2018

WeMos D1 Mini Pro

WEMOS D1 MINI PRO(V1.1.0) BASIC DEVELOPMENT

PARK, JEONG-UK

Contents

1. WeMos 란?!

1. WeMos 보드 소개

WeMos D1 mini Pro (v1.1.0)은 Arduino(아두이노) Uno 보드와 ESP8266 Wi-Fi 모듈을 혼합해서 Wi-Fi 사용이 가능하도록 제작한 보드이다. ESP-8266EX 기반의 제품으로 아두이노 및 NodeMCU와 호환된다.

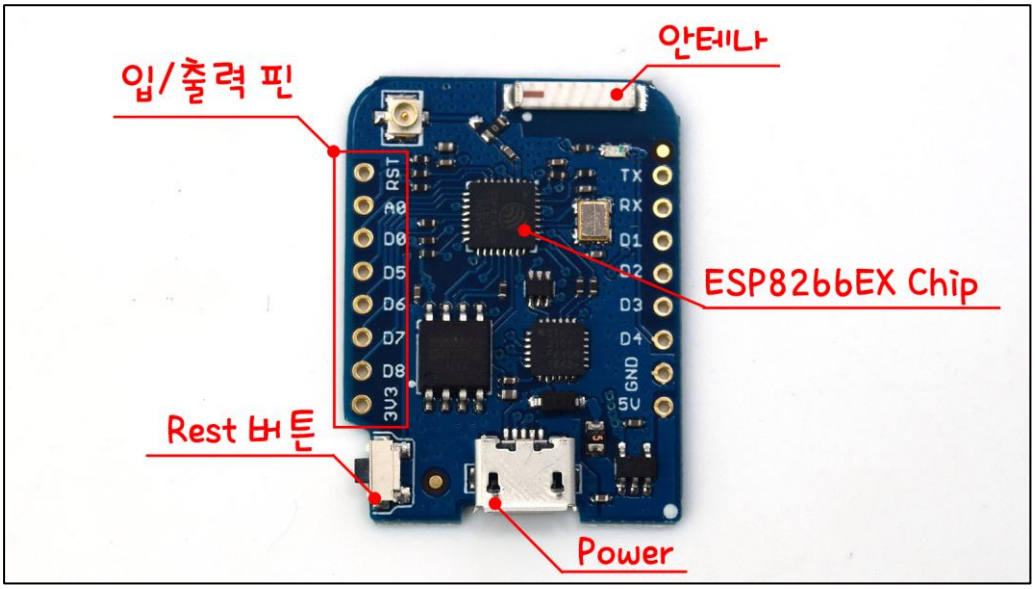

Figure 1. WeMos 개발 보드 외형 및 세부 항목 설명

기본적으로, 16MB의 Flash Memory(플래시 메모리)를 지원하며, 외부 안테나 커넥터[1]를 지원하며, 내부에 ESP-8266Ex에 기반한 내장형 세라믹 안테나[2]를 포함하고 Wi-Fi 및 Bluetooth 통신을 기본적으로 지원한다. 하단의 보드로 Micro USB 전원부에서 전원을 공급받으며, 각각 D0 ~ D9개의 Digital(디지털) Pin과, 1개의 Analog(아날로그) Pin, 각각 Tx, Rx, 3.3V 전압, 5V 전압 Pin, 1개의 그라운드(Ground; GND) Pin을 지원한다. 외부 환경으로부터 보드에 이상이 발생했을 때 Reset 버튼을 통한 자체 펌웨어 리셋을 사용할 수 있다.

ESP-8266EX는 안테나 스위치, RF 발룬, 전력 증폭기, 저잡음 수신 증폭기, 필터 및 전력 관리 모듈을 통합하고 있다. 또한 컴팩트 한 디자인은 PCB 크기를 최소화하고 최소한의 외부 회로를 구성하도록 할 수 있다.

[1] 좌측 상단(RST 핀 위)의 원형 모양의 외부 커넥터

[2] 우측 상단(Tx 핀 위)의 직사각형 모양의 안테나

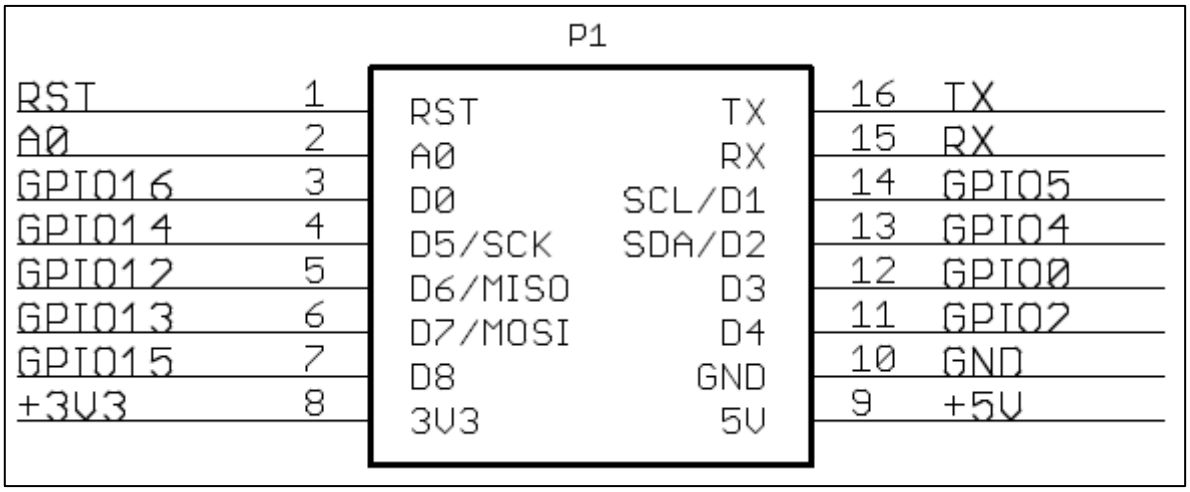

Figure 2. WeMos 보드 Pin Map (핀맵)

핀 맵은 각각 그림에서 나온 것 과 같이 1번 ~ 16번 핀으로 맵핑 해서 사용할 수 있으며, 직관적으로 D0 ~ D8, A0와 같이 이름 그대로 맵핑 해서 사용할 수도 있다.

Table 1. Pin

PIN	FUNCTION	ESP-8266 PIN
TX	TXD	TXD
RX	RXD	RXD
A0	Analog input, max 3.3V input	A0
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
G	Ground	GND
5V	5V	-
3V3	3.3V	3.3V
RST	Reset	RST

각 핀의 세부 정보 테이블을 참조하여 개발에 사용하자.

Table 2. Technical specs

MICROCONTROLLER	ESP-8266EX
OPERATING VOLTAGE	3.3V
DIGITAL I/O PINS	11
ANALOG INPUT PINS	1(Max input: 3.2V)
CLOCK SPEED	80MHz/160MHz
FLASH	16M bytes
LENGTH	34.2mm
WIDTH	25.6mm
WEIGHT	2.5g

Table 2.에서 WeMos 보드의 상세 스펙을 참조하여 개발에 사용 할 수 있다. 기타 세부 사항은 WeMos Wiki page를 참고하도록 한다.

2. WeMos 개발 환경 설정

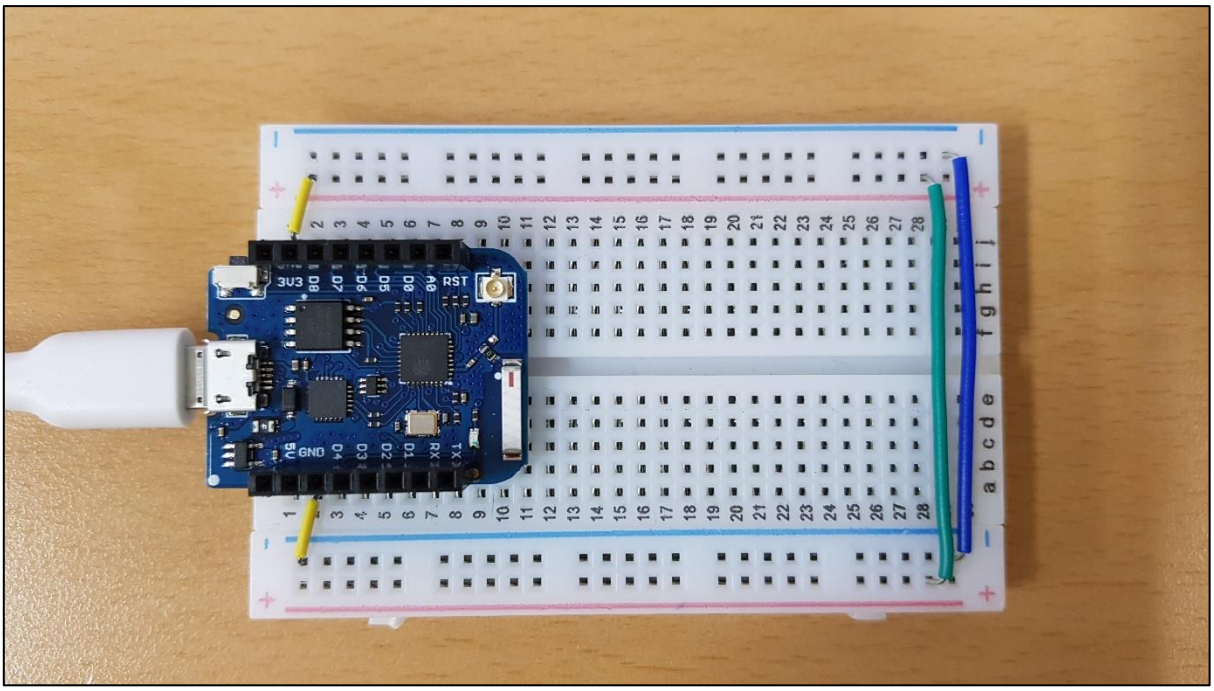

Figure 3. USB를 사용하여 컴퓨터와 WeMos 보드를 연결

일차적으로 사용하는 PC에 WeMoS 드라이버(Driver)를 설치하기 위해 키트에 동봉된 Micro USB를 사용하여 드라이버를 설치하고자 하는 PC와 연결한다. 이후 정상적인 연결을 확인하기 위해 윈도우(Windows) 장치 관리자를 통하여 정상적인 연결 상태를 확인한다. 윈도우 장치 관리자의 실행 방법은 [제어판] -> [장치 관리자]를 통하는 방법이 있고, “Win Key + R(실행)” 명령어를 다음과 같이 입력하여 접근하는 방법이 있다.

```
devmgmt.msc
```

