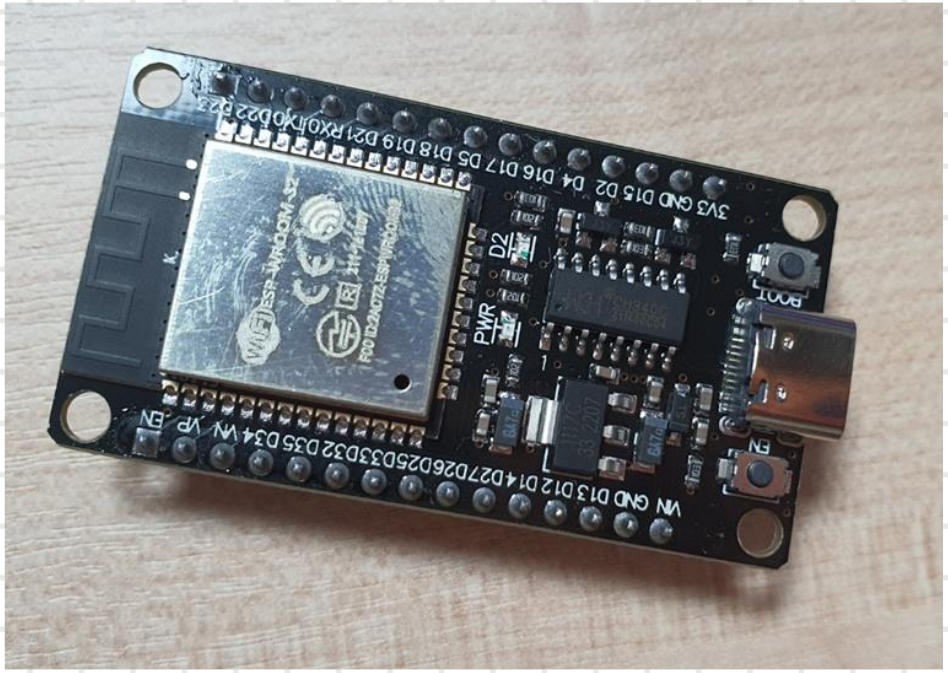
# 심화 강의 : ESP32

# ESP32



- ESP32는 아두이노와 같은 용도의 새로운 MCU(마이크로 콘트롤러 유닛) 도구
- 아두이노보다 고성능, 3.3V, 32비트
- 블루투스와 WiFi 모듈이 내장

# ESP32 구성요소

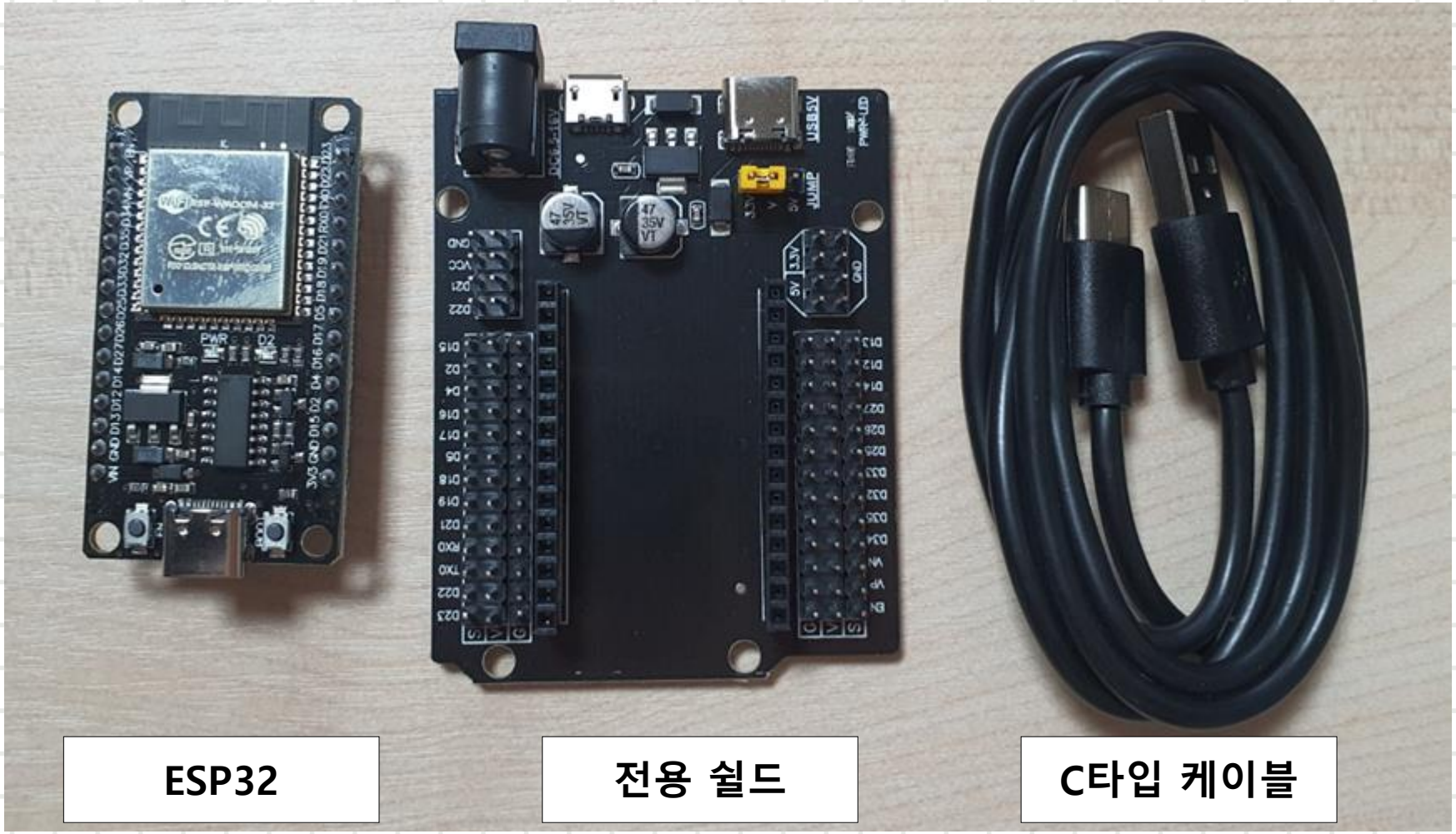
| | |
|---|---|
|  |  |
| ESP32 | 초음파센서 |
|  |  |
| 도트매트릭스 | 워터센서 |

| | | |
|---|---|---|
|  |  |  |
| 37종 센서 | 3W LED | USB 케이블 |
|  |  |  |
| 서보모터 | 신호등 | 점퍼케이블(FF) |

# ESP32 구성요소

37종 센서
-> INPUT_PULLUP 방식

# ESP32 준비



ESP32

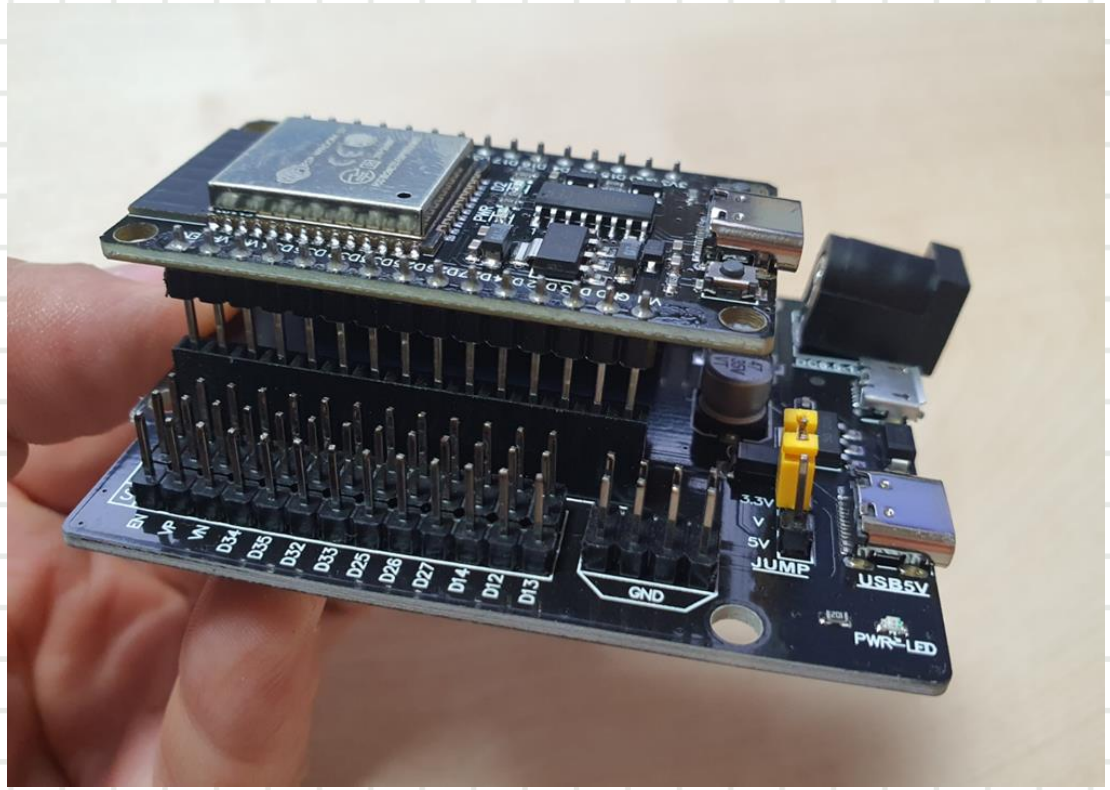
전용 쉴드

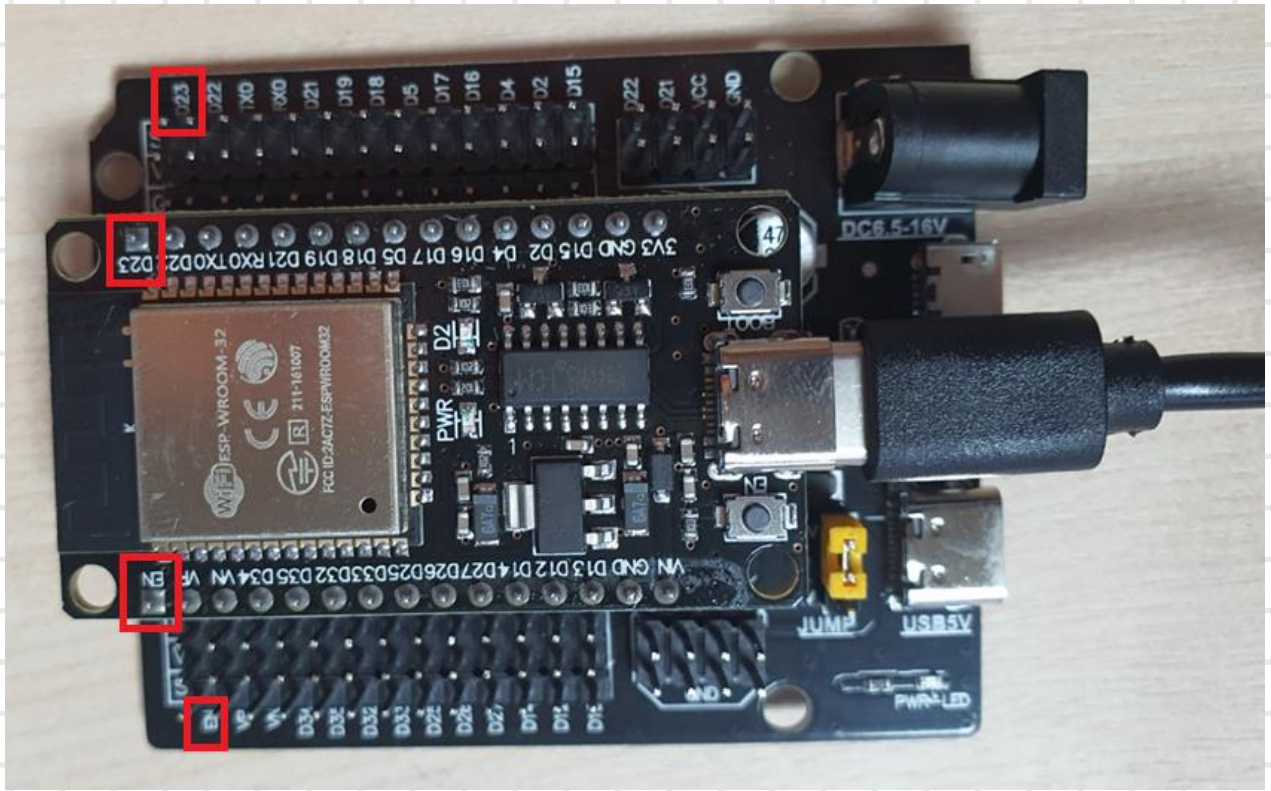
C타입 케이블

# ESP32 연결

<전용 쉴드와 ESP32 보드를 연결>

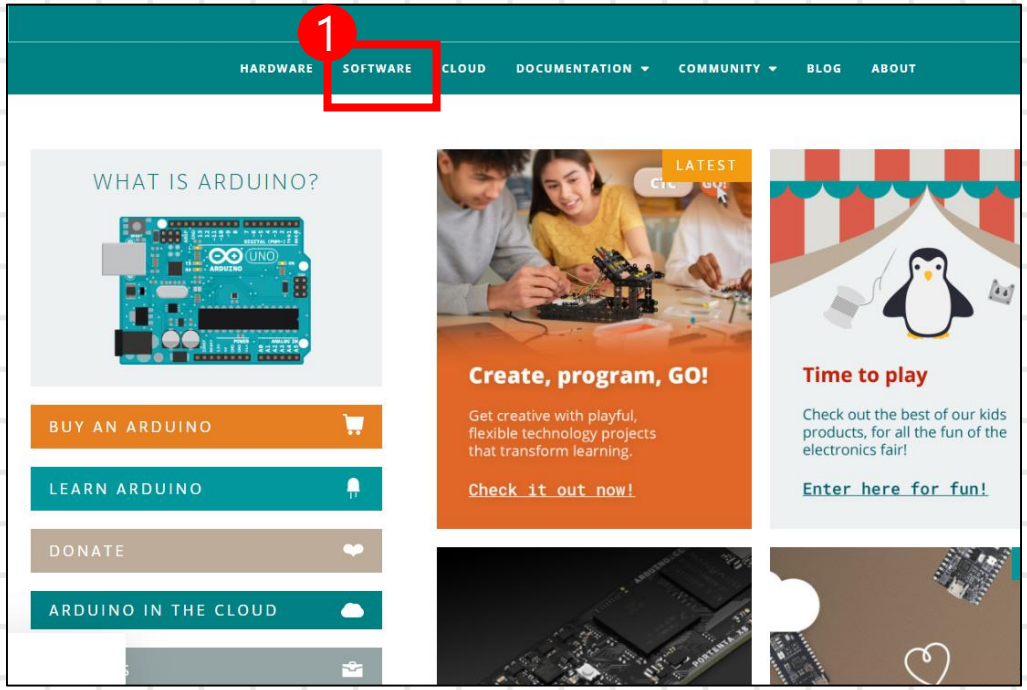
- 모든 핀이 다 연결되어야 하며, D23, EN 부분이 맞게 꽂기



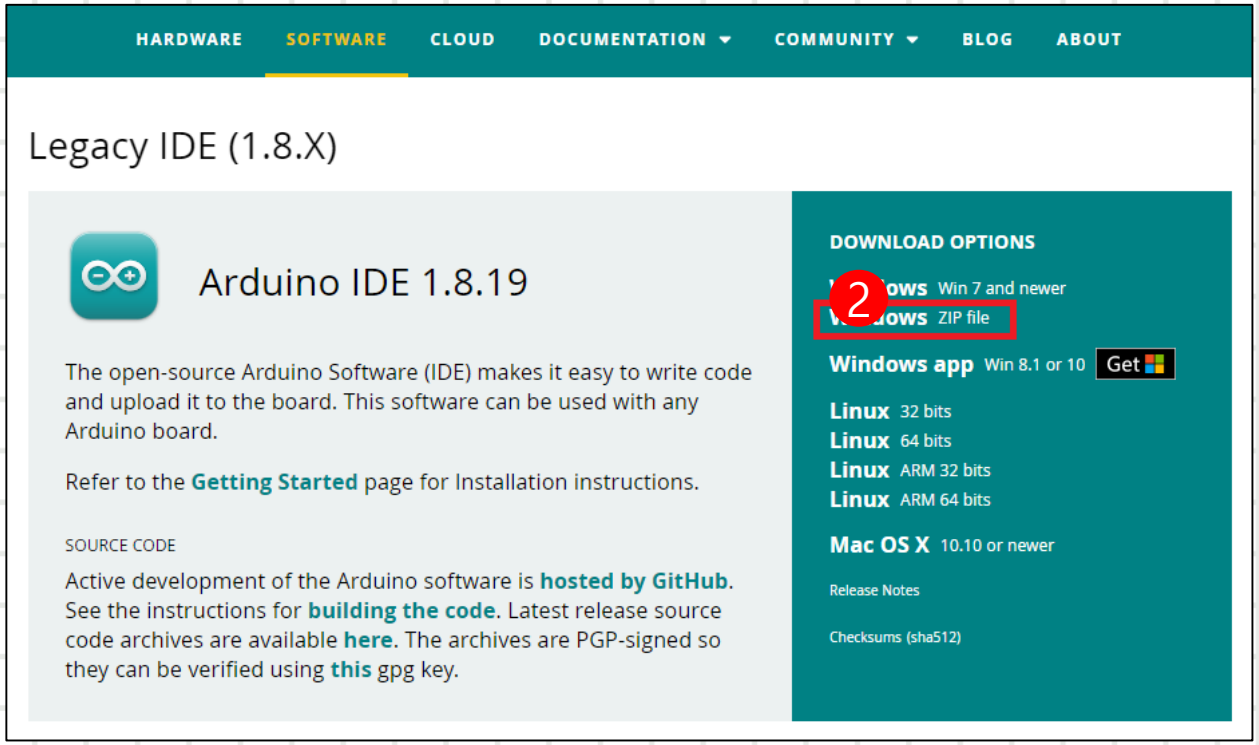



# 개발 툴 설치

- ESP32 독자적인 개발 툴이 제공되고 있으나 일반적으로 아두이노용 개발 툴(IDE)을 사용
- 아두이노 사이트(https://arduino.cc)에서 **개발 툴을 다운로드** 받아 설치

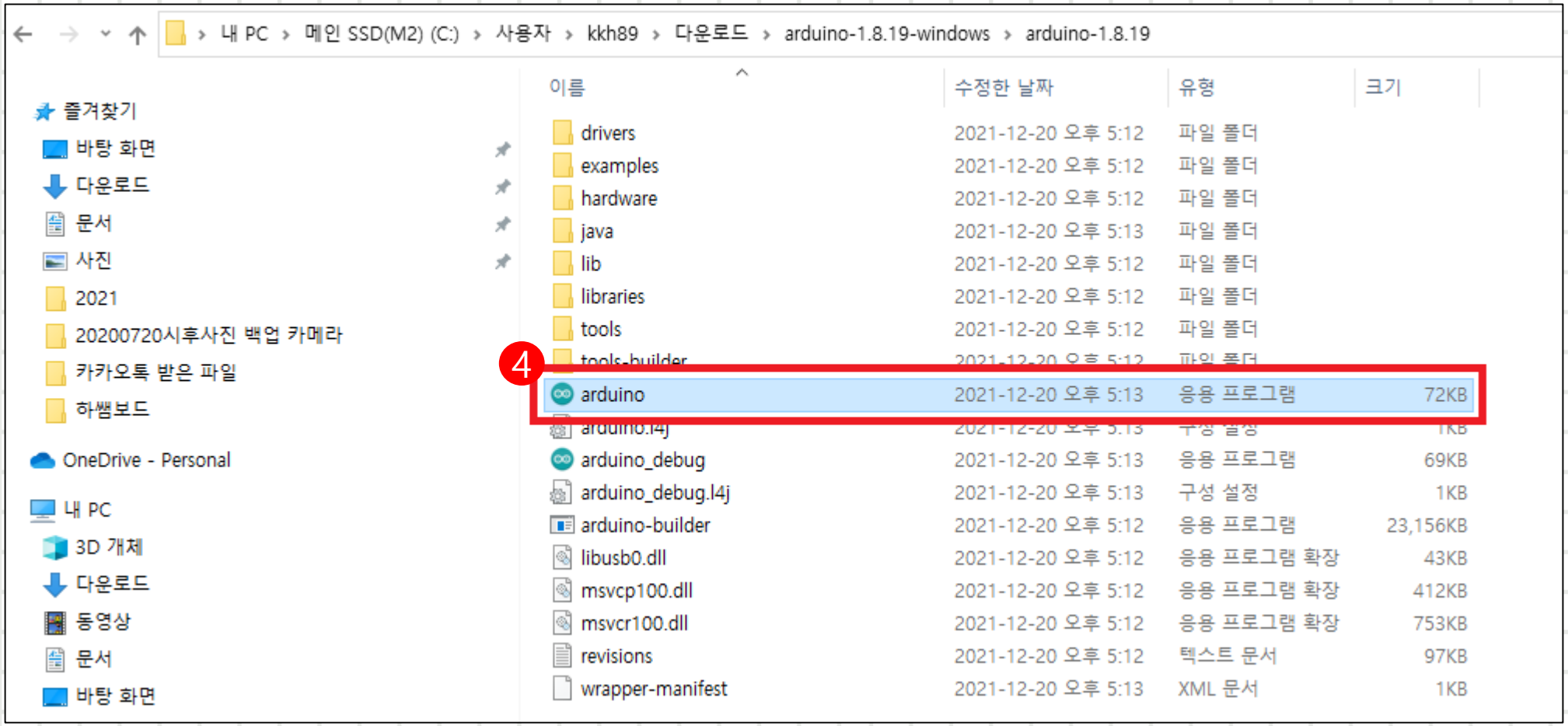


# 개발 툴 설치


HARDWARE
SOFTWARE
CLOUD
DOCUMENTATION
COMMUNITY
BLOG
ABOUT
Legacy IDE (1.8.X)
Arduino IDE 1.8.19
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.
SOURCE CODE
Active development of the Arduino software is hosted by GitHub. See the instructions for building the code. Latest release source code archives are available here. The archives are PGP-signed so they can be verified using this gpg key.
DOWNLOAD OPTIONS
Win 7 and newer
ZIP file
Windows app Win 8.1 or 10 Get
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits
Mac OS X 10.10 or newer
Release Notes
Checksums (sha512)
2


HARDWARE
SOFTWARE
CLOUD
DOCUMENTATION
COMMUNITY
BLOG
ABOUT
Support the Arduino IDE
Since the release 1.x release in March 2015, the Arduino IDE has been downloaded 65,509,391 times — impressive! Help its development with a donation.
$3
$5
$10
$25
$50
Other
3
JUST DOWNLOAD
CONTRIBUTE & DOWNLOAD

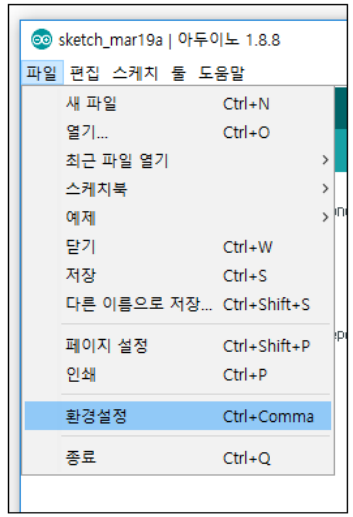
# 개발 툴 설치


내 PC › 메인 SSD(M2) (C:) › 사용자 › kkh89 › 다운로드 › arduino-1.8.19-windows › arduino-1.8.19
즐겨찾기
바탕 화면
다운로드
문서
사진
2021
20200720시후사진 백업 카메라
카카오톡 받은 파일
하쌤보드
OneDrive - Personal
내 PC
3D 개체
다운로드
동영상
문서
바탕 화면
이름
수정한 날짜
유형
크기
drivers 2021-12-20 오후 5:12 파일 폴더
examples 2021-12-20 오후 5:12 파일 폴더
hardware 2021-12-20 오후 5:12 파일 폴더
java 2021-12-20 오후 5:13 파일 폴더
lib 2021-12-20 오후 5:12 파일 폴더
libraries 2021-12-20 오후 5:12 파일 폴더
tools 2021-12-20 오후 5:12 파일 폴더
4
arduino 2021-12-20 오후 5:13 응용 프로그램 72KB
arduino_debug 2021-12-20 오후 5:13 응용 프로그램 69KB
arduino_debug.l4j 2021-12-20 오후 5:13 구성 설정 1KB
arduino-builder 2021-12-20 오후 5:12 응용 프로그램 23,156KB
libusb0.dll 2021-12-20 오후 5:12 응용 프로그램 확장 43KB
msvcp100.dll 2021-12-20 오후 5:12 응용 프로그램 확장 412KB
msvcr100.dll 2021-12-20 오후 5:12 응용 프로그램 확장 753KB
revisions 2021-12-20 오후 5:12 텍스트 문서 97KB
wrapper-manifest 2021-12-20 오후 5:13 XML 문서 1KB

# ESP32 사용을 위한 환경설정

- 아두이노가 아닌 ESP32를 사용하기 위한 환경설정이 필요

Cf. 크롬창에 아두이노 IDE에 ESP32 보드 추가하기로 검색 후 스크롤을 내려 아래 주소를 복사하여 사용하기




환경설정에서 '추가적인 보드 매니저 URLs' 항목에 주소를 입력해 줍니다.

