함
- 각 기기는 정격상태에서 가장 잘 동작할 수 있도록 설계된 것이므로 정격에 주의해서 사용해야 함

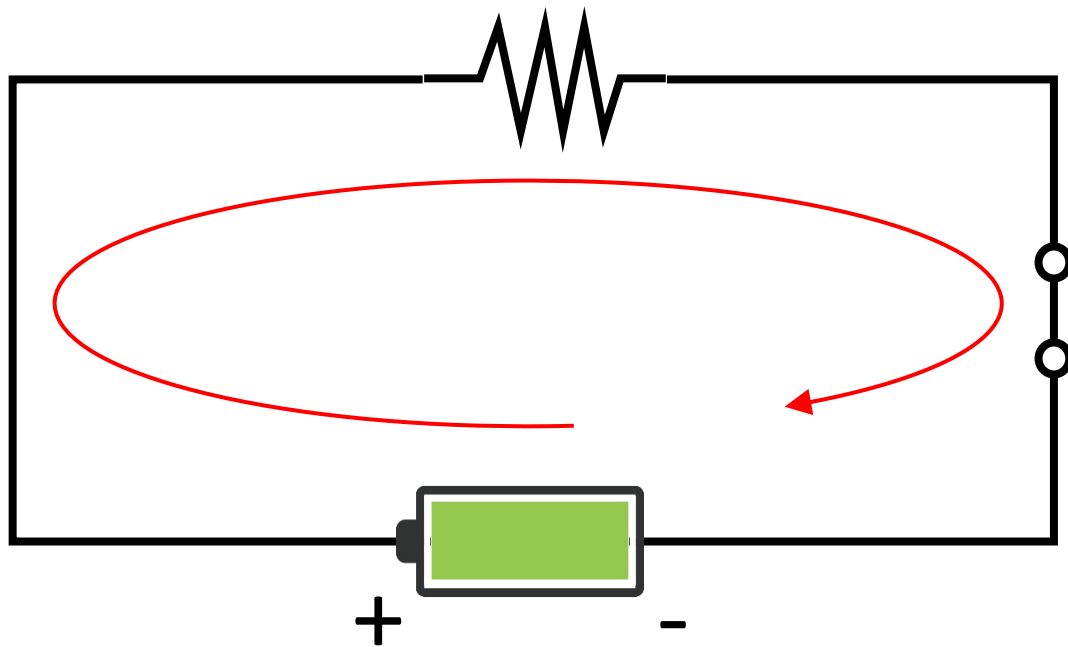








$R (\Omega)$

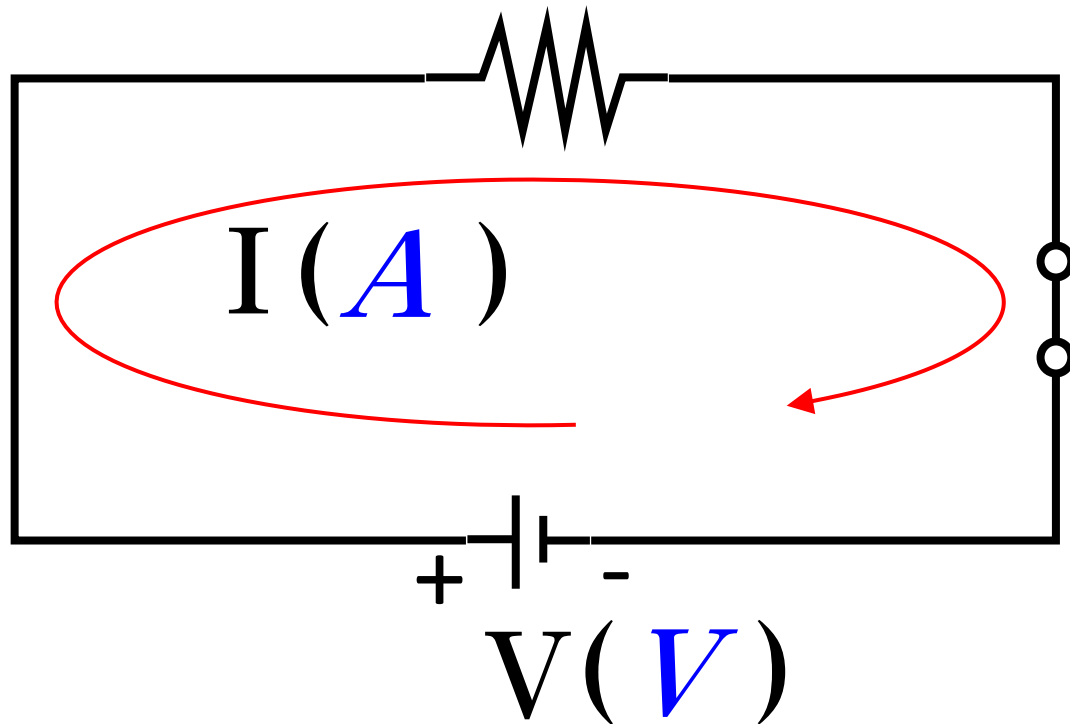




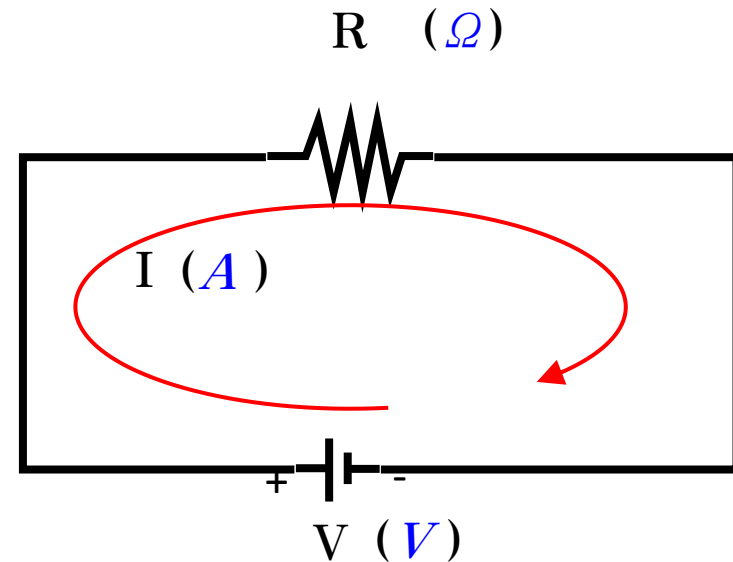
# 옴의 법칙 (Ohm's Law)

$$V = IR$$

$R (\Omega)$



- 전압(V)
  - 전류를 흐르게 하는 전기적인 압력
  - 단위 : 볼트 [V]
- 전류(I)
  - 단위 시간에 통과하는 전하의 양
  - 단위 : 암페어 [A]
- 저항(R)
  - 전류의 흐름을 방해하는 성질
  - 단위 : 옴 [ $\Omega$ ]

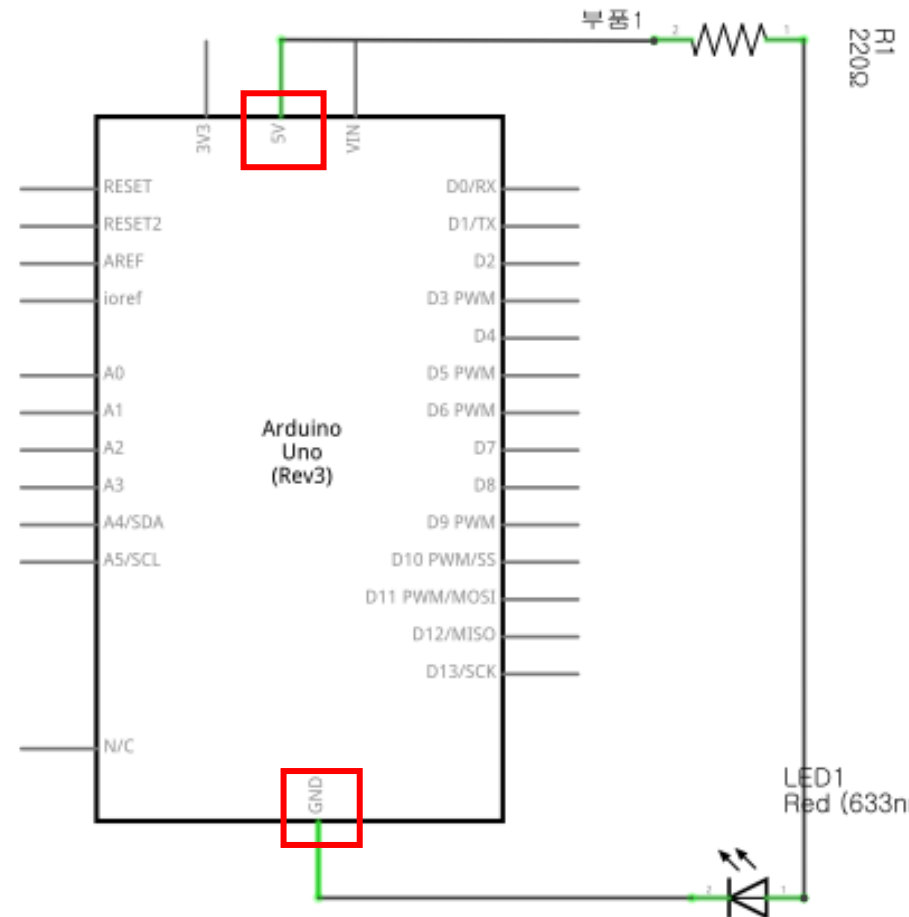


- 전원

- 회로에 전압을 만들어 줌
- 배터리의 (+)극이 전원
- VCC 등으로 표기

- 접지

- 전압의 크기를 결정
- 배터리의 (-)극이 접지
- GND로 표기
- 모든 회로는 같은 접지를 사용해야함

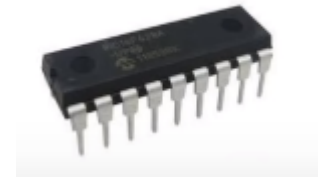
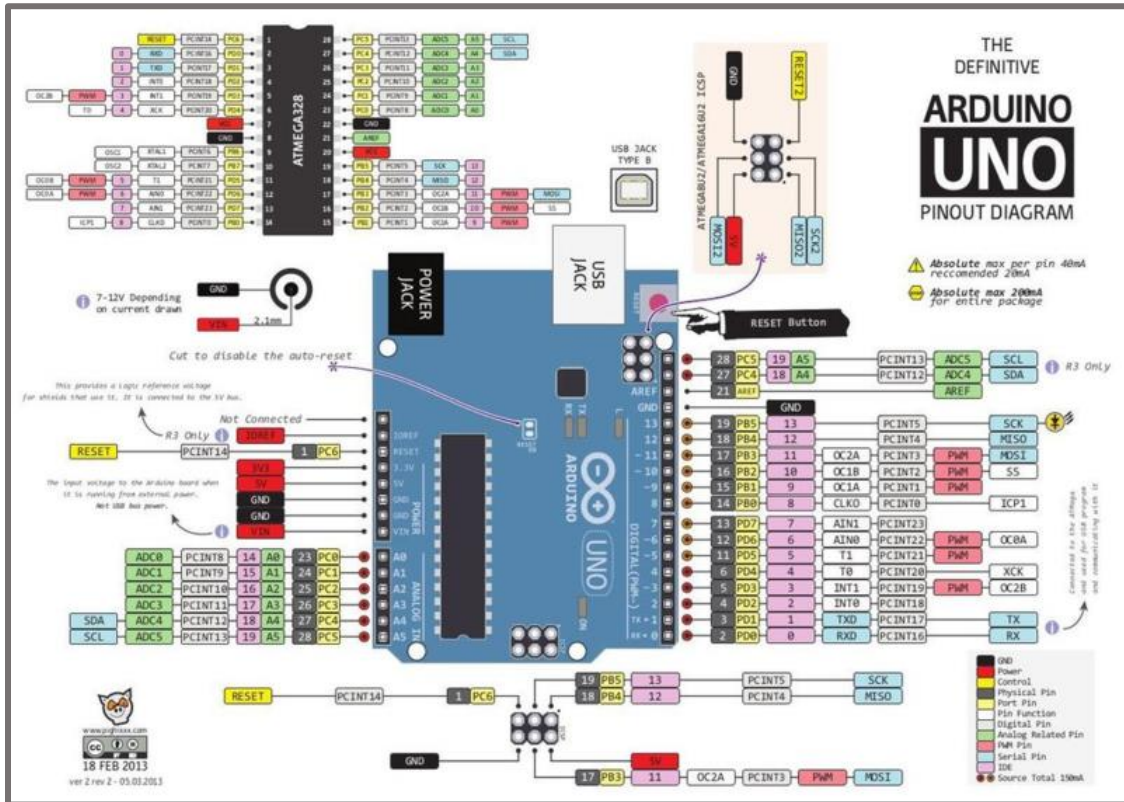