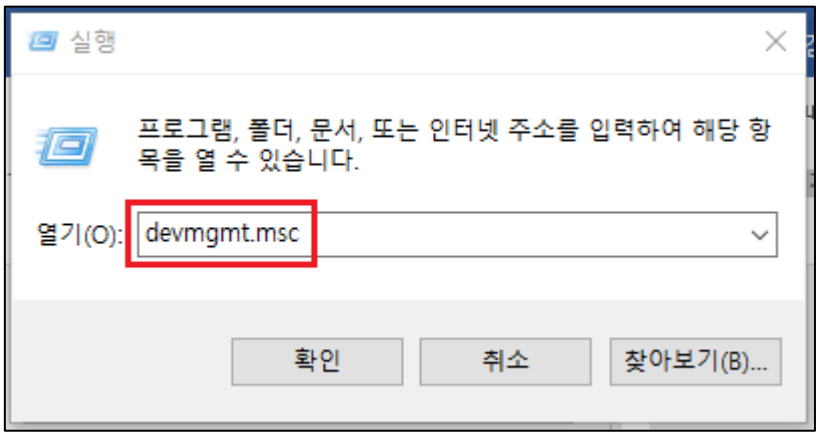

Figure 4. 실행 명령어 창

Figure 4.에서 확인 할 수 있는 것처럼 장치 관리자를 실행 하여 WeMos 연결 상태를 확인.

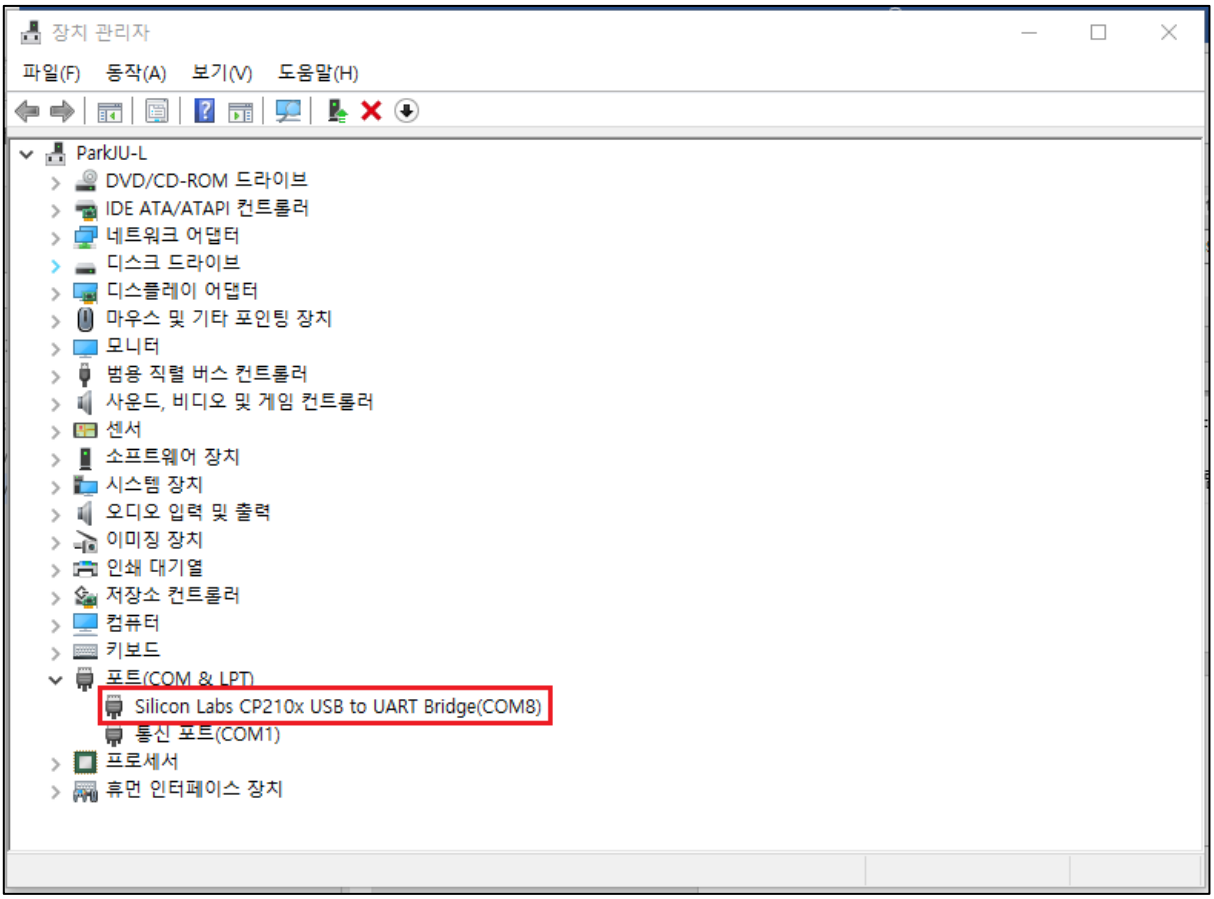

Figure 5. 장치관리자에서 WeMos 연결을 확인.

장치 관리자를 정상적으로 실행하면, Figure 5.에서 확인 할 수 있는 것처럼 COM 포트에서 잡힌 WeMos 보드를 확인 할 수 있다. 하지만 특별한 이유로 자동으로 드라이버가 설치되지 않는 경우 해당 링크를 통하여 수동적으로 드라이버를 설치하도록 한다.

드라이버가 정상적으로 설치된 후 개발 환경 설정을 위해 본래는 아두이노에 주로 사용하지만 WeMos에도 동일하게 사용할 수 있는 Arduino 통합 개발 환경(Integrated Development Environment; IDE) 프로그램을 링크를 통해 설치하여 코드 작성 및 업로드에 사용할 것이다.

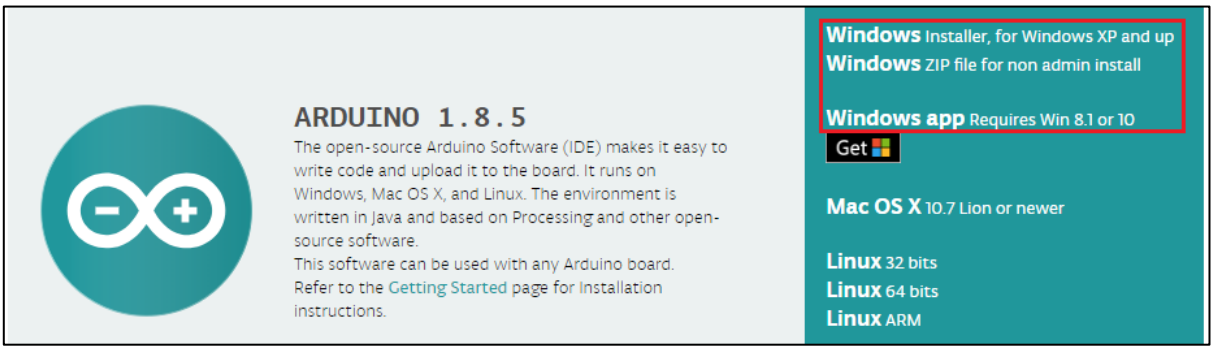

Figure 6. Arduino IDE 1.8.5 다운로드 페이지

아두이노 통합 개발 환경을 다운로드 누르게 되면, Arduino Sofeware를 후원하는 페이지가 나오게 되고, "Just Download" 버튼을 통하여 후원 없이 다운로드 하도록 한다.