주소가 여러 개라면 창 우측의 아이콘을 눌러 여러 개의 URL을 정리하여 입력할 수 있습니다.

> 아래 주소 입력창 하단 [비공식 보드 지원 URL 목록 ...] 부분을 눌러 지원 보드를 확인할 수 있습니다.

>> espressif 공식 ESP32 URL

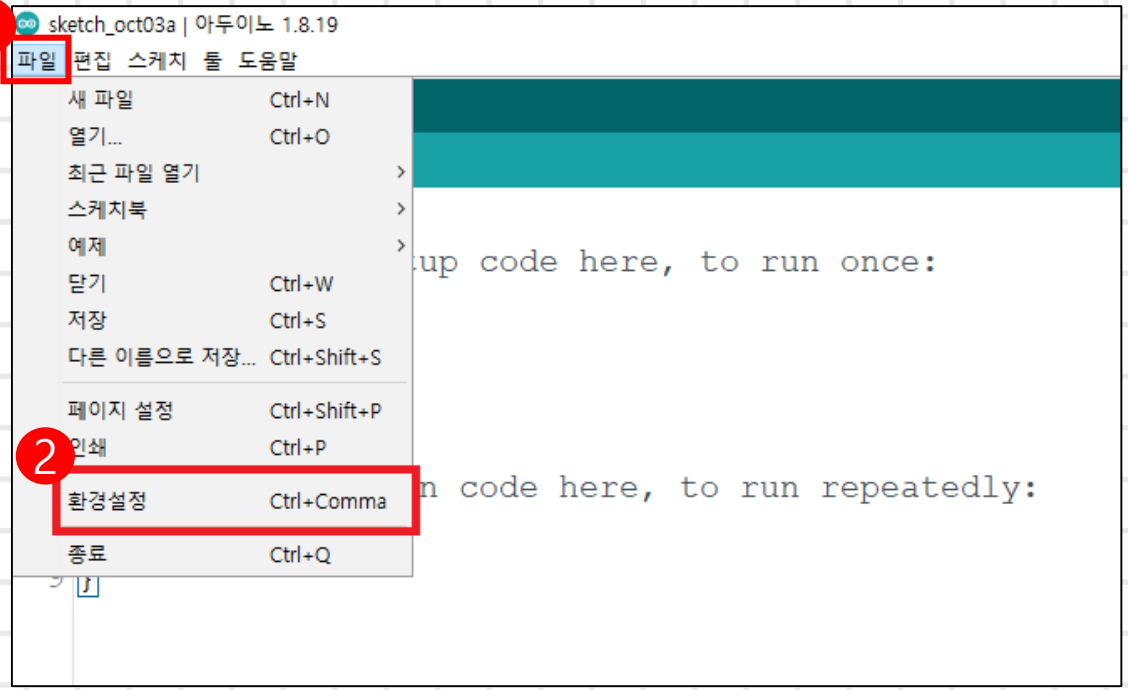
https://dl.espressif.com/dl/package_esp32_index.json

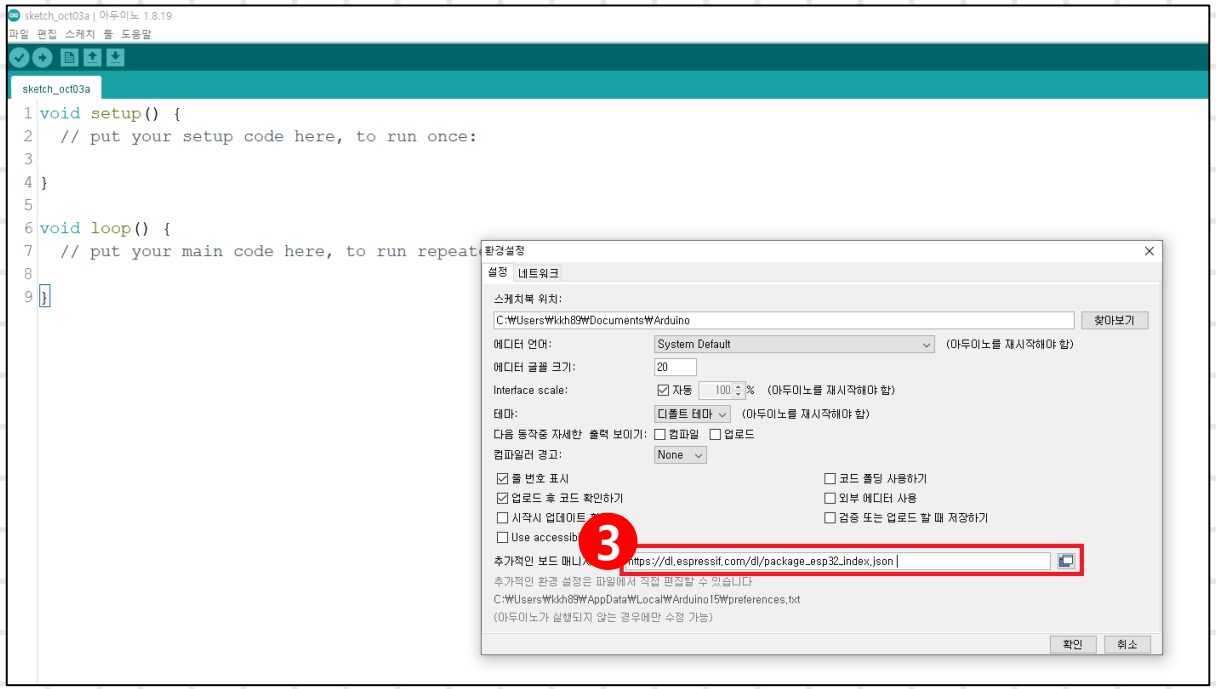
주소 복사

# ESP32 사용을 위한 환경설정

1) 파일 -> 환경설정에서 추가적인 보드 매니저 URLs에 주소를 입력

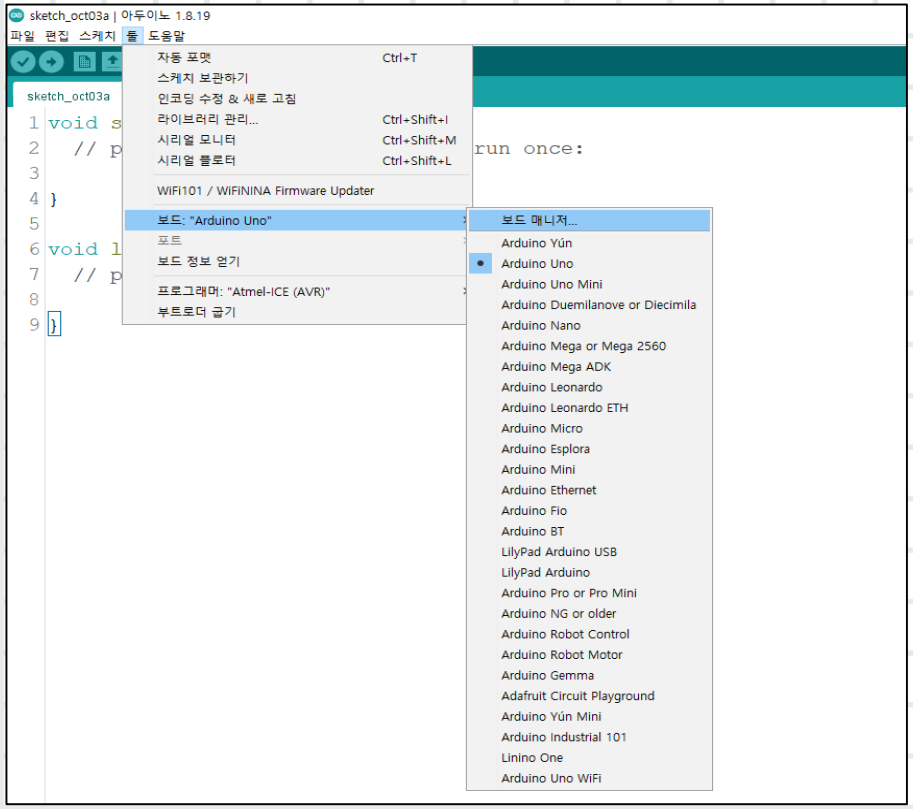
https://dl.espressif.com/dl/package_esp32_index.json



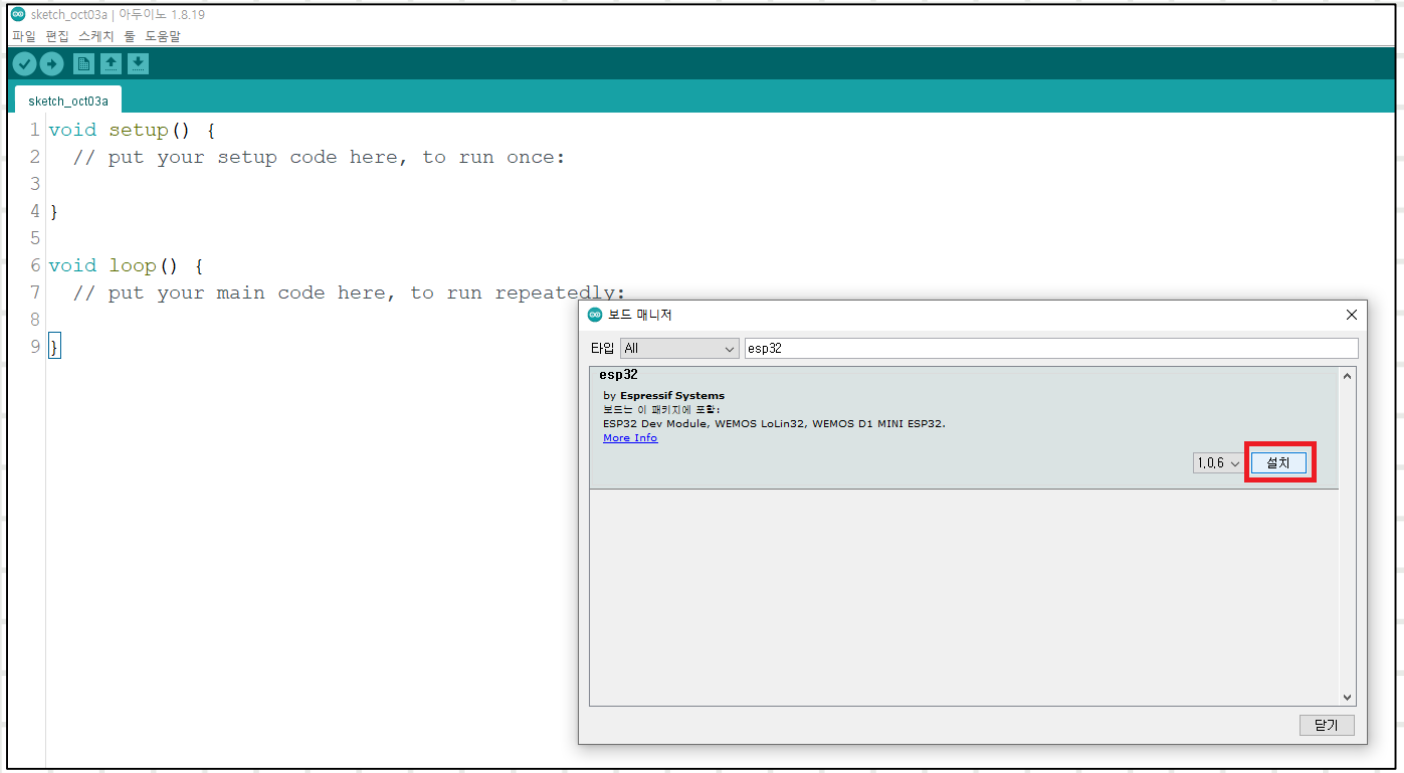
# ESP32 사용을 위한 환경설정

2) 툴 메뉴에서 보드>보드매니저를 선택



# ESP32 사용을 위한 환경설정

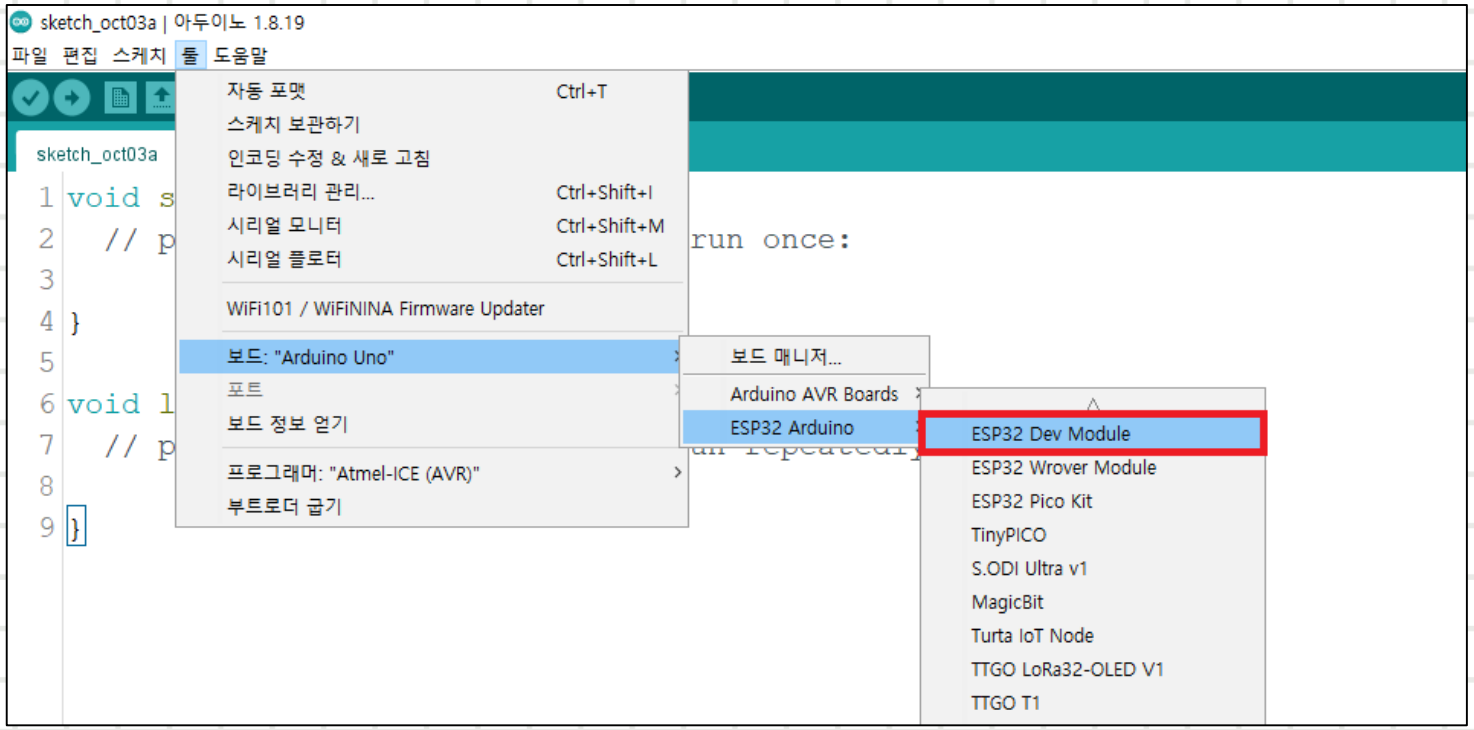
3) ESP32보드를 사용하기 위하여 **ESP32를 검색**하여 설치



# ESP32 사용을 위한 환경설정

4) 설치가 완료되면 다음과 같이 작업할 보드를 ESP32를 선택

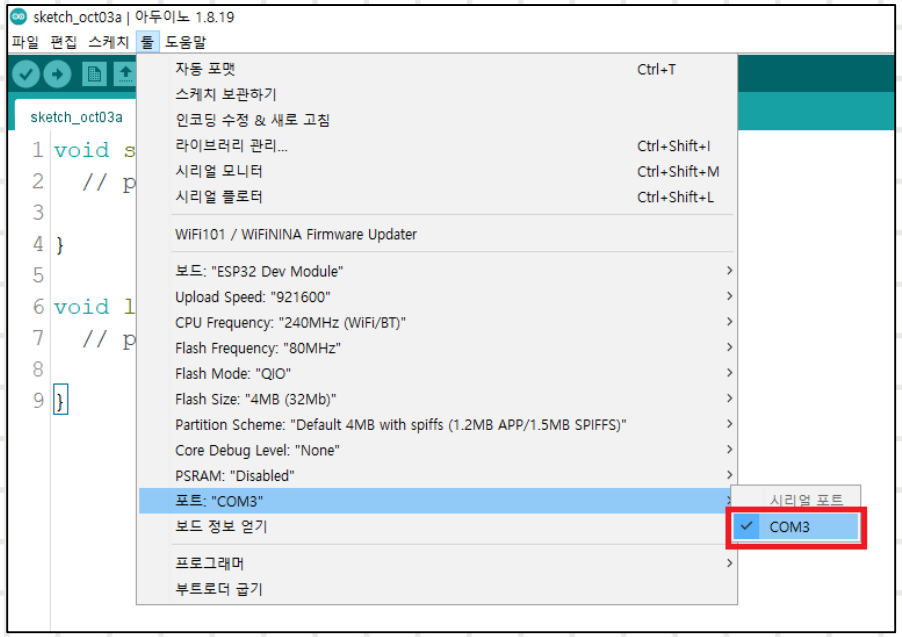
일반적으로 ESP32 Dev Module를 선택하기

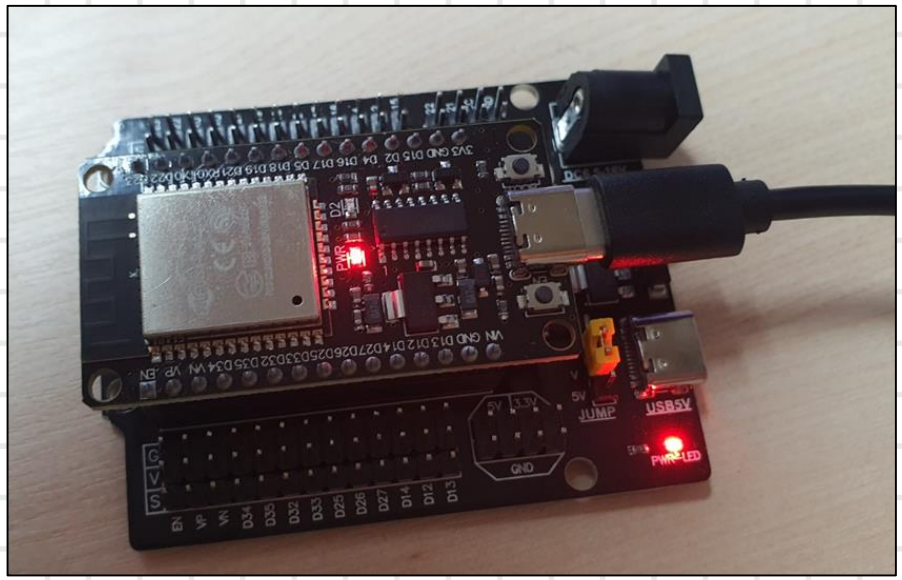


# ESP32 사용을 위한 환경설정

5) USB케이블로 ESP32를 연결하고 사용할 포트를 선택

-> ESP32를 USB케이블로 PC와 연결한 후 자신의 아두이노 보드 종류와 포트를 설정하기



# 프로그램 구조 및 설명

```
void setup() {
//이곳에 있는 코드는 한번만 실행
}
void loop() {
//이곳에 있는 코드는 반복해서 실행
}
```

## 주석

- 컴퓨터가 실행시키지 않는 문장
- 설명 시 이용
- // 한 줄일 때 사용
- /* ~~~*/ 여러 줄일 때 사용

# 프로그램 구조 및 설명

함수의 구조

```
반환값 함수이름 (매개변수)
{

실행문

}
```

## 함수

- 소스 코드에서 일정한 동작을 수행하는 코드
- 함수 호출 -> 매개변수(변수를 넘겨줌) -> 호출된 함수의 코드가 동작 -> 함수 종류 후 경우에 따라 반환 값을 반환