# 정격(定格)

- 모든 전자부품은 “정격”이 존재(제조자가 지정)
  - 정격 전압 : 해당 부품에 적합한 전압
  - 정격 전류 : 해당 부품에 적합한 전류
  - 해당 부품에 정격 전압 이상의 전류가 흐르면 파괴됨
  - 정격 출력 : 전압 $\times$ 전류(부품에 와트(W)로 표기)

부품의 데이터시트를 참조하여야 함

- 핀 : 회로를 연결하는 곳
- 핀 맵 : 핀 별로 용도를 정리해 놓은 것
- 아두이노의 핀 맵



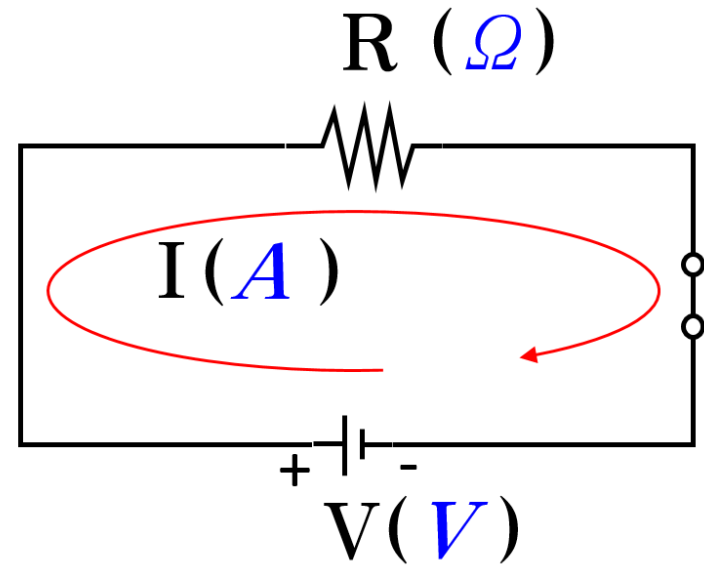
- 아두이노 출력 핀
  - 전원 역할을 하는 핀
  - 전압이 걸리고 전류가 나간다
  - 모터 구동, LED 불 켜기 등
  - $I = V / R$
- 아두이노 입력 핀
  - 전압을 측정하는 핀
  - 전류가 흘러 전압이 걸린다
  - 센서 값을 읽어들 일 때 사용
  - $V = I * R$

# 옴의 법칙 (Ohm's Law)

전기회로에 흐르는 전류는 전압에 비례하고 저항에 반비례

$$V = IR$$

$$I = \frac{V}{R}$$



전압( $V$  : Volt): 전류를 흐르게 하는 전기적인 압력, 단위 **볼트 [V]**

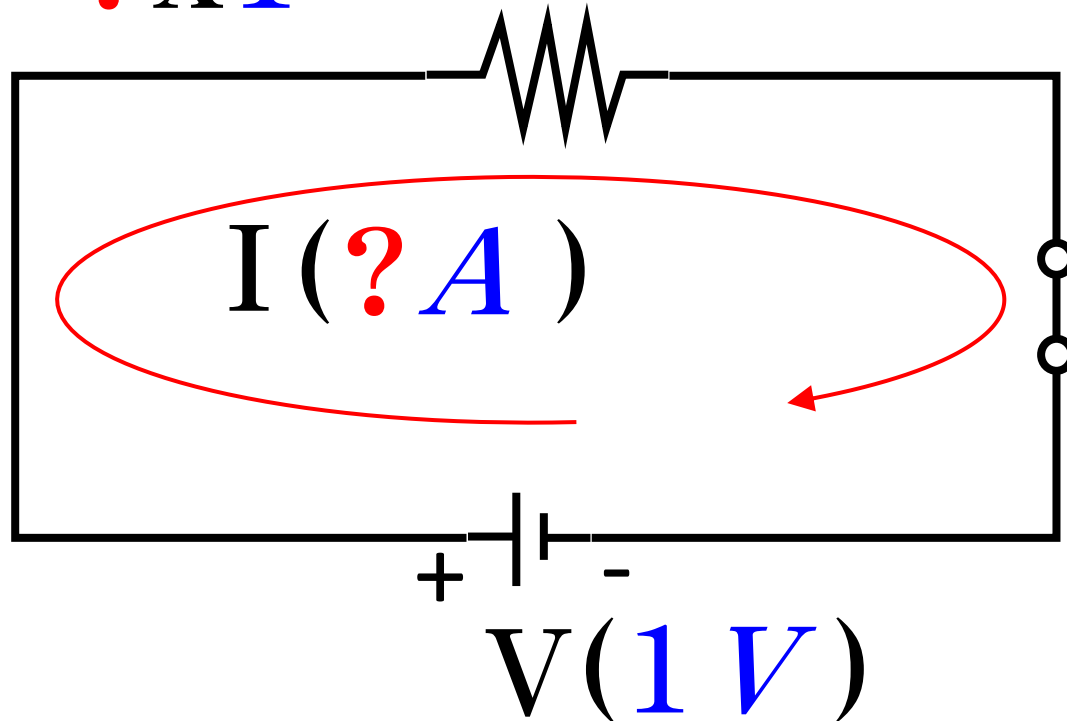
전류( $I$  : Intensity of Current): 단위 시간에 통과하는 전하의 양, 단위 **암페어 [A]**

저항( $R$  : Resistance): 전류의 흐름을 방해하는 성질, 단위 **옴 [ $\Omega$ ]**

# 옴의 법칙 (Ohm's Law)

$$V = IR$$
$$1 = ? \times 1$$

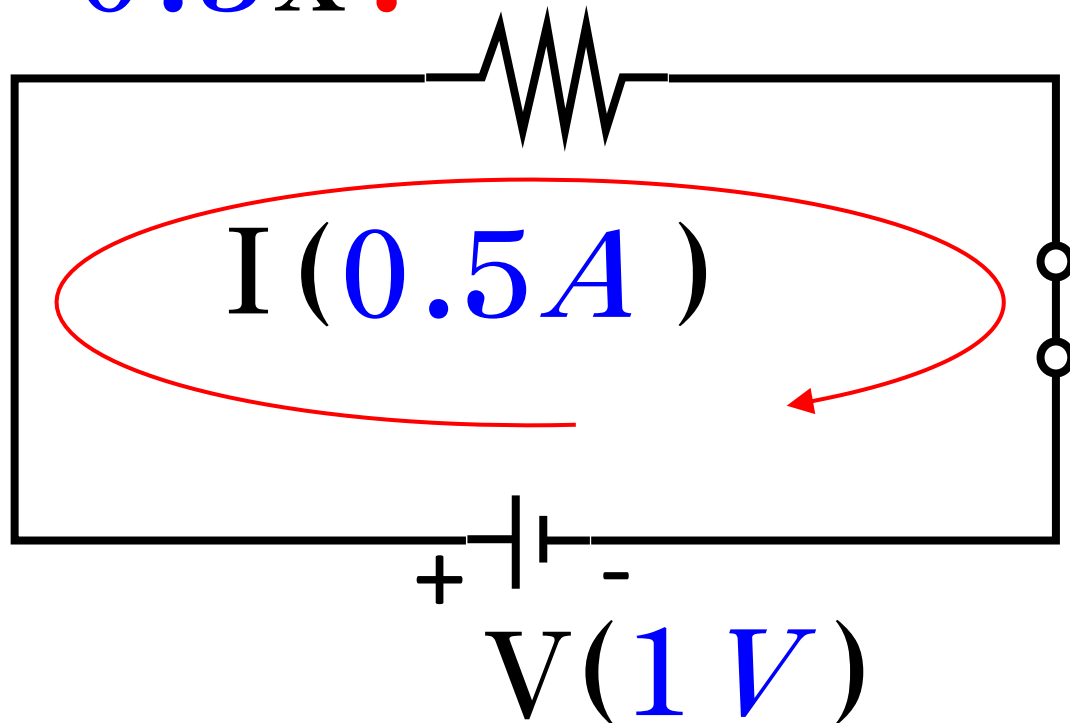
$$R (1 \Omega)$$





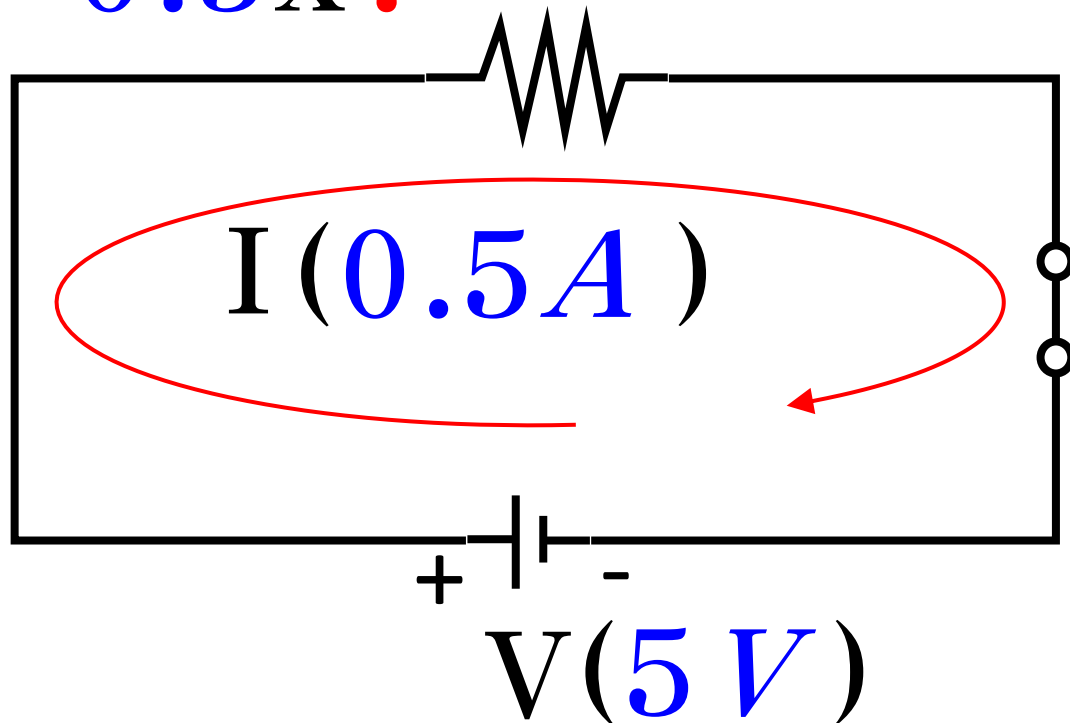
# 옴의 법칙 (Ohm's Law)

