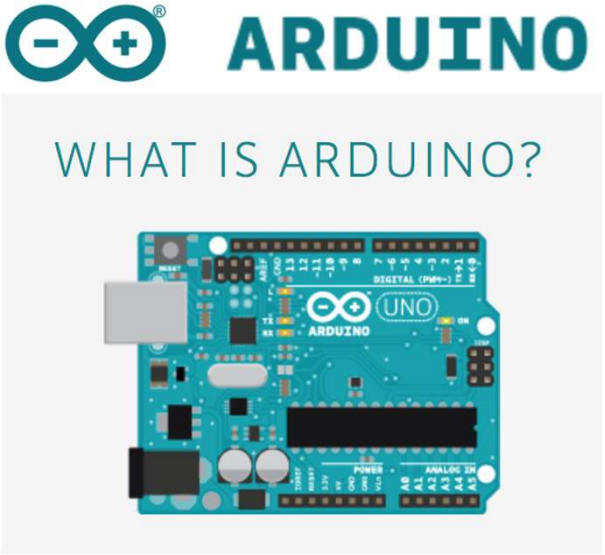
# 아두이노 실습

MAKIST
만·들·어·가·는·사·람·들

# Chapter 01
# 아두이노 설치하기




Step 1. 아두이노?

Step 2. 아두이노 설치하기

Step 3. 통신 드라이버 설치하기

Step 4. 소스파일 작성하기

Step 5. 확인(컴파일), 업로드

# 아두이노(ARDUINO)?

## 아두이노는 컴퓨터의 두뇌를 담당하는 하드웨어인 마이크로컨트롤러(MCU)와 마이크로컨트롤러를 제어하기 위한 소프트웨어(스케치라고 말한다) 제작 환경을 함께 묶어서 부르는 말입니다.

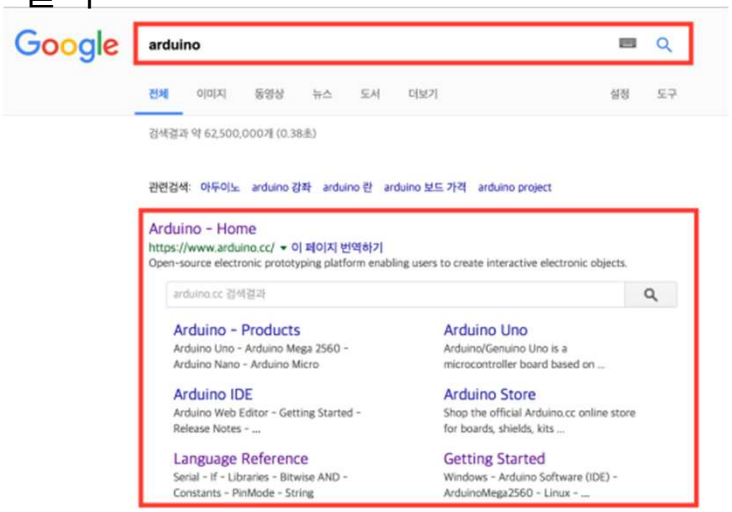
아두이노는 프로그램이나 전자회로 같은 하드웨어적 기술이 없는 사람도 일상생활에서 쉽게 사용 할 수 있도록 하기 위한 목적을 가지고 만들어졌습니다.
덕분에 아두이노를 익히는 것으로 어려운 Physical computing(물리적 컴퓨팅) 같은 복잡한 기술을 쉽게 사용 할 수 있습니다.

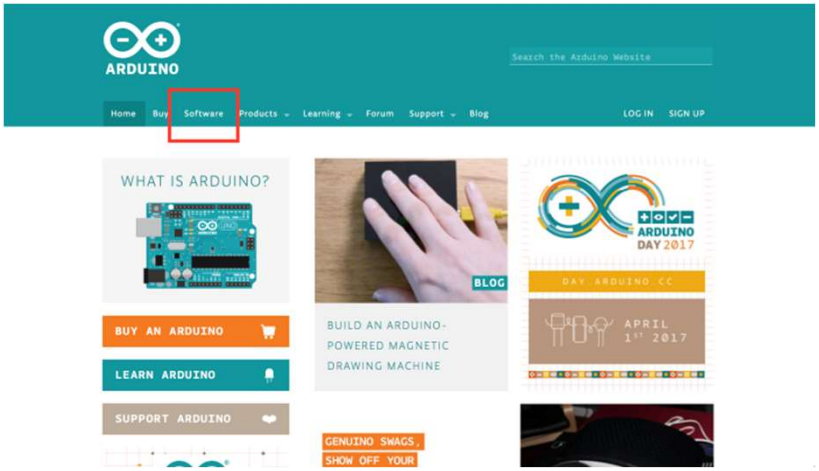
최근 IoT(사물인터넷), 쿼드콥터(드론), 3D프린터, 소프트웨어 의무교육 등의 키워드가 이슈화 되면서 아두이노도 함께 이슈화 되고 있습니다.
아두이노를 이용해서 IoT, 드론, 3D프린터, 웨어러블기기, 로봇 등등 모든 Make 활동이 가능하기 때문입니다.
그리고, 그 중심에는 아두이노를 활용하는 수많은 개발자들이 ' 오픈소스 플랫폼 ' 을 따르기에, 초보자들이 인터넷에서 쉽게 찾아 보고 따라 해볼 수 있기 때문입니다.

open source hardware

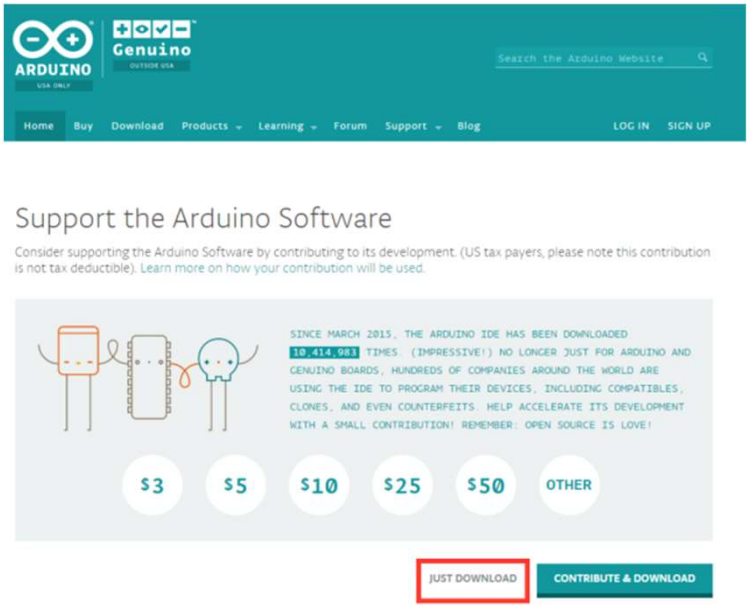
# 아두이노 IDE 설치하기

- 인터넷 주소창에 'http://www.arduino.cc' 을 직접 입력
- 구글에서 'ARDUINO' 로 검색 후 ARDUINO 공식 사이트의 시작화면에서 "SOFTWARE" 버튼 클릭



# 아두이노 IDE 설치하기

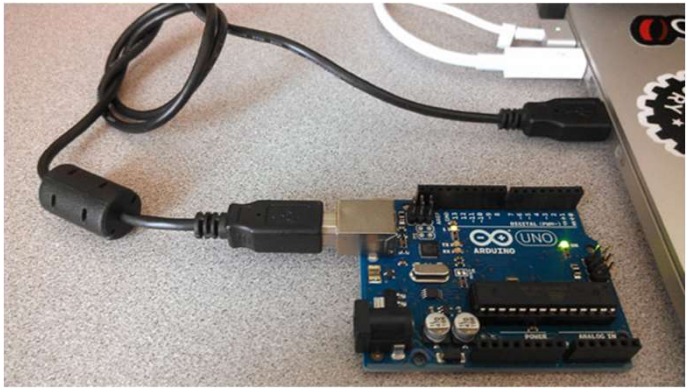

1. Windows installer 나 Mac OS 를 클릭

2. JUST DOWNLOAD를 클릭

3. 다운된 파일을 실행하면 설치과정이 진행됩니다.

# 통신 드라이버 설치하기

- PC에서 아두이노 보드 인식을 위해 USB Driver가 설치되어야 한다.
  - 정품의 경우는 ARDUINO IDE 설치시 자동 설치됨
  - FTDI계열 칩을 사용하는 호환 보드는 OS에서 자동 인식
  - 그 외의 경우는 제조사에서 제공하는 USB Driver 별도 설치 필요
- 포트 선택
  - Windows의 경우 "COM1", "COM2" 등의 이름 규칙



# 통신 드라이버 설치하기

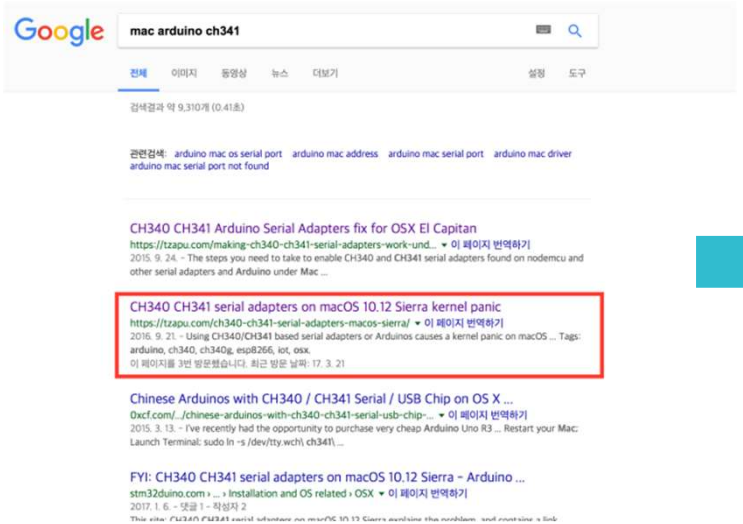
윈도우의 경우 검색입력창에서 'http://blog.naver.com/makist2015' 를 입력하셔서 메이키스트 블로그에 들어간 뒤 왼쪽 검색란에 'CH341'을 검색한 후 글에서 첨부파일을 다운 받습니다.

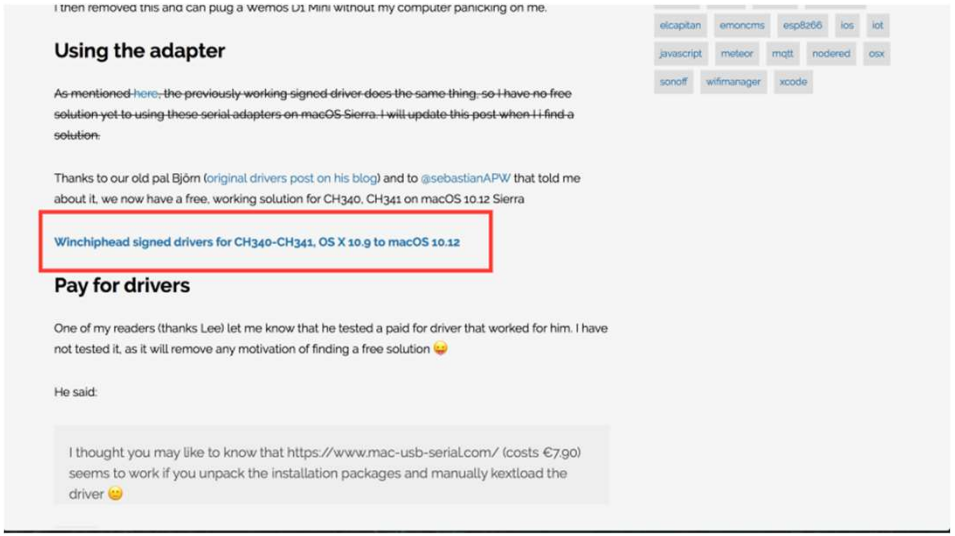


1. 첨부파일을 클릭
2. CH341을 내PC에 저장

## 통신 드라이버 설치하기

맥의 경우 검색입력창에서 'mac arduino ch314' 를 입력하셔서 나타난 두번째 사이트로 들어갑니다.