# 프로그램 구조 및 설명

```
void setup() {
//이곳에 있는 코드는 한번만 실행
}
void loop() {
//이곳에 있는 코드는 반복해서 실행
}
```

## setup()

- 전원이 켜졌을 때, 리셋 버튼을 눌렀을 때 한번만 실행
- 초기 변수, 핀 상태, 사용 라이브러리 시작 등을 지정

# 프로그램 구조 및 설명

```
void setup() {
//이곳에 있는 코드는 한번만 실행
}
void loop() {
//이곳에 있는 코드는 반복해서 실행
}
```

## loop()

- 전원이 켜있는 동안 계속 반복해서 동작

# ☆주석

☆ 미션

주석을 이용하여 아래 내용을 여러 번 실행하는 프로그래밍

-학번 : 10000

-이름 : 유00

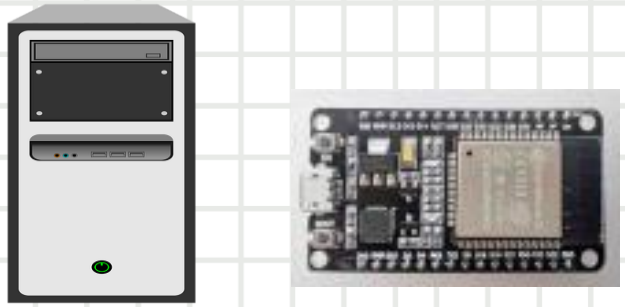
# 주석

```
// 주석을 이용하여 아래 내용을 여러 번 실행하는 프로그램
void setup() {
//이곳에 있는 코드는 한번만 실행
}
void loop() {
/*학번 : 10000
이름 : 유00*/
}
```

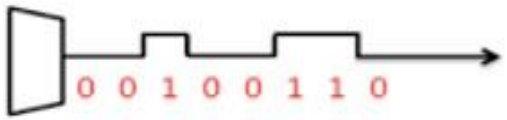
# 프로그램 구조 및 설명

| 기본 함수 | 설명 |
| --- | --- |
| pinMode(pin번호, 모드); | 핀의 상태를 입력 또는 출력으로 설정<br>- 모드 : INPUT, OUTPUT, INPUT_PULLUP<br>// INPUT_PULLUP(내부 풀업 저항 활성화) |
| digitalWrite(pin번호, 모드); | 지정된 핀에 디지털 HIGH 또는 LOW 값을 내보냄<br>- 모드 : HIGH(5V 출력 또는 내부 풀업 저항 사용), LOW(0V 출력) |
| digitalRead(pin번호); | 디지털 값을 읽음 |
| analogRead(pin번호); | 아날로그 값을 읽음 |
| analogWrite(pin번호, value); | 아날로그 값을 씀 |
| delay(value); | 프로그램을 지정된 밀리 초만큼 멈춤<br>1000은 1초를 의미 |

# 시리얼 통신(직렬 통신)



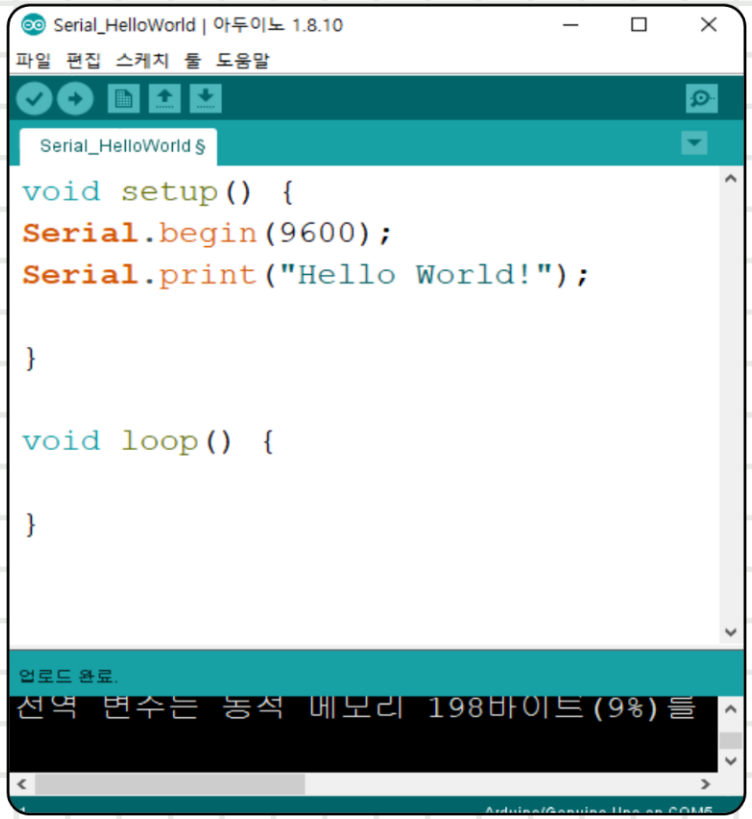
< ESP32와 시리얼 포트로 연결 >

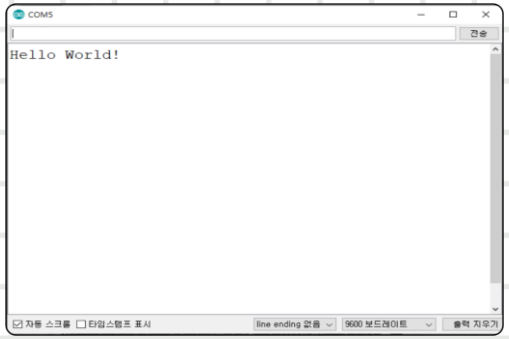


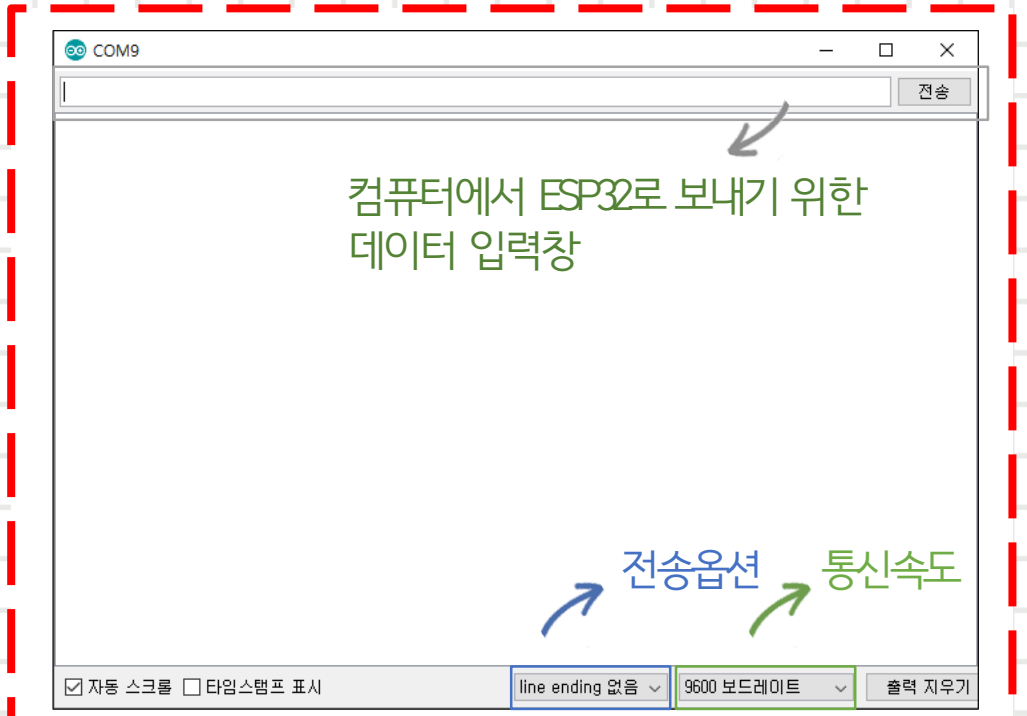

< 직렬 시리얼 통신의 예시 >

✓ ESP32는 컴퓨터와 USB 포트를 통해 시리얼 통신이라는 유선 통신 함.

# 시리얼 통신(직렬 통신)



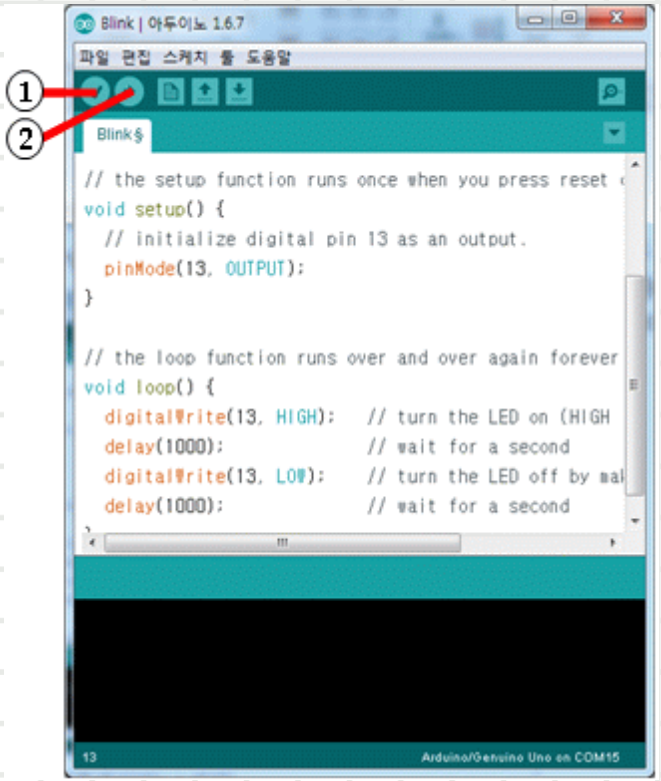




시리얼 통신 선언, 1초에 9600bit의 데이터를 주고받을 수 있는 통신 속도

# 시리얼 통신(직렬 통신)

## 컴파일과 업로드




### 컴파일

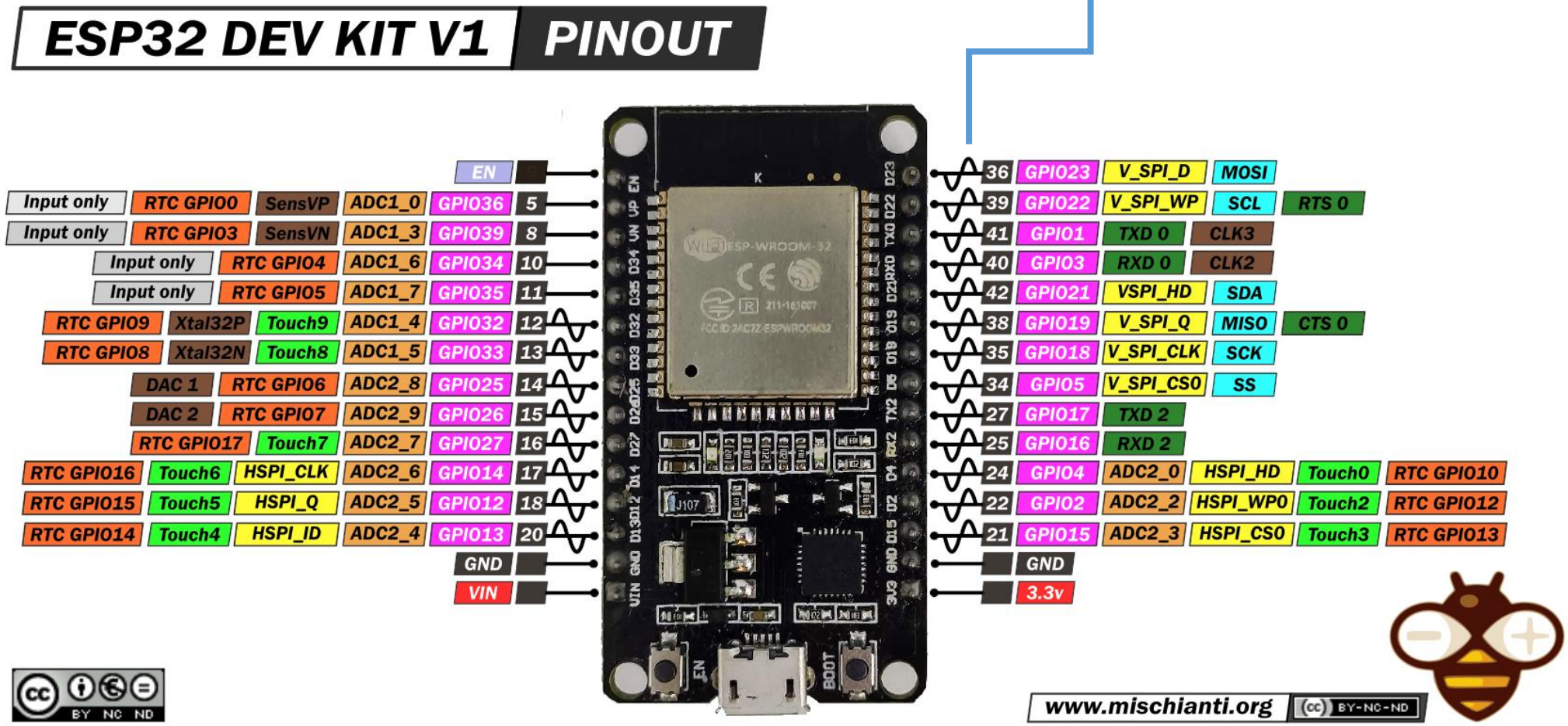
- 1번 클릭 혹은 ctrl + R
- 인간이 읽을 수 있는 소스코드를 컴퓨터가 이해할 수 있는 혹은 실행할 수 있는 언어로 변경하는 과정

### 업로드

- 1번 클릭 혹은 ctrl + U
- 컴파일하여 만들어진 내용을 ESP32에 업로드
- 업로드 실패 이유 : 보드/포트 설정 오류

# ESP32 핀 맵

PWM 방식 가능(아날로그처럼 이용)



# LED

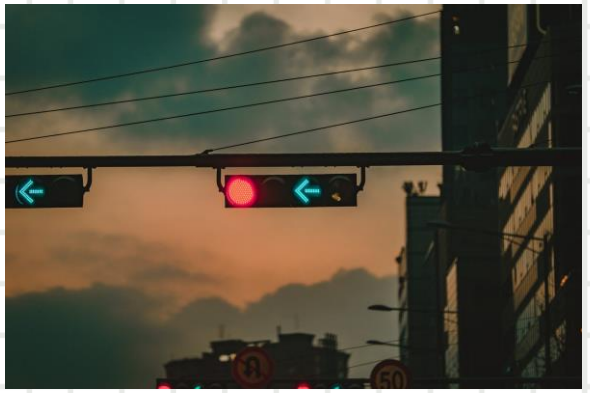
## LED의 역할

- 엑츄에이터
- 발광다이오드(Light Emitting Diode)
- 순방향으로 전압을 가했을 때 빛을 발생하는 반도체 부품
- 전기 에너지를 직접 빛 에너지로 변환(전기 낭비 X)

# LED

## LED의 사용


< 주차장 조명 >



< 신호등>

# 내부 LED

## 내부 LED

- 디지털 출력
- 디지털 2번 핀(연결된 내장 LED)
- 3.3V 보내면 LED 켜짐
- 0V 보내면 LED 꺼짐
- 테스트용으로 많이 사용

# 내부 LED

```
// 내부 LED를 1초 동안 켜고 1초 동안 끄기를 반복하는 내부 LED 프로그램
void setup() {
 pinMode(2, OUTPUT); //2번 핀을 출력 모드로 초기화
 }

 void loop() {
 digitalWrite(2, HIGH);
 //HIGH 대신 1 가능, LED 켜기, 3.3V 전기를 흘려 보냄
 delay(1000); //1초 기다리기
 digitalWrite(2, LOW);
 //LOW 대신 0 가능, LED 끄기, 0V 전기를 흘려 보냄
 delay(1000); //1초 기다리기
}
```

# ☆내부 LED

☆ 미션

2초 단위로 깜빡이는 내부 LED 프로그래밍

# 내부 LED

```
// 내부 LED를 2초 동안 켜고, 끄기를 반복하는 내부 LED 프로그램
void setup() {
 pinMode(2, OUTPUT); //2번 핀을 출력 모드로 초기화
 }

 void loop() {
 digitalWrite(2, HIGH);
 //HIGH 대신 1 가능, LED 켜기, 3.3V 전기를 흘려 보냄
 delay(2000); //2초 기다리기
 digitalWrite(2, LOW);
 //LOW 대신 0 가능, LED 끄기, 0V 전기를 흘려 보냄
 delay(2000); //2초 기다리기
}
```

# ☆내부 LED

☆ 미션

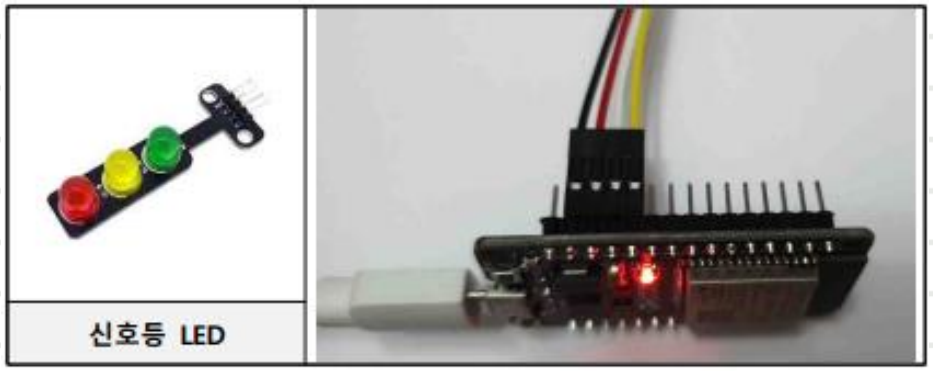
1번만 2초 단위로 깜빡이는 내부 LED 프로그래밍

-> setup() 함수 이용

# 내부 LED

```
 // 1번만 2초 동안 켜고, 끄는 내부 LED 프로그램
void setup() {
 pinMode(2, OUTPUT); //2번 핀을 출력 모드로 초기화
 digitalWrite(2, HIGH);
 //HIGH 대신 1 가능, LED 켜기, 3.3V 전기를 흘려 보냄
 delay(2000); //2초 기다리기
 digitalWrite(2, LOW);
 //LOW 대신 0 가능, LED 끄기, 0V 전기를 흘려 보냄
 delay(2000); //2초 기다리기
 }
 void loop() {
}
```

# 신호등 LED




## 신호등 LED

- 디지털 출력
- red, yellow, green, gnd의 4핀 모듈형 구조
- GPIO13, GPIO12, GPIO14, gnd 연결
- 3.3V 보내면 LED 켜짐
- 0V 보내면 LED 꺼짐

# 신호등 LED

```
// 빨간색 LED를 0.5초 동안 켜고 끄기를 반복하는 신호등 LED 프로그램
void setup() {
 pinMode(13, OUTPUT); //GPIO13핀을 출력모드로 사용함
 pinMode(12, OUTPUT); //GPIO12핀을 출력모드로 사용함
 pinMode(14, OUTPUT); //GPIO14핀을 출력모드로 사용함
}
void loop() {
 digitalWrite(13, HIGH); //red ON
 digitalWrite(12, LOW); //yellow OFF
 digitalWrite(14, LOW); //green OFF
 delay(500); //0.5초 대기
 digitalWrite(13, LOW); //red OFF
 digitalWrite(12, LOW); //yellow OFF
 digitalWrite(14, LOW); //green OFF
 delay(500); //0.5초 대기
}
```

# 신호등 LED

## 미션

빨간 불 -> 초록 불 -> 노란 불 순으로 0.5초 씩 켜고 꺼지는 신호등 LED 프로그래밍

# 신호등 LED

```
// 빨간 불 -> 초록 불 -> 노란 불 순으로 LED를 0.5초 씩 켜고 꺼지는 신호등 LED 프로그램
void setup() {
 pinMode(13, OUTPUT); //GPIO13핀을 출력모드로 사용함
 pinMode(12, OUTPUT); //GPIO12핀을 출력모드로 사용함
 pinMode(14, OUTPUT); //GPIO14핀을 출력모드로 사용함
}
void loop() {
 digitalWrite(13, HIGH); //red ON
 digitalWrite(12, LOW); //yellow OFF
 digitalWrite(14, LOW); //green OFF
 delay(500); //0.5초 대기
 digitalWrite(13, LOW); //red OFF
 digitalWrite(12, LOW); //yellow OFF
 digitalWrite(14, LOW); //green OFF
 delay(500); //0.5초 대기
 digitalWrite(13, LOW); //red OFF
 digitalWrite(12, HIGH); //yellow ON
 digitalWrite(14, LOW); //green OFF
 delay(500); //0.5초 대기
 digitalWrite(13, LOW); //red OFF
 digitalWrite(12, LOW); //yellow OFF
 digitalWrite(14, LOW); //green OFF
 delay(500); //0.5초 대기
 digitalWrite(13, LOW); //red OFF
 digitalWrite(12, HIGH); //yellow OFF
 digitalWrite(14, LOW); //green ON
 delay(500); //0.5초 대기
 digitalWrite(13, LOW); //red OFF
 digitalWrite(12, LOW); //yellow OFF
 digitalWrite(14, LOW); //green OFF
 delay(500); //0.5초 대기
}
```

# ☆신호등 LED

☆ 미션

GPIO15, 2, 4번으로 변경 후 1번만 2초 단위로 깜빡이는 내부 LED 프로그래밍

# 신호등 LED

Q. 여러 번 변경하지 않을 방법은 없을까?

-> 변수 사용

```
// 빨간 불 -> 초록 불 -> 노란 불 순으로 LED를 0.5초 씩 켜고 꺼지는 신호등 LED 프로그램
 void setup() {
pinMode(15, OUTPUT); //GPIO15핀을 출력모드로 사용함
pinMode(2, OUTPUT); //GPIO2핀을 출력모드로 사용함
pinMode(4, OUTPUT); //GPIO4핀을 출력모드로 사용함
}
void loop() {
 digitalWrite(15, HIGH); //red ON
 digitalWrite(2, LOW); //yellow OFF
 digitalWrite(4, LOW); //green OFF
 delay(500); //0.5초 대기
 digitalWrite(15, LOW); //red OFF
 digitalWrite(2, LOW); //yellow OFF
 digitalWrite(4, LOW); //green OFF
 delay(500); //0.5초 대기
```

```
 digitalWrite(15, LOW); //red OFF
 digitalWrite(2, HIGH); //yellow ON
 digitalWrite(4, LOW); //green OFF
 delay(500); //0.5초 대기
 digitalWrite(15, LOW); //red OFF
 digitalWrite(2, LOW); //yellow OFF
 digitalWrite(4, LOW); //green OFF
 delay(500); //0.5초 대기
 digitalWrite(15, LOW); //red OFF
 digitalWrite(2, LOW); //yellow OFF
 digitalWrite(4, HIGH); //green ON
 delay(500); //0.5초 대기
 digitalWrite(15, LOW); //red OFF
 digitalWrite(2, LOW); //yellow OFF
 digitalWrite(4, LOW); //green OFF
 delay(500); //0.5초 대기
}
```

# 변수

## 변수

- 변하는 수나 값을 저장하는 메모리 공간
- 같은 번호가 반복될 때 편리하게 사용 가능(한 번만 수정하면 됨)