$$V = IR$$
$$1 = 0.5 \times ? \quad R (? \Omega)$$



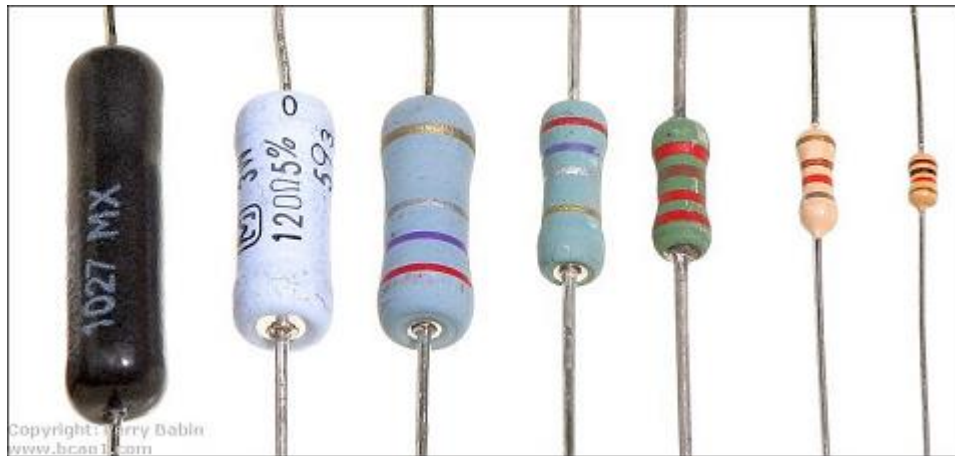
# 옴의 법칙 (Ohm's Law)

$$V = IR$$
$$5 = 0.5 \times ? \quad R (? \Omega)$$



# 저항(Resistor)

- 전류의 흐름을 방해하는 성질, 단위( $\Omega$ )
- 회로를 구성할 때 사용되는 부품



- 저항의 크기는 저항에 그려진 띠들의 색으로 표시

# 저항값 읽기



4색 코드 저항



5, 0, 2( $50 \times 100$ )

$50 \times 100 = 5,000\Omega$

$5,000\Omega = 5\text{ K}\Omega$  오차  $\pm 5\%$

5색 코드 저항



2, 6, 0, 3( $260 \times 1000$ )

$260 \times 1000 = 260,000\Omega$

$260,000\Omega = 260\text{ K}\Omega$  오차  $\pm 10\%$

흑  
갈  
적  
등  
황  
녹  
청  
자  
회  
백

검정색

갈 색

빨강색

주황색

노란색

초록색

파란색

보라색

회 색

하얀색

금 색

은 색

없음(무)

숫 자

0

1

2

3

4

5

6

7

8

9

$\Omega$ 단위

1

10

100

1K

10K

100K

1M

10M

100M

1G

오 차

5%

10%

20%

재미있는 아두이노 DIY

# 저항 값 읽기 연습

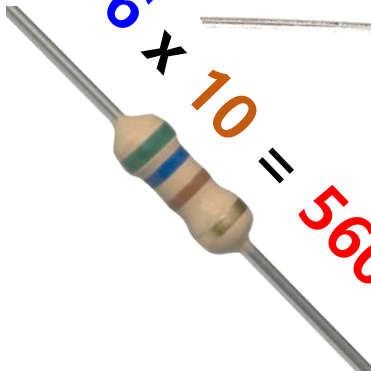
$$33 \times 10 = 330\Omega$$



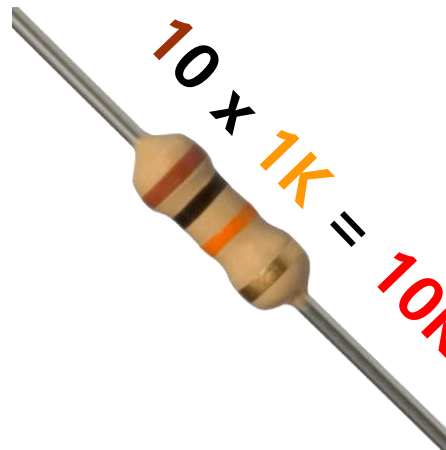
$$10 \times 100 = 1K\Omega$$



$$56 \times 10 = 560\Omega$$



$$10 \times 1K = 10K\Omega$$

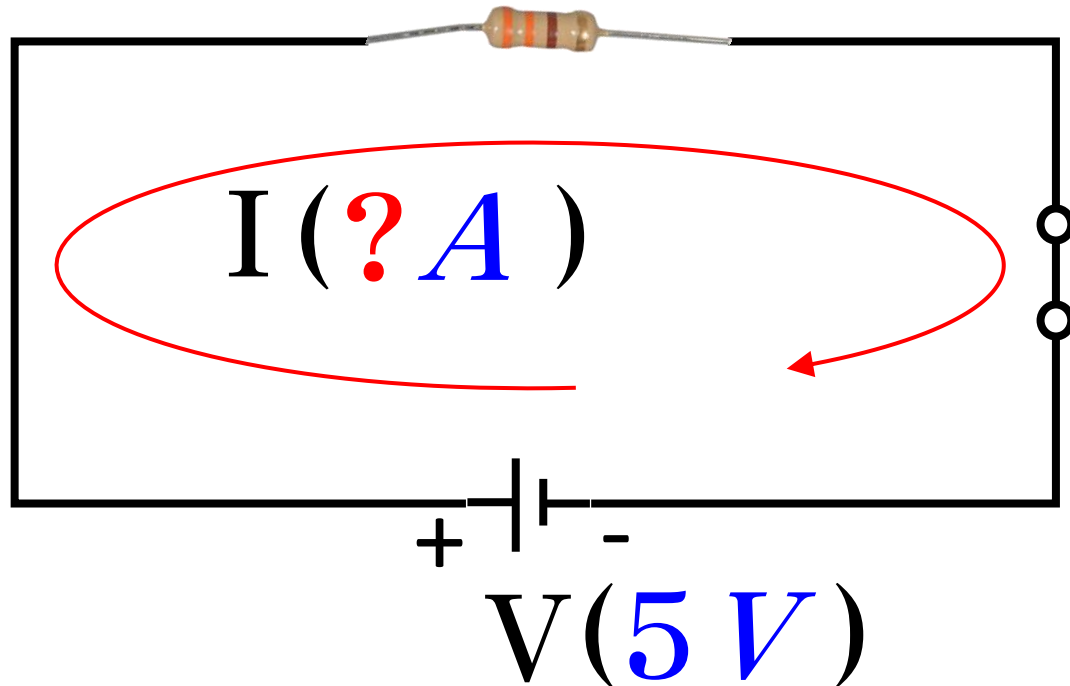


검정색	숫 자	옴단위	오 차
갈 색	0	1	
빨강색	1	10	
주황색	2	100	
노란색	3	1K	
초록색	4	10K	
파란색	5	100K	
보라색	6	1M	
회 색	7	10M	
하얀색	8	100M	
금 색	9	1G	5%
은 색			10%
없음(무)			20%

# 옴의 법칙 (Ohm's Law) (복습)

$$I = \frac{V}{R}$$

$$R \text{ (} 330 \Omega \text{)}$$



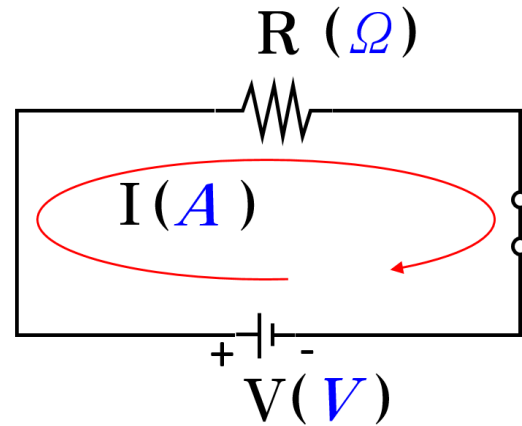
$$1A = 1000mA$$
$$1mA = \frac{1}{1000}A$$

$$I = \frac{5}{330} \approx 0.015A = 15mA$$

# 옴의 법칙 (Ohm's Law)

전기회로에 흐르는 전류는 전압에 비례하고 저항에 반비례

$$I = \frac{V}{R}$$



$$I = \frac{1V}{1\Omega} = 1A$$

$$I = \frac{1V}{0.1\Omega} = 10A$$

$$I = \frac{1V}{0.01\Omega} = 100A$$

$$I = \frac{1V}{0.0001\Omega} = 10000A$$



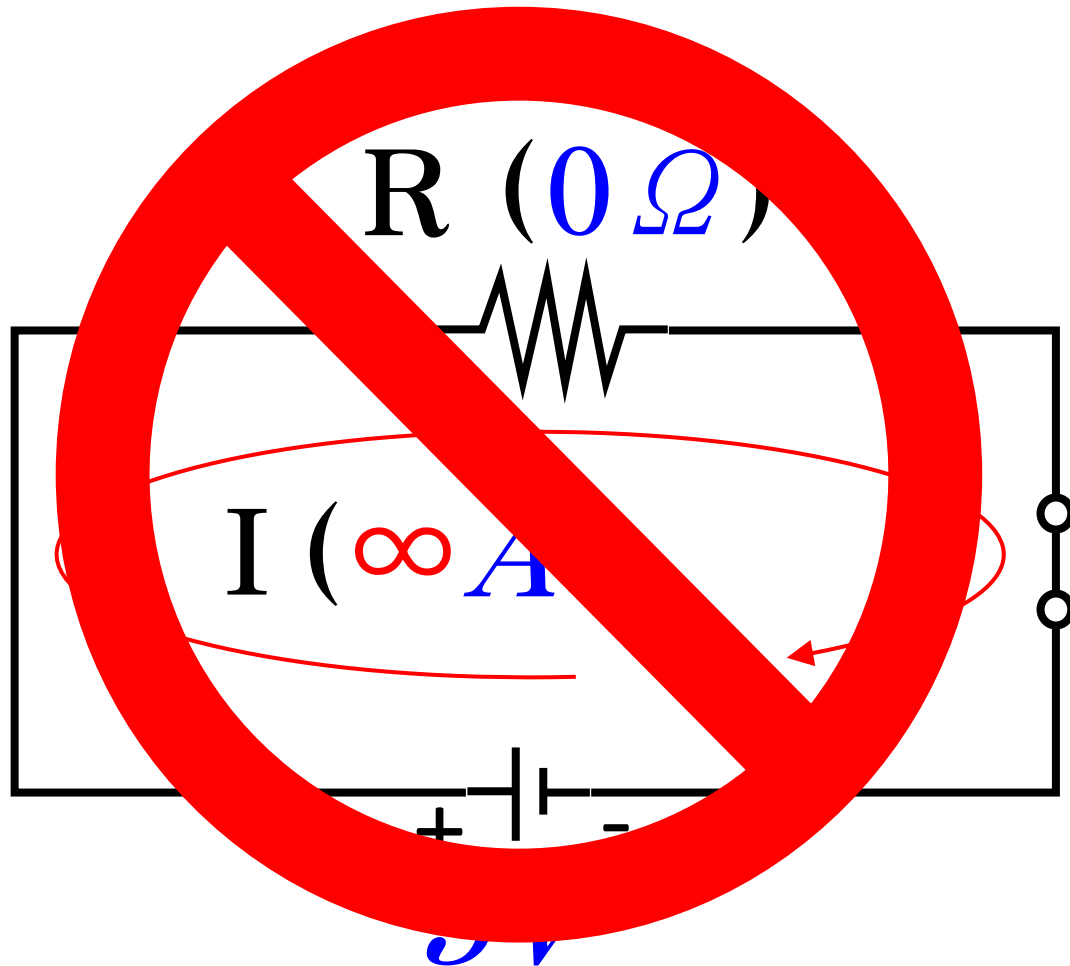
전기회로에서 저항 없이 전류가 흐르면 위험!