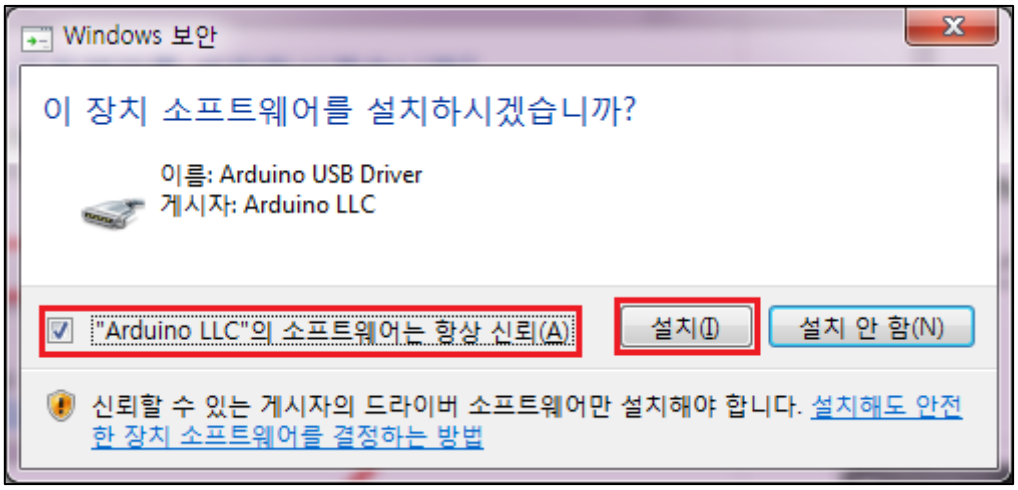

Figure 7. 소프트웨어 설치 동의

정상적으로 소프트웨어 설치 동의 메시지를 확인 해 준다. 소프트웨어 설치 메시지가 안 뜬다면 자동으로, 정상적으로 소프트웨어 설치가 진행되는 것이다. 이후 설치가 완료되었다면, 아두이노 IDE를 실행하고 연동 예제를 간단하게 따라한다.

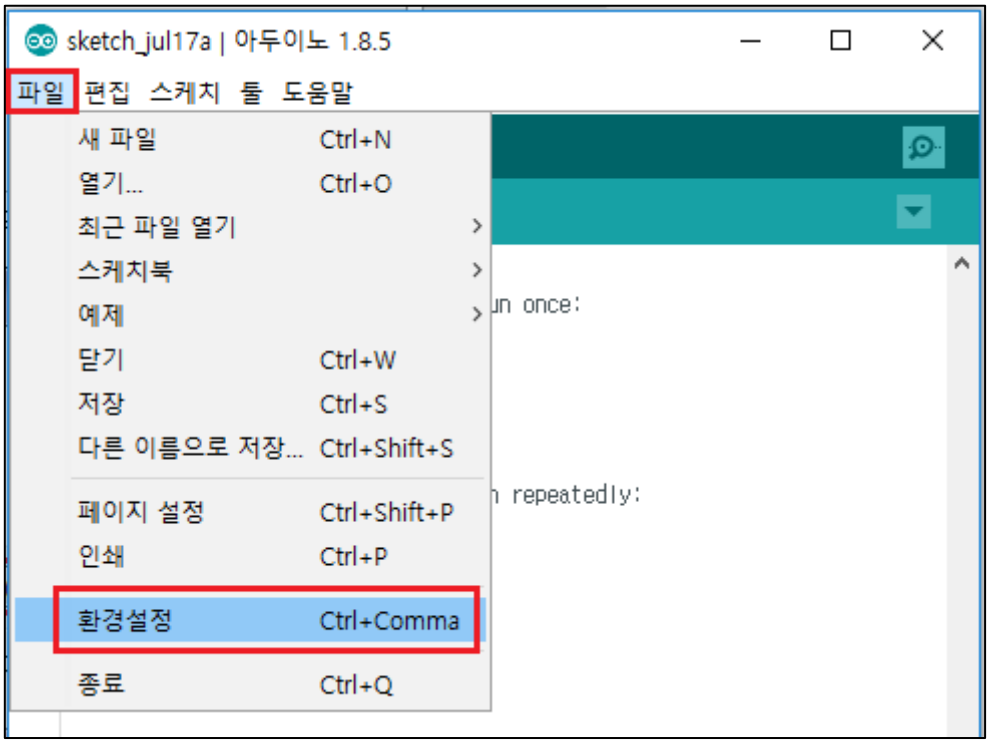

Figure 8. IDE 환경설정 위치

아두이노 IDE에서 WeMos 개발을 사용하기 위해서는 먼저 환경설정을 통한 WeMos 보드 관련 세팅을 설치해야 한다. Figure 8처럼 [파일] -> [환경설정]을 통하여 “보드 매니저”를 다운로드 할 수 있도록 한다.

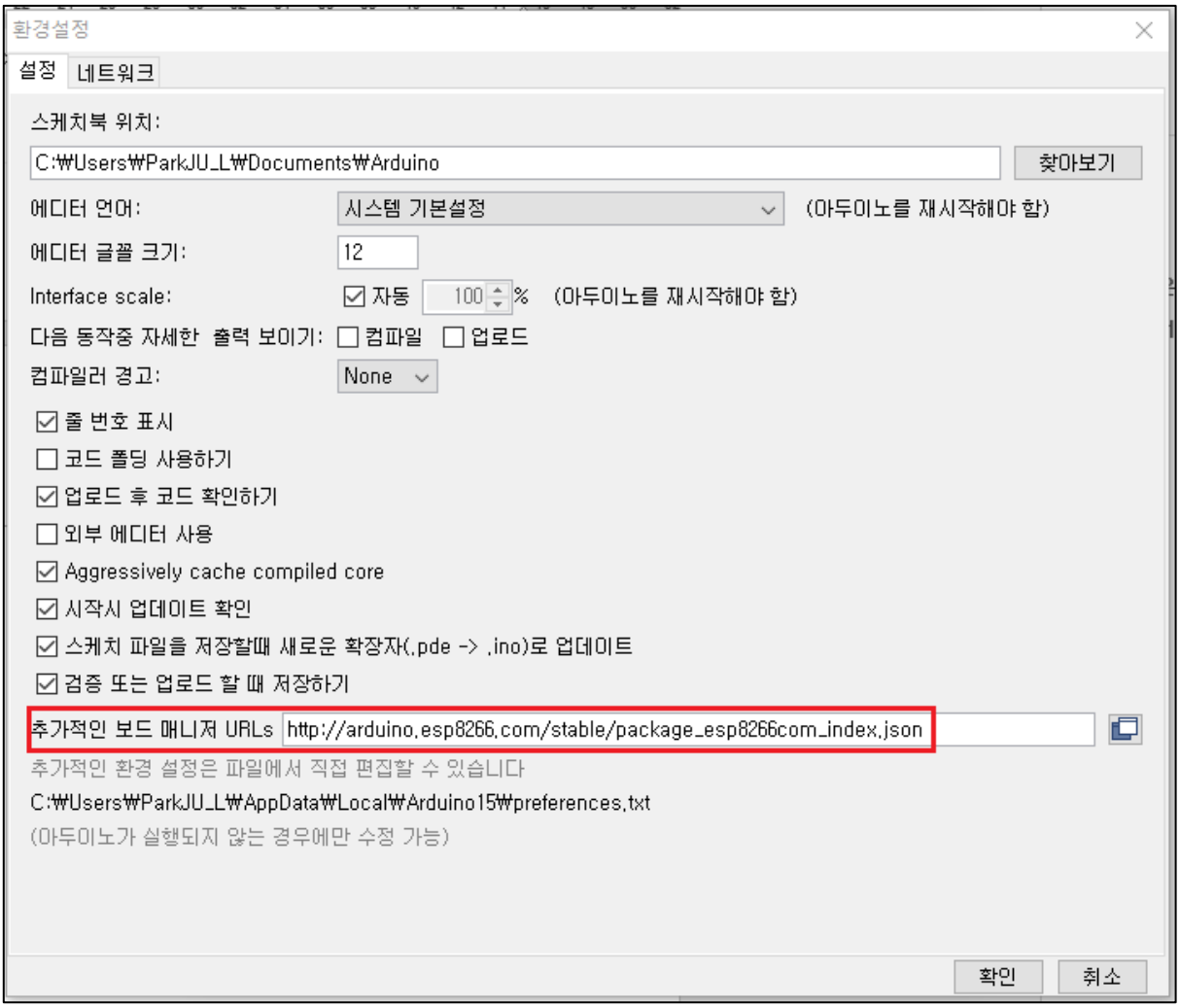

Figure 9. 보드 매니저 URL 설정

추가적인 보드 다운로드를 위해 보드 매니저 설정에 다음과 같이 입력한다.

http://arduino.esp8266.com/stable/package_esp8266com_index.json

해당 보드 매니저 URL을 설정하면 이제 IDE의 보드 매니저를 사용하여 WeMos 보드들의 칩셋 정보를 받아올 수 있다.

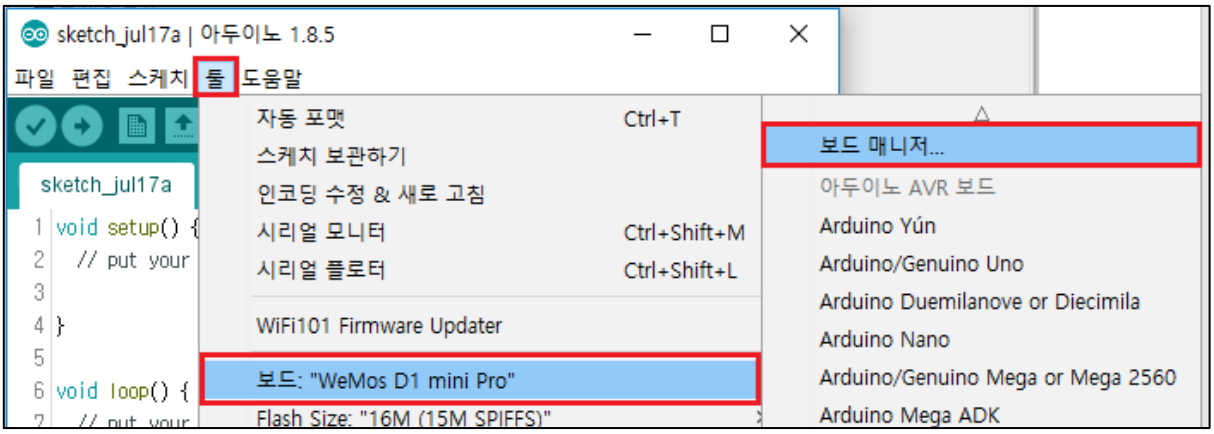

Figure 10. 보드 매니저 위치.