# 신호등 LED

```
// 변수를 사용하여 빨간 불 -> 초록 불 -> 노란 불 순으로 LED를 0.5초 씩 켜고 꺼지는 신호등 LED 프로그램
int red = 15;
int yellow = 2;
int green = 4;
 void setup() {
 pinMode(red, OUTPUT); //GPIO15핀을 출력모드로 사용함
 pinMode(yellow, OUTPUT); //GPIO2핀을 출력모드로 사용함
 pinMode(green, OUTPUT); //GPIO4핀을 출력모드로 사용함
}
void loop() {
 digitalWrite(red, HIGH); //red ON
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, LOW); //green OFF
 delay(500); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, LOW); //green OFF
 delay(500); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, HIGH); //yellow ON
 digitalWrite(green, LOW); //green OFF
 delay(500); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, LOW); //green OFF
 delay(500); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, HIGH); //green ON
 delay(500); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, LOW); //green OFF
 delay(500); //0.5초 대기
}
```

# ☆내부 LED

☆ 미션

time 변수를 사용하여 빨간 불 -> 초록 불 -> 노란 불 순으로 LED를 0.5초 씩 켜고 꺼지는 신호등 LED 프로그램

# 신호등 LED

// time 변수를 사용하여 빨간 불 -> 초록 불 -> 노란 불 순으로 LED를 0.5초 씩 켜고 꺼지는 신호등 LED 프로그램

```
int red = 15;
int yellow = 2;
int green = 4;
int time = 500;
 void setup() {
 pinMode(red, OUTPUT); //GPIO15핀을 출력모드로 사용함
 pinMode(yellow, OUTPUT); //GPIO2핀을 출력모드로 사용함
 pinMode(green, OUTPUT); //GPIO4핀을 출력모드로 사용함
}
void loop() {
 digitalWrite(red, HIGH); //red ON
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, LOW); //green OFF
 delay(time); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, LOW); //green OFF
 delay(time); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, HIGH); //yellow ON
 digitalWrite(green, LOW); //green OFF
 delay(time); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, LOW); //green OFF
 delay(time); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, HIGH); //green ON
 delay(time); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, LOW); //green OFF
 delay(time); //0.5초 대기
}
```

# 신호등 LED

Q. 빛의 밝기를 조절할 순 없을까?

-> 3W LED 사용

// time 변수를 사용하여 빨간 불 -> 초록 불 -> 노란 불 순으로 LED를 0.5초 씩 켜고 꺼지는 신호등 LED프로그램

```
int red = 15;
int yellow = 2;
int green = 4;
int time = 500;
 void setup() {
 pinMode(red, OUTPUT); //GPIO15핀을 출력모드로 사용함
 pinMode(yellow, OUTPUT); //GPIO2핀을 출력모드로 사용함
 pinMode(green, OUTPUT); //GPIO4핀을 출력모드로 사용함
}
void loop() {
 digitalWrite(red, HIGH); //red ON
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, LOW); //green OFF
 delay(time); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, LOW); //green OFF
 delay(time); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, HIGH); //yellow ON
 digitalWrite(green, LOW); //green OFF
 delay(time); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, LOW); //green OFF
 delay(time); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, HIGH); //green ON
 delay(time); //0.5초 대기
 digitalWrite(red, LOW); //red OFF
 digitalWrite(yellow, LOW); //yellow OFF
 digitalWrite(green, LOW); //green OFF
 delay(time); //0.5초 대기
}
```

# PWM 방식

## PWM(Pulse Width Modulation)

- 펄스 폭을 조절해 전류를 조정
- 디지털 출력을 이용해 아날로그 출력 효과를 냄
- 0V~3.3V 사이의 주파수를 만듦
- analogWrite(pin번호, value); 형태로 사용

# PWM 방식

## PWM(Pulse Width Modulation)

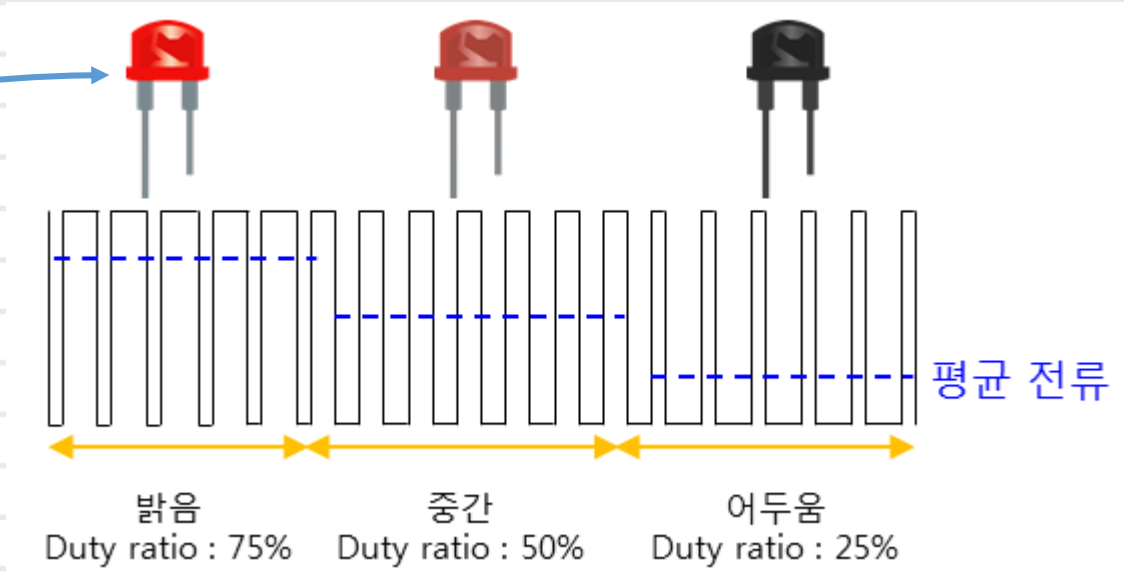
- 주파수[HZ] : 1초에 진동하는 횟수
- Duty ratio[%] : 한 주기 안에서 신호가 on되어 있는 비율

=> 즉, PWM은 Duty ratio를 조절한다는 의미

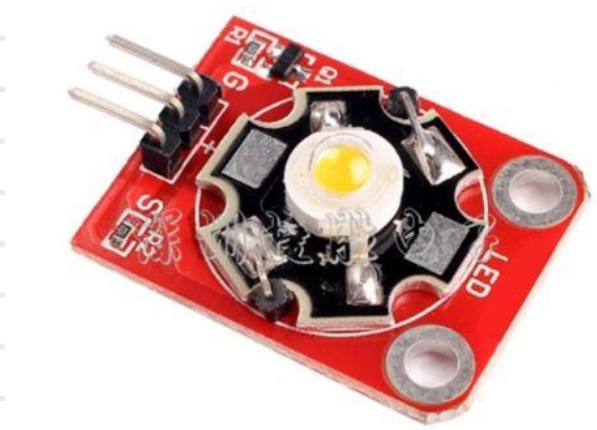
# PWM 방식

## PWM(Pulse Width Modulation)

HIGH 구간이 김



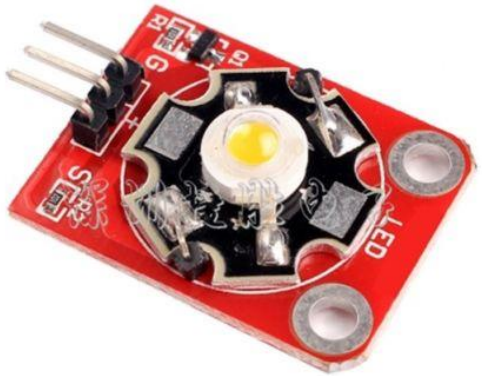
# 3W LED




## 3W LED

- 디지털 출력
- 색온도: 6000~7000K
- 전류: 700~750mA
- 공급전압: 3.3V ~ 5V
- 밝은 불빛을 내는 전구(육안 위험)

# 3W LED




<**3W LED** 연결>

1. 검정선은 GND선으로 전자회로에서는 GND를 음극(-)
2. 가운데 +는 VIN(5V) 또는 3.3V에 연결
3. S는 Signal로 전원을 켜거나, 끄는 등의 신호가 연결되는 것으로 GPIO13핀에 연결

# 3W LED

```
// 0.5초 씩 켜고 꺼지는 3W LED프로그램
void setup() {
ledcSetup(0, 5000, 8);              //0번 채널에 5000Hz로 8비트로 세팅
ledcAttachPin(13, 0);               //GPIO13핀을 0채널로 지정
}
void loop() {
ledcWrite(0, 255);                  //0채널에 255(최대값) 밝기로 PWM 출력
delay(500);                         //밝기가 0.5초 유지
ledcWrite(0, 30);                   //0채널에 30 밝기로 PWM 출력
delay(500);                         //밝기가 0.5초 유지
}
```

# 내부 LED

## 미션

0.5초 씩 켜고 꺼지는 3W LED프로그램을 이용하여 값을 바꿔가며 빛의 밝기 확인해보기

# 3W LED

```
// 0.5초 씩 켜고 꺼지는 3W LED 프로그램을 활용하여 빛의 밝기 확인해보기
void setup() {
ledcSetup(0, 5000, 8);                  //0번 채널에 5000Hz로 8비트로 세팅
ledcAttachPin(13, 0);                   //GPIO13핀을 0채널로 지정
}
void loop() {
ledcWrite(0, 255);                      //0채널에 255(최대값) 밝기로 PWM 출력
delay(500);                             //밝기가 0.5초 유지
ledcWrite(0, 30);                       //0채널에 30 밝기로 PWM 출력
delay(500);                             //밝기가 0.5초 유지
}
```

# 3W LED

## 미션

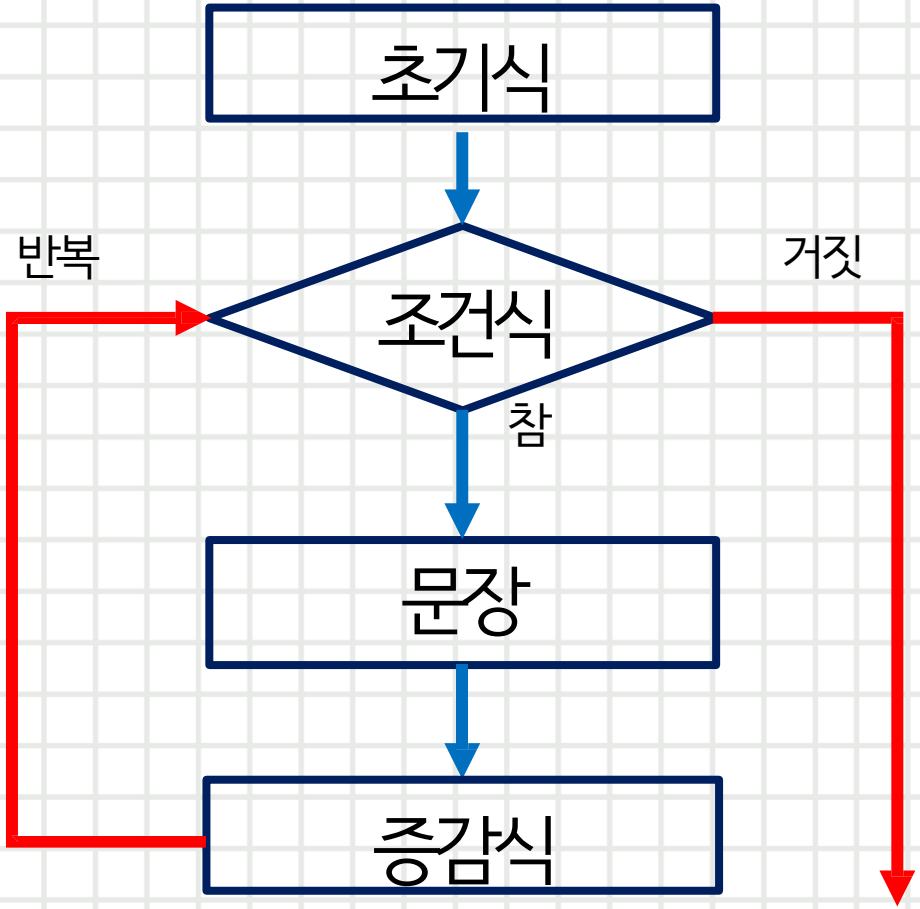
0.5초 씩 빛의 세기가 점점 밝아지는 3W LED 프로그래밍

# 3W LED

Q. 코드를 줄일 순 없을까?
-> 반복문 필요

```
// 0.5초 씩 빛의 세기가 점점 밝아지는 3W LED 프로그램
void setup() {
ledcSetup(0, 5000, 8);              //0번 채널에 5000Hz로 8비트로 세팅
ledcAttachPin(13, 0);               //GPIO13핀을 0채널로 지정
}
void loop() {
ledcWrite(0, 0);                  //0채널에 255(최대값) 밝기로 PWM 출력
delay(500);                         //255밝기가 0.5초 유지
ledcWrite(0, 1);                  //0채널에 254 밝기로 PWM 출력
delay(500);                         //밝기가 0.5초 유지
ledcWrite(0, 2);                  //0채널에 254 밝기로 PWM 출력
delay(500);                         //밝기가 0.5초 유지
...
}
```

# 반복문(for)




for의 기본문장 구조

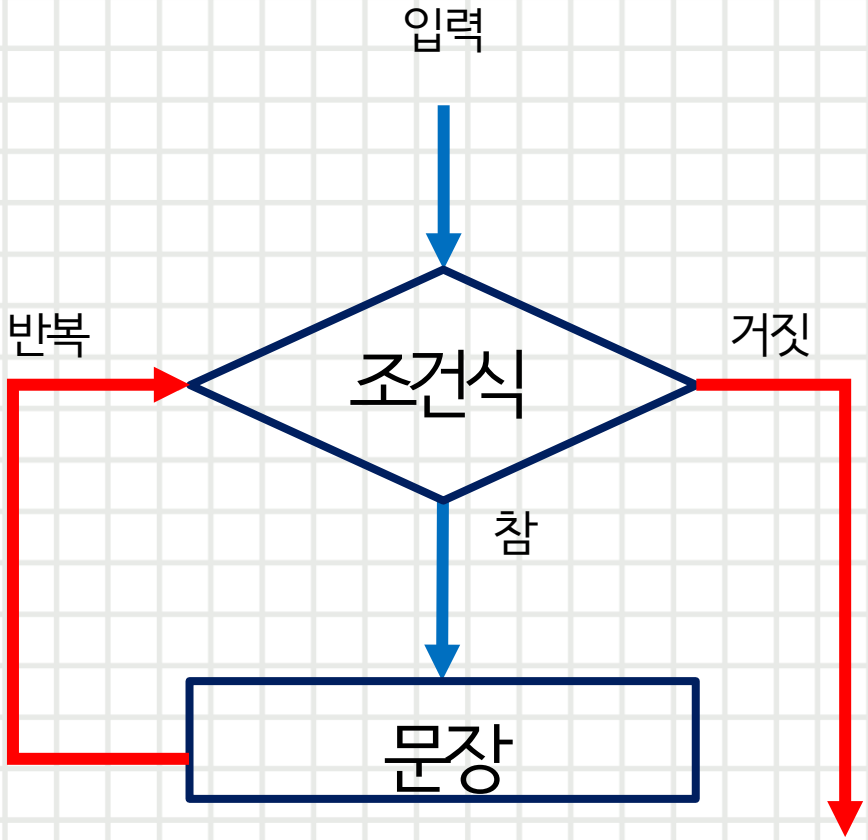
```
for(초기식; 조건식; 증감식)
{
  문장
}
```

# 반복문(for)

## For문을 사용한 경우(예시)

```
 /*for문을 사용하여 조건식이 10미만일 때, 0 ~ 9까지만  출력하여 시리얼 모니터로 출력 값 확인*/
void setup() {
  Serial.begin(9600);
  for(int i=0; i<10; i++){
    Serial.print("for i : ");
    Serial.print(i);
    Serial.println(", Hello World");
  }
}

void loop() {

}
```

# 반복문(while)




while의 기본 문장구조

```
while(조건식)
{
   문장
}
```

# 반복문(while)

## while문을 사용한 경우(예시)

```
  /*w h i l e 문을사용하여 조건식이 10미만일 때, 0 ~ 9까지만 출력하여시리얼모니터로출력값확인*/
void setup() {
  Serial.begin(9600);
  int i=0;
  while(i<10){
    Serial.print("while i : ");
    Serial.print(i);
    Serial.println(", Hello World");
    i++;
  }
}

void loop() {

}
```

# 3W LED

```
// 반복문을 활용한 0.5초 씩 빛의 세기가 점점 밝아지는 3W LED 프로그램
void setup() {
ledcSetup(0, 5000, 8);
ledcAttachPin(13, 0);
}
void loop() {
for(int w=0 ; w<256 ; w++) {    //for()는 주어진 횟수만큼 반복
    ledcWrite(0, w);            //13핀에 밝기가 증가하며 PWM출력
    delay(5);
    }
}
```

# 3W LED

## 미션

빛의 세기가 점점 어두워지는 3W LED 프로그래밍

# 3W LED

```
// 반복문을 활용한 0.5초 씩 빛의 세기가 점점 어두워지는 3W LED 프로그램
void setup() {
ledcSetup(0, 5000, 8);
ledcAttachPin(13, 0);
}
void loop() {
for(int w=0 ; w<256 ; w++) {     //for()는 주어진 횟수만큼 반복
   ledcWrite(0, 255-w);          //13핀에 밝기가 감소하며 PWM 출력
   delay(5);
   }
}
```

# 3W LED(PWM 출력)

미션. While문으로 바꿔보기
-> 반복문

```
void setup() {
ledcSetup(0, 5000, 8);
ledcAttachPin(13, 0);
}
void loop() {
for(int w=0 ; w<256 ; w++) {    //for()는 주어진 횟수만큼 반복
    ledcWrite(0, w);            //13핀에 밝기가 증가하며 PWM출력
    delay(5);
    }
for(int w=0 ; w<256 ; w++) {     //for()는 주어진 횟수만큼 반복
   ledcWrite(0, 255-w);         //13핀에 밝기가 감소하며 PWM 출력
   delay(5);
   }
}
```

# 3W LED

☆ 미션

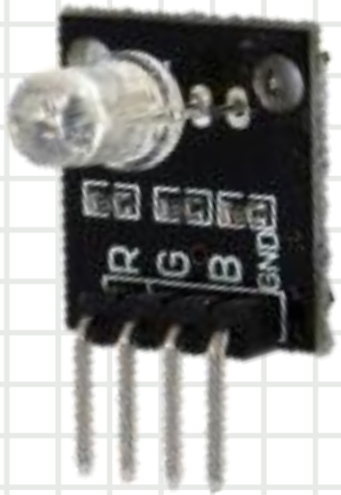
while()반복문을 활용하여 빛의 세기가 점점 밝아지다가 어두워지는 3W LED 프로그래밍

# 3W LED(PWM 출력)

Q. 다른 색을 나타낼 순 없을까?
-> 3색 LED

```
// while()반복문을 활용하여 빛의 세기가 점점 밝아지다가 어두워지는 3W LED 프로그래밍
void setup() {
ledcSetup(0, 5000, 8);
ledcAttachPin(13, 0);
}
void loop() {
int w=0;
while(w<256) {    //for()는 주어진 횟수만큼 반복
    ledcWrite(0, w);        //13핀에 밝기가 증가하며 PWM출력
    delay(5);
    w++;
    }
w=0;
while(w<256) {     //for()는 주어진 횟수만큼 반복
  ledcWrite(0, 255-w);  //13핀에 밝기가 감소하며 PWM 출력
  delay(5);
  w++;
  }
}
```

# 3색 LED




## 3색 LED

- 디지털 출력
- RGB LED라고 불림
- LED 부품 안에 3가지 색을 내는 LED가 들어 있음
- 각각 색을 따로 제어하여 다양한 색을 출력

# 3색 LED

## 3색 LED의 사용

< 분수대 조명 >

< 무대 조명 >

# 3색 LED


## 빛의 삼원소

- 빛을 합성하여 다양한 색을 표현할 수 있음.

# 3색 LED

우리의 방식

**캐소드(Cathode)** 공통 핀(Common)이 GND로 묶여 있는 형태의 회로방식

LED 점등 신호

```
digitalWrite(Red,   HIGH);
digitalWrite(Green, LOW);
digitalWrite(Blue,  LOW);
```

빨강

초록

파랑

붉은색 LED 점등 시 붉은색 연결 핀에 **켜기(HIGH)**신호를 보내야 점등이 됨.

**애노드(Anode)** 공통 핀(Common)이 VCC로 묶여 있는 형태의 회로방식

LED 점등 신호

```
digitalWrite(Red,   LOW);
digitalWrite(Green, HIGH);
digitalWrite(Blue,  HIGH);
```

빨강

초록

파랑

붉은색 LED 점등 시 붉은색 연결 핀에 **끄기(LOW)**신호를 보내야 점등이 됨.

# 3W LED

## 미션

파랑색, 자홍색 3색 LED 프로그래밍

# 3색 LED

```
// 파랑색, 자홍색 3색 LED 프로그램
int red = 15;
int green = 2;
int blue = 4;
int time = 1000;
void setup() {
 pinMode(red, OUTPUT); //GPIO15핀을 출력모드로 사용함
 pinMode(green, OUTPUT); //GPIO2핀을 출력모드로 사용함
 pinMode(blue, OUTPUT); //GPIO4핀을 출력모드로 사용함
}
void loop() {
 // 파랑색
 digitalWrite(red, LOW);
 digitalWrite(green, LOW);
 digitalWrite(blue, HIGH );
 delay(time); //0.5초 대기
// 자홍색
 digitalWrite(red, HIGH);
 digitalWrite(green, LOW);
 digitalWrite(blue, HIGH );
}
```

Q. 그 외의 색은 나타낼 수 없을까?
-> PWM 방식

# 3색 LED

우리의 방식

**캐소드(Cathode)** 공통 핀(Common)이 GND로 묶여 있는 형태의 회로방식

**애노드(Anode)** 공통 핀(Common)이 VCC로 묶여 있는 형태의 회로방식

LED 점등 PWM

```
analogWrite(Red,   0);
analogWrite(Green, 0);
analogWrite(Blue,  255);
```

빨강
초록
파랑

LED 점등 PWM

```
analogWrite(Red,   255);
analogWrite(Green, 255);
analogWrite(Blue,  0);
```

빨강
초록
파랑

# 3색 LED

☆ 미션

네이버 -> rgb색상표 검색

무지개 색깔의 3색 LED 프로그래밍

# 3색 LED


rgb 색상표
통합
이미지
VIEW
지식iN
인플루언서
동영상
쇼핑
뉴스
어학사전
지도
색상 팔레트
#FF5E00
조회
색상 코드
RGB R 255 G 94 B 0
HSB H 22 ° S 100 % B 100 %
CMYK C 0 % M 63 % Y 100% K 0 %
* 직접입력 또는 색상팔레트에서 색을 선택하세요.
조회
색상 및 코드값은 그래픽 도구별로 다를 수 있습니다.