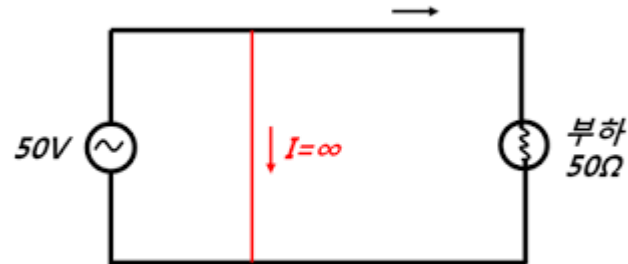
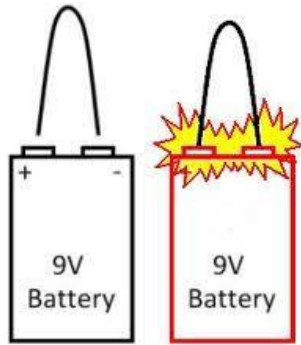


# 단락(short, 합선) <= 주의!

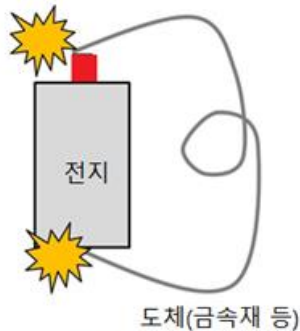
전기회로에서 부하(저항)없이 +, -가 연결되는 회로



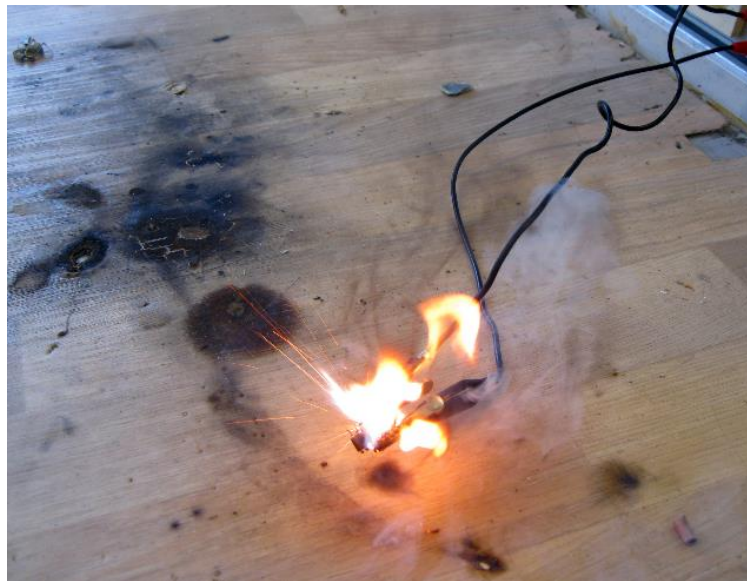
# 단락(short, 합선) 예 <= 주의!



전지 단락이란?

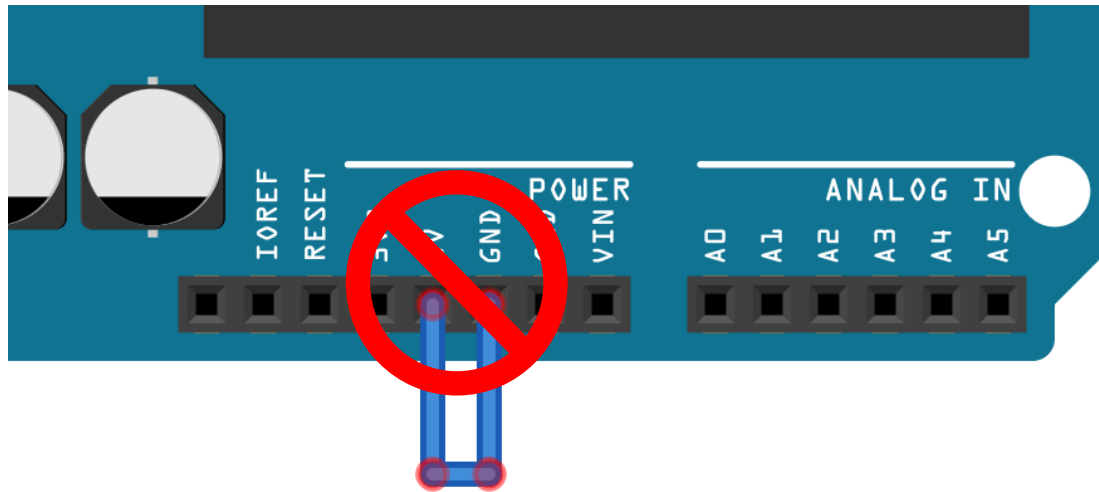


단락: 전지의 + 와 -를  
도체로 연결하는 것



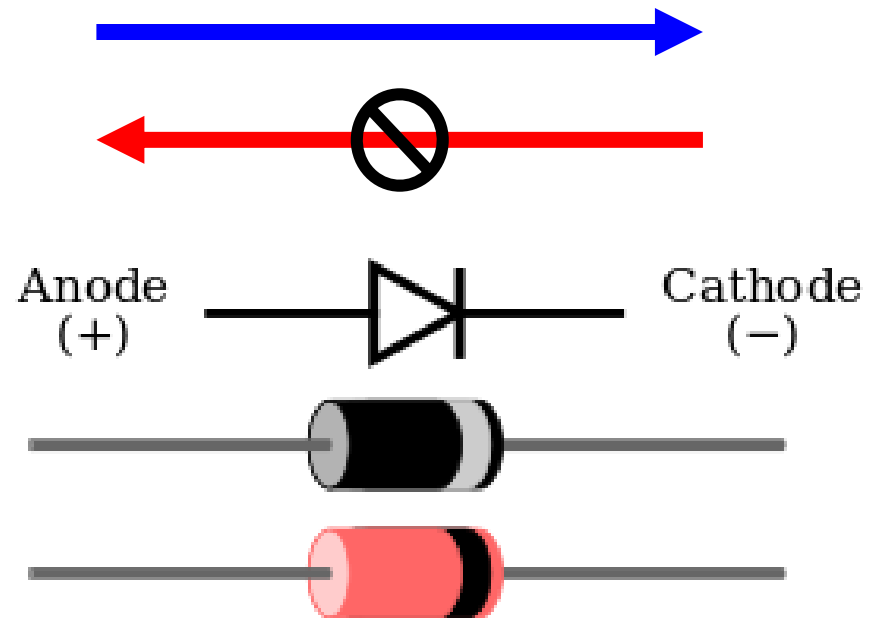
부품에서 열 발생, 타는 냄새, 연기 등이 나면 즉시 전원 차단!  
=> 전원 플러그 뽑기, 컴퓨터와 연결된 USB 케이블 분리 등

# 연결 절대 금지!



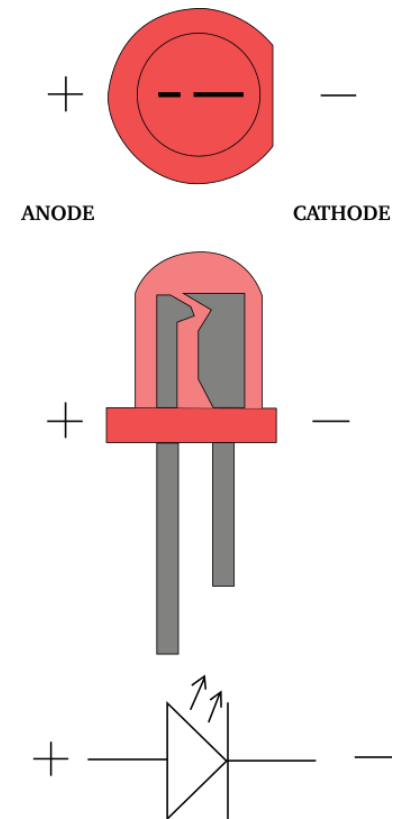
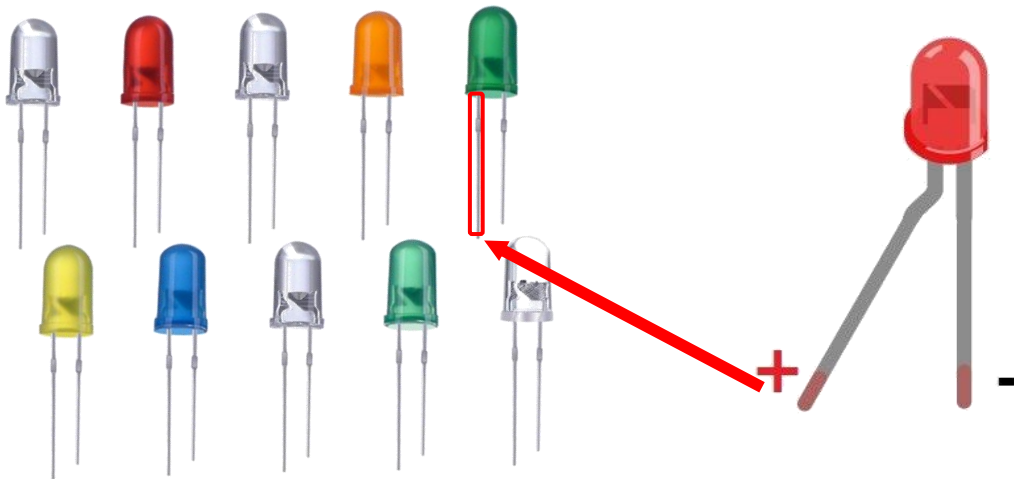
# 다이오드(diode)

- 전류가 한 방향(순방향)으로만 흐르는 소자 (극성이 있음. 저항은 극성이 없음)
- 교류->직류 변환, 역 전압 방지 회로 등에 활용



# LED (Light Emitting Diode)

- 순방향 전압이 걸리면 빛을 내는 다이오드
- 간단한 정보를 표시하는 용도로 활용



# LED 저항 값 계산

색상	구 분	최소전압	최대전압	전류(일반)	전류(최대)
적●	Red	1.8V	2.3V	20 mA	50 mA
등●	Orange	2.0V	2.3V	30 mA	50 mA
황●	Real Yellow	2.0V	2.8V	20 mA	50 mA
초●	emerald Green	1.8V	2.3V	20 mA	50 mA
초●	Real Green	3.0V	3.6V	20 mA	50 mA
청●	sky Blue	3.4V	3.8V	20 mA	50 mA
청●	Real Blue	3.4V	3.8V	20 mA	50 mA
자●	Pink	3.4V	3.8V	20 mA	50 mA
백○	White	3.4V	4.0V	20 mA	50 mA