IDE의 보드 매니저에 접근하여 새로운 보드 목록을 다운로드 할 것이다.

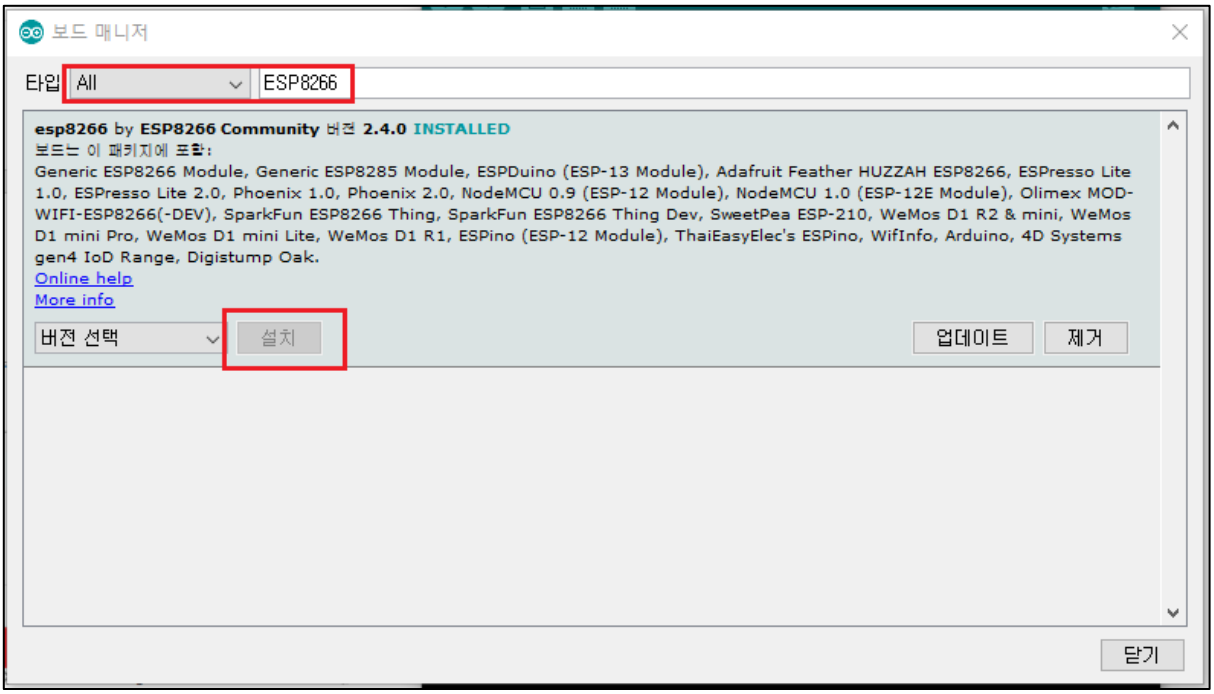

Figure 11. 보드매니저에서 ESP-8266을 다운로드.

이제 검색창을 이용하여 “ALL”(모든항목)에서 ESP-8266을 검색하여 다운로드 한다. 해당 Figure 11에서는 이미 필자가 다운로드 한 상태라 설치는 활성화 되지 않고, 업데이트와 제거만 사용 가능하지만 정상적으로는 설치 버튼을 클릭하여 라이브러리를 다운로드 하면 된다. 설치 이후 버전명 우측에 청록색의 “INSTALLED”를 확인 할 수 있으며, “업데이트” 버튼이 활성화되지 않는 경우 최신 버전으로 다운로드 되어있기 때문에 업데이트 불가능한 상태를 의미한다.

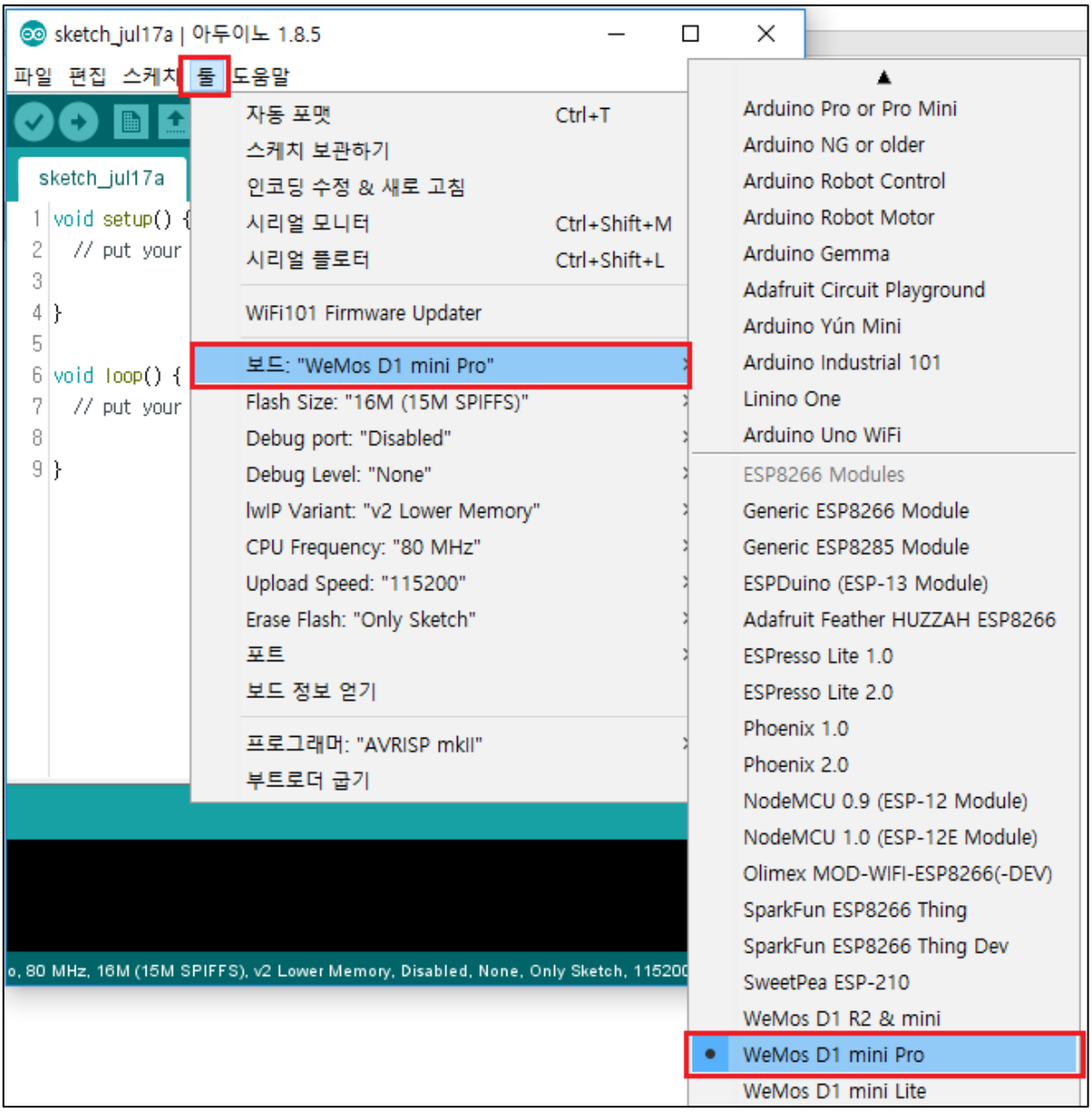

Figure 12. WeMos D1 mini Pro 보드 설정

보드 설치가 정상적으로 완료되었다면. [툴] -> [보드] 에서 정상적으로 WeMos 보드들[3]을 확인할 수 있고, 해당 문서에서 사용할 WeMos D1 mini Pro 보드를 선택하여 앞으로의 소스코드 작성에 사용할 수 있다.

[3] 해당 Figure 12.에서는 D1 R2 ~ D1 mini Lite 까지 확인 할 수 있다.

이제 기본적으로 WeMos 보드가 정상적으로 작동하는지 테스트하기 위해 테스트 코드를 업로드 하여 가장 기본적인 LED 테스트를 진행 한다.