# 3색 LED

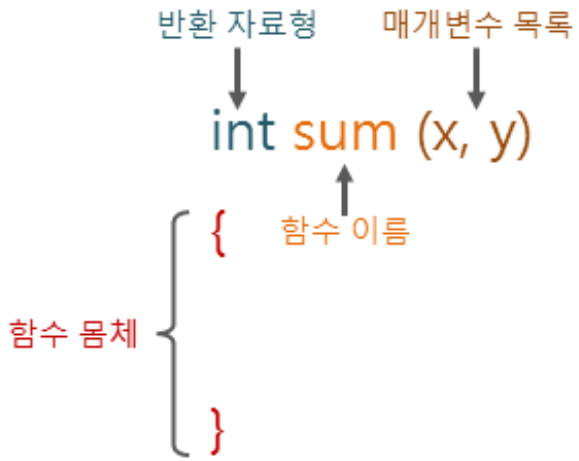
```
// 무지개 색깔의 3색 LED 프로그램
int red = 15;
int yellow = 2;
int green = 4;
int time = 1000;
void setup() {
 pinMode(red, OUTPUT); //GPIO15핀을 출력모드로 사용함
 pinMode(yellow, OUTPUT); //GPIO2핀을 출력모드로 사용함
 pinMode(green, OUTPUT); //GPIO4핀을 출력모드로 사용함
}
void loop() {
 analogWrite(redPin, 255);
 analogWrite(greenPin, 0);
 analogWrite(bluePin, 0);     //red
 delay(time); //0.5초 대기
 ...
}
```

Q. 반복되는 코드를 줄일 수 있는 방법은 없을까?
-> 함수

# 3색 LED

## 함수

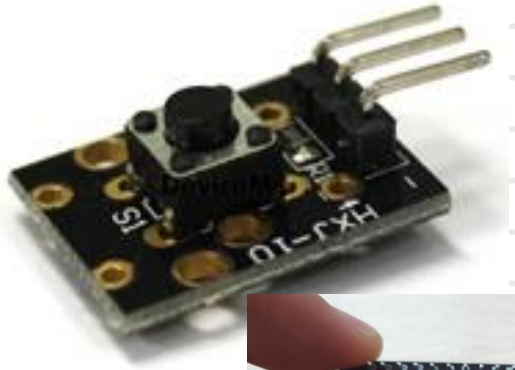
- 하나의 특별한 목적의 작업을 수행하기 위해 독립적으로 설계된 프로그램 코드의 집합

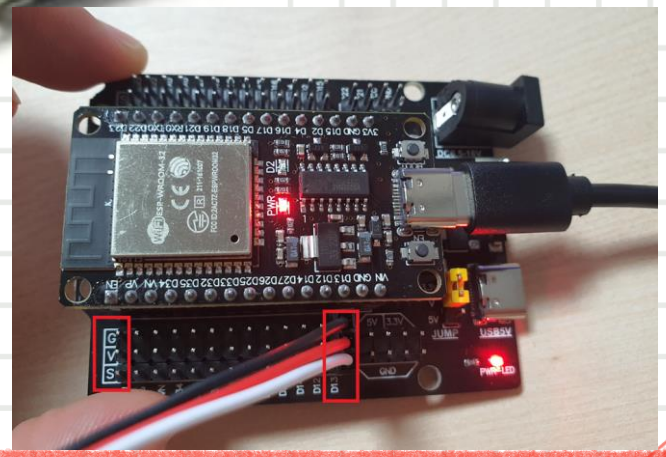


# 3색 LED

```
// 함수를 활용한 무지개 색깔의 3색 LED 프로그래밍
int redPin = 15;
int greenPin = 2;
int bluePin = 4;
int time = 1000;
void setup() {
 pinMode(redPin, OUTPUT); //GPIO15핀을 출력모드로 사용함
 pinMode(greenPin, OUTPUT); //GPIO2핀을 출력모드로 사용함
 pinMode(bluePin, OUTPUT); //GPIO4핀을 출력모드로 사용함
}
void loop() {
setColor(255, 0, 0); // red
delay(1000);
setColor(0, 255, 0); // green
delay(1000);
setColor(0, 0, 255); // blue
delay(1000);
 setColor(255, 255, 0); // yellow
 delay(1000);
 setColor(80, 0, 80); // purple
 delay(1000);
 setColor(0, 255, 255); // aqua
 delay(1000);
 }

void setColor(int red, int green, int blue)
{
 analogWrite(redPin, red);
 analogWrite(greenPin, green);
 analogWrite(bluePin, blue);
}
```

# 버튼





## 버튼

- 디지털 입력 센서
- 택트 스위치라고도 불림
- 동작 시 딸각 거리는 스위치
- 모듈형

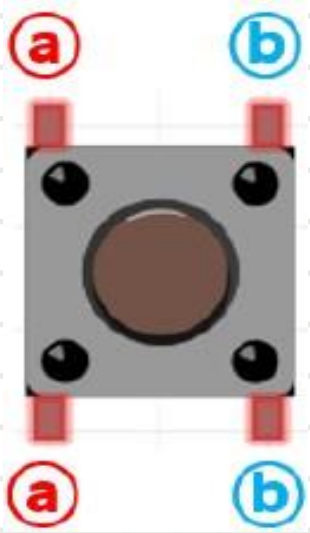
# 버튼

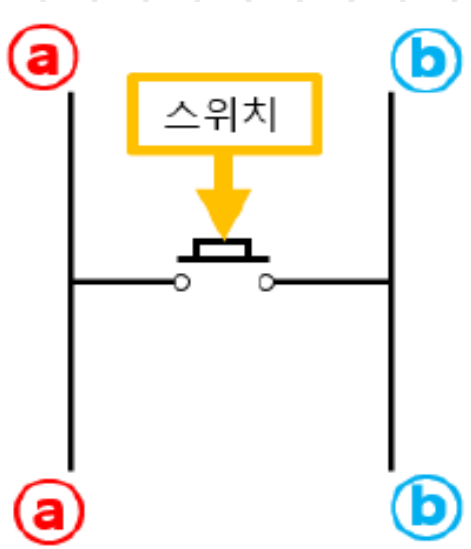
## 버튼의 역할

- 마우스
- 전자제품 전원

# 버튼

택트 스위치 외관




택트 스위치 내부 결선도

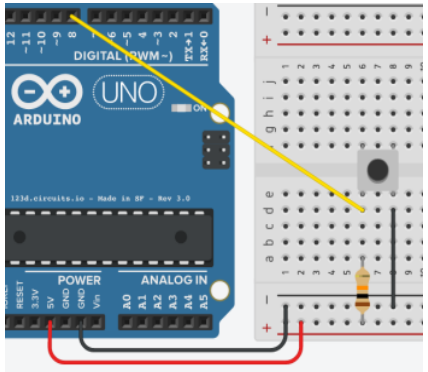
✓ 택트 스위치는 단순한 버튼식 스위치로 버튼을 눌렀을 때 (a) 와 반대편 (b)가 연결됨

# 버튼

## 버튼의 종류

☆ 풀업 저항

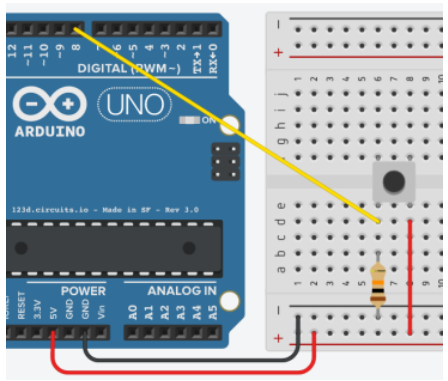
- 버튼 누르지 않음 -> HIGH, 누름 -> LOW




저항 : vcc(5V)
입출력핀
나머지 : GND

☆ 풀다운 저항

- 버튼 누르지 않음 -> LOW, 누름 -> LOW




저항 : GND
입출력핀
나머지 : vcc(5V)

# 버튼

## 풀업, 풀다운 방식 테스트

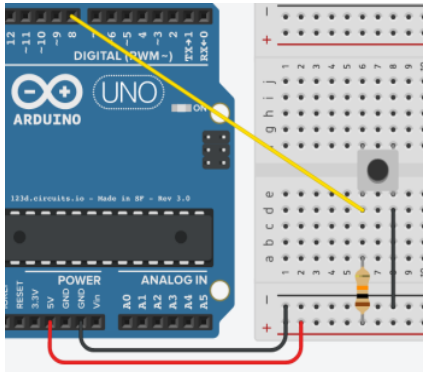
```
void setup() {
Serial.begin(9600); //아두이노-PC 통신시작(9600 속도)
pinMode(13, INPUT);              //버튼연결 GPIO13핀을 입력모드로 사용
}
void loop() {
int button = digitalRead(13);              //13핀의 값을 읽어서 변수에 넣음
Serial.println(button);                    //센서의 값(변수)을 PC로 보냄
delay(100);                                //0.1초 대기(0.1초에 1회 센서 읽음)
}
```

# 버튼

## 버튼의 종류

☆ 풀업 저항

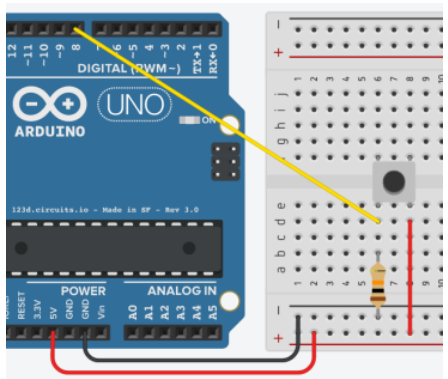
- 버튼 누르지 않음 -> HIGH, 누름 -> LOW



저항 : vcc(5V)
입출력핀
나머지 : GND

☆ 풀다운 저항

- 버튼 누르지 않음 -> LOW, 누름 -> LOW

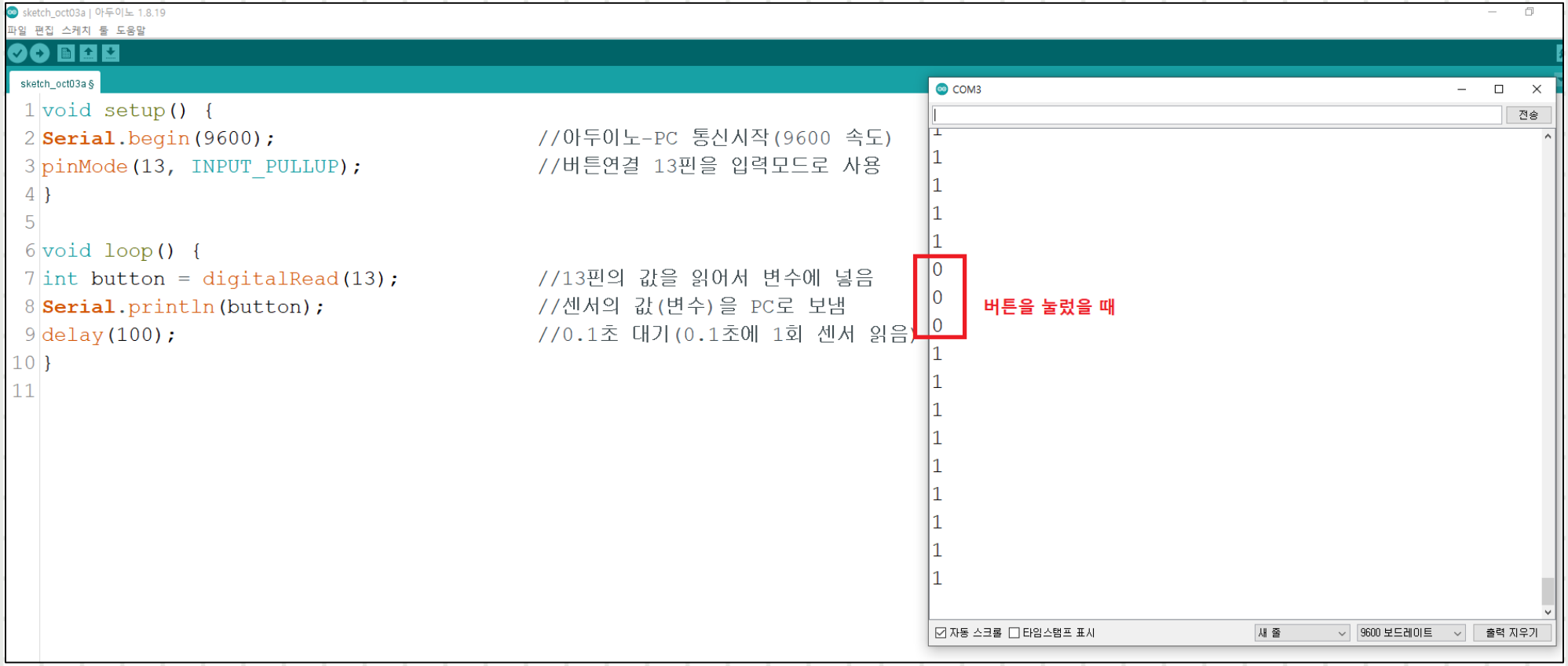


저항 : GND
입출력핀
나머지 : vcc(5V)

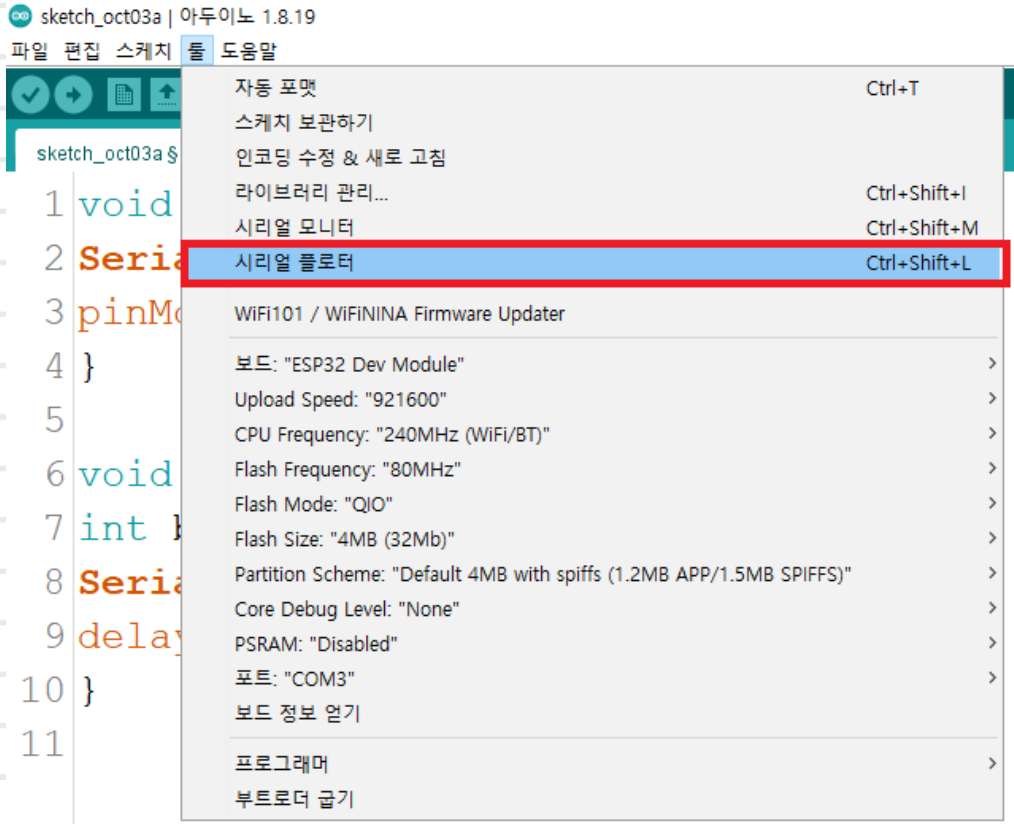
# 버튼

```
void setup() {
Serial.begin(9600);                    //아두이노-PC 통신시작(9600 속도)
pinMode(13, INPUT_PULLUP);             //버튼연결 GPIO13핀을 입력모드로 사용
}
void loop() {
int button = digitalRead(13);          //13핀의 값을 읽어서 변수에 넣음
Serial.println(button);                //센서의 값(변수)을 PC로 보냄
delay(100);                            //0.1초 대기(0.1초에 1회 센서 읽음)
}
```

# 버튼


sketch_oct03a | 아두이노 1.8.19
파일 편집 스케치 툴 도움말
sketch_oct03a §
1 void setup() {
2 Serial.begin(9600); //아두이노-PC 통신시작(9600 속도)
3 pinMode(13, INPUT_PULLUP); //버튼연결 13핀을 입력모드로 사용
4 }
5
6 void loop() {
7 int button = digitalRead(13); //13핀의 값을 읽어서 변수에 넣음
8 Serial.println(button); //센서의 값(변수)을 PC로 보냄
9 delay(100); //0.1초 대기(0.1초에 1회 센서 읽음)
10 }
11
COM3
전송
1
1
1
1
1
0
0
0
1
1
1
1
1
1
1
1
1
1
버튼을 눌렀을 때
자동 스크롤
타임스탬프 표시
새 줄
9600 보드레이트
출력 지우기

# 버튼




```
//아두이노-PC 통신시작(9600 속도)
//버튼연결 13핀을 입력모드로 사용



//13핀의 값을 읽어서 변수에 넣음
//센서의 값(변수)을 PC로 보냄
//0.1초 대기(0.1초에 1회 센서 읽음)
```

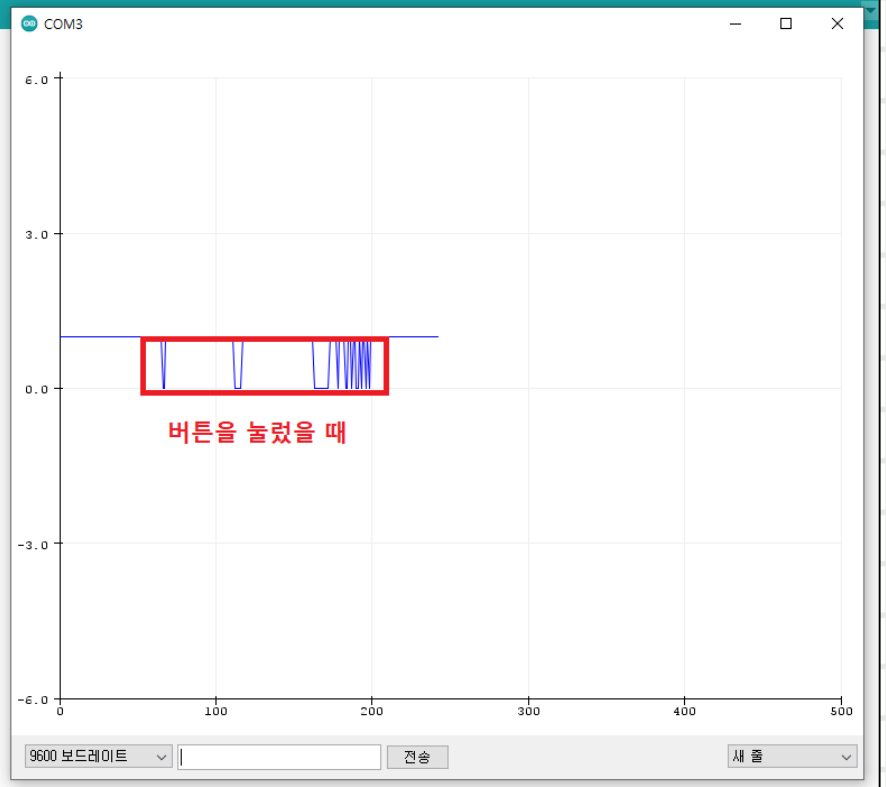
# 버튼

Q. 만약 버튼을 눌렀을 때 빛이 나오도록 할 수 없을까?
-> 조건문

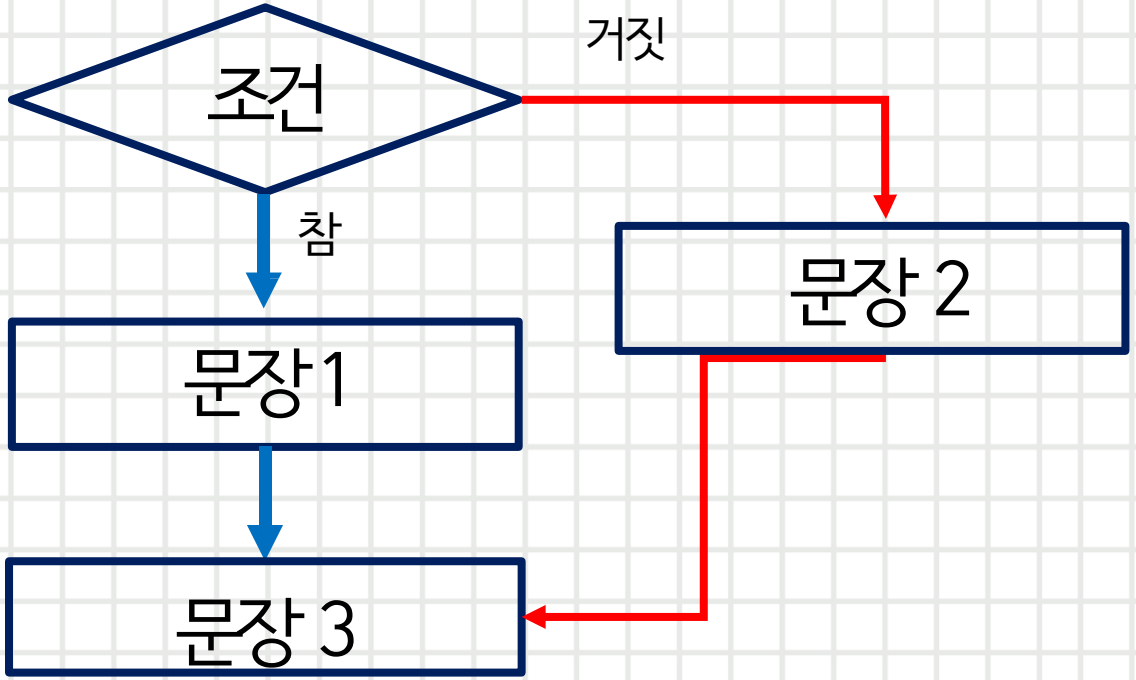
sketch_oct03a | 아두이노 1.8.19
파일 편집 스케치 툴 도움말

sketch_oct03a §

```
1  void setup() {
2  Serial.begin(9600);                  //아두이노-PC 통신시작(9600 속도)
3  pinMode(13, INPUT_PULLUP);           //버튼연결 13핀을 입력모드로 사용
4  }
5
6  void loop() {
7  int button = digitalRead(13);        //13핀의 값을 읽어서 변수에 넣음
8  Serial.println(button);              //센서의 값(변수)을 PC로 보냄
9  delay(100);                          //0.1초 대기(0.1초에 1회 센서 읽음)
10 }
11
```