정확한 스펙은 LED의 데이터 시트를 살펴봐야하나 데이터 시트가 없는 경우 표 참고

옴의 법칙

$$V = IR$$

$$R = V/I$$

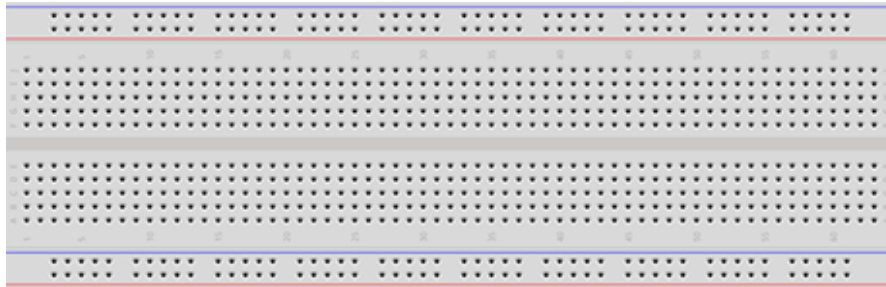
## • LED 저항값 계산

- 저항값 = (입력 전압 - LED 전압) / LED 전류
- 일반 :  $R = (5-2)/0.02 \rightarrow R = 150 \Omega$
- 최대 :  $R = (5-2)/0.05 \rightarrow R = 250 \Omega$

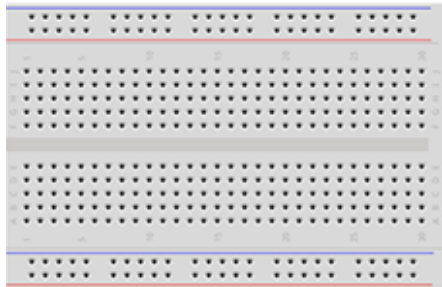
➔ 보통 **220  $\Omega$**  사용

# 브레드 보드(bread board)

- 납땜 없이 소자를 꼽아 회로를 쉽게 구성할 수 있음



830홀



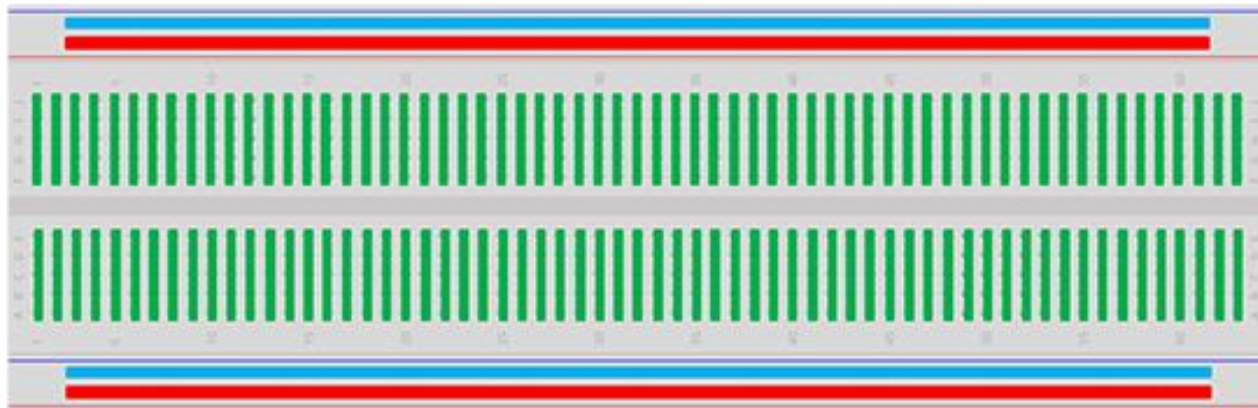
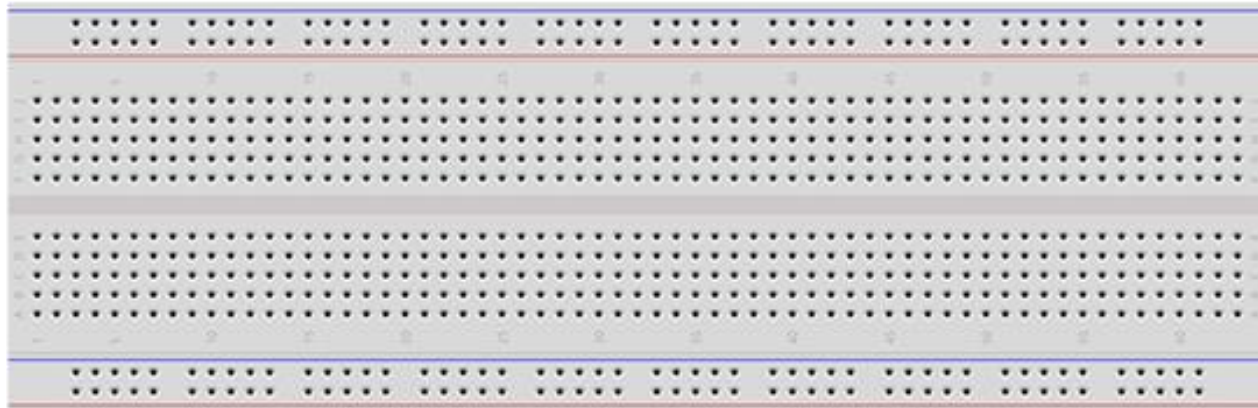
400홀



170홀

# 브레드 보드 내부 연결 상태

830층





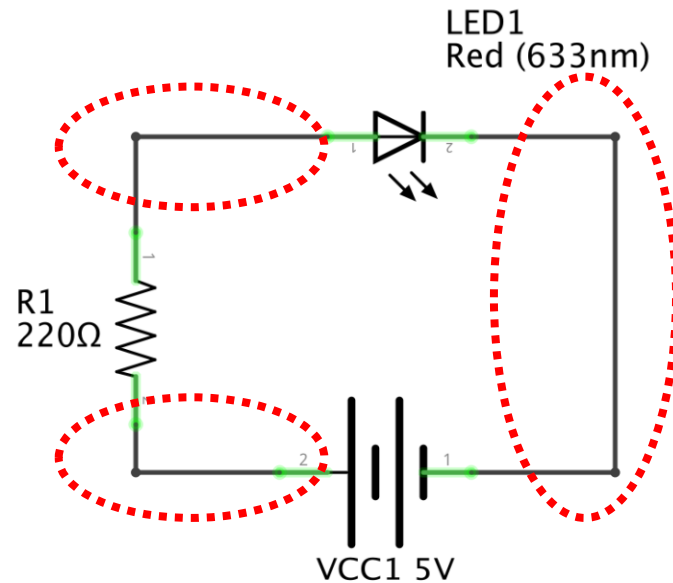
The logo for fritzing, featuring the word "fritzing" in a white, lowercase, sans-serif font. The letters are slightly rounded and have a modern, clean appearance. The text is centered within a solid red rectangular background. Below the red rectangle, there is a thin, dark horizontal line.