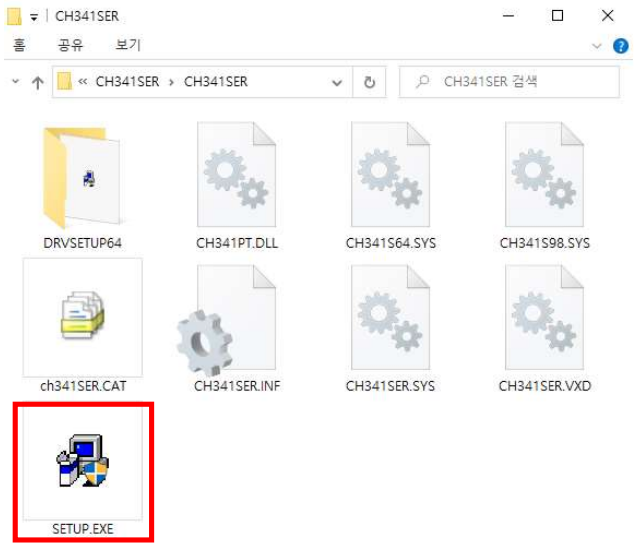


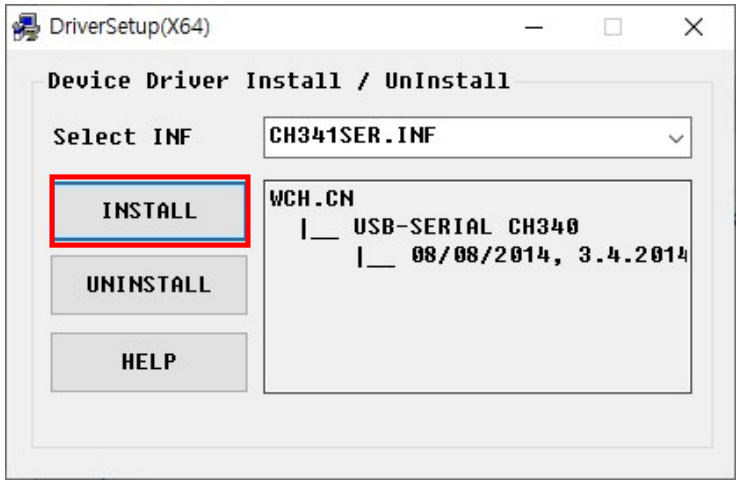

1. 파란색 링크 클릭
2. 다운된 드라이버 설치

## 통신 드라이버 설치하기

압축파일을 푼 후 SETUP.EXE 파일을 설치해 줍니다.



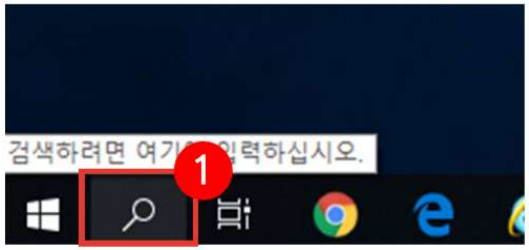

1. SETUP.EXE 파일 클릭



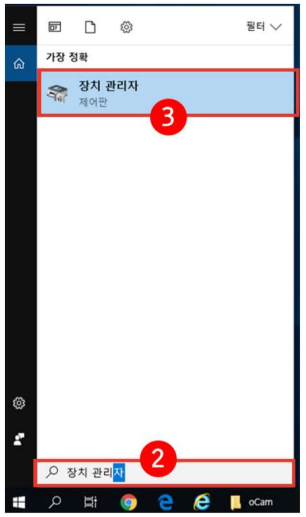

1. INSTALL 클릭
2. 설치가 끝나면 창을 닫습니다.

## 통신 드라이버 설치하기

준비된 USB 케이블로 아두이노와 PC를 연결한 후 '장치관리자'를 실행합니다.




1. 시작버튼옆 '검색' 아이콘 클릭



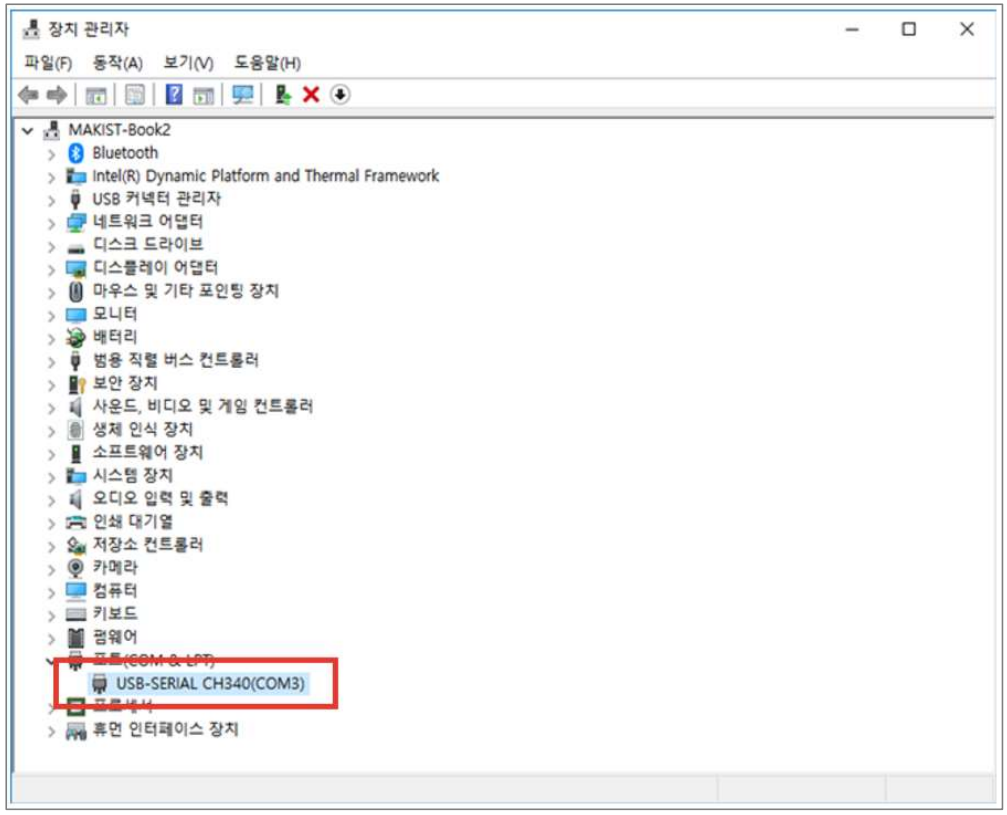

2. 검색어 '장치관리자'를 입력

3. 검색된 '장치 관리자'를 선택

장치 관리자 화면의 하단에 '포트(COM & LPT)' 항목 아래 'USB-SERIAL CH340(COM..)' 항목이 있는지 확인후 기억합니다.

만약, USB-SERIAL' 또는 'Arduino Uno' COM 포트가 보이지 않는다면 위의 설치 과정을 다시 진행하거나, PC의 다른 USB 포트에 연결 해봐야 합니다.

# 통신 드라이버 설치하기

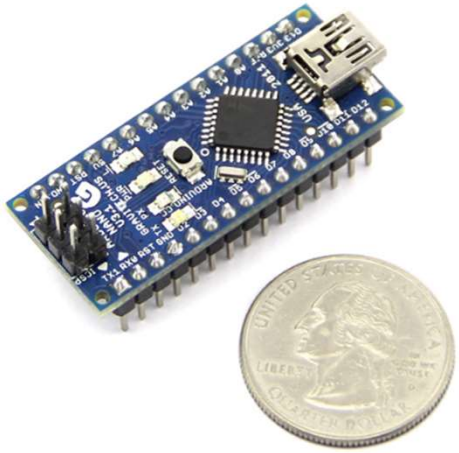



확인 후 창을 닫습니다.