# 조건문(if)



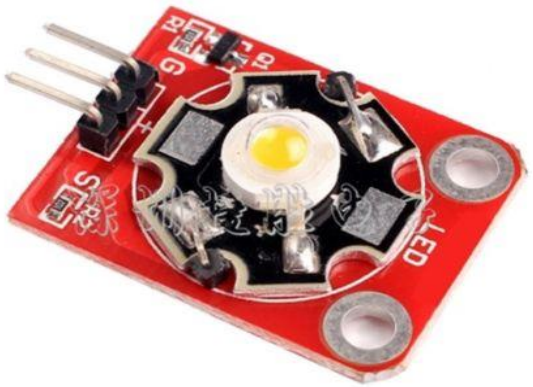

If-else의 기본문장 구조

```
if(조건식)
{
   문장1
}
else
{
   문장 2
}
문장 3
```

# 조건문(if)

```
/* if 문을 사용하여 조건 a==b (참 = 1) 이면 " Hello World!"  출력, a==b (거짓=0) 이면 " Hello!" 출력*/
int a=49;
int b=0;
void setup() {
  Serial.begin(9600);
}

void loop() {
 if(Serial.available()){
   b=Serial.read();
   Serial.println(b);
   if(a==b){
     Serial.println("hello, world");
   }
   else{
     Serial.println("hello");
   }
 }
}
```

# 3W LED + 버튼




# 3W LED + 버튼

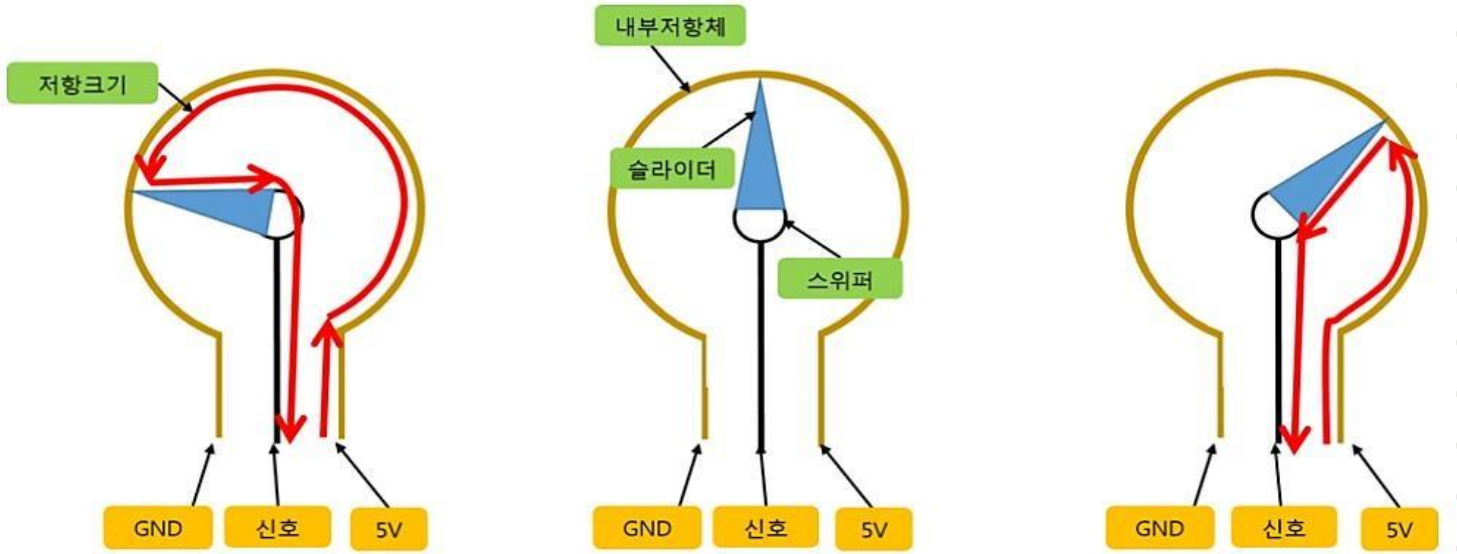
```
// 조건문을 활용한 3W LED, 버튼 프로그램
void setup() {
   pinMode(13, INPUT_PULLUP);              //13핀 버튼
   pinMode(12,OUTPUT);                     //12번 핀 3W LED
}
void loop() {
   int button = digitalRead(13);
   if(button==1){
       digitalWrite(12,0);                 // 버튼을 안 누르면 꺼진다.
   }else{
     digitalWrite(12,1);                   // 버튼을 누르면 켜진다.
      }
}
```

# 가변저항


## 가변저항

- 아날로그 입력 센서
- 포텐셔미터, 볼륨이라고 불림
- 저항 값의 변화가 조절 가능한 저항기
- 일반 저항 소자와 달리 3개의 핀(VCC, VOUT, GND)으로 구성
- 슬라이더를 회전 시켜서 저항 값을 조절

# 가변저항




- ✓ 내부 저항체를 통해 전기가 흐르며 저항의 기능을 하게 되는 원리
- ✓ GND 핀과 3.3V 핀은 각각 가변 저항의 내부 저항체와 연결됨.
- ✓ 스위퍼를 회전 시키면 가변 저항 내부 슬라이더가 움직여 저항 값 조절

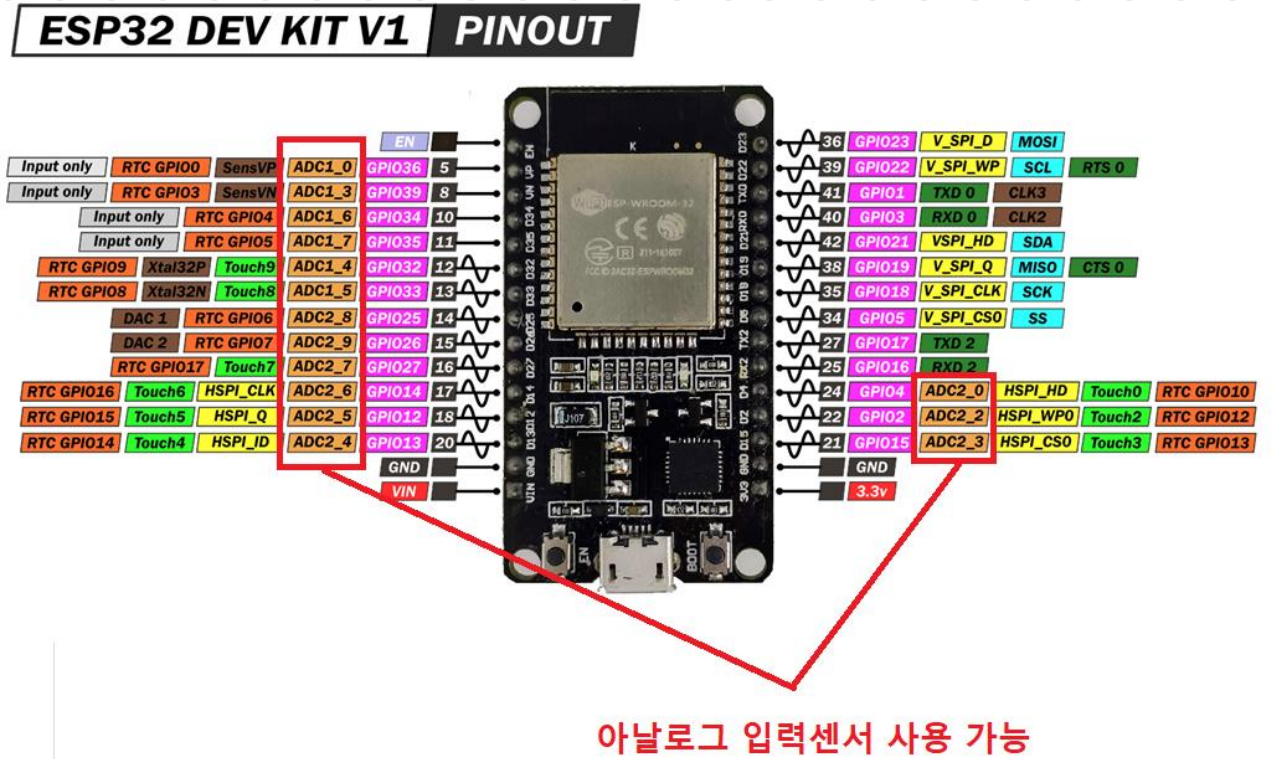
**Tip!**

VCC와 GND 바꿔 연결 가능
→ 스위퍼의 회전 방향과 저항 값의 증감 방향이 바뀌게 됩니다.

# 가변저항

## 가변저항의 사용

# 가변저항

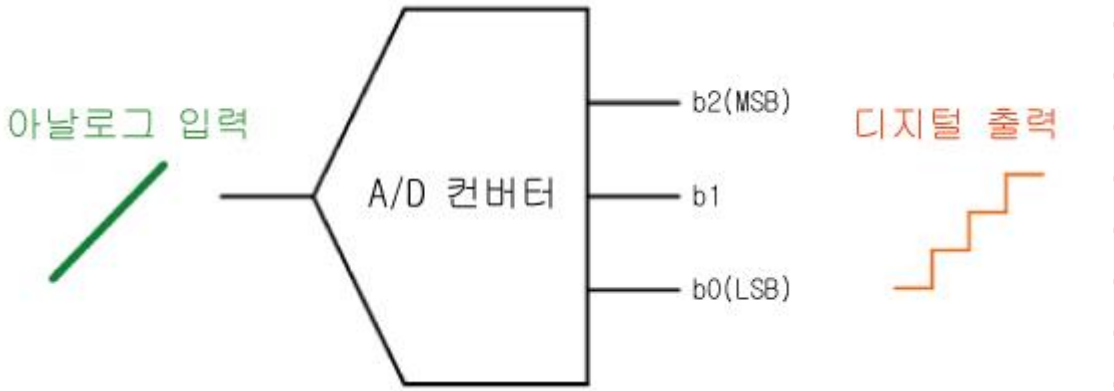



- ESP32에서 아날로그 입력은 0V~3.3V의 값을 구분
- **3.3V를 12비트 단계로 구분하여 값을 인식**
- 핀맵 기준으로 ADC핀에서 가능(거의 대부분에서 아날로그 센서 신호를 인식 가능)

# 가변저항

<ADC핀>

- 아날로그 신호의 진폭을 이산적인 주기로 추출하여, 부호로 표시된 디지털 신호로 변환



# 가변저항

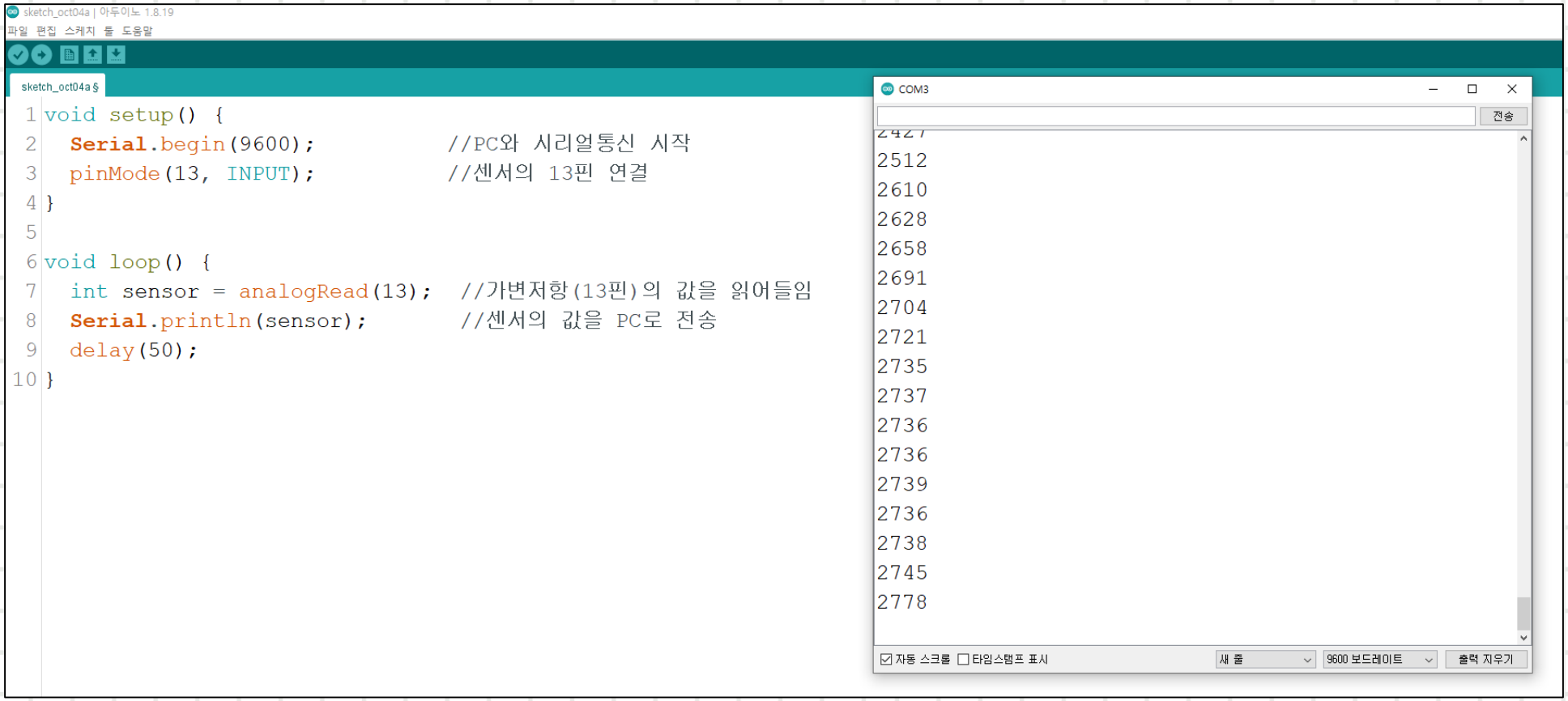
```
void setup() {
Serial.begin(9600);                    //PC와 시리얼통신 시작
pinMode(13, INPUT);                    //센서의 13핀 연결
}
void loop() {
int sensor = analogRead(13);           //가변저항(13핀)의 값을 읽어 들임
Serial.println(sensor);                //센서의 값을 PC로 전송
delay(50);
}
```

# ☆가변저항

☆ 미션

가변저항의 최댓값, 최솟값을 확인해보기

# 가변저항


sketch_oct04a | 아두이노 1.8.19
파일 편집 스케치 툴 도움말
sketch_oct04a §
1 void setup() {
2 Serial.begin(9600); //PC와 시리얼통신 시작
3 pinMode(13, INPUT); //센서의 13핀 연결
4 }
5
6 void loop() {
7 int sensor = analogRead(13); //가변저항(13핀)의 값을 읽어들임
8 Serial.println(sensor); //센서의 값을 PC로 전송
9 delay(50);
10 }
COM3
전송
2512
2610
2628
2658
2691
2704
2721
2735
2737
2736
2736
2739
2736
2738
2745
2778
자동 스크롤
타임스탬프 표시
새 줄
9600 보드레이트
출력 지우기

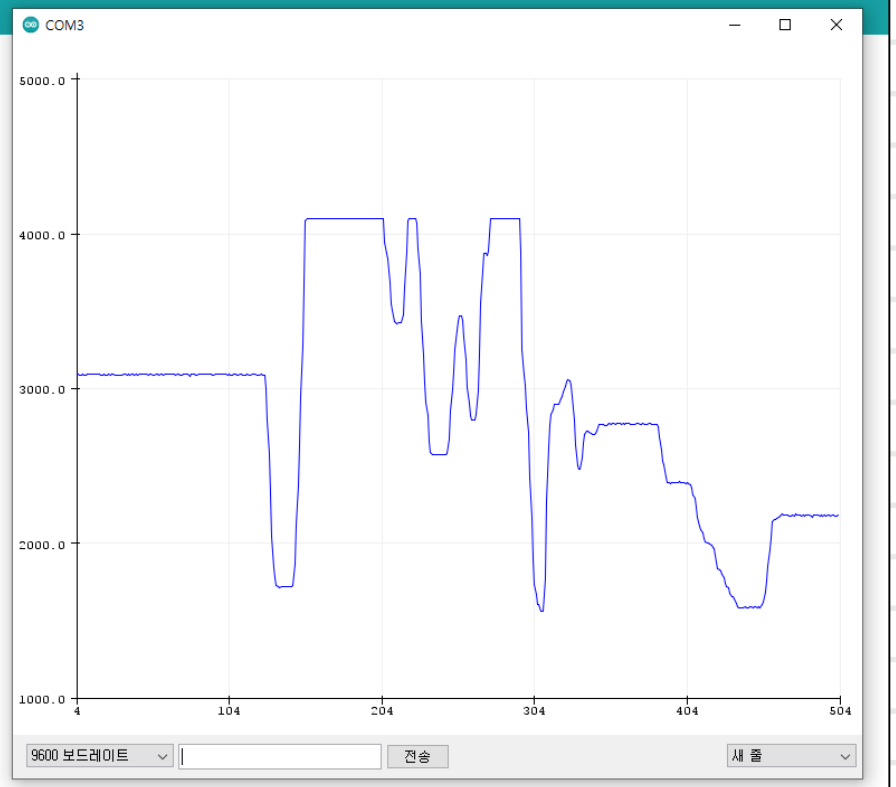
# 가변저항

sketch_oct04a | 아두이노 1.8.19

파일 편집 스케치 툴 도움말

sketch_oct04a §

```
1  void setup() {
2    Serial.begin(9600);           //PC와 시리얼통신 시작
3    pinMode(13, INPUT);           //센서의 13핀 연결
4  }
5
6  void loop() {
7    int sensor = analogRead(13);  //가변저항(13핀)의 값을 읽어들임
8    Serial.println(sensor);       //센서의 값을 PC로 전송
9    delay(50);
10 }
```



# 가변저항 + 3W LED


# 가변저항 + 3W LED

Q. Led가 꺼지고 켜지기를 반복함
→ map() 함수 필요

```
void setup() {
   ledcSetup(0, 5000, 8);
   ledcAttachPin(12, 0);
   pinMode(13, INPUT);                    //센서의 13핀 연결
}
void loop() {
   int sensor = analogRead(13);          //가변저항(13)의 값을 읽어 들임
   ledcWrite(0, sensor);
   delay(50);
}
```

# 가변저항

```
void loop()  {
   int val = analogRead(Analog_pin);
   val = map(val, 0, 4095 , 0, 255);
   analogWrite(Led_pin, val);
}
```

int val = analogRead(Analog_pin);

변수 val에 가변 저항 값을 저장

val = map(val, 0, 4095, 0, 255);

변환 된 값 변환할 수 현재 범위의 하한 값 현재 범위의 상한 값 목표 범위의 하한 값 목표 범위의 상한 값

저항에서 들어오는 값(0 ~ 1023)을
LED 밝기 값 (0 ~ 255)으로 변환

analogWrite(Led_pin, val);

변환된 값을 LED로 출력

# 가변저항 + 3W LED

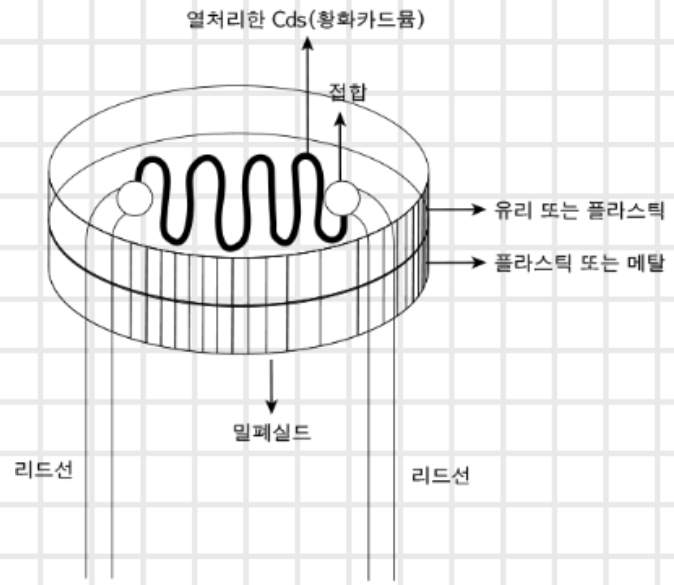
```
void setup() {
   ledcSetup(0, 5000, 8);
   ledcAttachPin(12, 0);
   pinMode(13, INPUT);                    //센서의 13핀 연결
}
void loop() {
   int sensor = analogRead(13);          //가변저항(13)의 값을 읽어 들임
   sensor = map(sensor, 0, 4095, 0, 255);
   ledcWrite(0, sensor);
   delay(50);
}
```

# 조도 센서

## 조도 센서

- 아날로그 센서
- 빛 센서, 광 센서라고도 불림
- 측정된 빛의 세기에 따라 저항 값이 변하는 센서
- 빛이 강하면 저항 값이 작아지고, 빛이 약하면 저항 값이 커짐

# 조도 센서




<조도 센서 구조도>

- ✓ 광에너지(빛)를 받으면 내부에 움직이는 전자가 발생하여 전도율이 변함.
- ✓ 황화카드뮴(Cds) 소자 사용
- ✓ 빛을 강하게 받으면 조도 센서는 저항 값이 작아져 전류가 많이 흐름.
- ✓ 빛을 약하게 받으면 저항 값이 커져 전류를 방해하게 됨.