fritzing

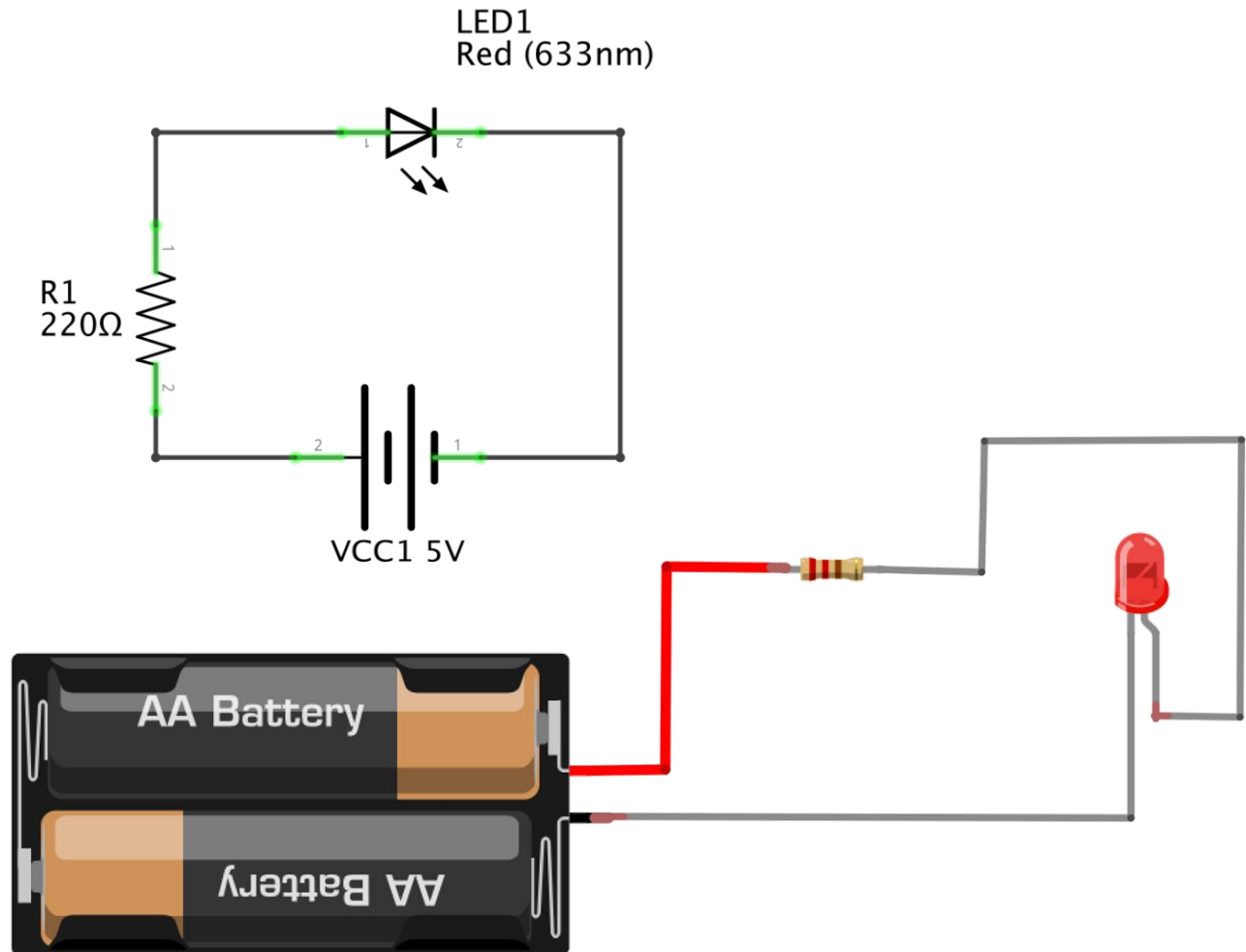
Breadboard 내부 연결 살펴보기

회로도 → 실제 회로 구성

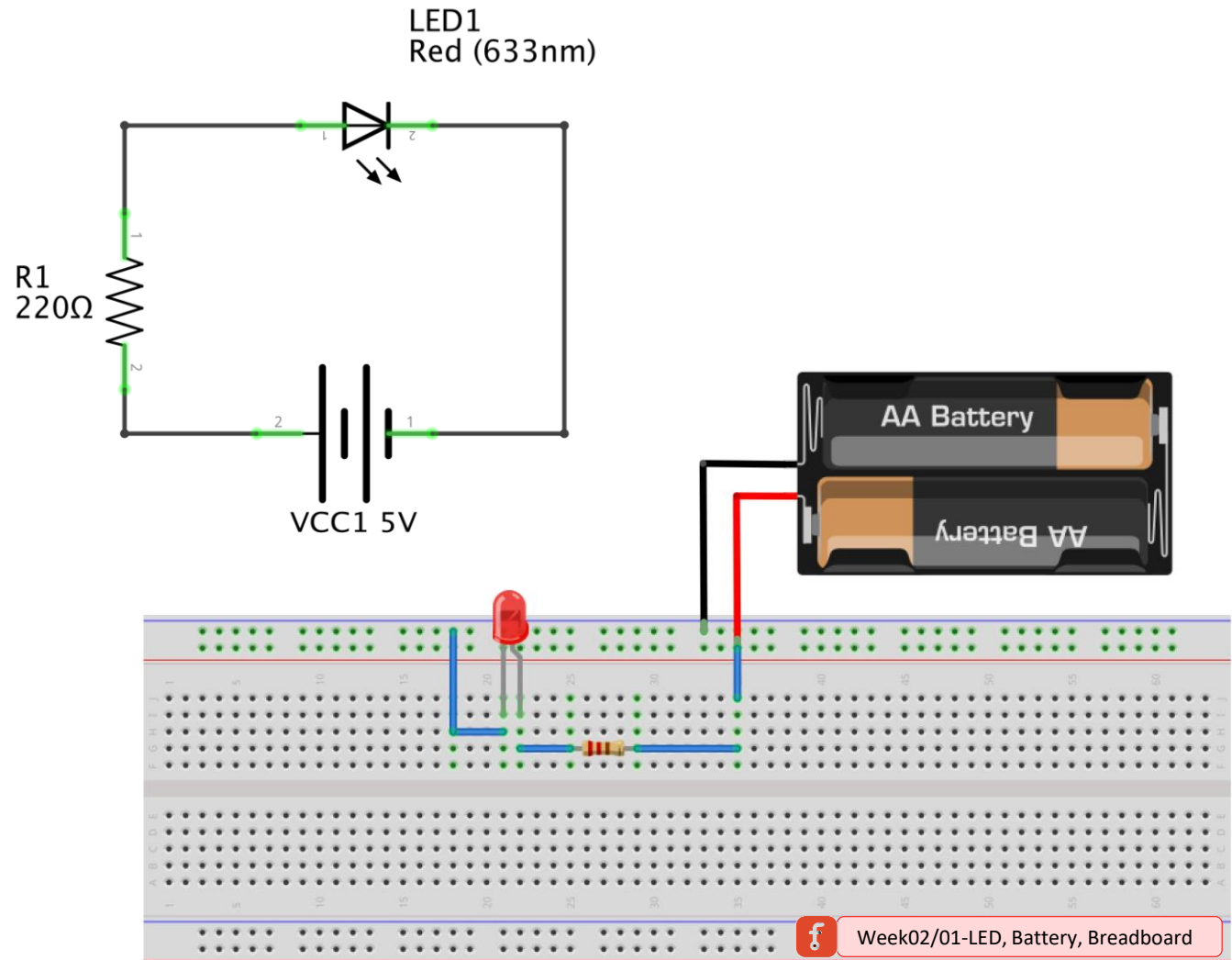
# LED 동작 회로 (브레드 보드 미사용)



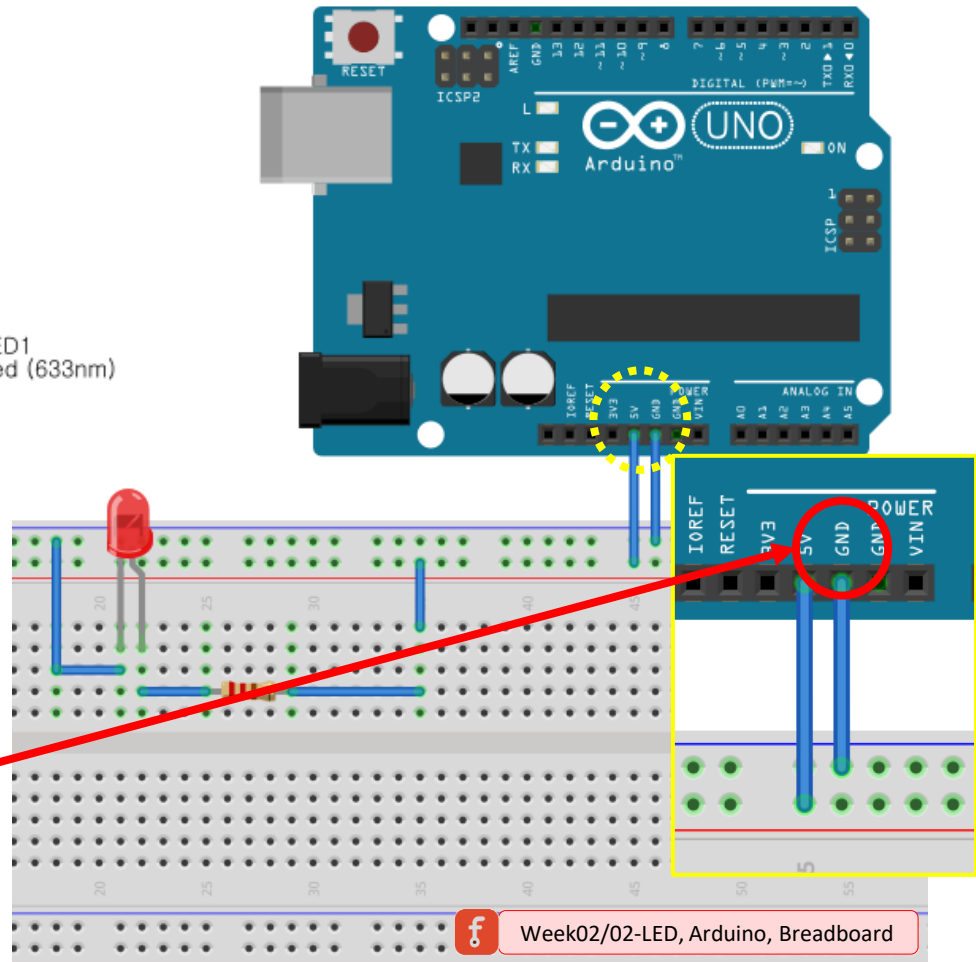
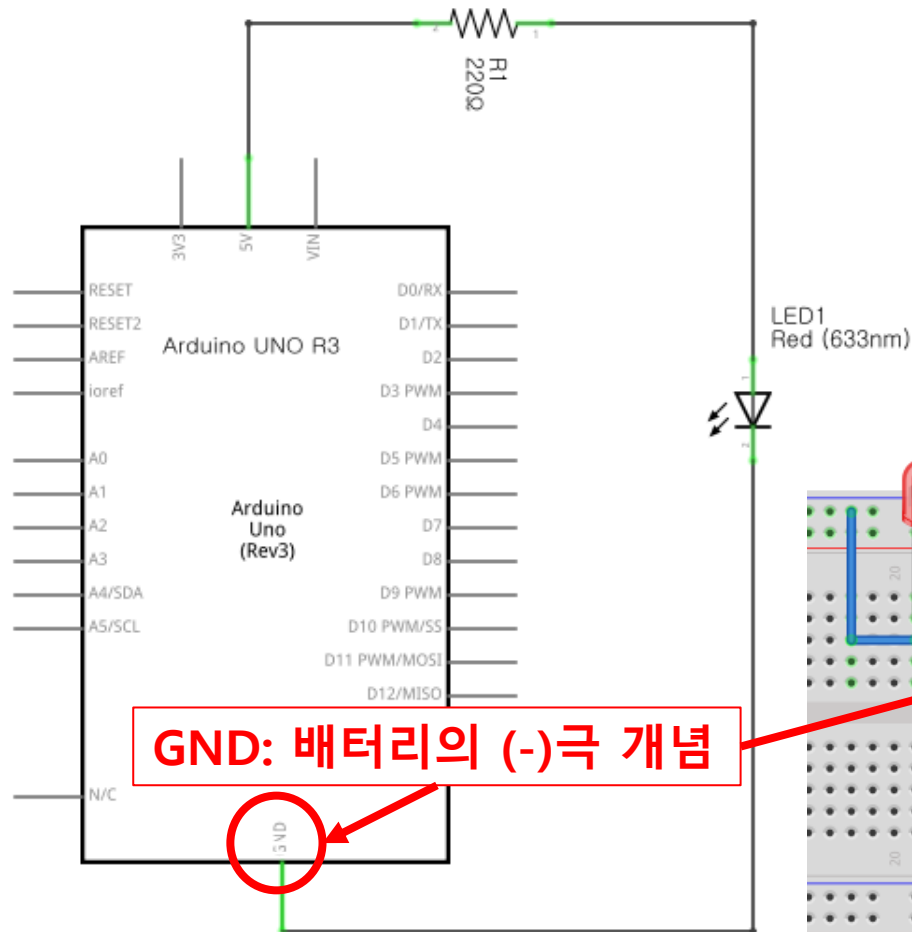
# LED 동작 회로 (브레드 보드 미사용)



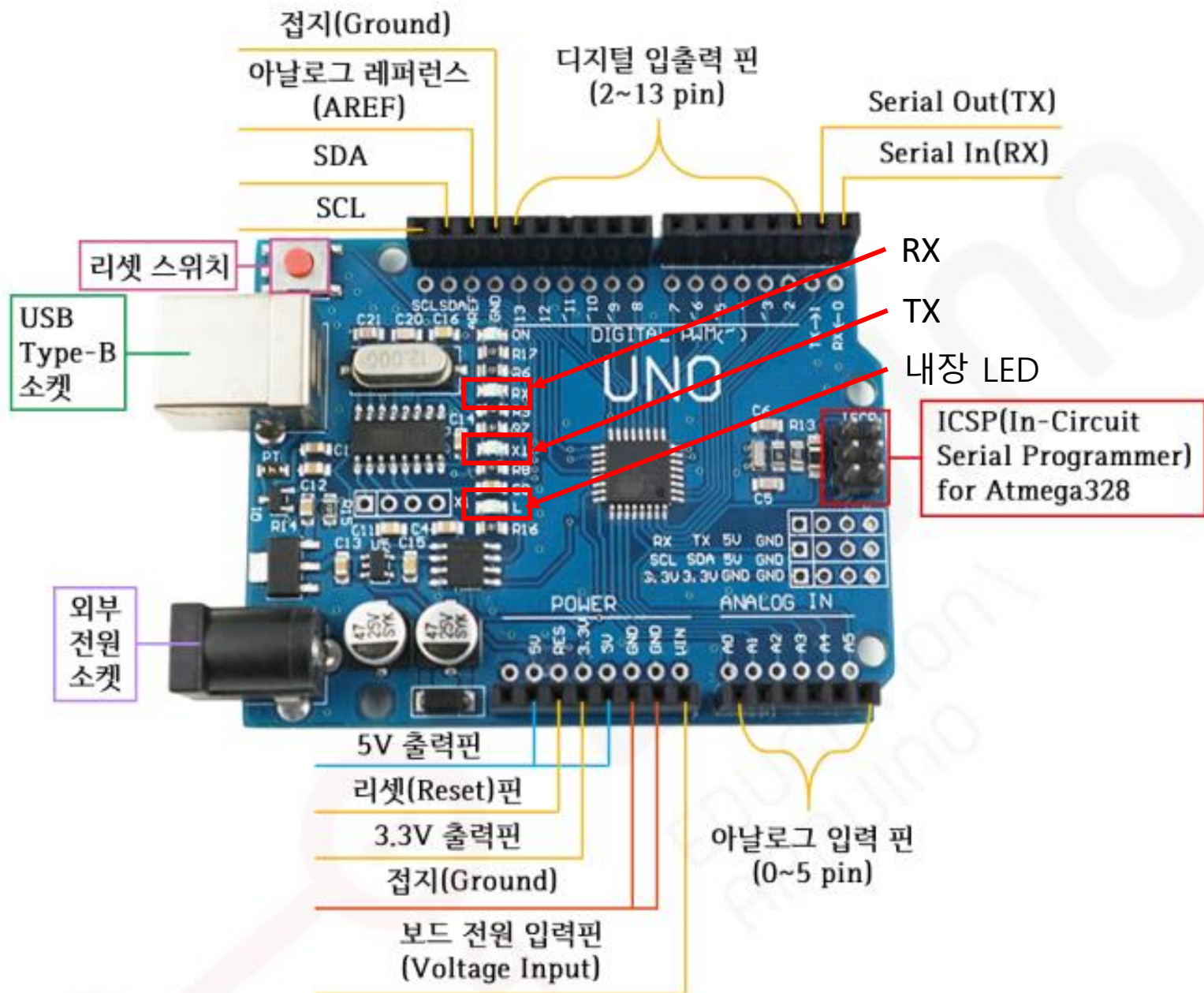
# LED 동작 회로 (브레드 보드 사용)