# 소스파일 작성하기

## 보드 선택

- 아두이노 보드 시리즈



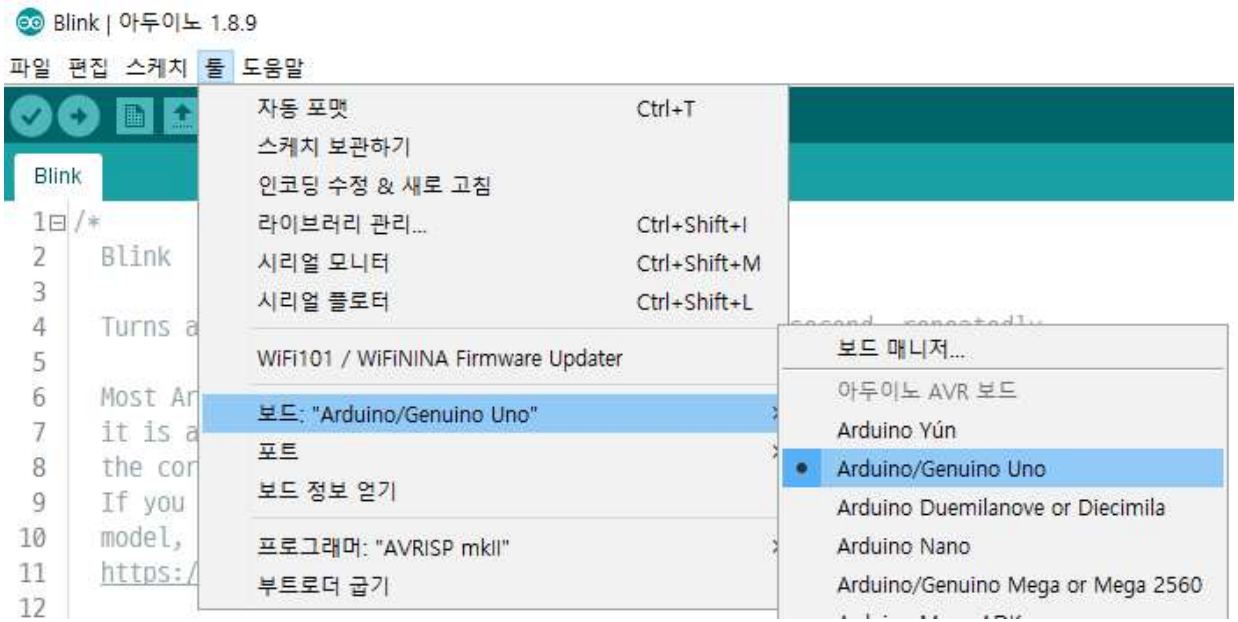
아두이노 Nano

아두이노 Mega


아두이노 Yun

# 소스파일 작성하기

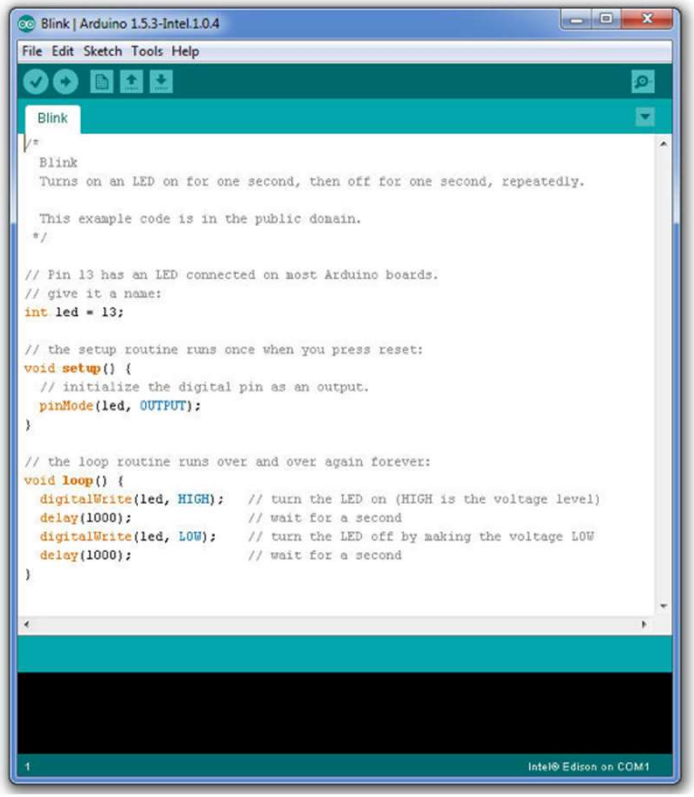
## 보드 선택




툴 -> 보드 -> Arduino/Genuino Uno 선택

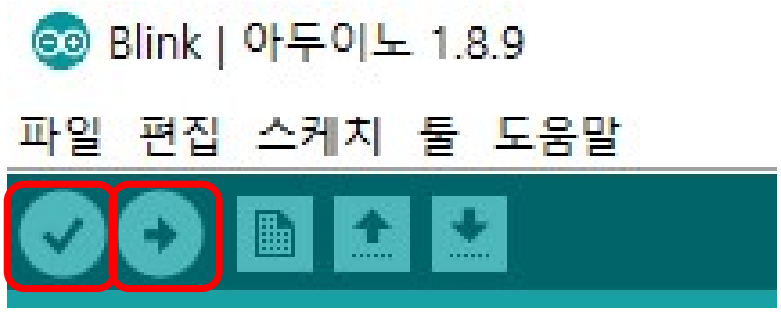
# 소스파일 작성하기

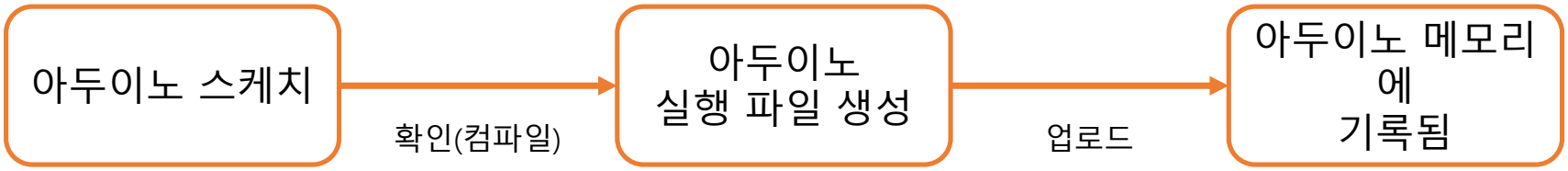
- 아두이노 프로그램 소스를 부르는 명칭
- C/C++ 언어로 작성
- 스케치 명명규칙
  - 숫자가 제일 앞에 오면 안된다.
  - 특수 문자가 포함되면 안된다. ( _ 제외)
  - 띄어쓰기를 하면 안된다.
  - 한글을 사용하면 안된다.



## 확인(컴파일), 업로드

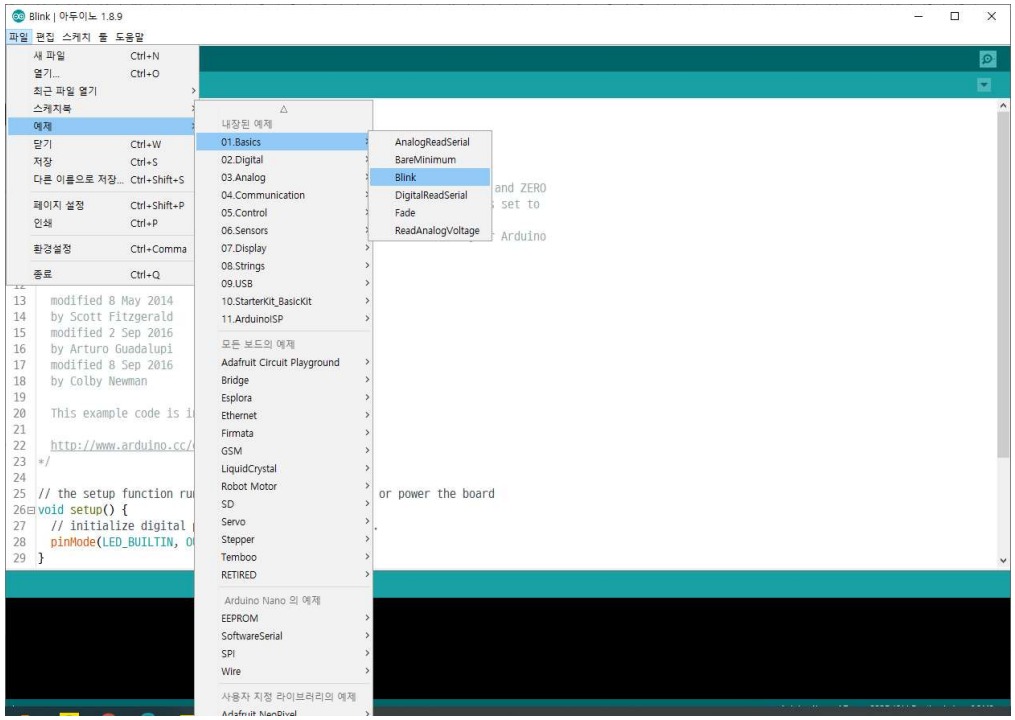
- 아두이노 프로그램을 보드에 옮기는 과정
- 한번 업로드되면 전원이 꺼져도 유지됨
- 보드 전원이 켜지면 바로 실행됨
- 리셋 버튼을 누르면 업로드된 프로그램 다시 실행



# 실습

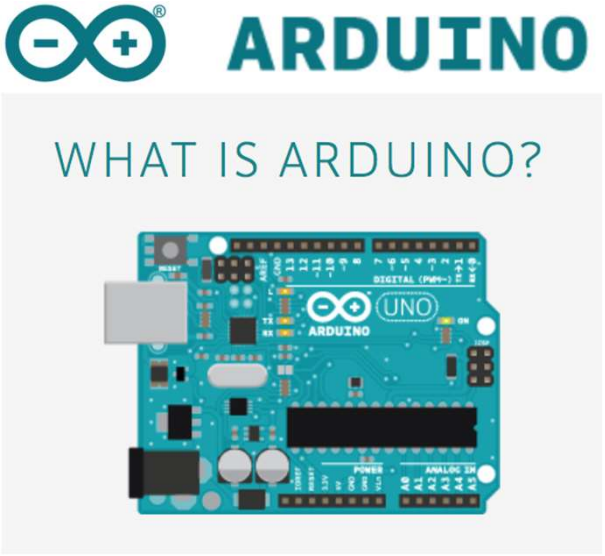
## LED Blink 예제 실행




좌측 상단의에서 파일 -> 예제 -> 01. Basic -> Blink 열기

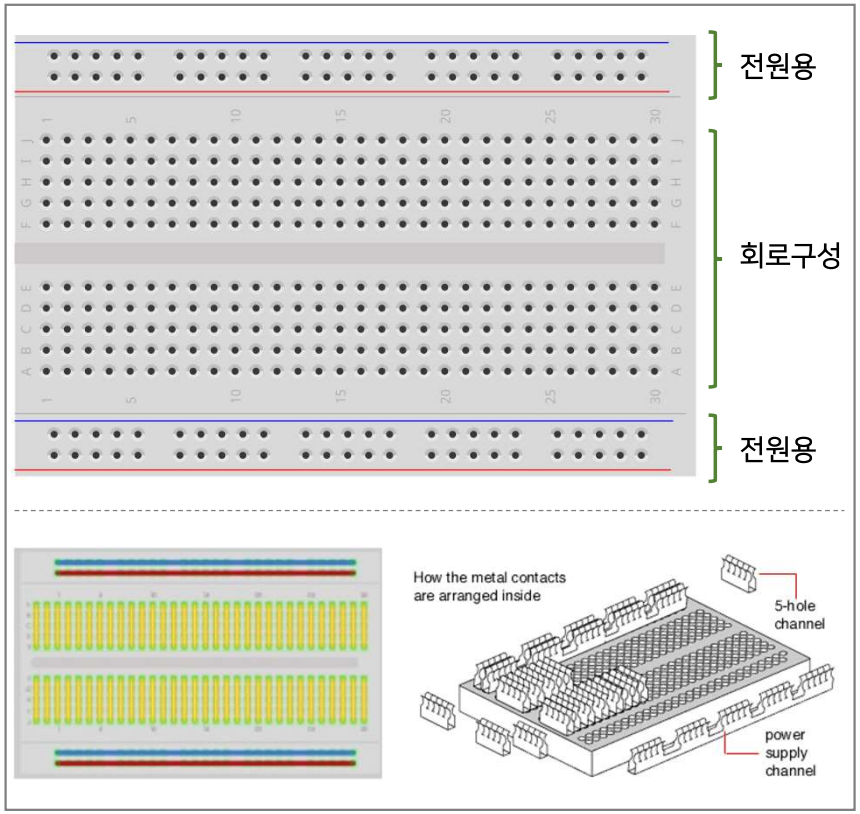
# Chapter 02

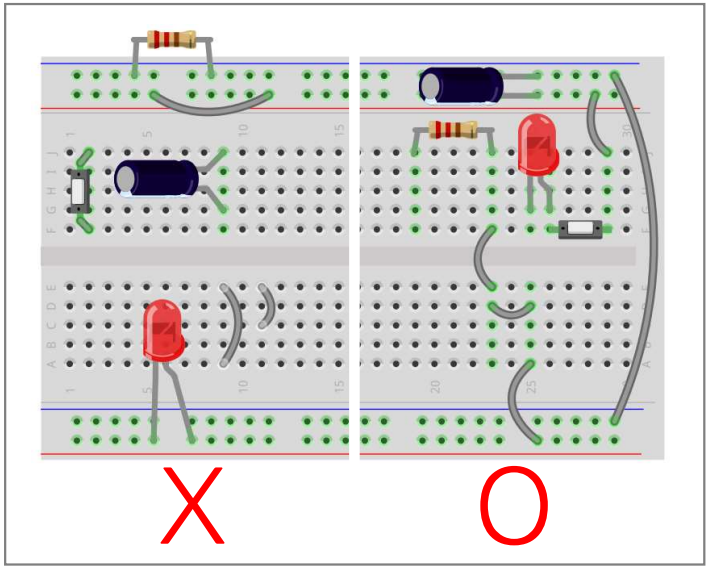
# 브래드보드 이해하기




Step 1. 브래드보드 이해하기

# 브레드보드(빵판) 이해하기



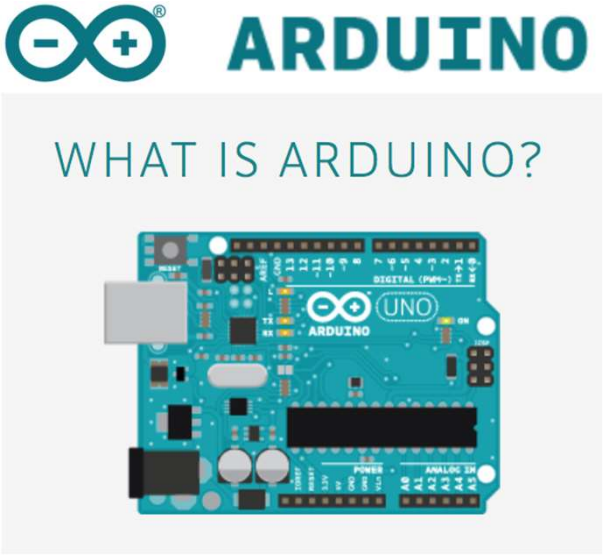




브레드보드 사용시 쇼트가 나지 않도록 주의해서 사용해야 합니다.

브레드보드는 설계한 회로가 정상적으로 작동하는지를 확인하기 위해서 사용합니다.
납땜 없이 자유롭게 회로를 구성 할 수 있습니다.