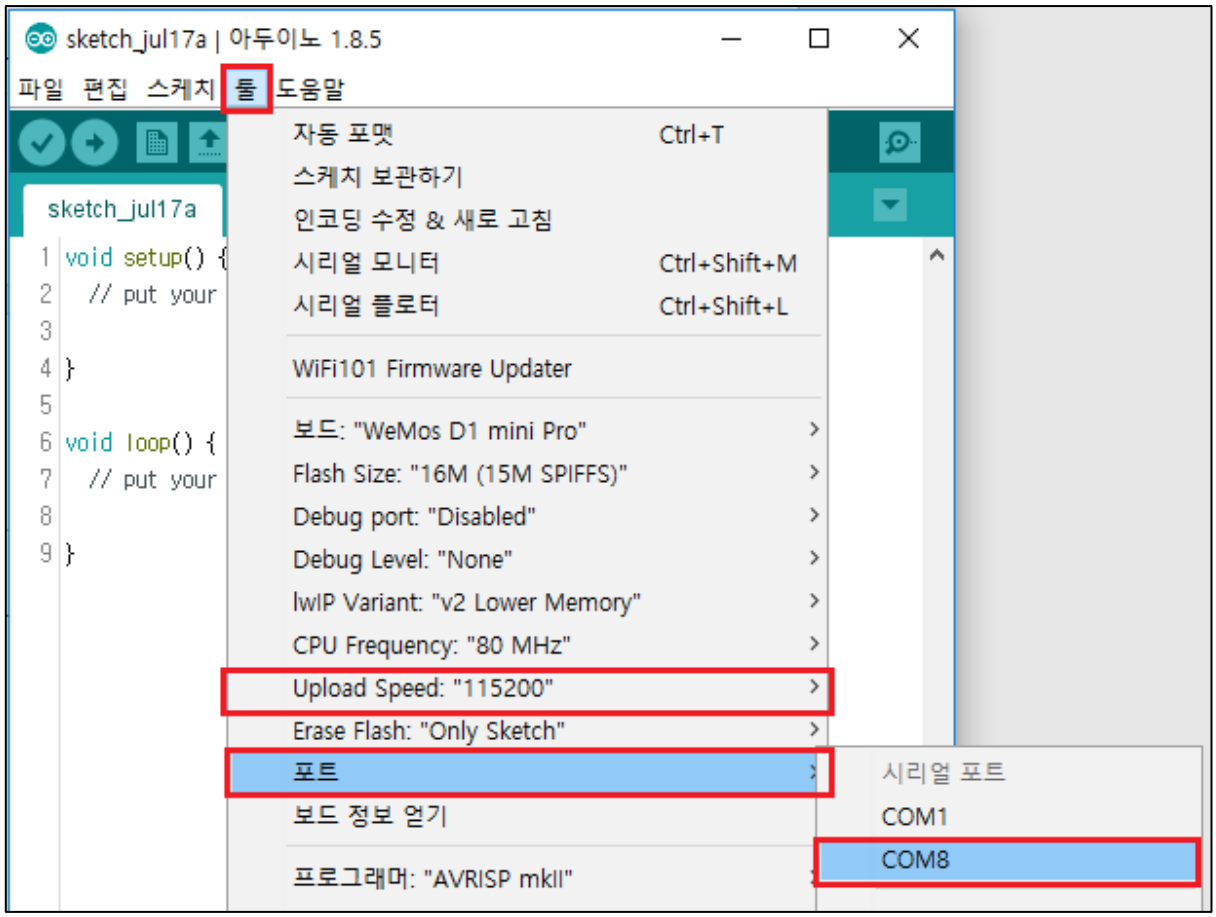

Figure 13. 포트 설정.

정상적인 연결을 위해 Baud Rate(보드 레이트) 설정을 115200으로 설정한 뒤 포트를 연결한 WeMos에 맞게 설정한다. 해당 Figure 13에서는 COM 포트 8번으로 연결했지만, 연결하는 PC와 WeMos 보드에 따라 COM 포트가 다르므로 정확히 확인 후 연결하도록 한다(문서 상단의 Figure 5 에서 확인했던 COM 포트를 사용하면 된다).

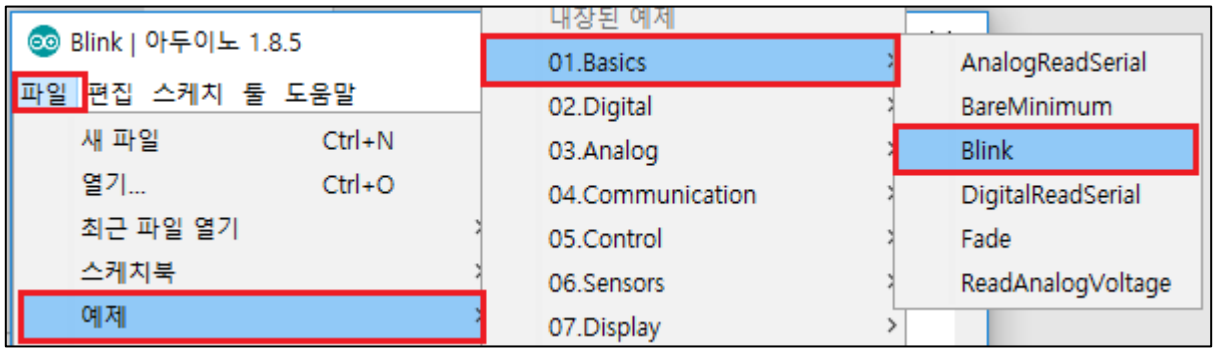

Figure 14. Blink 예제 사용.

가장 기초가 되는 예제 중 On-Board(온-보드) LED를 사용한 Blink 예제를 테스트할 것이다.