# 조도 센서

## 조도 센서의 사용

# 조도 센서

```
void setup() {
Serial.begin(9600);                    //PC와 시리얼통신 시작
pinMode(13, INPUT_PULLUP);             //센서의 13핀 연결
}
void loop() {
int sensor = analogRead(13);           //조도센서(13핀)의 값을 읽어
들임
Serial.println(sensor);                //센서의 값을 PC로 전송
delay(50);
}
```

# 조도 센서

☆ 미션

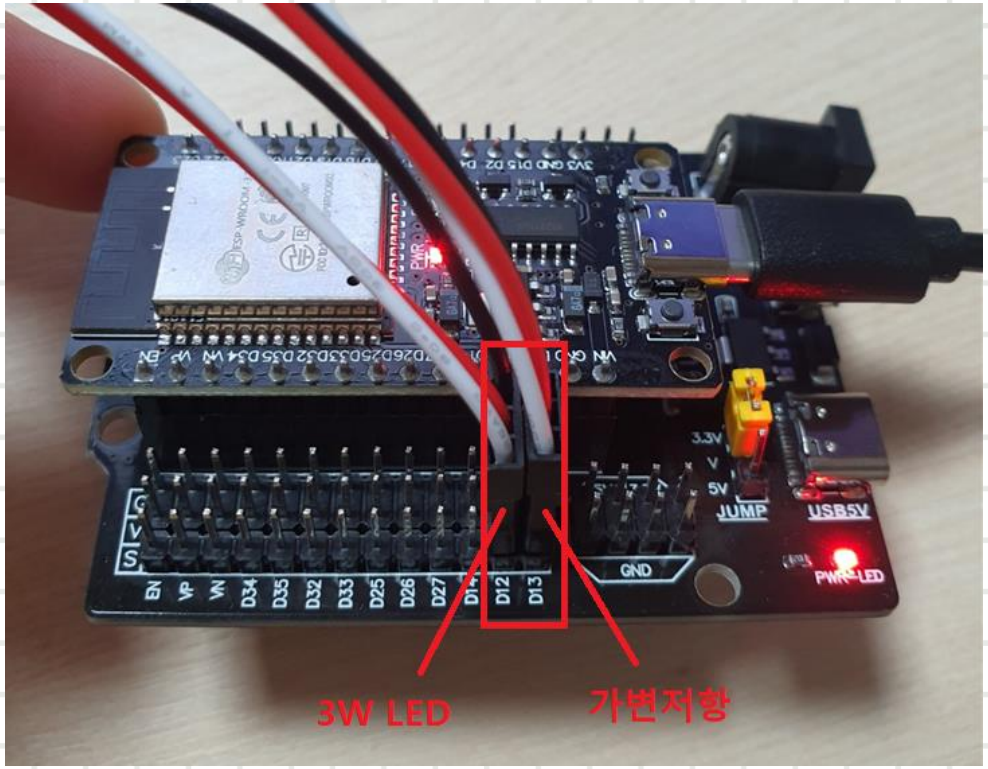
조건문을 사용하여 조도 센서 값에 따라 어두워지면 led 끄고 밝으면 켜는 프로그래밍

# 조도 센서

```
// 조건문을 사용하여 조도 센서 값에 따라 어두워지면 led 끄고 밝으면 켜는 프로그램
void setup() {
pinMode(13, INPUT_PULLUP);     //센서의 13핀 연결
ledcSetup(0, 5000, 8);                 //0번 채널에 5000Hz로 8비트로 세팅
ledcAttachPin(27, 0);                  //GPIO27핀을 0채널로 지정
}
void loop() {
int sensor = analogRead(13);                  //조도센서(13핀)의 값을 읽어 들임
if(sensor>700) ledcWrite(0, 255);
else ledcWrite(0, 30);
delay(50);
}
```

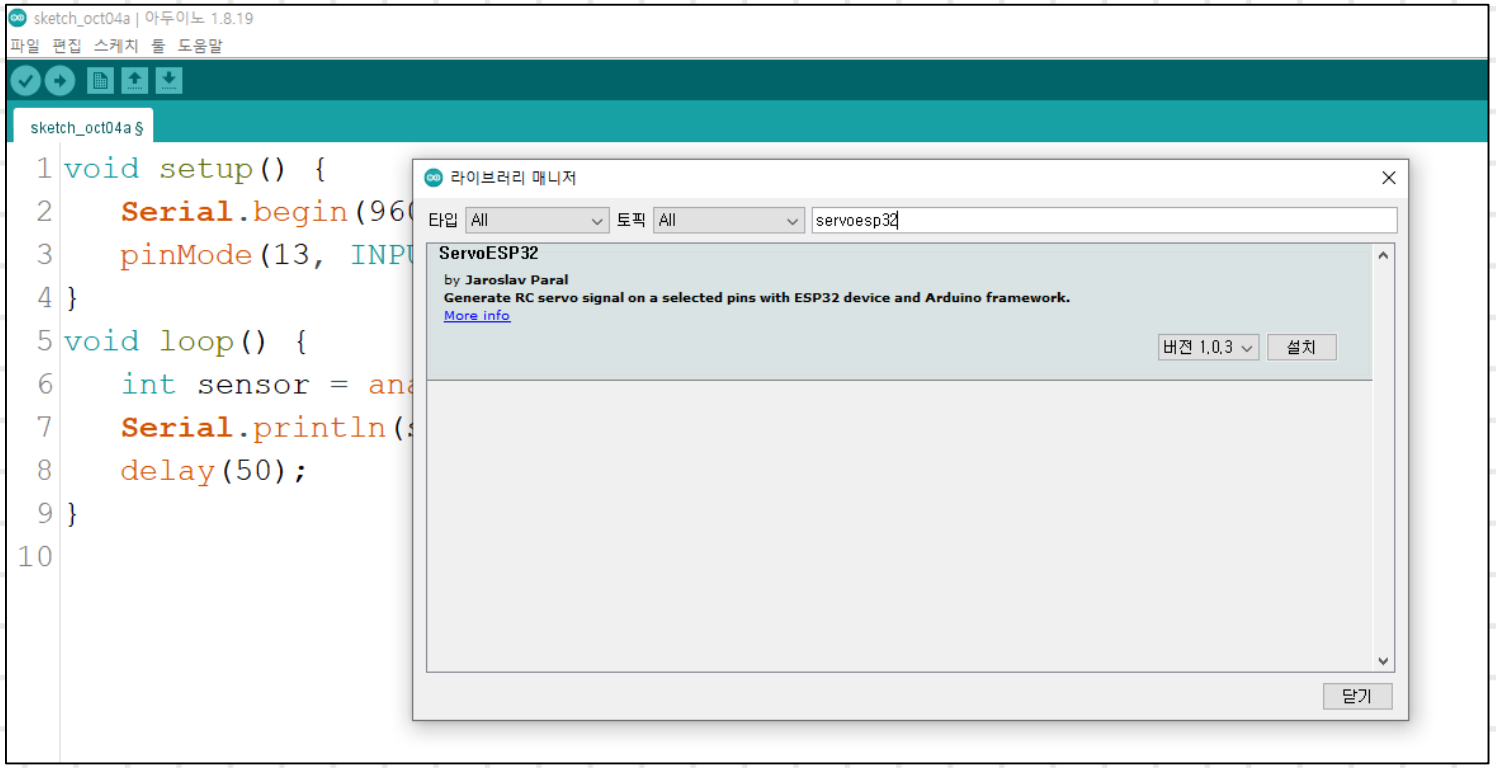
# 서보 모터

- 서보 모터를 쉽게 사용할 수도 도와주는 라이브러리 필요



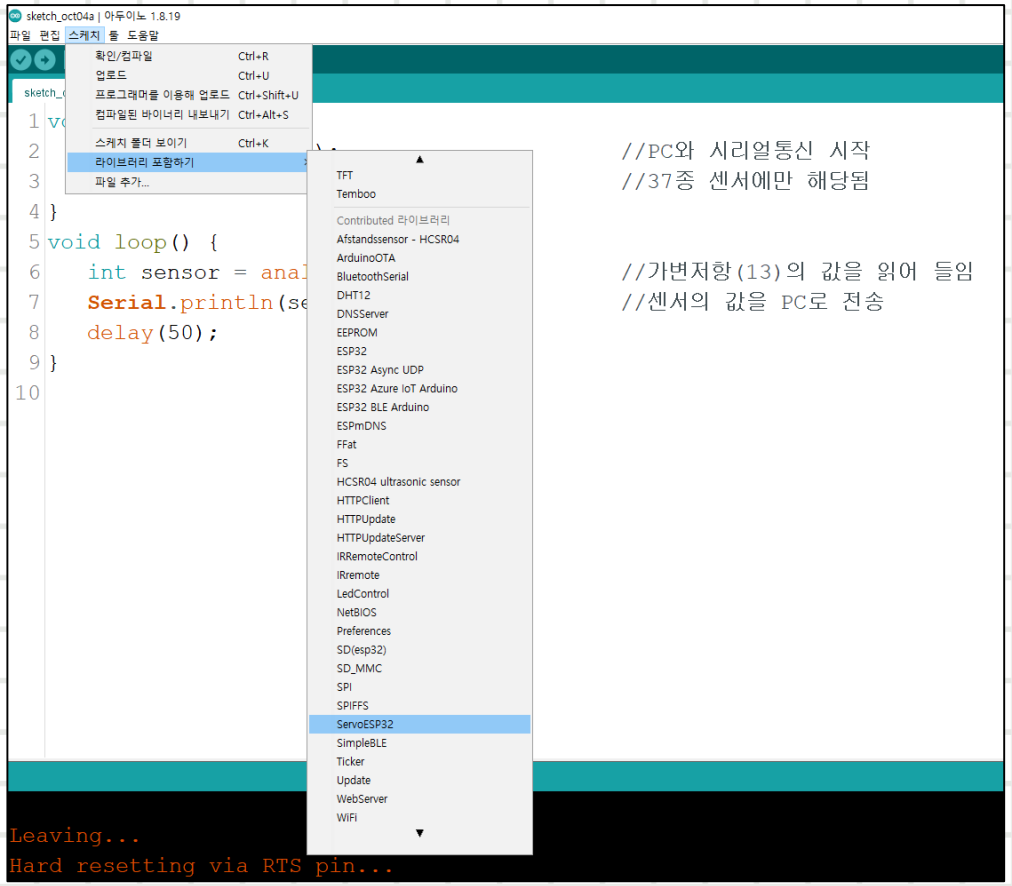
# 서보 모터

1) 메뉴의 스케치>>라이브러리 포함하기>>라이브러리 관리를 선택하고 ServoEsp32를 검색해서 설치



# 서보 모터

2) 메뉴의 스케치>>라이브러리 포함하기를 이용해서 ServoEsp32 라이브러리를 선택



# 서보 모터

```
#include <Servo.h>                    //서보모터 라이브러리를 사용함
Servo svo;                            //서보모터를 svo라 함
void setup() {
   svo.attach(13);                    //13핀에 서보모터 연결
   pinMode(15, INPUT);
}
void loop() {
   svo.write(0);                      //서보모터 0도로 이동
   delay(1000);                       //서보모터가 충분히 이동할 수 있도록 1초 대기
   svo.write(170);                    //서보모터 170도로 이동
   delay(1000);                       //서보모터가 충분히 이동할 수 있도록 1초 대기
}
```

# 서보 모터

sketch_oct04a | 아두이노 1.8.19

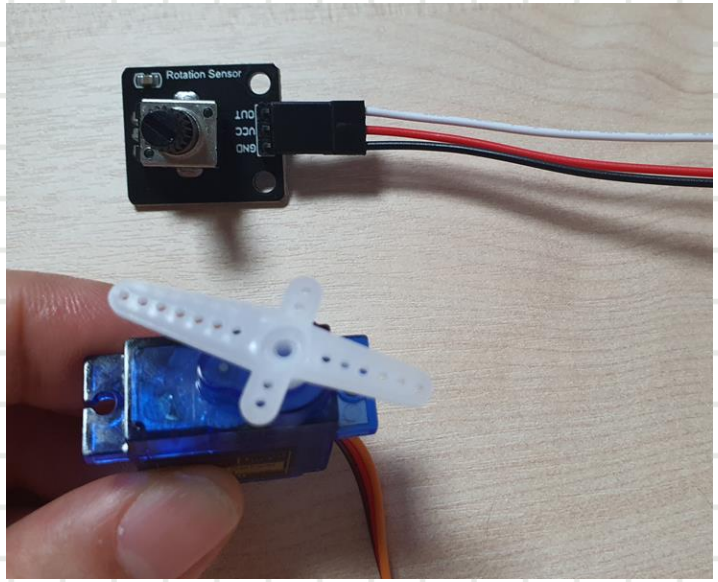
파일 편집 스케치 툴 도움말

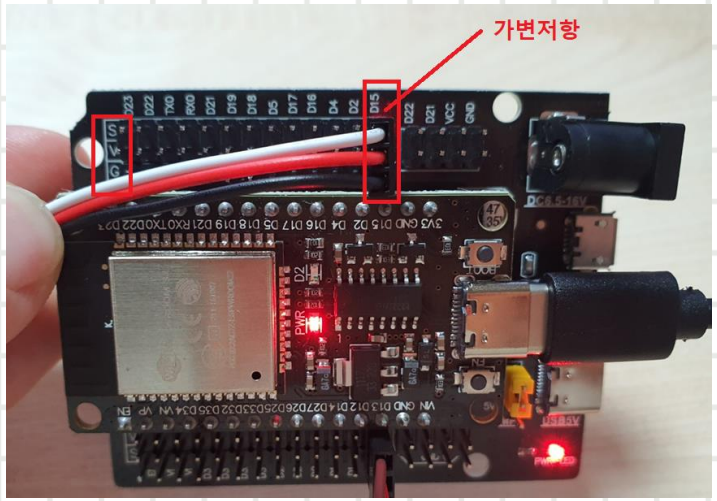
sketch_oct04a §

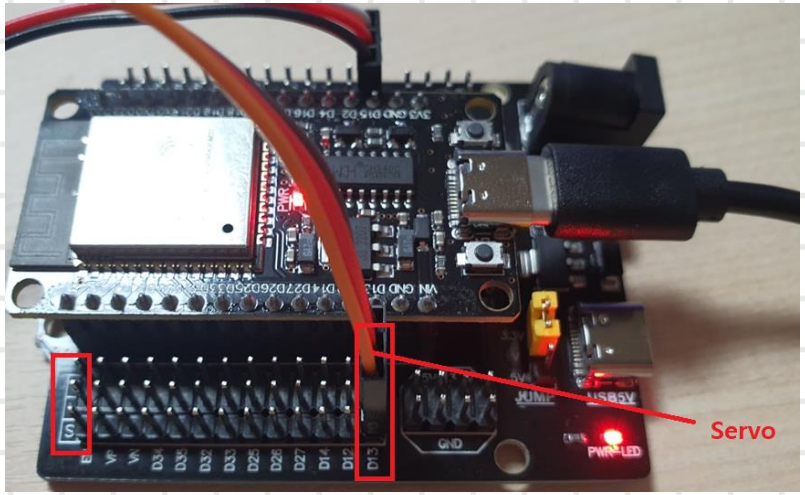
```
#include <Servo.h>          //서보모터 라이브러리를 사용함
Servo svo;                  //서보모터를 svo라 함

void setup() {
  svo.attach(13);           //13핀에 서보모터 연결
  pinMode(15, INPUT);
}

void loop() {
  svo.write(0);             //서보모터 0도로 이동
  delay(1000);              //서보모터가 충분히 이동할 수 있도록 1초 대기
  svo.write(170);           //서보모터 170도로 이동
  delay(1000);              //서보모터가 충분히 이동할 수 있도록 1초 대기
}
```

서보 모터 -> 이론 상 0~180도를 이동할 수 있지만, 180도 부근에선 많은 부하가 발생하여 진동이 발생하거나 열이 발생하기 때문에 **0~170도 정도를 사용하는 것을 권장**

# 서보 모터 + 가변저항



# 서보 모터 + 가변저항

```
#include <Servo.h>                                          //서보 모터 라이브러리를 사용함
Servo svo;                                                  //서보 모터를 svo라 함
void setup() {
   svo.attach(13);                                          //13핀에 서보 모터 연결
   pinMode(15, INPUT);                                      //15핀에 가변저항 연결
}
void loop() {
   int sensor = analogRead(15);
   int angle = map(sensor, 0, 4095, 170, 0);                //0~4095 => 170~0도
   svo.write(angle);
}
```

# 서보 모터 + 가변저항

sketch_oct04a | 아두이노 1.8.19

파일 편집 스케치 툴 도움말

sketch_oct04a §

```
#include <Servo.h>                                  //서보모터 라이브러리를 사용함
Servo svo;                                          //서보모터를 svo라 함

void setup() {
  svo.attach(13);                                   //13핀에 서보모터 연결
  pinMode(15, INPUT);                               //15핀에 가변저항 연결
}

void loop() {
  int sensor = analogRead(15);
  int angle = map(sensor, 0, 4095, 170, 0);     //0~4095 => 170~0도
  svo.write(angle);
}
```

서보 모터와 가변저항이 움직임이 서로 반대로 움직임(가변저항을 왼쪽으로 돌리면 서보 모터는 오른쪽으로 돎) 움직임을 맞추기 위해 **map 함수에서 0~170도가 아닌 170~0도로 뒤집어 사용**

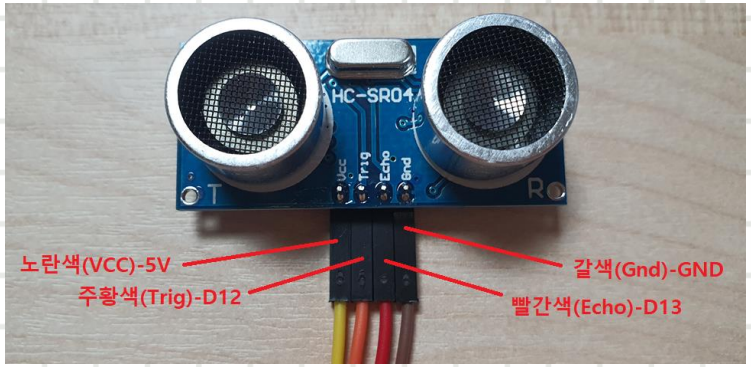
# 초음파 센서

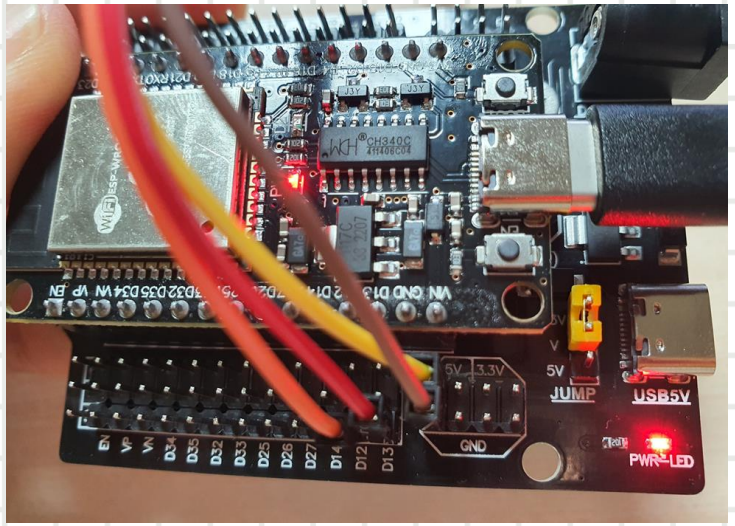

## 초음파 센서

- 초음파를 이용하여 거리를 측정하는 센서
- Trig : 초음파가 발사됨.
- Echo : 발사된 초음파가 사물에 부딪혀 반사되어 Echo 부분으로 돌아옴.
- 초음파 발생 -> 장애물과 부딪힘 -> 반사되어 돌아오는 시간을 측정(거리 계산)
- 거리 측정에 사용
- 초음파 : 인간이 들을 수 없는 범위의 20kHZ이상의 주파수

# 초음파 센서

- 초음파를 방출하여 장애물에 반사되어 되돌아오기까지의 시간을 측정하는 원리로 거리를 측정하는 센서


| 초음파센서(HC-SR04) | Vcc-5V / Trig-D12 / Echo-D13 / Gnd-GND |
|---|---|

주의사항 : HC-SR04는 5V로 작동하기 때문에 VCC핀을 ESP32의 VIN핀 또는 5V에 연결해야 함!

# 초음파 센서

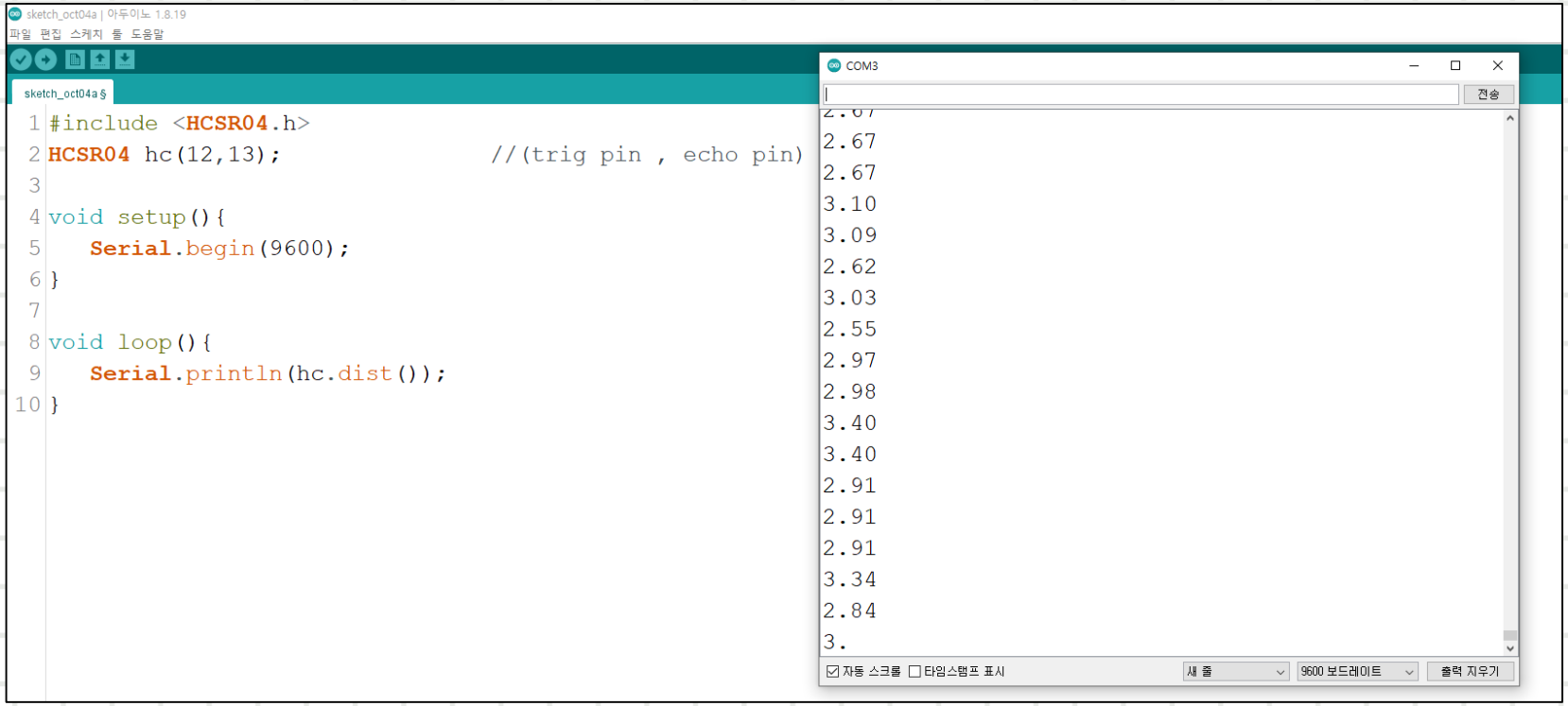
1) 메뉴의 스케치>>라이브러리 포함하기>>라이브러리 관리를 선택하고 ServoEsp32를 검색해서 설치


* HCSR04로 검색하면 여러 가지 라이브러리가 나온다. 자신이 사용하기 편한 라이브러리를 설치해도 되지만, 여기에서는 HC-SR04 ultrasonic sensor을 설치

# 초음파 센서

```
#include <HCSR04.h>
HCSR04 hc(12,13);                    //hc(trig pin , echo pin)

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

void loop(){
   Serial.println(hc.dist());
}
```