# Chapter 03

# LED 사용해보기




Step 1. LED 점멸

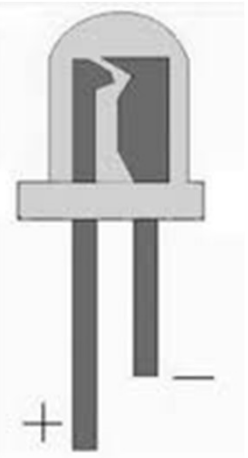
Step 2. PWM을 이용하여 밝기 바꾸기

Step 3. 애노드 캐소드 이해

# LED 점멸

## LED

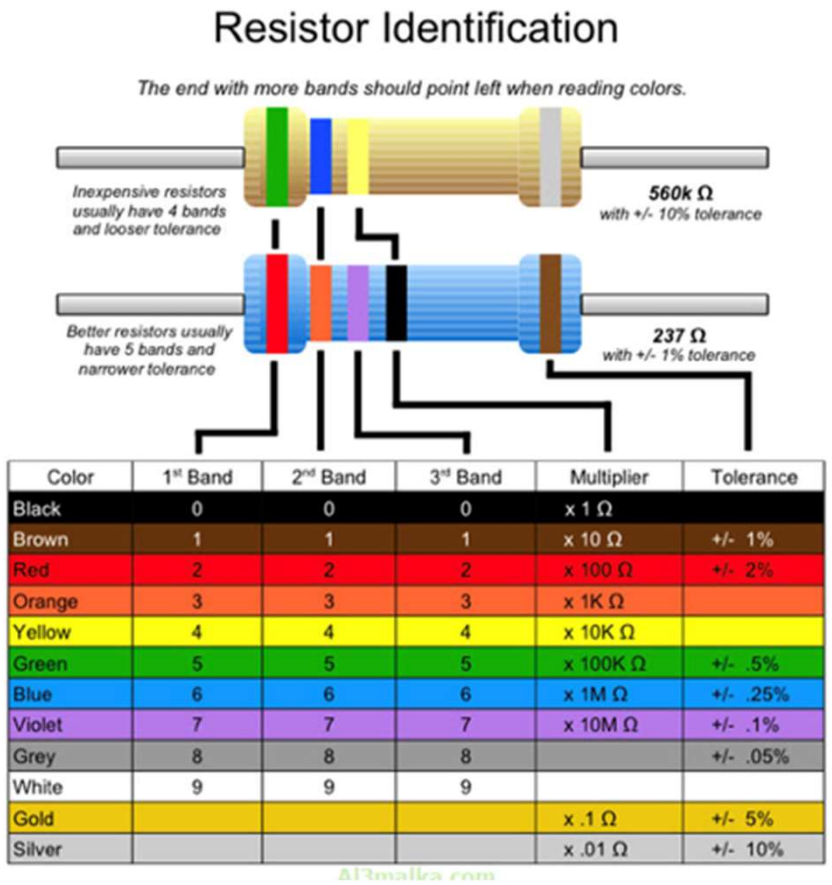
- 전류가 흐르면 빛이난다.
  (전류가 셀수록 빛이 밝아진다.)
- 핀 극성에 주의
- 정격
  - 약 0.1W = 5V * 0.02A (5V인가 시 20mA가 최대)
  - 정격을 지키기 위해 저항으로 조절한다.
    (R = V / I이므로 최소 250옴이상의 저항이 필요하다.)
  - 정격을 지키지 않으면 타게된다.



# LED 점멸

## 저항(Resistor)

- 띠 색깔로 저항 크기 표시

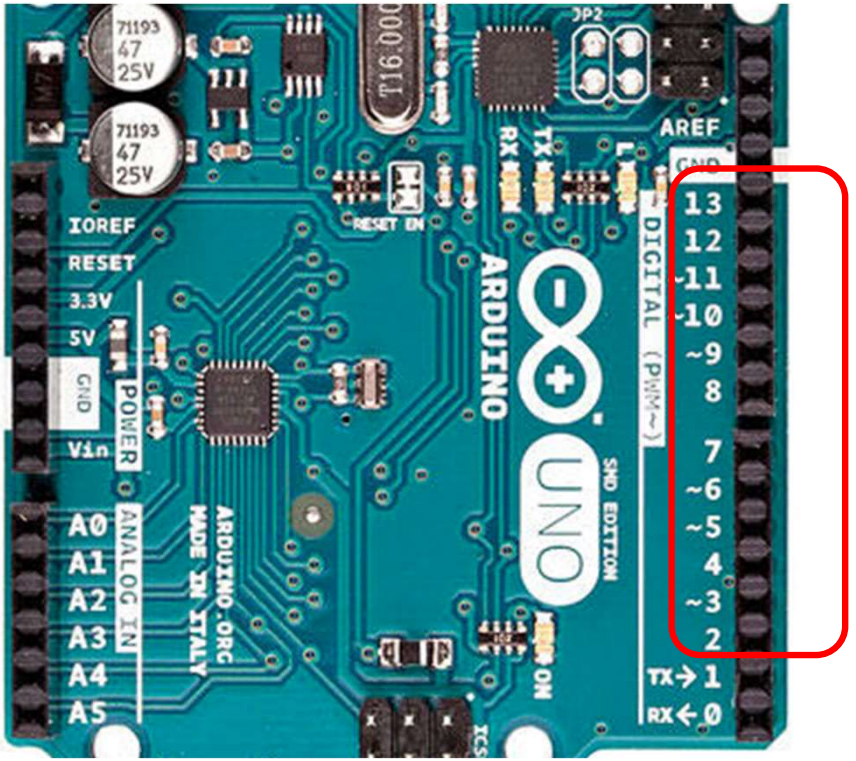



| Color | 1st Band | 2nd Band | 3rd Band | Multiplier | Tolerance |
|---|---|---|---|---|---|
| Black | 0 | 0 | 0 | x 1 Ω | |
| Brown | 1 | 1 | 1 | x 10 Ω | +/- 1% |
| Red | 2 | 2 | 2 | x 100 Ω | +/- 2% |
| Orange | 3 | 3 | 3 | x 1K Ω | |
| Yellow | 4 | 4 | 4 | x 10K Ω | |
| Green | 5 | 5 | 5 | x 100K Ω | +/- .5% |
| Blue | 6 | 6 | 6 | x 1M Ω | +/- .25% |
| Violet | 7 | 7 | 7 | x 10M Ω | +/- .1% |
| Grey | 8 | 8 | 8 | | +/- .05% |
| White | 9 | 9 | 9 | | |
| Gold | | | | x .1 Ω | +/- 5% |
| Silver | | | | x .01 Ω | +/- 10% |

## LED 점멸

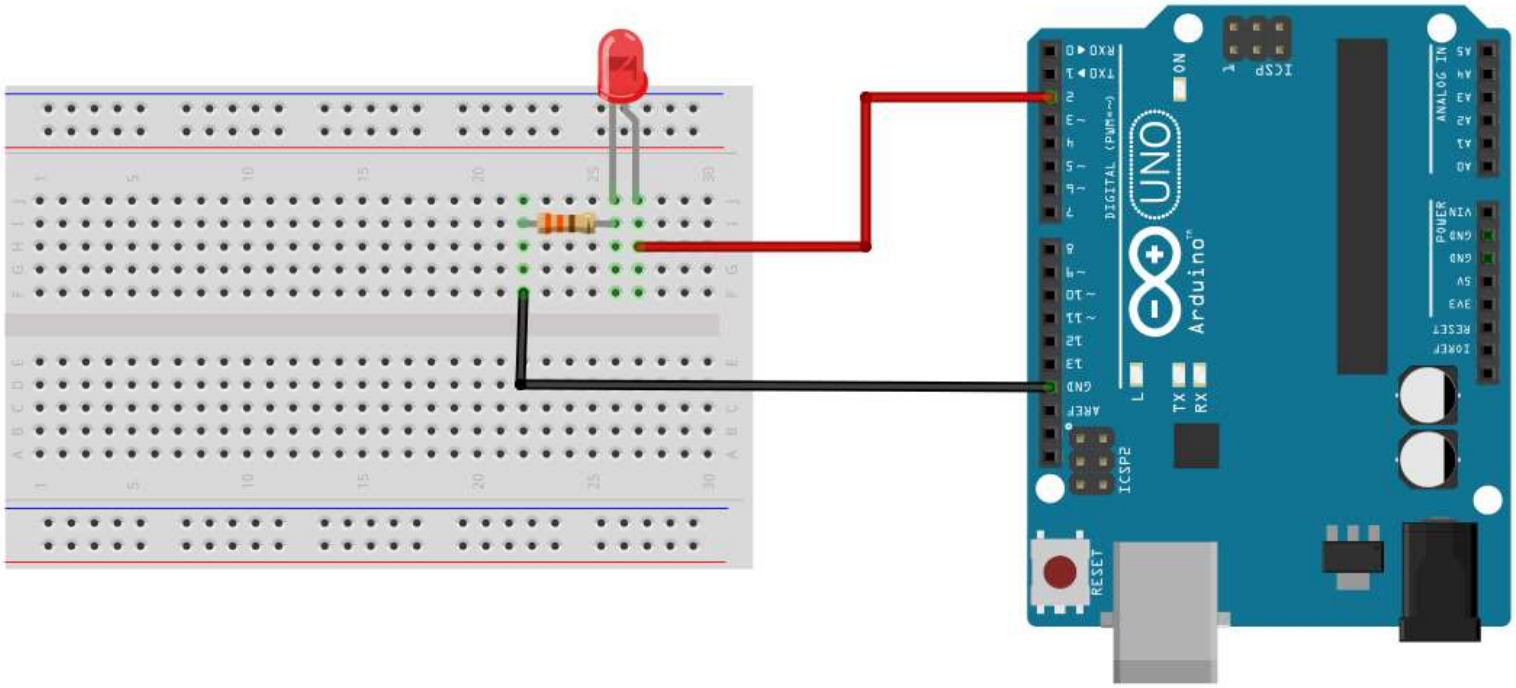
### 아두이노 디지털 출력

- 디지털 핀 사용
  - D2 ~ D13까지 12개 사용 가능
- 출력 핀으로 설정
  - 입력과 출력 공용
- 디지털 제어
  - 5V 혹은 0V 상태 제어 가능



# LED 점멸

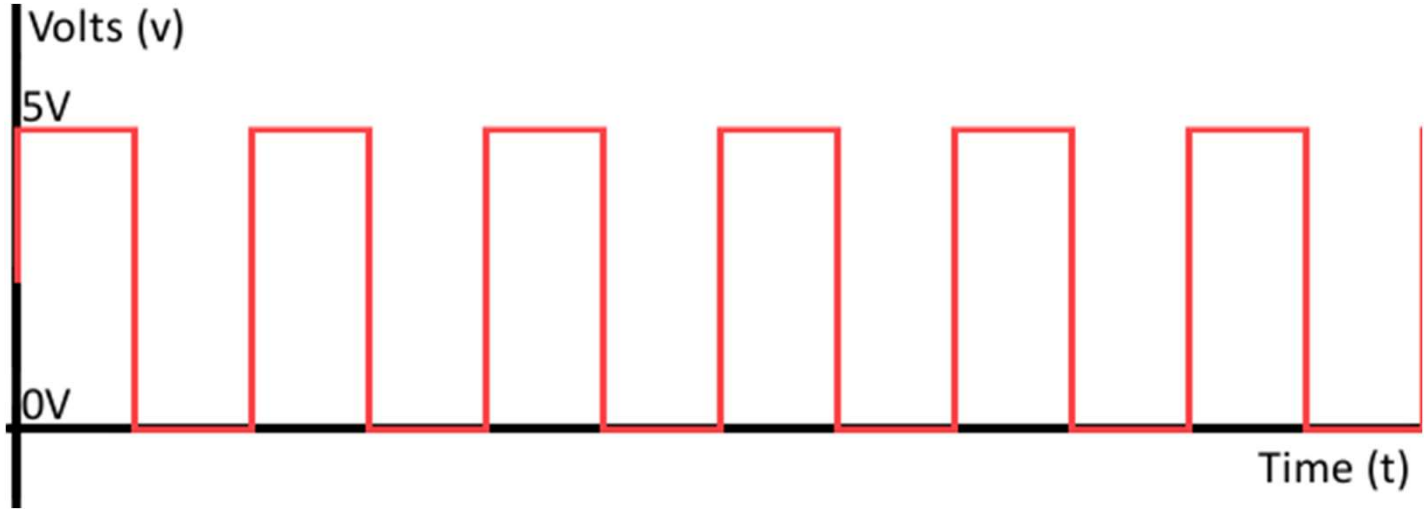
## LED 회로 구현



# LED 점멸

## LED 깜박이기

### 디지털 신호만들기



# LED 점멸

## setup / loop

- setup
  - 보드에 전원이 켜지면 한번만 실행
  - 초기화관련 코드 구현
- loop
  - setup실행 이후에 반복적으로 실행
  - 반복적으로 수행할 코드 구현

# LED 점멸

## pinMode

- 아두이노 디지털 핀의 출력 혹은 입력으로 용도 결정
- C언어 문법
  - void pinMode(int pin, int mode);
  - 반환값 없음, 인자 2개
- 사용방법
  - pin: 아두이노 디지털 핀 번호
  - mode: 출력 혹은 입력으로 사용할 지 여부
    - OUTPUT: 출력핀으로 설정
    - INPUT: 입력핀으로 설정
    - INPUT_PULLUP: 내부 풀업 입력핀으로 설정
  - 예) pinMode(3, OUTPUT); // 3번 핀을 출력으로 설정