# LED 동작 회로 (아두이노 전원 사용)



# 아두이노 우노 UNO SMD R3 호환보드



아두이노 기본 함수



# pinMode(*pin\_no*, *mode*)

- 지정한 핀(pin\_no)이 **입력** 또는 **출력**으로 동작하도록 설정
- 사용 예)
  - pinMode(7, **OUTPUT**); //7번 핀을 **출력모드**로 설정
  - pinMode(6, **INPUT**); //6번 핀을 **입력모드**로 설정

```
int ledPin = 9;           //핀 번호를 변수에 지정하여 활용

void setup()
{
    pinMode(ledPin, OUTPUT); //ledPin(7)을 출력으로 지정
}
```

# **digitalWrite**(*pin\_no*, *value*)

- **출력모드**로 설정한 디지털 핀에 **HIGH** 또는 **LOW** 값을 출력
- 사용 예)
  - digitalWrite(7, **HIGH**); //7번 핀에 **HIGH**값 출력
  - digitalWrite(7, **LOW**); //7번 핀에 **LOW**값 출력

```
int ledOut = 7;           //핀 번호를 변수에 지정하여 활용

void setup()
{
    pinMode(ledOut, OUTPUT);
    digitalWrite(ledOut, HIGH); //7번 핀에 HIGH 출력
}
```

# delay (ms)

- 지정한 시간(ms)만큼 프로그램 실행이 중단됨(대기)
- 시간 지정 단위: 1/1000초 (ms)
  - 1000 => 1초, 500 => 0.5초, 100 => 0.1초
- 사용 예)
  - delay(**1000**); //1초 대기
  - delay(**100**); //0.1초 대기

```
int ledPin = 13;    //아두이노 내장 LED(13번)를 사용

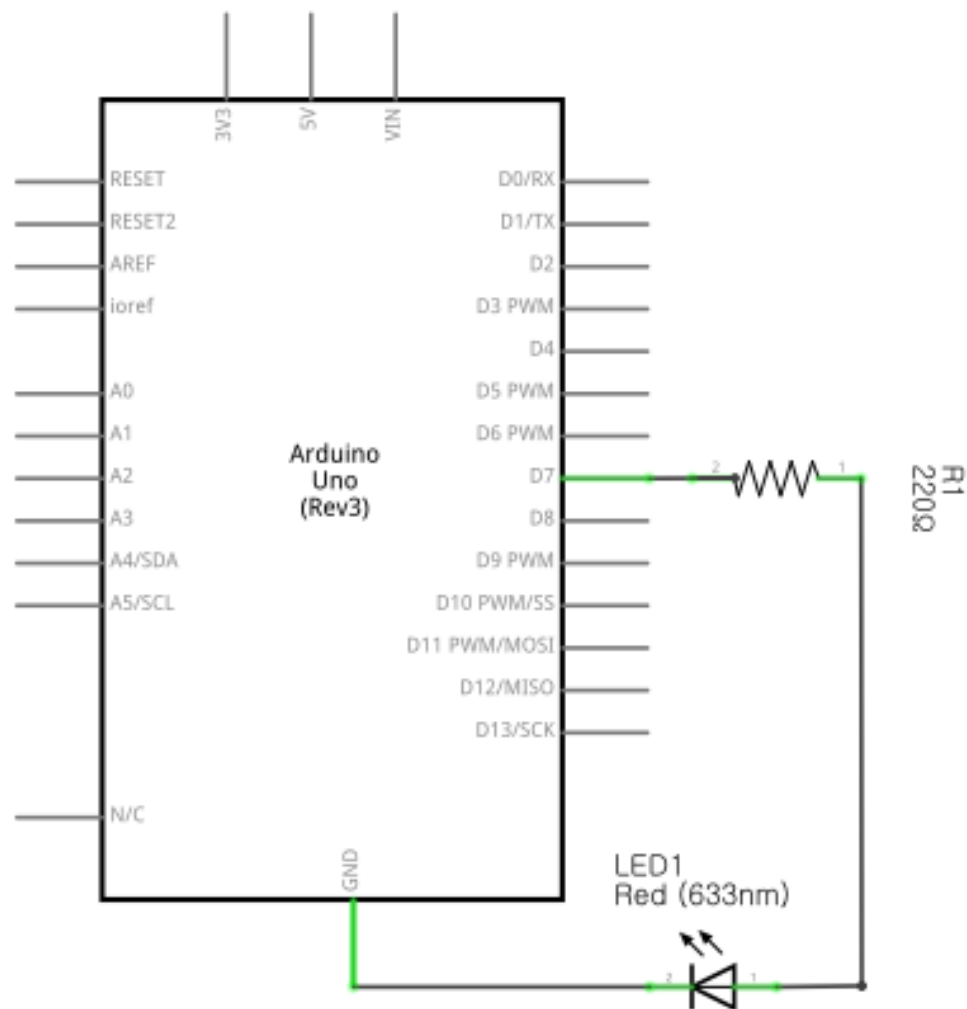
void setup()
{
    pinMode(ledPin, OUTPUT);    //13번 핀을 출력으로 지정
}
void loop()
{
    digitalWrite(ledPin, HIGH); //LED On
    delay(1000);                //1초 대기
    digitalWrite(ledPin, LOW);  //LED Off
    delay(500);                 //0.5초 대기
}
```

# 아두이노로 LED 제어하기

아두이노 보드 7번 핀을 이용하여 LED를 1초 간격으로 점멸시킨다.