# 초음파 센서

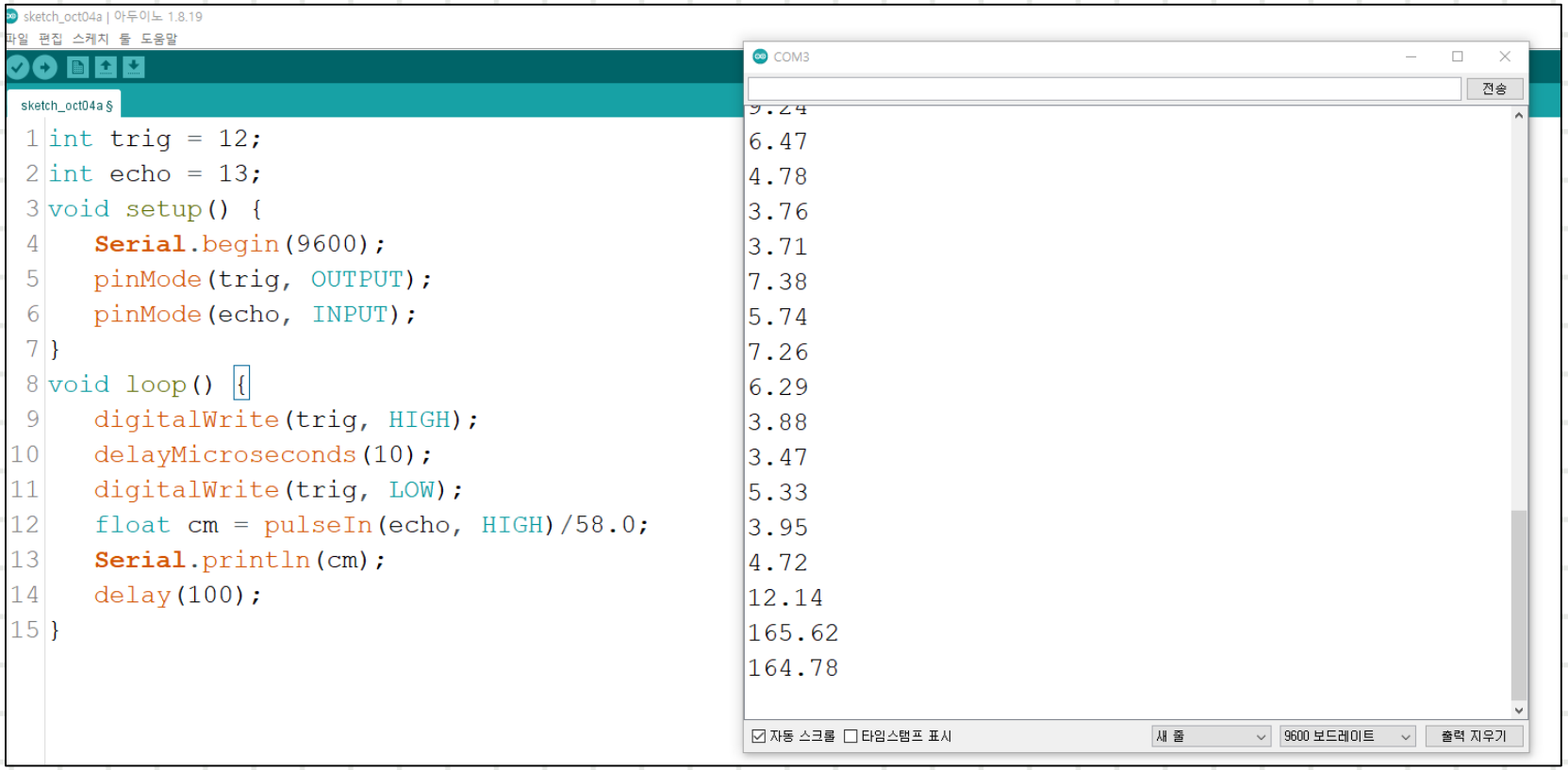



* HC-SR04는 약 3cm~200cm 정도의 거리 측정

# 초음파 센서

```
#include <Servo.h>                                          //서보 모터 라이브러리를 사용함
Servo svo;                                                  //서보 모터를 svo라 함
void setup() {
   svo.attach(13);                                          //13핀에 서보 모터 연결
   pinMode(15, INPUT);                                      //15핀에 가변저항 연결
}
void loop() {
   int sensor = analogRead(15);
   int angle = map(sensor, 0, 4095, 170, 0);                //0~4095 => 170~0도
   svo.write(angle);
}
```

# 초음파 센서


sketch_oct04a | 아두이노 1.8.19
파일 편집 스케치 툴 도움말
sketch_oct04a §
1 int trig = 12;
2 int echo = 13;
3 void setup() {
4 Serial.begin(9600);
5 pinMode(trig, OUTPUT);
6 pinMode(echo, INPUT);
7 }
8 void loop() {
9 digitalWrite(trig, HIGH);
10 delayMicroseconds(10);
11 digitalWrite(trig, LOW);
12 float cm = pulseIn(echo, HIGH)/58.0;
13 Serial.println(cm);
14 delay(100);
15 }
COM3
전송
6.47
4.78
3.76
3.71
7.38
5.74
7.26
6.29
3.88
3.47
5.33
3.95
4.72
12.14
165.62
164.78
자동 스크롤
타임스탬프 표시
새 줄
9600 보드레이트
출력 지우기

# 초음파 센서

```
// 초음파 센서로 입력되는 값을 시리얼 모니터로 출력하는 프로그램
#define TRIG 7 //TRIG 핀 설정 (초음파 보내는 핀)
#define ECHO 6 //ECHO 핀 설정 (초음파 받는 핀)
void setup() {
  Serial.begin(9600);
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
}
void loop()
{
  long duration, distance;

  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);

  duration = pulseIn (ECHO, HIGH);
  //물체에 반사되어 돌아온 초음파의 시간을 변수에 저장
  //34000*초음파가 물체로 부터 반사되어 돌아오는 시간 /1000000 / 2(왕복 값이 아니라 편도값이기때문에 나누기 2)
   distance = duration * 17 / 1000;
  //PC모니터로 초음파 거리값을 확인
Serial.println(duration );
 //초음파가 반사되어 돌아오는 시간
  Serial.print( " \nDIstance :  " );
  Serial.print(distance);
 //측정된 물체로부터 거리값(cm값)
  Serial.println(" Cm");
  delay(1000);
}
```

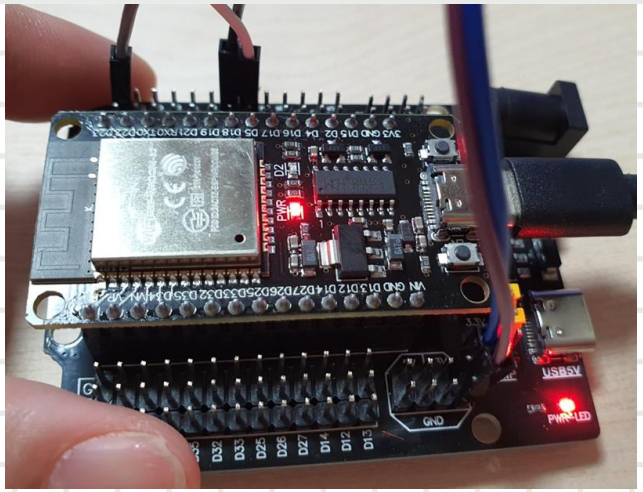
# 도트 매트릭스

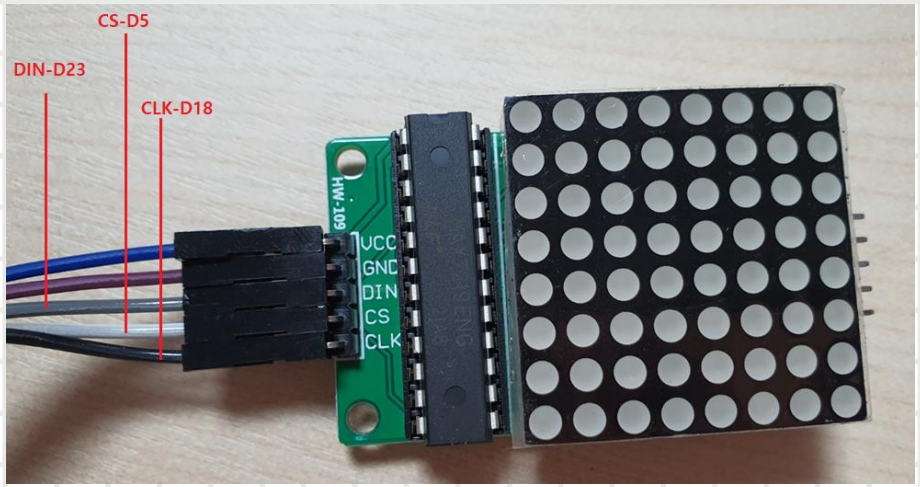
## 도트 매트릭스

- 디지털 출력
- 숫자, 문자, 이미지를 표시하기 위해 사용
- 가로, 세로에 led를 배치하여 각 점을 발광하는 방법

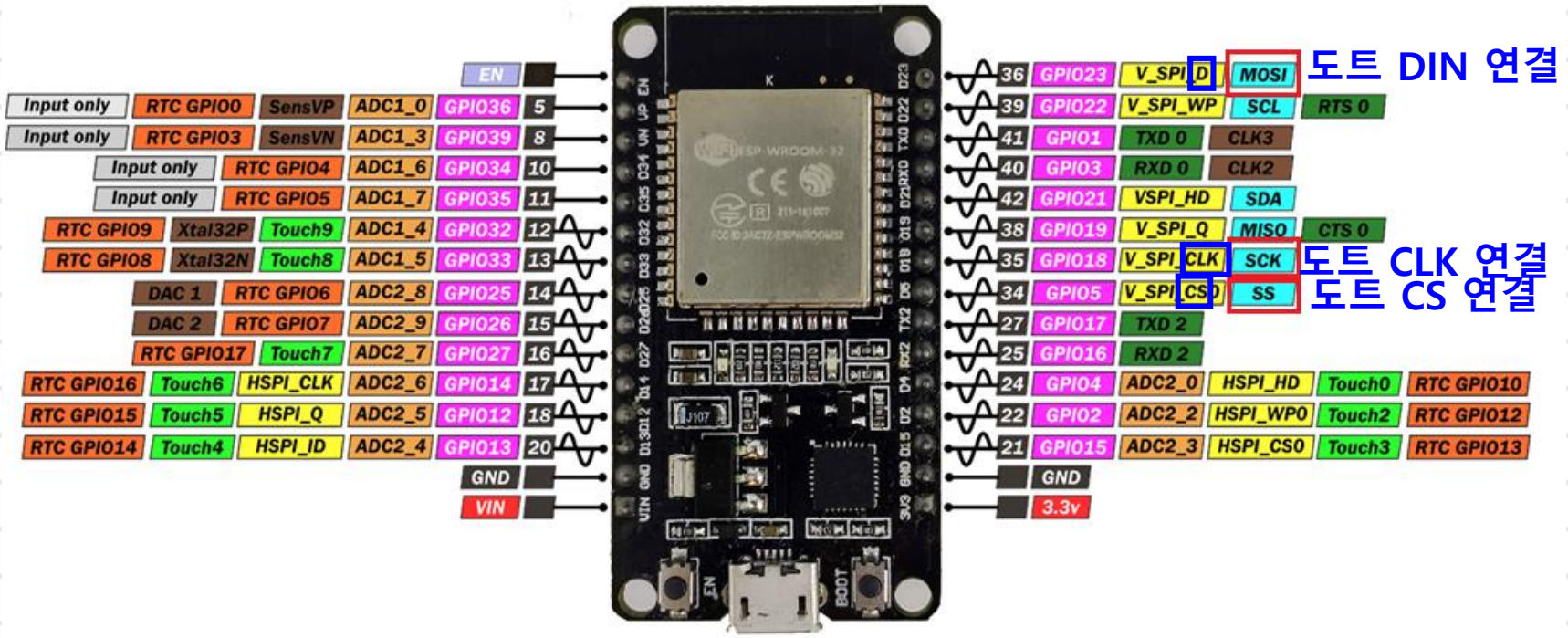
# 도트 매트릭스

- 기능이 부여된 핀에 각각 연결하여 사용





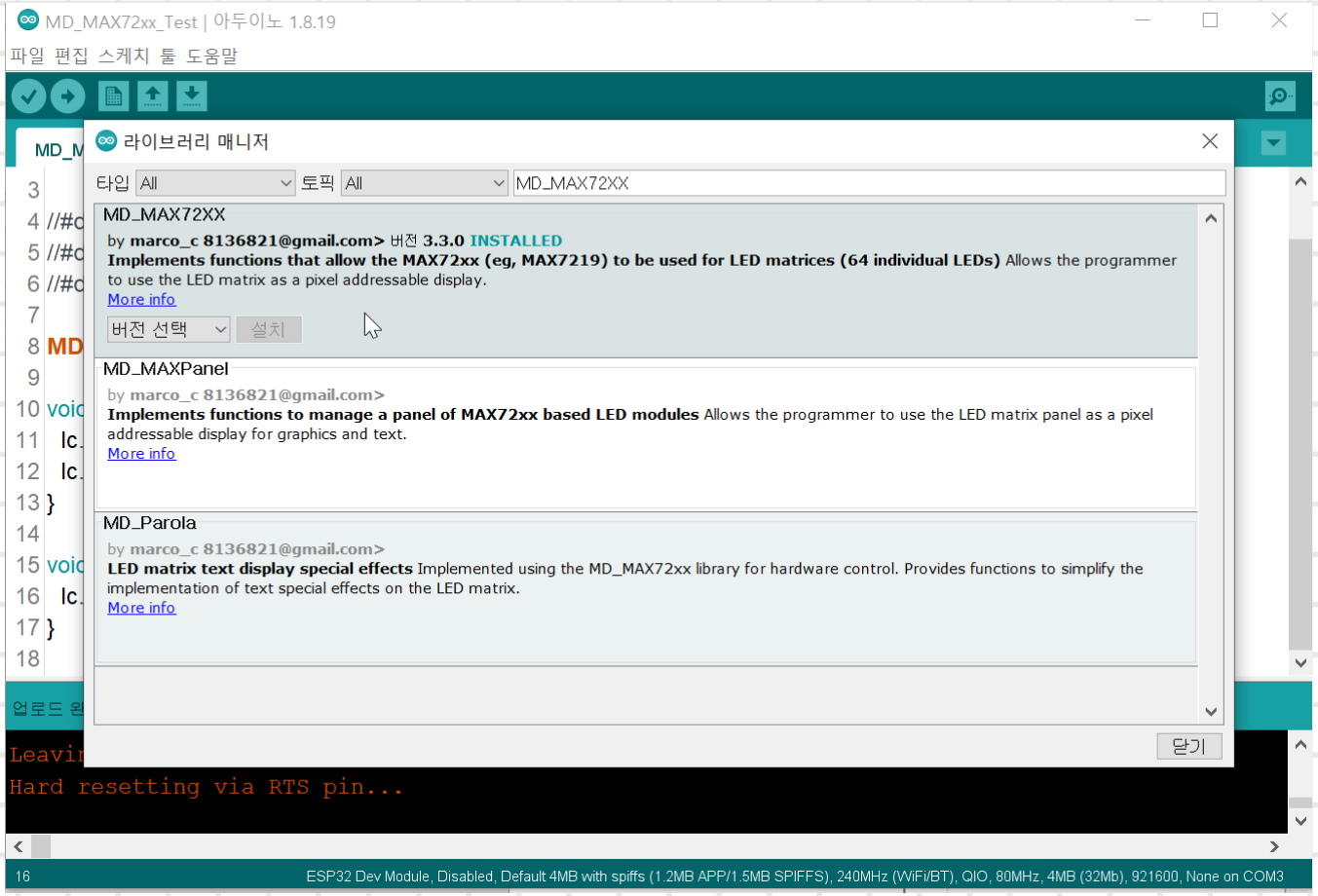
# 도트 매트릭스



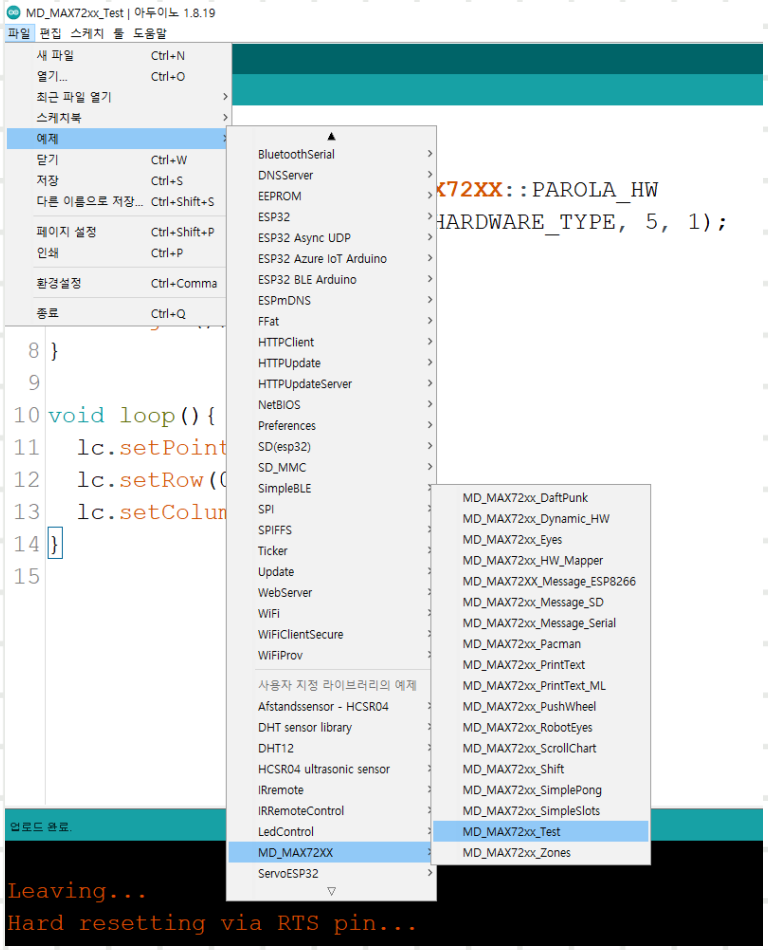

- MOSI : Master Out, Slave In – 마스터에서 데이터를 출력하기 위한 신호 선
- MISO : Master In, Slave Out – 슬레이브에서 데이터를 출력하기 위한 신호 선
- SCK : Clock 신호 선
- SS : Slave Select – 데이터를 송수신할 슬레이브를 선택하기 위한 신호 선

# 도트 매트릭스

1) 메뉴의 스케치>>라이브러리 포함하기>>라이브러리 관리를 선택하고 MD_MAX72XX를 검색해서 설치



# 도트 매트릭스

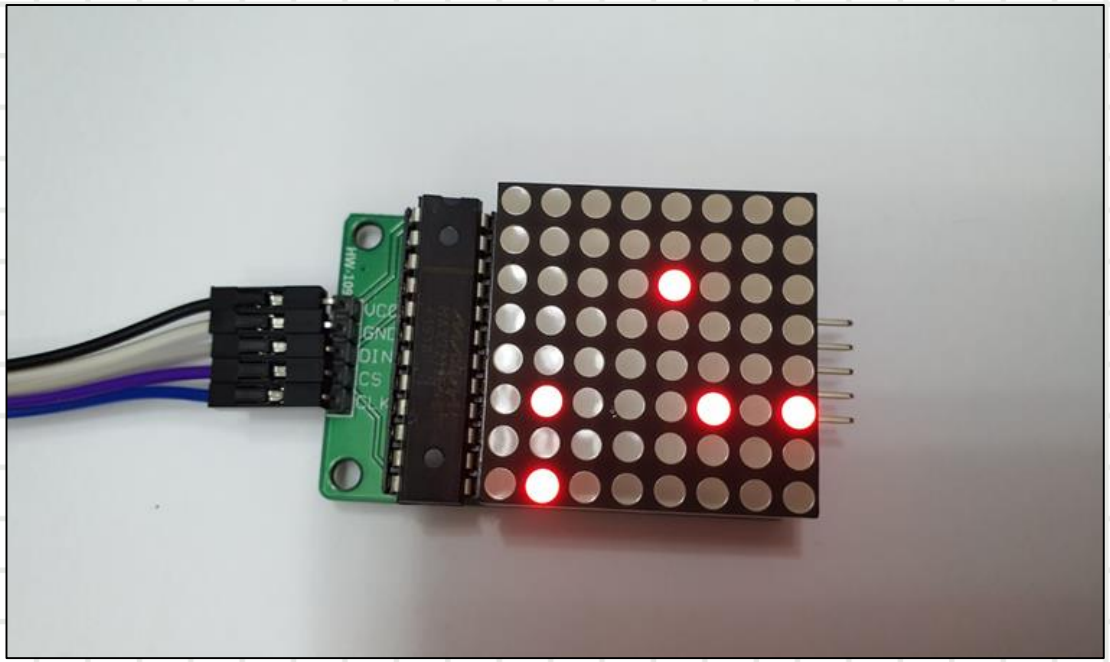



2) 파일>>예제를 이용해서 활용 확인

# 도트 매트릭스

```
#include <MD_MAX72xx.h>
#define HARDWARE_TYPE MD_MAX72XX::PAROLA_HW
MD_MAX72XX lc = MD_MAX72XX(TYPE, 5, 1);
void setup(){
  lc.begin();
}
void loop(){
  lc.setPoint(2, 4, true);                  //(2,4)위치의 LED를 켜기
  lc.setRow(0, 5, B10100000);               //5행 LED를 10100000 형태로 켜기
  lc.setColumn(0, 1, B10100000);      //1열 LED를 10100000 형태로 켜기
}
```

# 도트 매트릭스



# 도트 매트릭스

[도트매트릭스 코드 변환기
https://dot2bit.netlify.app](https://dot2bit.netlify.app)

# 도트 매트릭스

## 미션

하트 모양 프로그래밍

# 도트 매트릭스


하쌤보드 ESP32 & Arduino
도트매트릭스 코드 변환기
By Kiwoon
아두이노 ESP32
7행: B10000001
6행: B11100111
5행: B01100110
4행: B01111110
3행: B01111110
2행: B00111100
1행: B00111100
0행: B00011000
모두 선택
지우기
반전
행(setRow()) 코드
열(setColumn()) 코드
선택
################## 행(setRow())코드 변환 결과 (ver 2.0.1) ###################
lc.setRow(0, 7, B10000001);
lc.setRow(0, 6, B11100111);
lc.setRow(0, 5, B01100110);
lc.setRow(0, 4, B01111110);
lc.setRow(0, 3, B01111110);
lc.setRow(0, 2, B00111100);
lc.setRow(0, 1, B00111100);
lc.setRow(0, 0, B00011000);

# 질문해 주세요.

# 참고 문헌


- https://www.eduino.kr/
- www.mischianit.org (mischianti.org)
- 하쌤보드로 ESP32 고수되기(하주원 저)