# LED 점멸

## digitalWrite

- 아두이노 디지털 핀의 출력 상태 제어
- C언어 문법
  - void digitalWrite(int pin, int state);
  - 반환값 없음, 인자 2개
- 사용방법
  - pin: 아두이노 디지털 핀 번호
  - state: 출력 핀의 상태
    - HIGH: 5V 상태
    - LOW: 0V 상태
  - 예) digitalWrite(3, LOW); // 3번 핀을 0V로 제어

# LED 점멸

## delay

- 지정한 시간동안 프로그램이 멈춤
- C언어 문법
  - void delay(unsigned long ms);
  - 반환값 없음, 인자 1개
- 사용방법
  - ms: 멈출 시간, millisecond(1/1000초) 단위
  - 예) delay(1000); // 1000ms = 1sec 초 동안 멈춤

# LED 점멸

## C언어 변수 타입별 사용 수 범위

| 타입 | 크기 (Byte) | 수 범위 |
|---|---|---|
| char | 1 | -128 ~ 127 |
| unsigned char | 1 | 0 ~ 255 |
| int | 2 | -32768 ~ 32767 |
| unsigned int | 2 | 0 ~ 65535 |
| long | 4 | -2147483648 ~ 2147483647 |
| unsigned long | 4 | 0 ~ 4294967295 |
| float | 4 | 1.2E-38 ~ 3.4E+38 |

# LED 점멸

## LED Blink 소스코딩

```
#define LED_PIN    2
#define HIGH_TIME    1000 //msec
#define LOW_TIME     1000 //msec

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

void loop()
{
  digitalWrite(LED_PIN, HIGH);
  delay(HIGH_TIME);

  digitalWrite(LED_PIN, LOW);
  delay(LOW_TIME);
}
```

# PWM을 이용하여 밝기 바꾸기

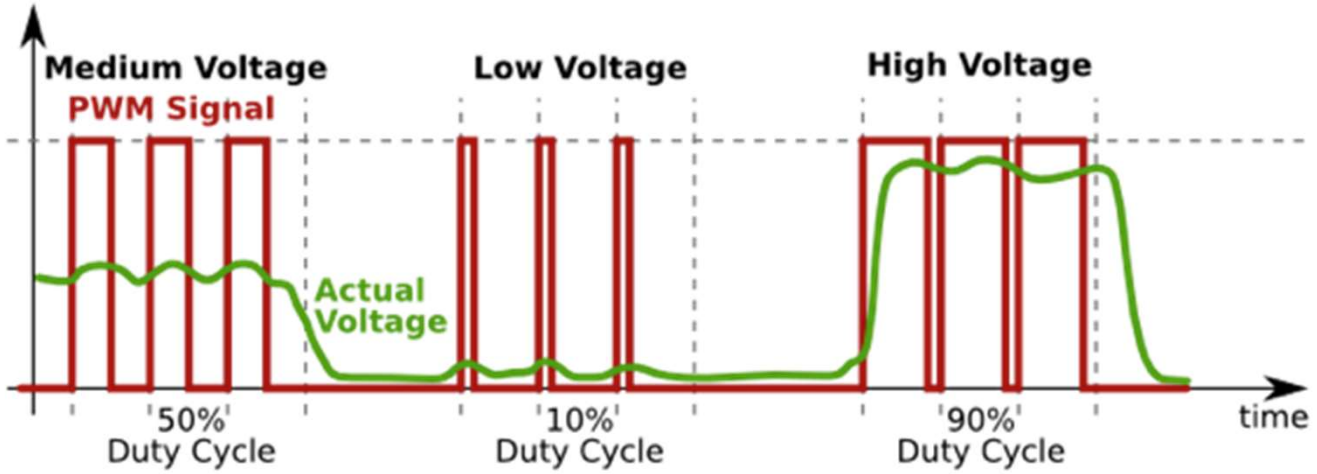
## LED 부드럽게 깜박이기

- 아날로그 신호만들기

# PWM을 이용하여 밝기 바꾸기

## PWM 신호

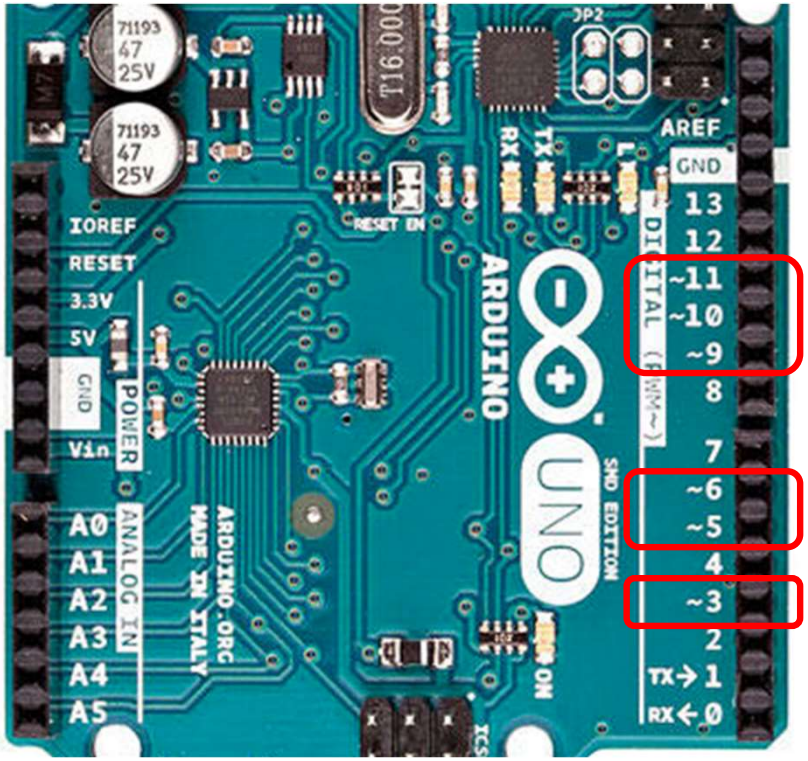
- 디지털 신호로 아날로그 신호를 만드는 기법
- PWM(Pulse Width Modulation)
  - 펄스 폭에 정보를 싣는 신호



# PWM을 이용하여 밝기 바꾸기

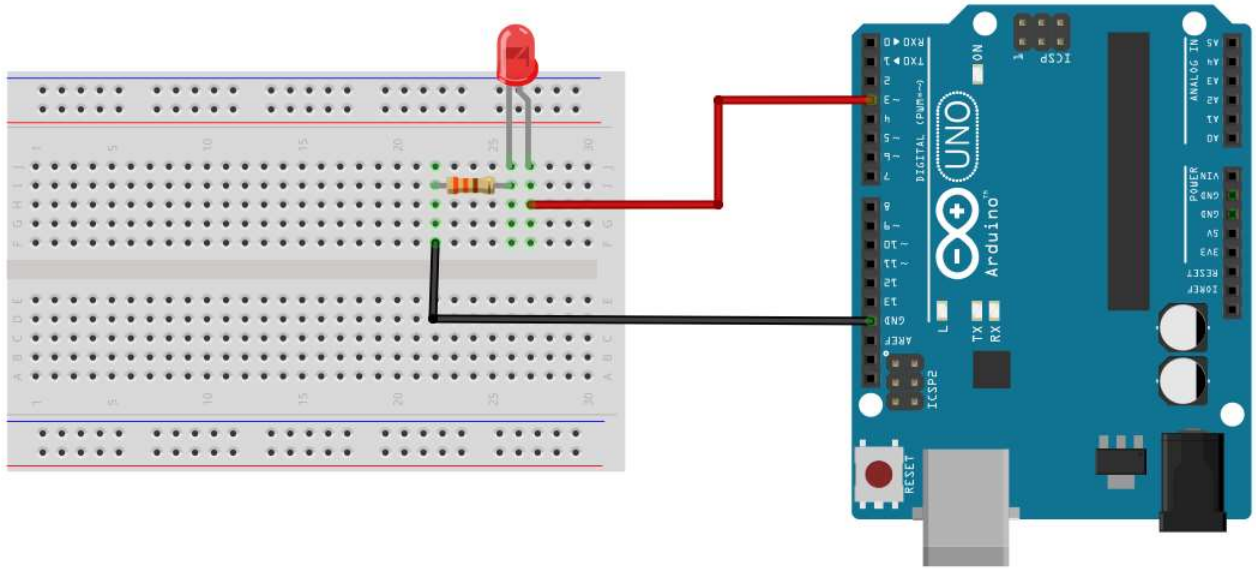
## 아두이노 PWM 핀

- ~표시 핀
  - Uno보드의 경우 6개
    (3, 5, 6, 9, 10, 11번 핀)



# PWM을 이용하여 밝기 바꾸기

## LED Analog 제어 회로 구현



# PWM을 이용하여 밝기 바꾸기

## analogWrite

- 아두이노 PWM 출력 상태 제어
- C언어 문법
  - void analogWrite(int pin, int value);
  - 반환값 없음, 인자 2개
- 사용방법
  - pin: 아두이노 PWM 핀 번호
  - value: PWM의 펄스 폭 (0 ~ 255)
  - 예) analogWrite(3, 50); // PWM 3번 핀을 약 20%로 제어

# PWM을 이용하여 밝기 바꾸기

## C언어 For 반복문

- 지정한 수만큼 반복한다.

```
for (int i=0; i<255; i++)
{
  // 반복할 코드
}
```

# PWM을 이용하여 밝기 바꾸기

## LED Analog 제어 소스코딩

```
#define LED_PIN    3
#define HIGH_TIME 500 //msec
#define LOW_TIME   500 //msec
#define STEP_TIME 10 //msec

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

void loop()
{
  for(int i=0; i<256; i++)
  {
    analogWrite(LED_PIN, i);
    delay(STEP_TIME);
  }

  digitalWrite(LED_PIN, HIGH);
  delay(HIGH_TIME);
```

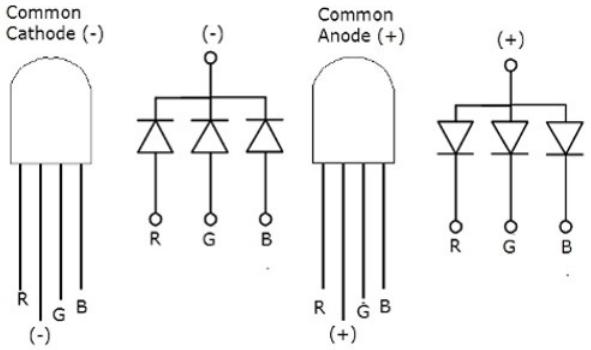
```
  for(int i=0; i<255; i++)
  {
    analogWrite(LED_PIN, 255 - i);
    delay(STEP_TIME);
  }

  digitalWrite(LED_PIN, LOW);
  delay(LOW_TIME);
}
```