Blink.ino

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA
and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN
is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on
your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the
board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW)
  delay(1000);                      // wait for a second
  }
```

예제를 실행하면 소스를 따로 작성할 필요없이 해당 “Blink” 코드를 사용할 수 있다.

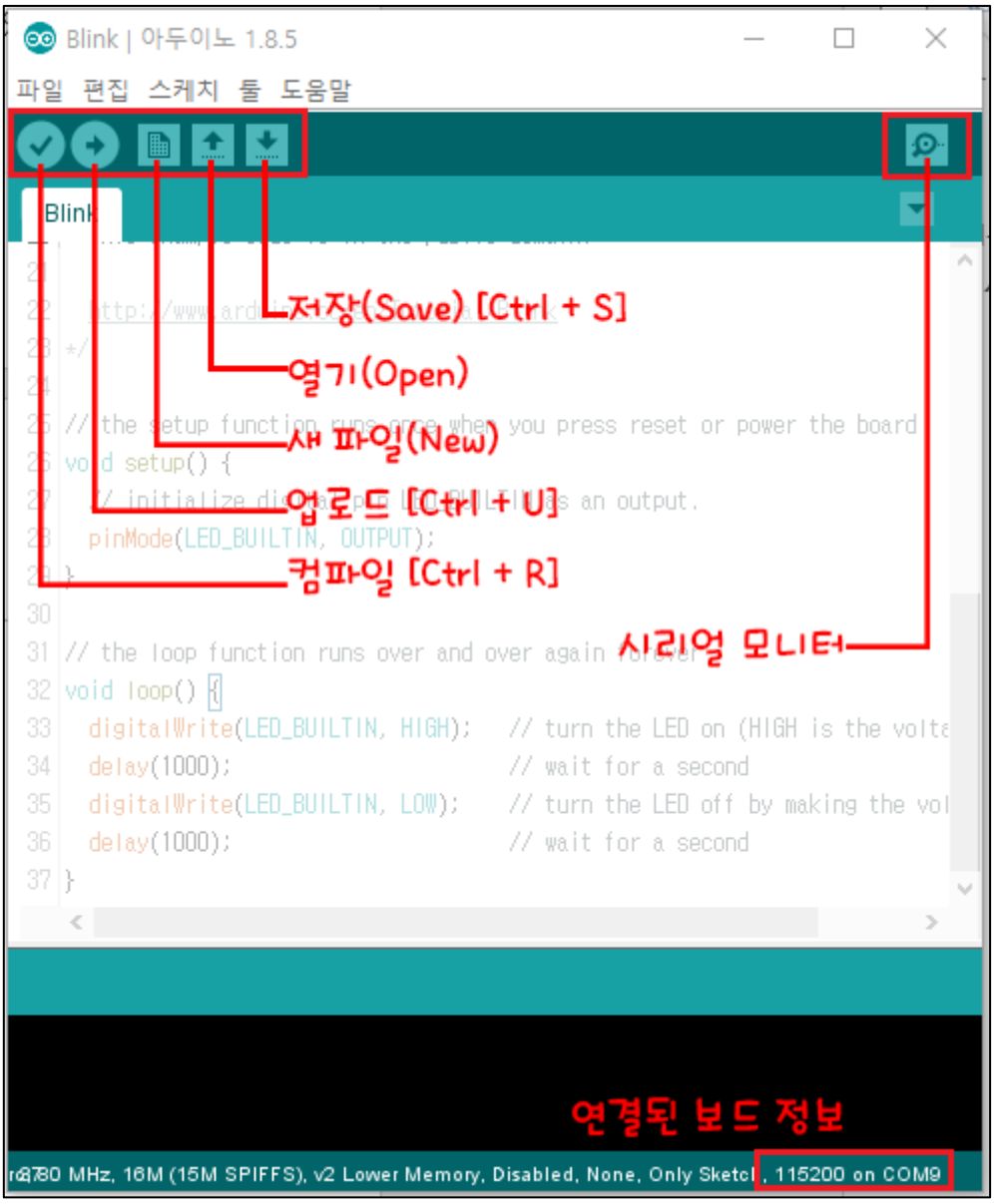

Figure 15. Arduino IDE 인터페이스 설명

아두이노 IDE의 인터페이스에 관련된 사항은 Figure 15를 통해 정확히 확인할 수 있다. 왼쪽 상단부터 각각 컴파일, 업로드, 새 파일, 열기, 저장, 시리얼 모니터, 하단에 연결된 보드 정보를 확인할 수 있다. 또한 가장 자주 사용하는 업로드와 저장의 경우 "[ ]"에 써 있는 단축키를 외워서 개발 속도향상에 도움을 받도록 한다. 이어서 Blink의 소스를 WeMos 펌웨어에 업로드 하기 위해 업로드 버튼을 누르거나 "Ctrl + U" 단축키를 사용하여 업로드를 사용해 본다. 소스의 경우 본래 컴파일을 실행하여 소스의 에러가 있는지 판단하는 것이 최우선이지만 예제소스라 에러가 발생할 일 없고, 업로드 버튼을 누르면 자동으로 1회 컴파일 후 WeMos 보드로 전송되게 설계되어 있기 때문에 해당 예제에서는 상관없다.

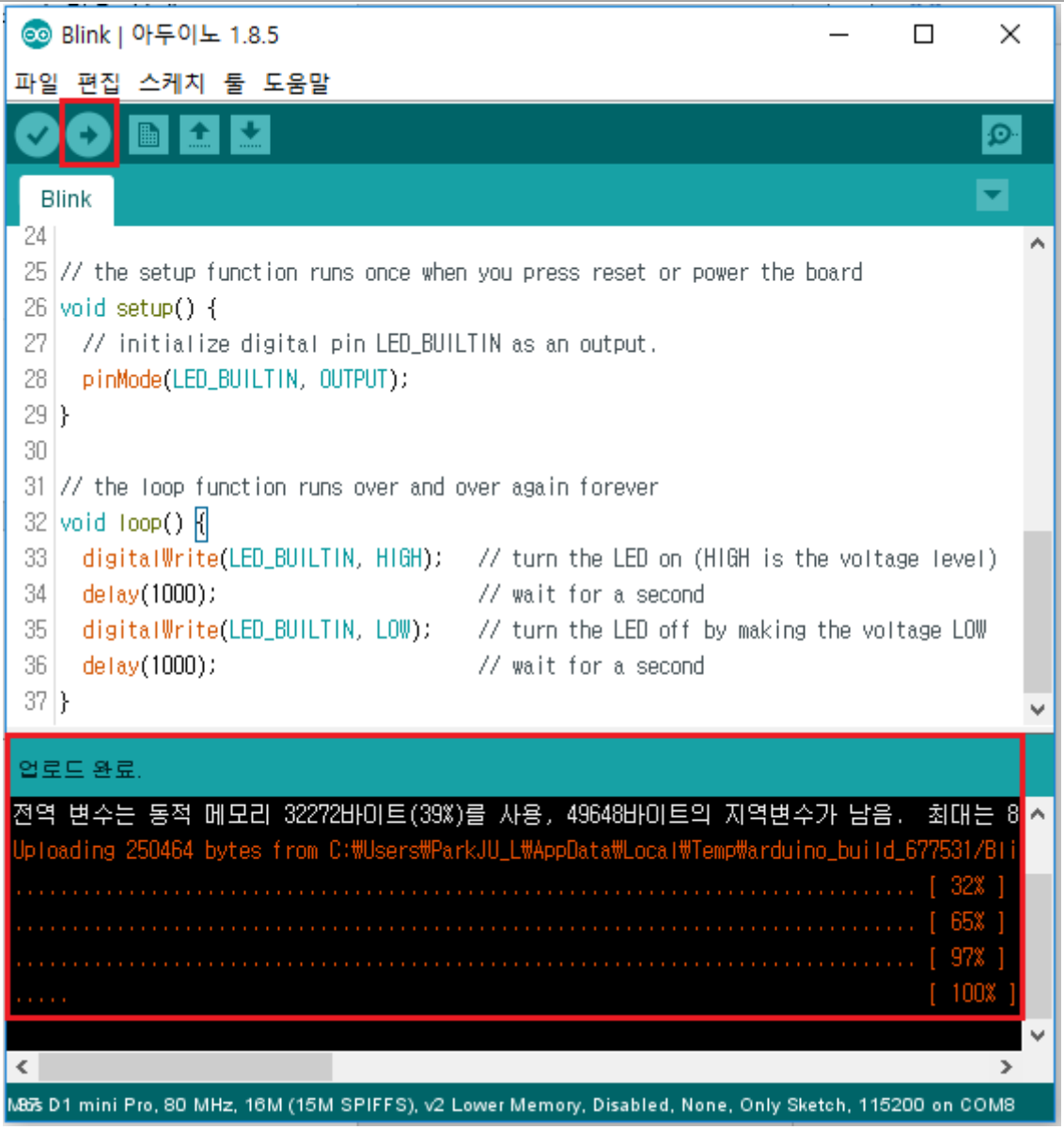

Figure 16. 업로드 완료 메시지 확인.

업로드 버튼이 눌리고 COM 포트와 정상적인 업로딩이 완료되면 아래 하단의 콘솔창에서 업로드 되고 있는 화면을 실시간으로 확인할 수 있고, 로딩 게이지가 100%가 완료되면 정상적으로 펌웨어에 소스가 올라갔다는 증거이다. 소스의 해설이 필요 없을 것 같아서 간략하게만 설명하면,

- Line 1~25: 현재 Blink 소스에 대한 주석
- Line 26, 32: 아두이노 프로그램 작성시 반드시 필요한 셋업(프로그램 시작시 실행)과 루프(셋업이 완료되고 무한대로 실행)되는 부분
- Line 28: pinMode(a, b) a핀을 OUTPUT으로 사용하도록 핀 모드를 설정
- Line 33~36: LED로 설정된 핀에 High와 Low 값을 1초 간격으로 발생시켜 깜빡이게 함

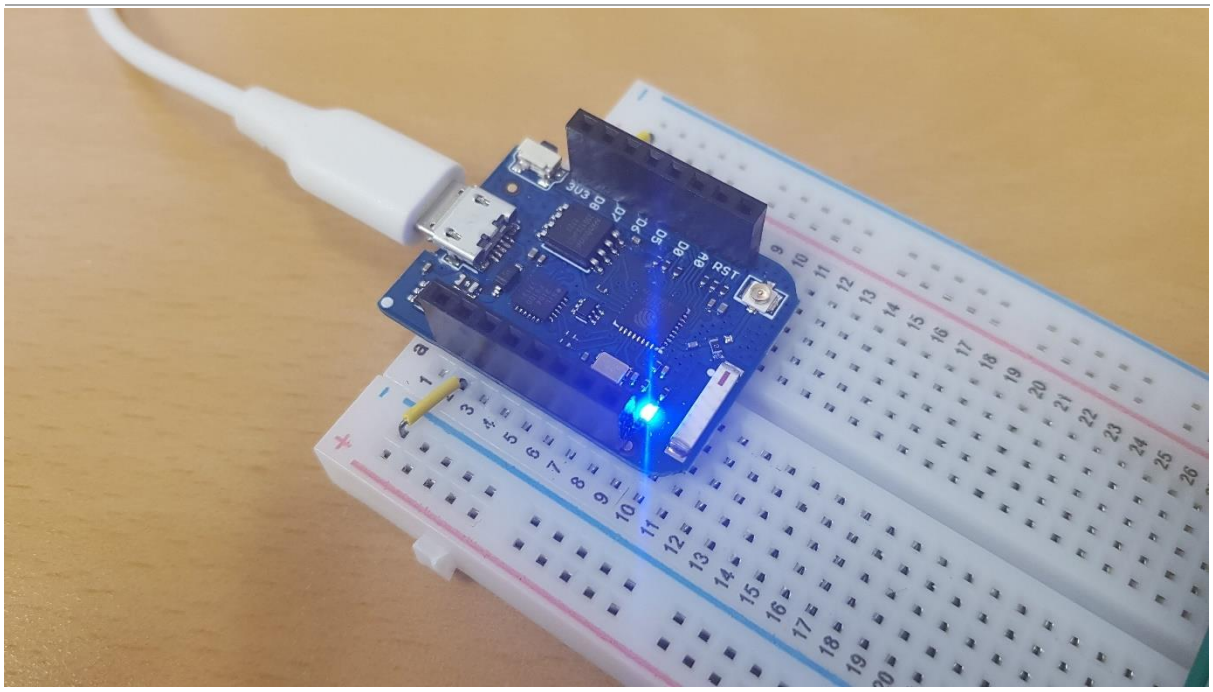

Figure 17. 예제 결과 확인

정상적인 컴파일, 정상적인 업로드가 완료 된다면 해당 Figure 17의 그림과 같이 WeMos 보드에 기본적으로 탑재되어 있는 Blue LED에 불이 들어왔다가 나갔다가 반복하면 깜빡이는 것을 확인할 수 있으며, 해당 과정까지 모두 완료 되었다면 정상적으로 WeMos를 통한 개발이 가능하다.

따라서 해당 개발 문서의 뒷부분을 완벽하게 따라하며 WeMos 개발 보드의 실습 내용을 경험해 보도록 한다.

2. WeMos 센서 및 실습 예제

1. LED(Light Emitting Diode)

가장 기본이 되는 실습은 Digital Output과 Digital Input을 사용한 센서 값의 제어이다. 가장 먼저 실습할 것은 한 개의 LED를 사용하여 Digital Output을 실습할 것이다.

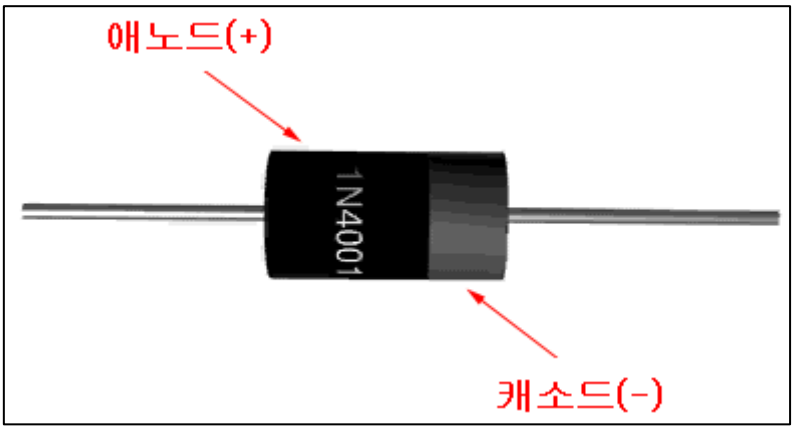

Figure 18. 다이오드 구성

기본적으로 LED를 이해하기 위해서는 다이오드(Diode)를 이해해야 한다. 다이오드는 게르마늄이나 규소(Si)로 만들어지고, 주로 한쪽 방향으로 전류가 흐르도록 제어하는 반도체 소자를 통칭한다. 보통 정류, 발광 등 특성을 지니는 반도체 소자이며 대부분의 다이오드는 P-N 접합으로 두 개의 전극을 갖는 반도체 결정체(Crystalline)이다[4]. 다이오드는 기본적으로 반도체의 P-N 접합을 바탕으로 두고 있기 때문에 P-N 다이오드에서 전류는 P형 반도체(Anode)에서 N형 반도체(Cathode)로만 흐를 수 있게 되어 있다.

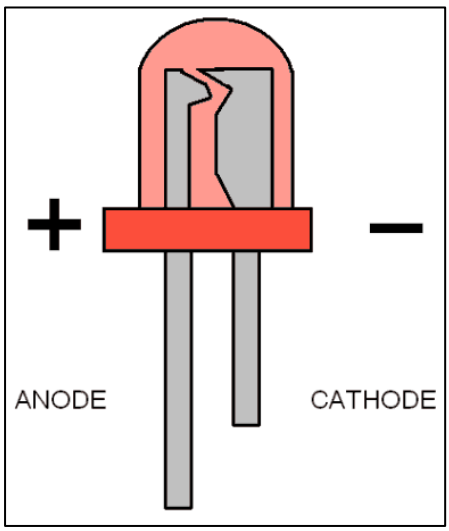

Figure 19. LED 구조

LED 역시 Light(빛을) Emitting(발하는) Diode(다이오드)라는 이름 그대로, 왼쪽 그림 Figure 19에서 볼 수 있듯 모든 전류의 방향인 +(양극)에서 –(음극)으로 전류가 이동할 때 각각 좌측의 애노드면에서 우측의 캐소드면으로 한쪽 방향으로 전류가 흐르도록 제어하며 애노드와 캐소드의 접합면에서 발광이 일어난다. 이러한 원리로 LED가 빛을 낼 수 있는 것이다.

[4] 출처: https://en.wikipedia.org/wiki/Diode

Figure 20. 설명을 돕기 위한 LED의 애노드/캐소드 접합 면 예시

이론에 대한 글과 해당 Figure 20을 통하여 이해를 더욱 더하도록 한다.

✓ 회로 구성(Schematic) 및 핀 구성(Pin)

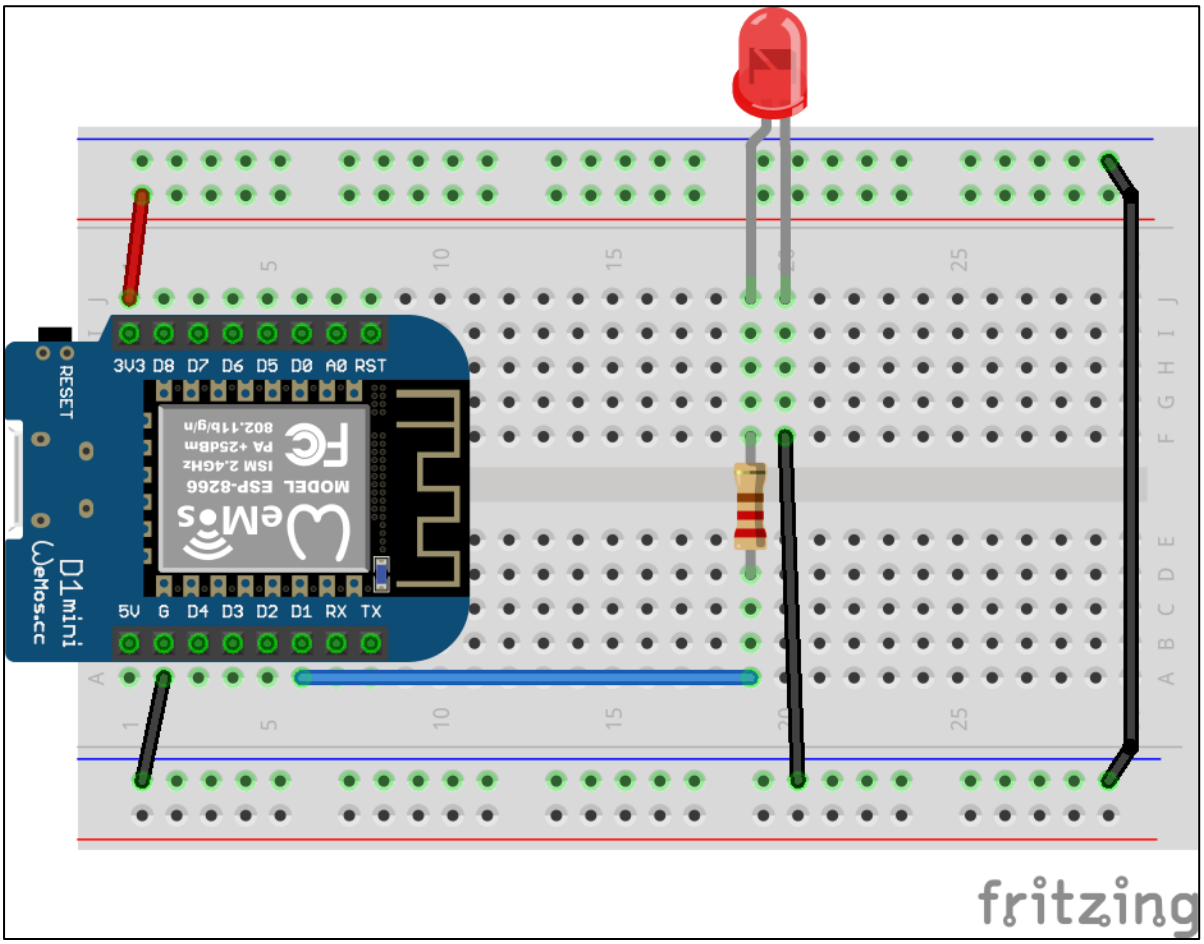

Figure 21. LED 회로 구성

Figure 21과 같이 회로를 구성한다.

Table 3. LED 핀 구성

LED	WEMOS BOARD
VCC	3V3
GND	GND

✓ **프로그램 소스(Source Code)**

01_LED.ino