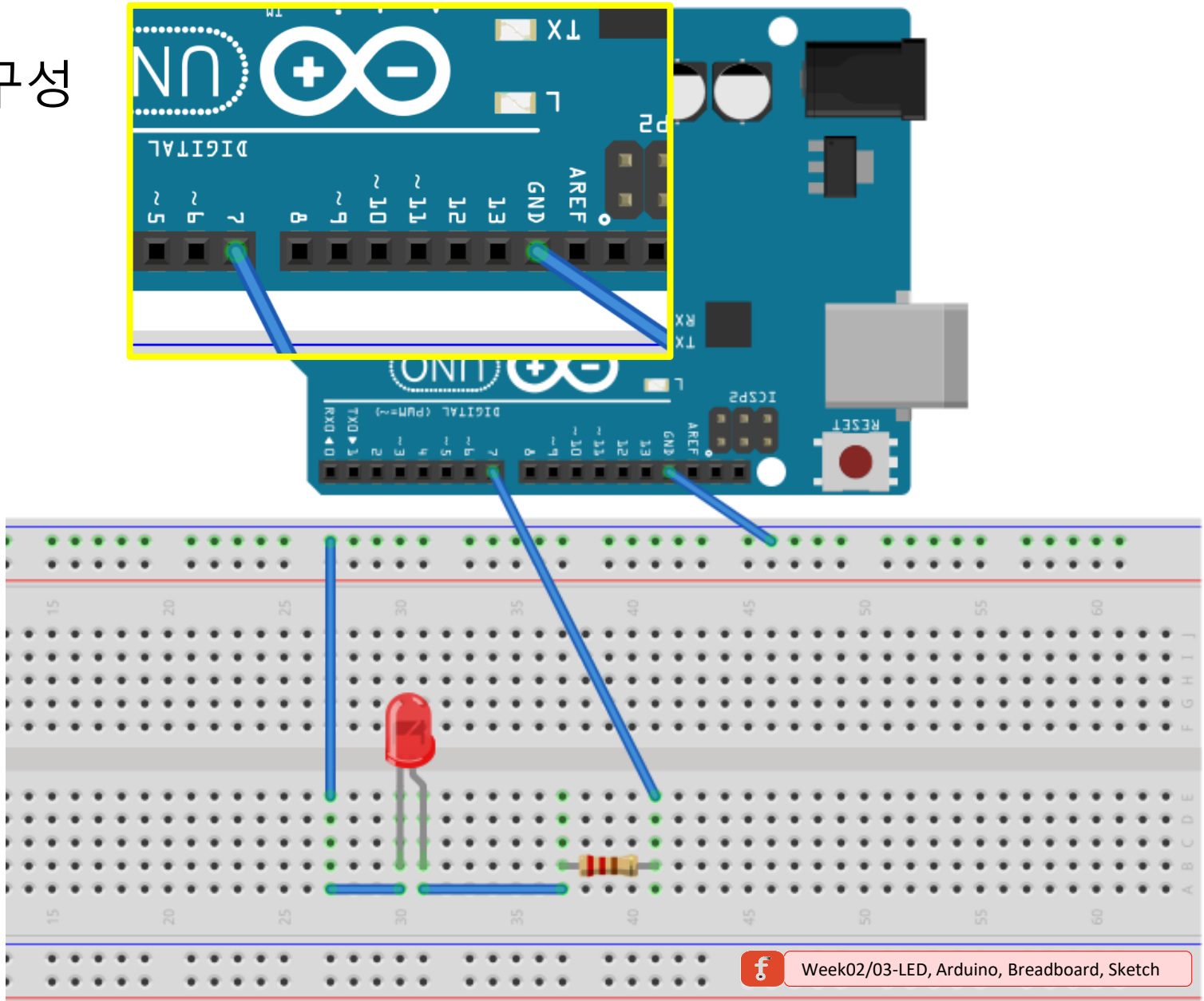
# 아두이노로 LED 제어하기

## 회로도



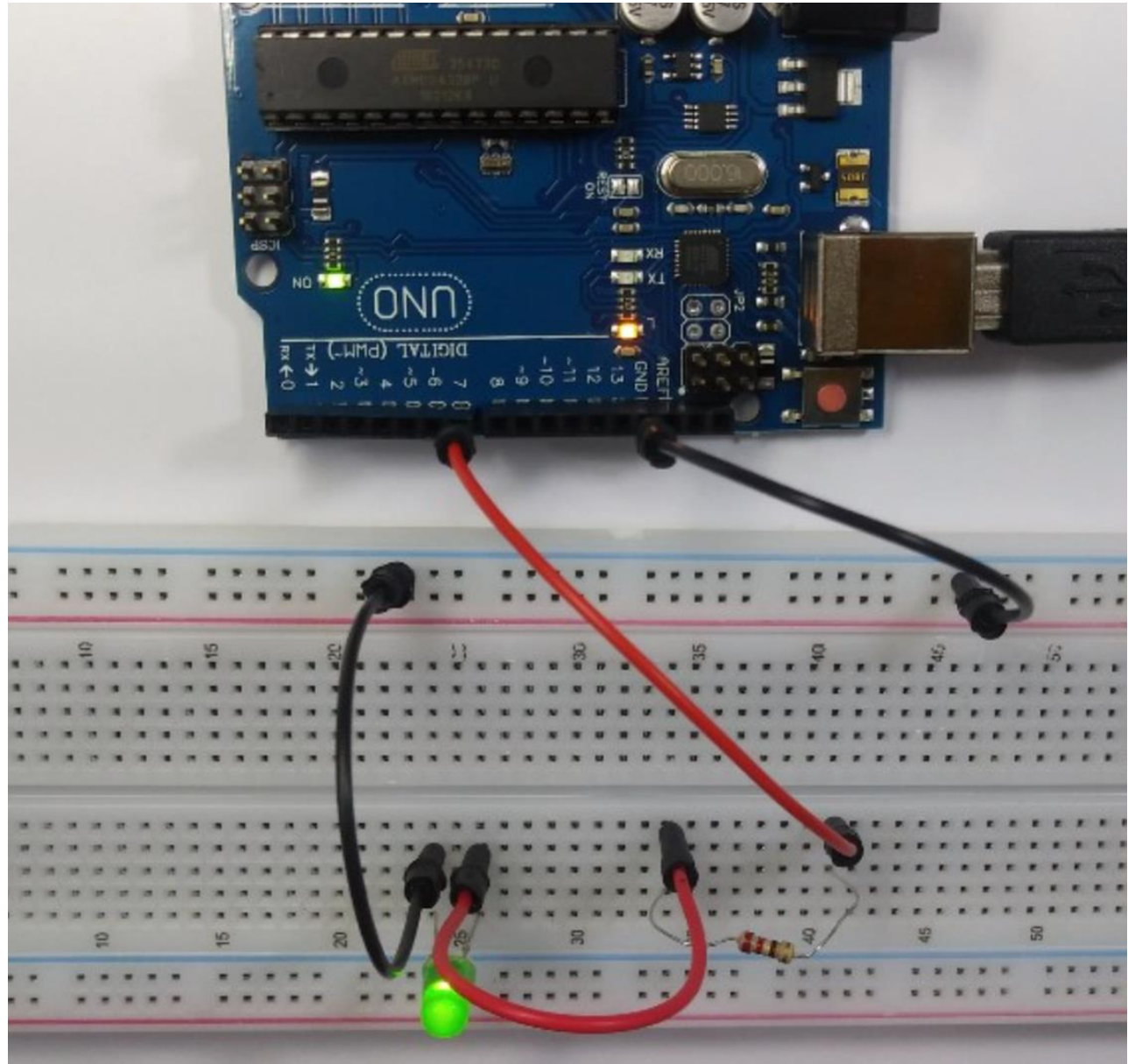
# 아두이노로 LED 제어하기

## 회로 구성



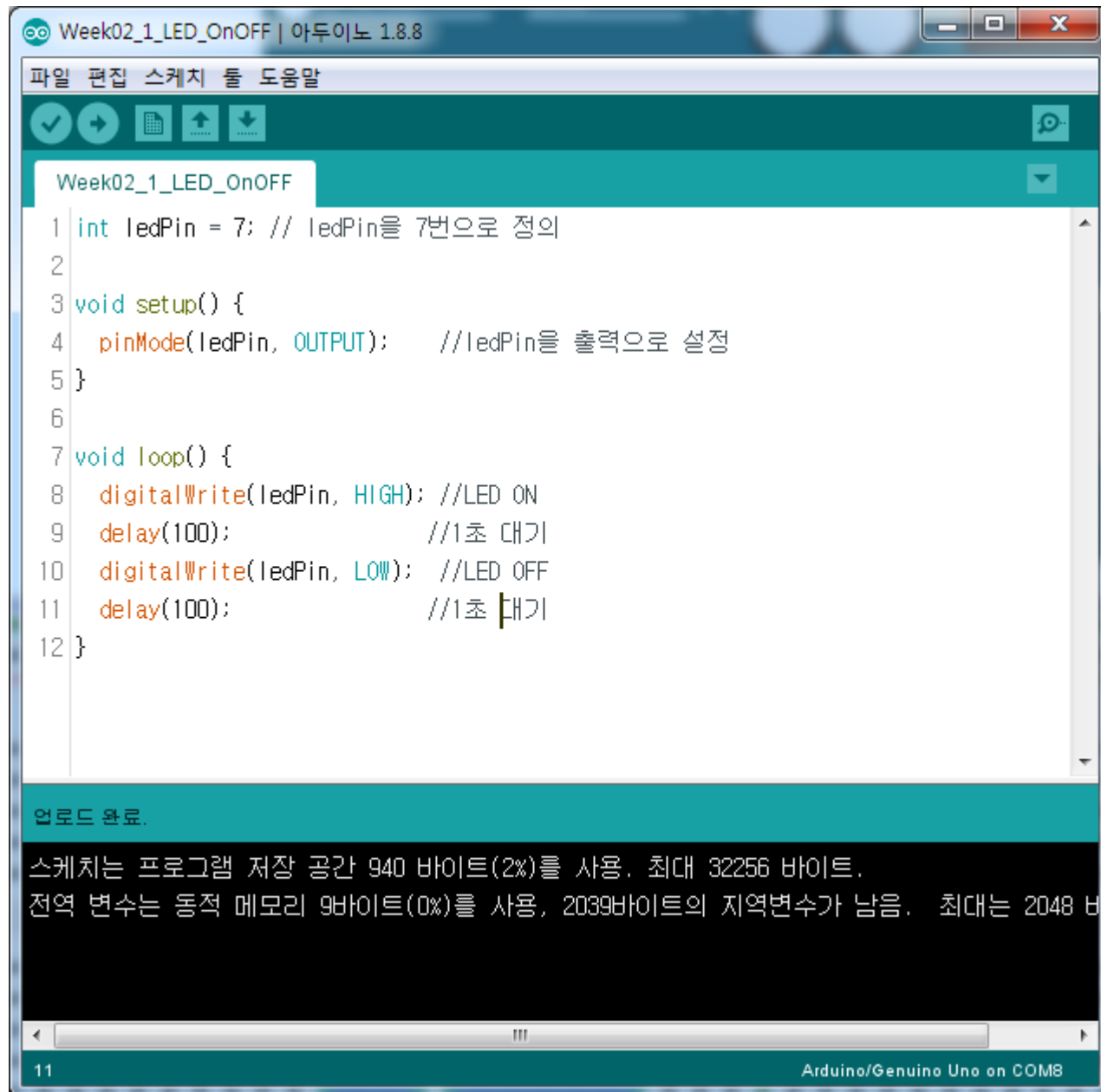
# 아두이노로 LED 제어하기

## 실제 구성



# 아두이노로 LED 제어하기

## sketch



The screenshot shows the Arduino IDE interface. The title bar reads "Week02\_1\_LED\_OnOFF | 아두이노 1.8.8". The menu bar includes "파일", "편집", "스케치", "툴", and "도움말". The toolbar contains icons for saving, running, uploading, and downloading. The sketch name "Week02\_1\_LED\_OnOFF" is displayed in the top right. The code editor contains the following C++ code:

```
1 int ledPin = 7; // ledPin을 7번으로 정의
2
3 void setup() {
4   pinMode(ledPin, OUTPUT); //ledPin을 출력으로 설정
5 }
6
7 void loop() {
8   digitalWrite(ledPin, HIGH); //LED ON
9   delay(100); //1초 대기
10  digitalWrite(ledPin, LOW); //LED OFF
11  delay(100); //1초 대기
12 }
```

Below the code editor, a status bar indicates "업로드 완료." (Upload complete.). Below that, a message box states: "스케치는 프로그램 저장 공간 940 바이트(2%)를 사용, 최대 32256 바이트. 전역 변수는 동적 메모리 9바이트(0%)를 사용, 2039바이트의 지역변수가 남음. 최대는 2048 바이트." (The sketch uses 940 bytes (2%) of program storage space, maximum 32256 bytes. Global variables use 9 bytes (0%) of dynamic memory, 2039 bytes of local variables remain. Maximum is 2048 bytes.). The bottom status bar shows "11" and "Arduino/Genuino Uno on COM8".



# Http://fritzing.org/home/

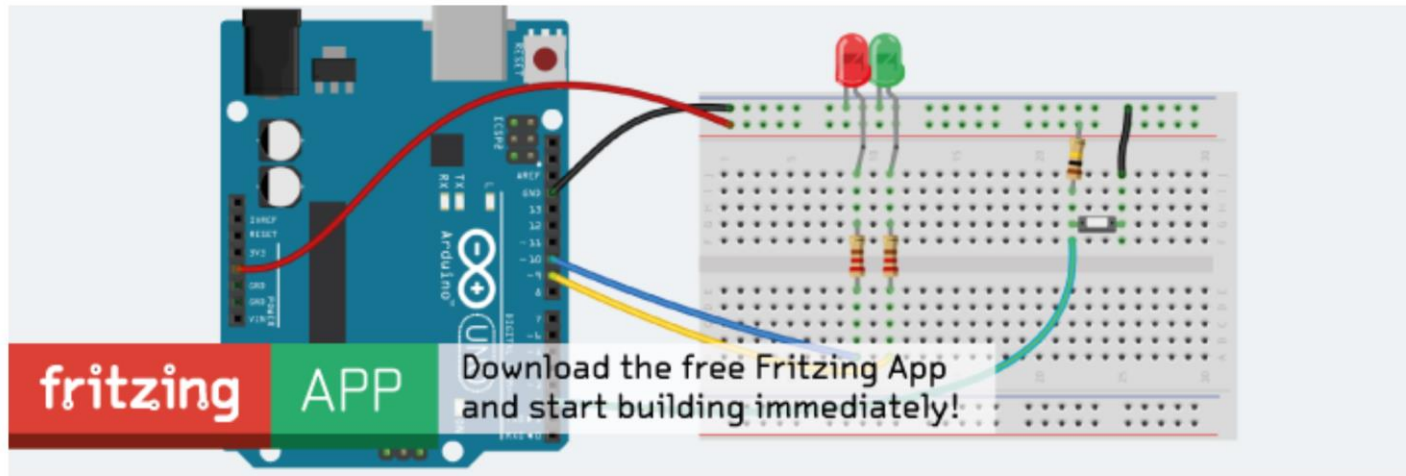
fritzing.org/home/

**fritzing** electronics  
made easy

Projects Parts Download Learning Services Contribute

FORUM

FAB



Fritzing is an **open-source hardware initiative** that makes electronics accessible as a creative material for anyone. We offer a software tool, a community website and services in the spirit of

## Download and Start

Download our **latest version 0.9.3b** released on June 2, 2016 and start right away.

## Produce your own board