# 애노드 캐소드 이해

## 애노드 캐소드 알아보기

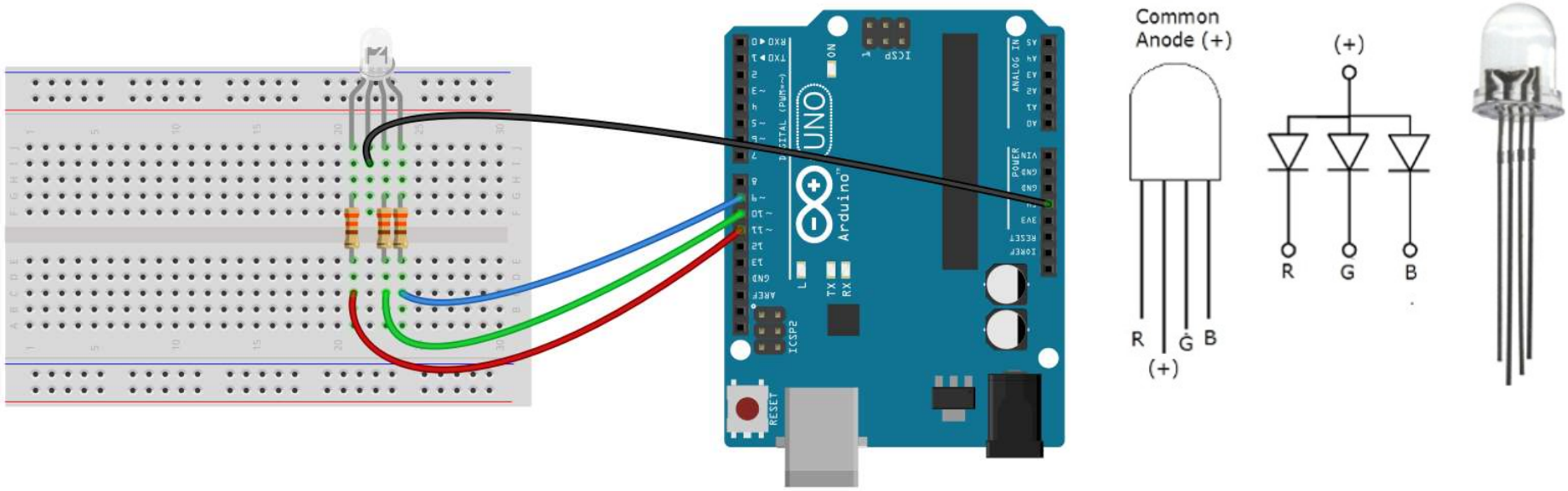
- Anode : (+)극으로 전자를 방출하거나 산화(oxidation) 반응이 일어나는 전극
  - 양극이 모두 묶여 한꺼번에 전원을 주면 반대방향의 캐소드에 MCU포트가 물려 LOW신호를 주면 동작하는 방식
- Cathod : (-)극으로 전자가 들어오거나 환원(reduction) 반응이 일어나는 전극
  - 음극이 모두 묶여 그라운드로 향하게 되는데 MCU에서 HIGH신호를 주면 동작하는 방식

*** 키트에서 RGB LED는 애노드 방식이며, 단색 LED는 캐소드 방식이다.**



# 애노드 캐소드 이해

## RGB LED 회로 구현

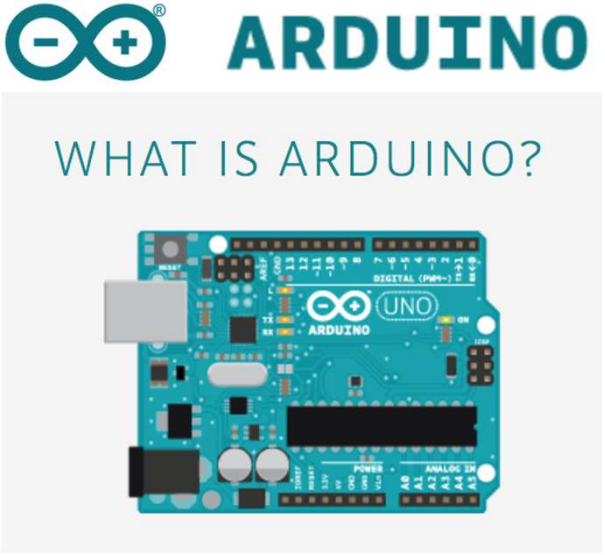


# 애노드 캐소드 이해

## RGB LED 소스코딩

```
#define LED_R_PIN    11
#define LED_G_PIN    10
#define LED_B_PIN    9

#define TIME    500 //msec

void setup()
{
    pinMode(LED_R_PIN, OUTPUT);
    pinMode(LED_G_PIN, OUTPUT);
    pinMode(LED_B_PIN, OUTPUT);
}

void loop()
{
    digitalWrite(LED_R_PIN, LOW);
    digitalWrite(LED_G_PIN, HIGH);
    digitalWrite(LED_B_PIN, HIGH);
    delay(TIME);

    digitalWrite(LED_R_PIN, HIGH);
    digitalWrite(LED_G_PIN, LOW);
    digitalWrite(LED_B_PIN, HIGH);
    delay(TIME);

    digitalWrite(LED_R_PIN, HIGH);
    digitalWrite(LED_G_PIN, HIGH);
    digitalWrite(LED_B_PIN, LOW);
    delay(TIME);

    digitalWrite(LED_R_PIN, LOW);
    digitalWrite(LED_G_PIN, LOW);
    digitalWrite(LED_B_PIN, LOW);
    delay(TIME);

    digitalWrite(LED_R_PIN, HIGH);
    digitalWrite(LED_G_PIN, HIGH);
    digitalWrite(LED_B_PIN, HIGH);
    delay(TIME);
}
```

# Chapter 04
# 스위치 사용해보기



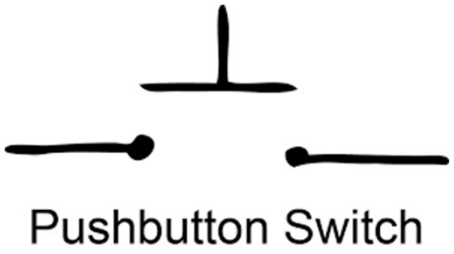

Step 1 . 택트 스위치 사용해보기

Step 2 . 슬라이드 스위치 사용해보기

# 택트 스위치 사용해보기

## 택트 스위치

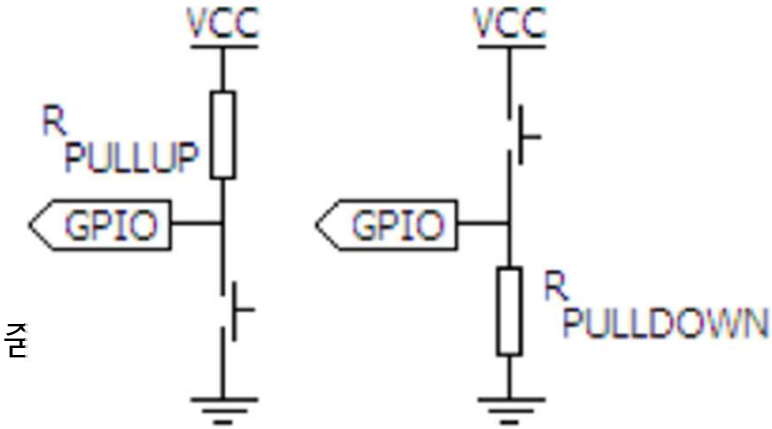
- 손으로 회로를 차단시키거나 연결시킬 수 있는 부품
- 핀 극성은 없다.
- 정격
  - 고려하지 않아도 됨



# 택트 스위치 사용해보기

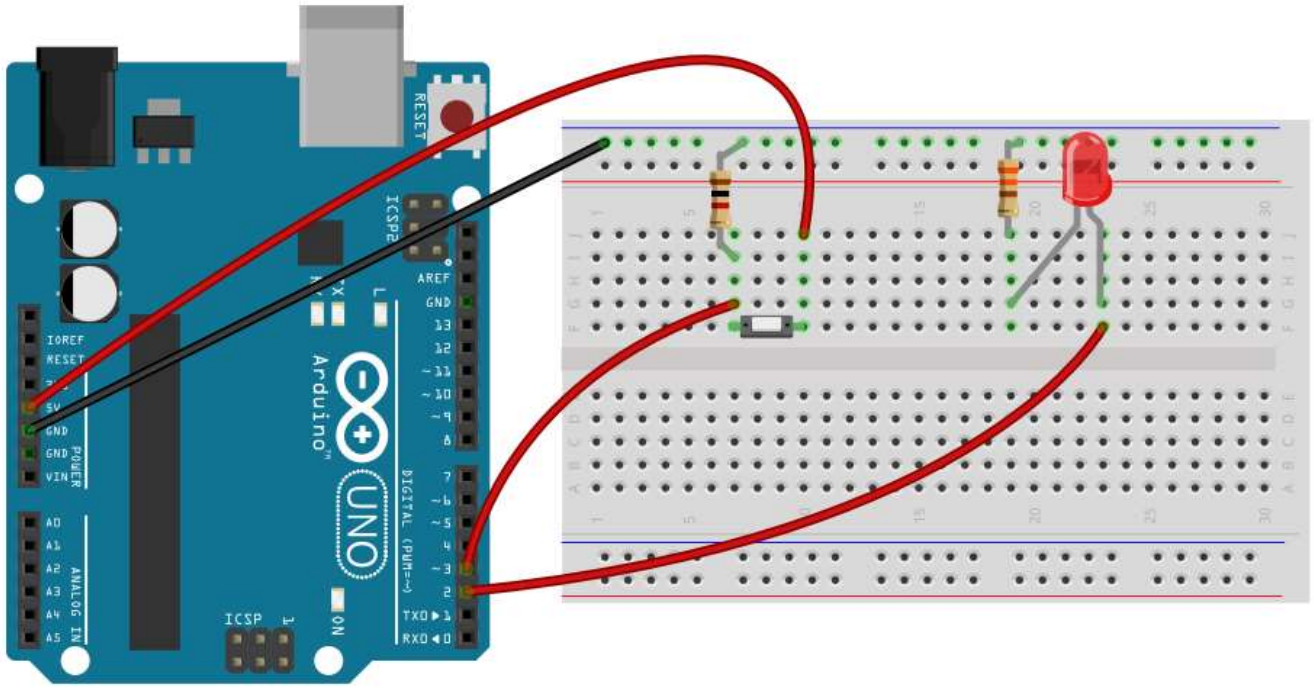
## 풀업 / 풀다운 회로

- Button의 경우 누르지 않았을 때 회로가 끊어진 상태이다.
- 이때의 전압은 애매한 크기를 가지기에 알 수가 없다.
- 이를 해결해주는 것이 풀업 / 풀다운 회로이다.
  - 풀업(Pull-up): 애매한 전압을 5V로 만들어 줌
    - 안 누름: 5V
    - 누름: 0V
  - 풀다운(Pull-down): 애매한 전압을 0V로 만들어 줌
    - 안 누름: 0V
    - 누름: 5V



# 택트 스위치 사용해보기

Button Pull-down 회로 구현 (테스트를 위해 LED 회로 추가)



# 택트 스위치 사용해보기

## pinMode

- 아두이노 디지털 핀의 출력 혹은 입력으로 용도 결정
- C언어 문법
  - void pinMode(int pin, int mode);
  - 반환값 없음, 인자 2개
- 사용방법
  - pin: 아두이노 디지털 핀 번호
  - mode: 출력 혹은 입력으로 사용할 지 여부
    - OUTPUT: 출력핀으로 설정
    - INPUT: 입력핀으로 설정
    - INPUT_PULLUP: 내부 풀업 입력핀으로 설정
  - 예) pinMode(3, OUTPUT); // 3번 핀을 출력으로 설정

# 택트 스위치 사용해보기

## digitalRead

- 아두이노 디지털 핀의 입력 상태 확인
- C언어 문법
  - int digitalRead(int pin);
  - 반환값 있음, 인자 1개
- 사용방법
  - pin: 아두이노 디지털 핀 번호
  - 반환값 : 입력 핀의 상태
    - HIGH: 5V 상태
    - LOW: 0V 상태
  - 예) int state = digitalRead(3); // 3번 핀의 상태를 state 변수에 저장

# 택트 스위치 사용해보기

## Button으로 LED 제어 (Pull-down방식)

- 버튼이 눌리면 LED 켜기
  - 버튼의 상태가 5V이면 LED 출력 5V
- 버튼이 안 눌리면 LED 끄기
  - 버튼의 상태가 0V이면 LED 출력 0V

# 택트 스위치 사용해보기

## C언어 if 조건문

- 조건에 맞으면 실행한다.
- if / else if / else 순으로 사용
  - if : 처음 조건이 맞다면
    (필수 사용)
  - else if : 그 다음 조건이 맞다면
    (선택적 사용, 여러 개 사용 가능)
  - else : 앞의 조건들이 안 맞는다면
    (선택적 사용)

```
if (a == 1)
{
  // a가 1이면 실행할 코드
}
else if (a == 2)
{
  // a가 2이면 실행할 코드
}
else if (a == 3)
{
  // a가 3이면 실행할 코드
}
else
{
  // a가 나머지 값이면 실행할 코드
}
```