```
const int PIN_LED = D1;

// 프로그램 시작시 초기화 작업
void setup() {
  pinMode(PIN_LED, OUTPUT);
}

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

✓ **결과 확인(Result)**

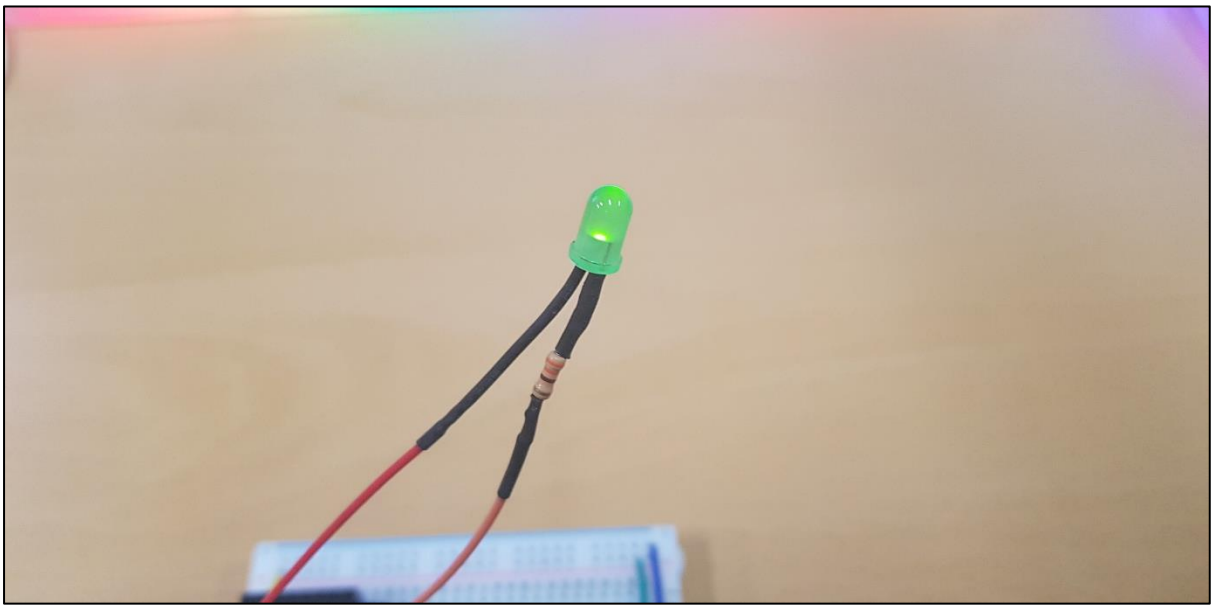

Figure 22. LED 결과

LED 결과 확인.