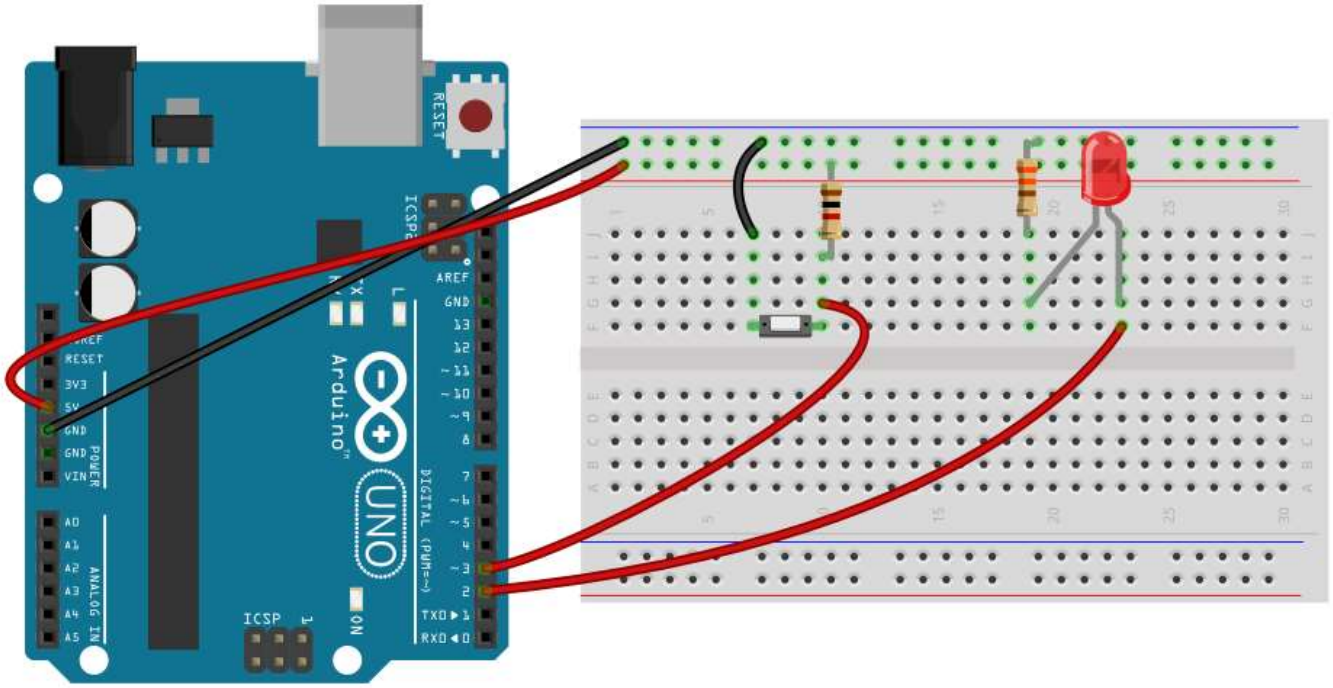
# 택트 스위치 사용해보기

## Button으로 LED 제어 소스코딩(Pull-down방식)

```
#define LED_PIN     2
#define BUTTON_PIN  3


void setup()
{
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);
}

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

# 택트 스위치 사용해보기

## Button Pull-down을 Pull-up으로 회로 수정



# 택트 스위치 사용해보기

## Button으로 LED 제어 (Pull-up방식)

- 버튼이 눌리면 LED 켜기
  - 버튼의 상태가 0V이면 LED 출력 5V
- 버튼이 안 눌리면 LED 끄기
  - 버튼의 상태가 5V이면 LED 출력 0V

# 택트 스위치 사용해보기

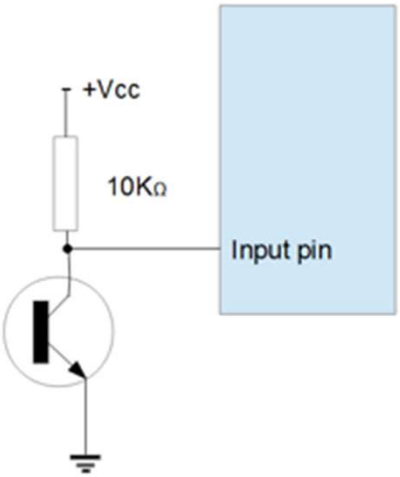
## Button으로 LED 제어 소스코딩(Pull-up방식)

```
#define LED_PIN     2
#define BUTTON_PIN  3


void setup()
{
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);
}

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

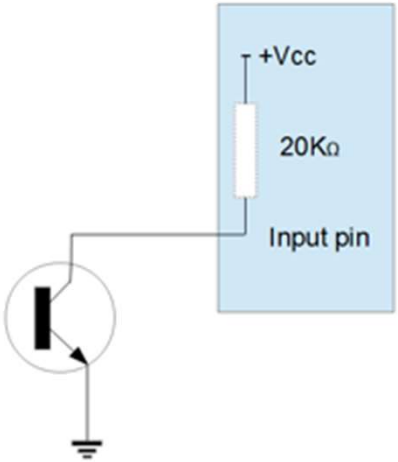
# 택트 스위치 사용해보기

## 내부 풀업

- 아두이노 보드는 풀업 회로가 내장되어 있다.
- pinMode 설정 시 INPUT_PULLUP 모드로 사용



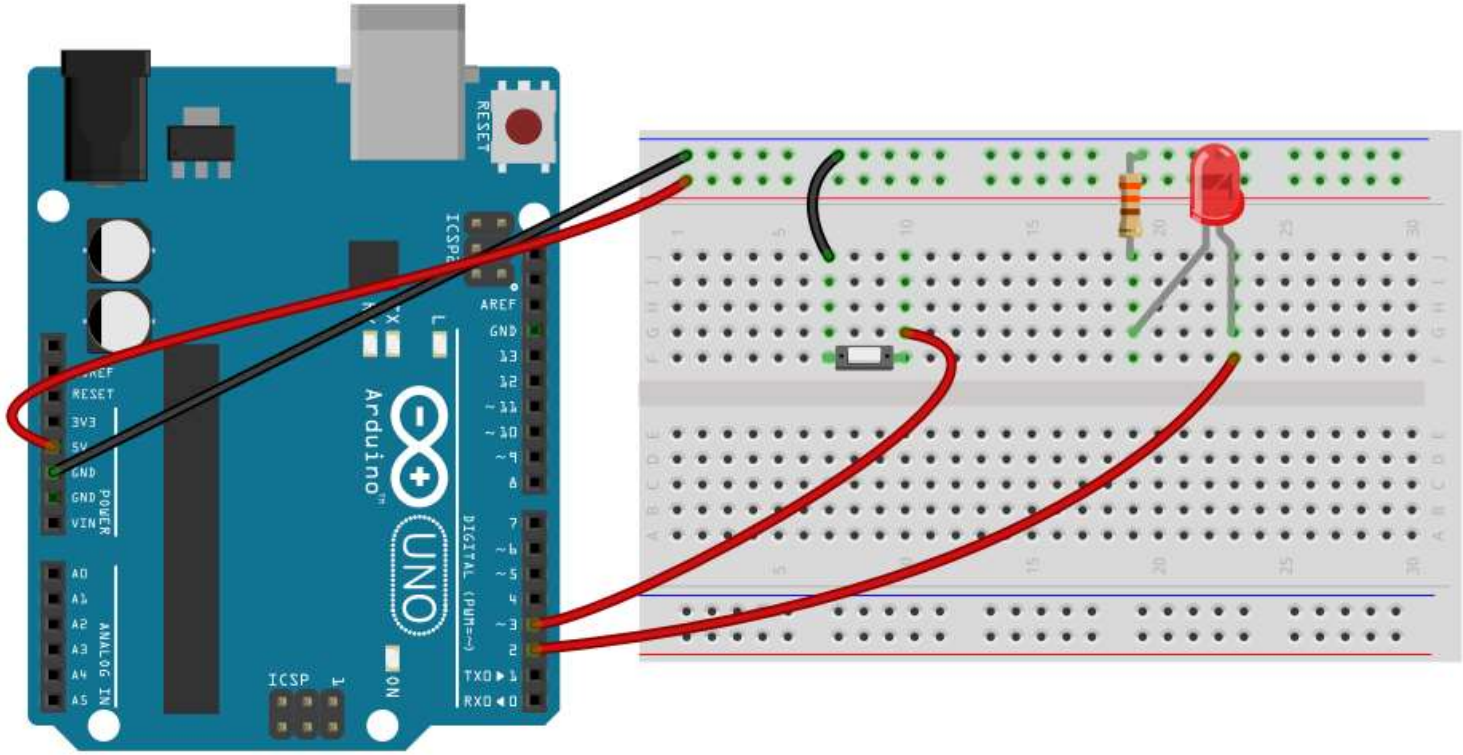

External pullup resistor




Internal pullup resistor with pin mode INPUT_PULLUP

# 택트 스위치 사용해보기

## Button Pull-up회로를 내부 Pull-up 회로로 수정



# 택트 스위치 사용해보기

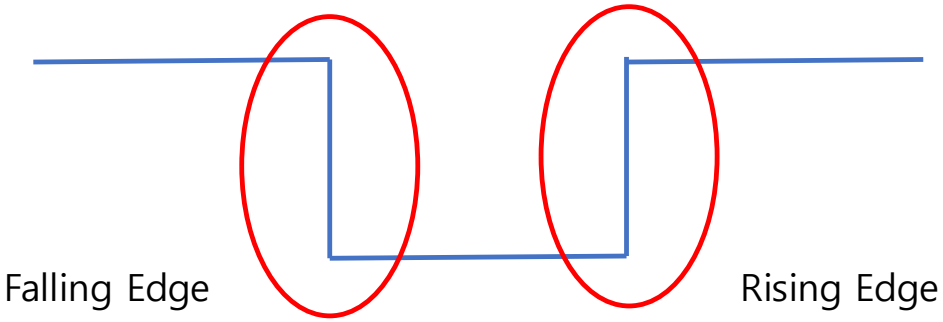
## 토글(Toggle) 제어

- 2가지 상태를 번갈아가며 제어하는 방식
  - 참이면 거짓으로 거짓이면 참으로 바꿈
- 버튼을 누를때마다 LED ON/OFF 제어

# 택트 스위치 사용해보기

## 에지(Edge) 입력

- 버튼 누름 상태가 아닌 누르거나 뗄 때를 체크하는 것
  - Button Down: 누르는 순간
  - Button Up: 떼는 순간



# 택트 스위치 사용해보기

## Button으로 LED 토글 방식 제어

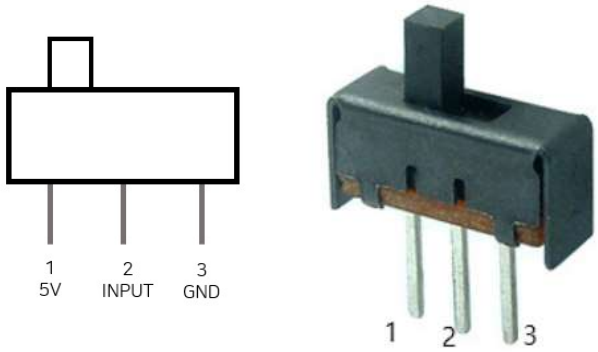
```
#define LED_PIN     2
#define BUTTON_PIN  3

int led_state = LOW;
int pre_button_state = HIGH;

void setup()
{
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
}

void loop()
{
  int button_state = digitalRead(BUTTON_PIN);
```

```
  if(button_state != pre_button_state)
  {
    if(button_state == LOW &&
     pre_button_state == HIGH) // Rising Edge
    {
      if(led_state == LOW) led_state = HIGH;
      else led_state = LOW;
    }
    pre_button_state = button_state;
  }
  digitalWrite(LED_PIN, led_state);
}
```

# 슬라이드 스위치 사용해보기

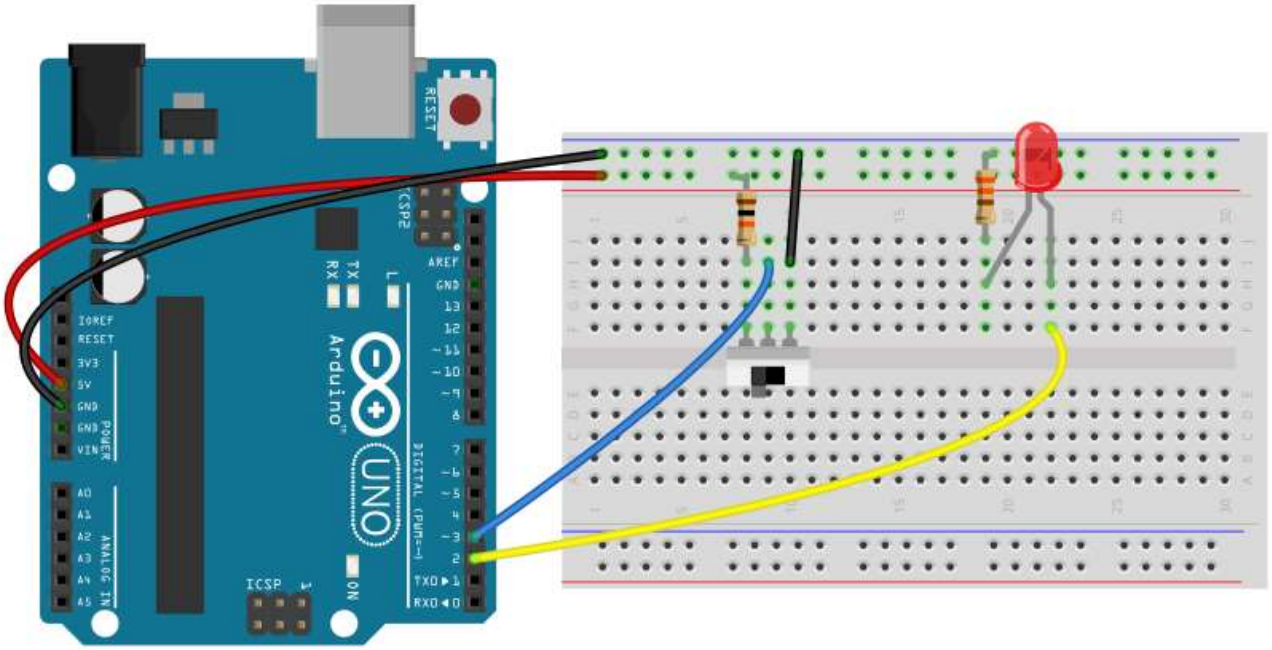
## 슬라이드 스위치

- 정적인 성질을 가짐
- 전원 스위치용으로 많이 사용됨



# 슬라이드 스위치 사용해보기

슬라이드 스위치 회로구현(테스트를 위해 LED 회로 추가)



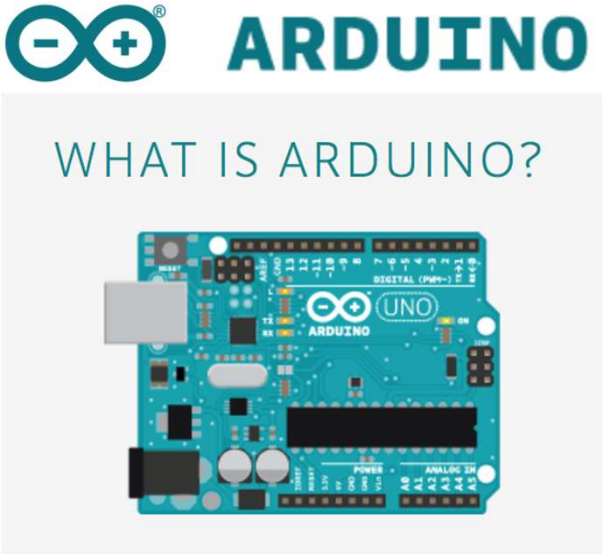
# 슬라이드 스위치 사용해보기

## 슬라이드 스위치 소스코딩

```
#define LED_PIN     2
#define BUTTON_PIN  3

void setup()
{
    pinMode(LED_PIN, OUTPUT);
    pinMode(BUTTON_PIN, INPUT);
}

void loop()
{
    int button_state = digitalRead(BUTTON_PIN);

    if (button_state == HIGH) {
        digitalWrite(LED_PIN, HIGH);
    }
    else {
        digitalWrite(LED_PIN, LOW);
    }
}
```

# Chapter 05

# 피에조 부저 사용해보기




Step 1. 피에조 부저를 사용하여 노래 만들기

# 피에조 부저를 사용하여 노래 만들기

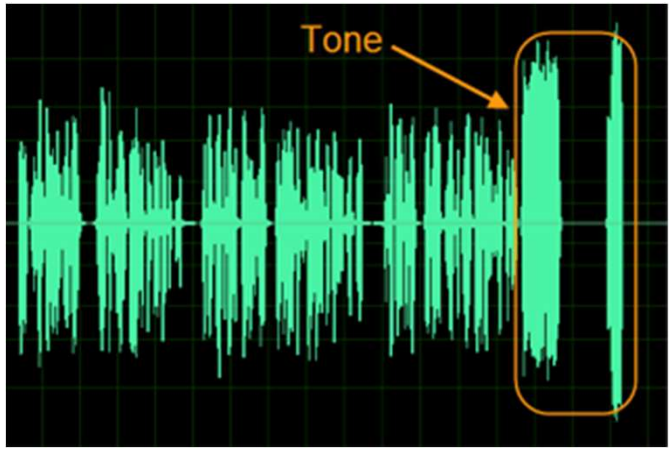
## 피에조 부저 (piezo buzzer)

- 전기적 신호를 소리로 바꾸는 전자 소자
- 정격
  - 정격 전압: 4 ~ 6V
  - 정격 전류: 40mA
  - 5V 입력 시 125옴 이상 사용

# 피에조 부저를 사용하여 노래 만들기

## 버저 제어 방법

- 전기적 신호의 주파수에 따라 음의 높낮이 제어
  - 저주파: 낮은 음
  - 고주파: 높은 음
  - 기본 '라' 음이 440Hz



# 피에조 부저를 사용하여 노래 만들기

## 아두이노 Tone라이브러리 사용

- 버저를 쉽게 사용할 수 있도록 라이브러리가 만들어져 있다.
- Tone 함수 호출
  - tone: 아두이노 핀에 음계 주파수를 출력함
  - noTone: 주파수 출력 멈춤

# 피에조 부저를 사용하여 노래 만들기

## tone

- 아두이노 핀에 음계 주파수 출력
- C언어 문법
  - void tone(int pin, unsigned int frequency);
  - 반환값 없음, 인자 2개
- 사용방법
  - pin: 아두이노 핀 번호
  - frequency: 음계 주파수
  - 예) tone(3, 440); // 3번 핀에 440Hz 주파수 출력

# 피에조 부저를 사용하여 노래 만들기

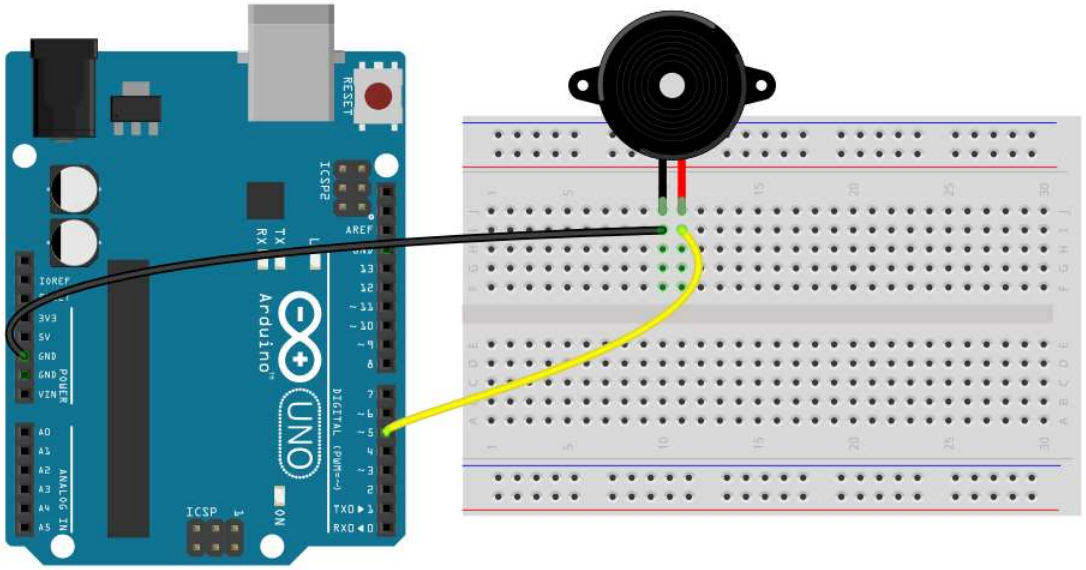
## 음계표

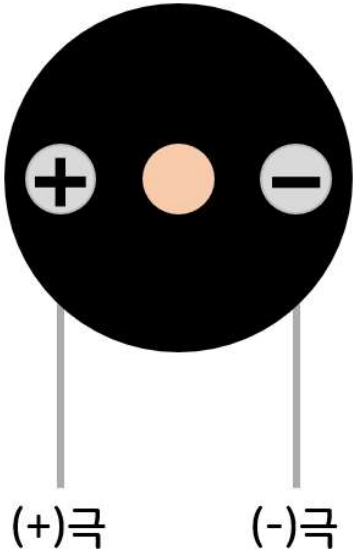
( 단위 : Hz )

| 옥타브 / 음계 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| C(도) | 32.7032 | 65.4064 | 130.8128 | 261.6256 | 523.2511 | 1046.502 | 2093.005 | 4186.009 |
| C# | 34.6478 | 69.2957 | 138.5913 | 277.1826 | 554.3653 | 1108.731 | 2217.461 | 4434.922 |
| D(레) | 36.7081 | 73.4162 | 146.8324 | 293.6648 | 587.3295 | 1174.659 | 2349.318 | 4698.636 |
| D# | 38.8909 | 77.7817 | 155.5635 | 311.1270 | 622.2540 | 1244.508 | 2489.016 | 4978.032 |
| E(미) | 41.2034 | 82.4069 | 164.8138 | 329.6276 | 659.2551 | 1318.510 | 2637.020 | 5274.041 |
| F(파) | 43.6535 | 87.3071 | 174.6141 | 349.2282 | 698.4565 | 1396.913 | 2793.826 | 5587.652 |
| F# | 46.2493 | 92.4986 | 184.9972 | 369.9944 | 739.9888 | 1479.978 | 2959.955 | 5919.911 |
| G(솔) | 48.9994 | 97.9989 | 195.9977 | 391.9954 | 783.9909 | 1567.982 | 3135.963 | 6271.927 |
| G# | 51.9130 | 103.8262 | 207.6523 | 415.3047 | 830.6094 | 1661.219 | 3322.438 | 6644.875 |
| A(라) | 55.0000 | 110.0000 | 220.0000 | 440.0000 | 880.0000 | 1760.000 | 3520.000 | 7040.000 |
| A# | 58.2705 | 116.5409 | 233.0819 | 466.1638 | 932.3275 | 1864.655 | 3729.310 | 7458.620 |
| B(시) | 61.7354 | 123.4708 | 246.9417 | 493.8833 | 987.7666 | 1975.533 | 3951.066 | 7902.133 |

# 피에조 부저를 사용하여 노래 만들기

## 피에조 부저 회로구현





# 피에조 부저를 사용하여 노래 만들기 (영상이 어려우면 아래 코드로 연습하세요~^^)

## 피에조 부저 소스코딩

```
#define BUZZER_PIN  11
void setup() {
    pinMode(BUZZER_PIN, OUTPUT);
}

void loop() {
    //옥타브 = 4
    tone(BUZZER_PIN, 262);  // 도
    delay(500);
    tone(BUZZER_PIN, 277); // 도#
    delay(500);
    tone(BUZZER_PIN, 294); //레
    delay(500);
    tone(BUZZER_PIN, 311); //레#
    delay(500);
    tone(BUZZER_PIN, 330); //미
    delay(500);
    tone(BUZZER_PIN, 349);  //파
    delay(500);
```

```
    tone(BUZZER_PIN, 370);  // 파#
    delay(500);
    tone(BUZZER_PIN, 392);  // 솔
    delay(500);
    tone(BUZZER_PIN, 415);  // 솔#
    delay(500);
    tone(BUZZER_PIN, 440);  //라
    delay(500);
    tone(BUZZER_PIN, 466);  // 라#
    delay(500);
    tone(BUZZER_PIN, 494);  // 시
    delay(500);
    tone(BUZZER_PIN, 523);  //도(옥타브5)
    delay(500);
}
```