2. Switch

스위치(Switch) 혹은 푸쉬 버튼(Push Button)이라고 불리는 Digital Input을 수행할 우리 주변에서 접하기 쉬운 입력 제어를 실습할 것이다.

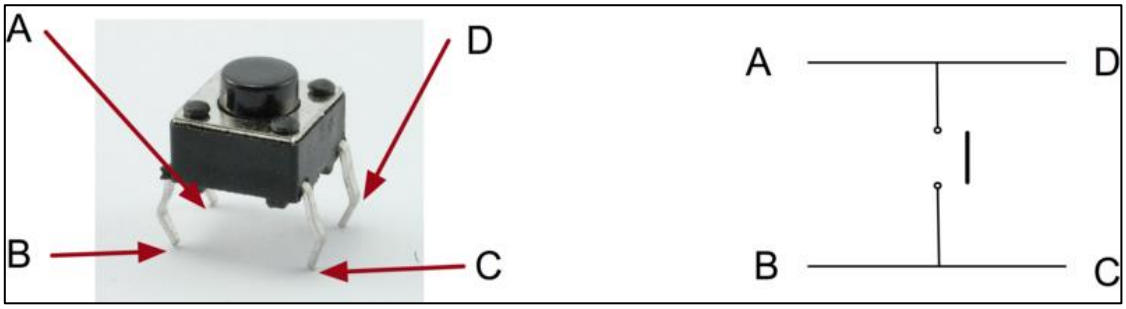

Figure 23. Switch 구조

기본적으로 스위치는 Figure 23에서 볼 수 있는 것처럼 4개의 다리를 가지고 있고, Figure 23의 우측에서 확인할 수 있는 것처럼 각각 연속한 다리의 회로가 끊어져 있다. 스위치는 상단과 하단의 회로를 브레드보드에 구성하여 버튼을 누르면 중간의 회로가 연결되고 양극에서 음극으로 전류가 흐르도록 설계되어 있다.

스위치 회로를 구성함에 앞서 중요한 개념을 알고 넘어가야 하는데 그것이 바로 "Pull-Up Switch"와 "Pull-Down Switch"이다.

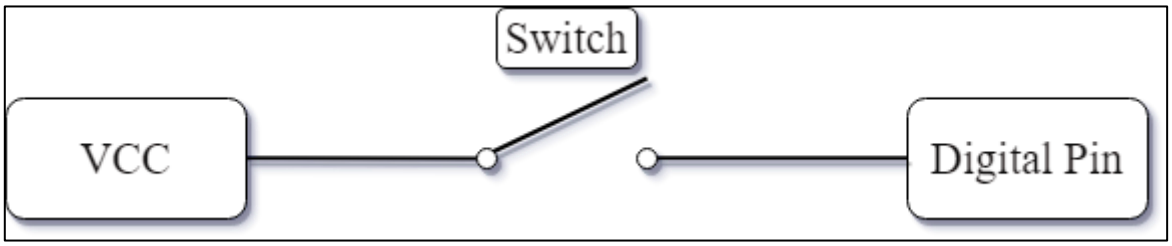

Figure 24. 부동(Floating) 상태의 이해

풀-업, 풀-다운 상태를 공부하기 전에 스위치의 기본적인 특징인 "부동(Floating)" 상태를 이해해야 한다. 부동은 사전적으로 "회로 상태가 변환되었을 때 회로 전체가 전위가 대지에 대하여 변동하도록 되어 있는 것[5]"으로 정의한다. 쉽게 Figure 24를 사용하여 설명할 수 있는데, 얼핏 보기에 Figure 24의 가운데 스위치를 붙이면 회로가 작동하여 "Digital Pin"에 "1"이라는 값이 출력 될 것 같지만, 실제로 스위치가 열린 상태에는 "0"값도 아니고, "1"값도 아닌 이른바 "Floating(부동)" 상태가 된다. 따라서 스위치가 연결되었다 하더라도 전류가 통하긴 하지만 그 값이 출력되지 않는 것이다. 이때, 저항과 회로의 구성으로 스위치의 상태를 정의해 주는 것이 바로 "Pull-Up"과 "Pull-Down" 두 가지 방법이다.

[5] 출처: https://terms.naver.com/entry.nhn?docId=1920633&cid=50324&categoryId=50324

✓ **Pull-Up(풀-업) vs Pull-Down(풀-다운)**

Table 4. Pull-Up과 Pull-Down 비교

상태	PULL-UP	PULL-DOWN
ON	0 (Low)	1 (High)
OFF	1 (High)	0 (Low)

Pull-Up 스위치와 Pull-Down 스위치는 근본적으로 ON/OFF 상태의 High/Low 값이 정반대로 동작한다는 특징이 있다. 설명은 하단을 통해 이해하도록 한다.

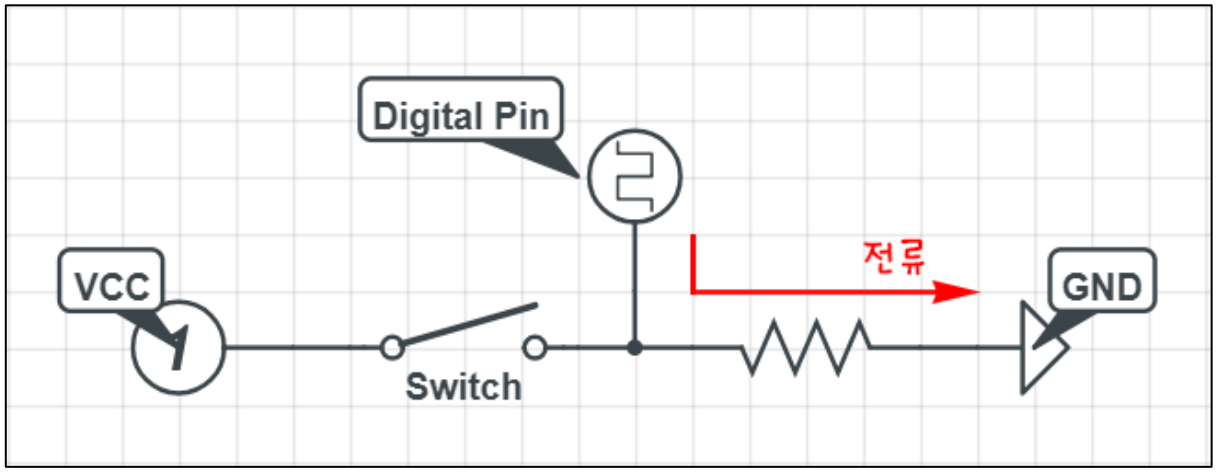

Figure 25. Pull-Down 스위치 Open

먼저 가장 많이 사용하는 Pull-Down 스위치를 살펴보겠다. Pull-Down 회로도는 "VCC(3.3V)" 쪽에 스위치(Switch)가 연결 되어있고, 반대쪽 "GND(Ground)" 쪽에 저항이 연결되어 있는 형태를 띈다. 스위치가 열려 있는 OFF(=Open) 상태일 때는 VCC로부터 전압을 전달받지 못하므로 GND에 연결된 Digital Pin의 상태는 0(Low) 상태를 가리킨다.

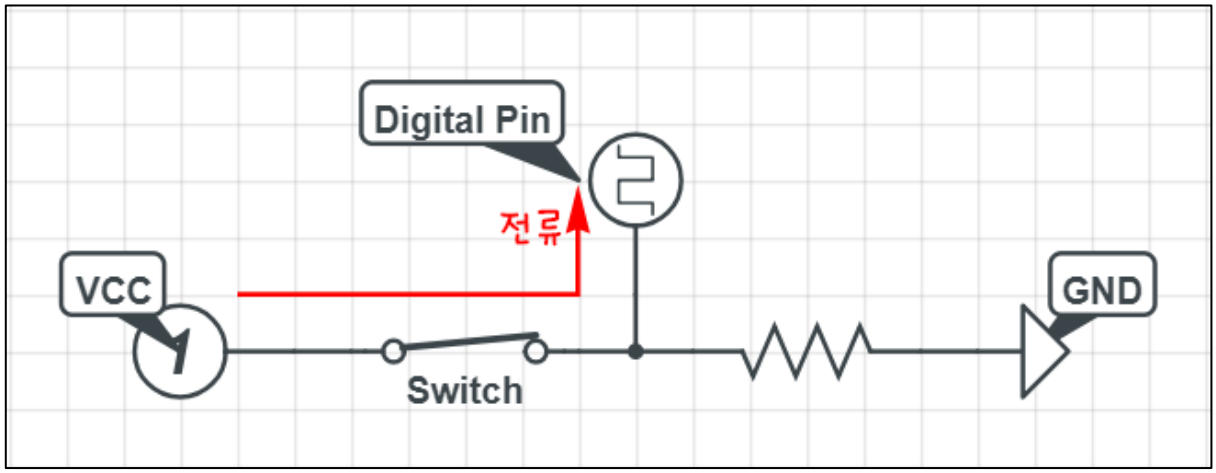

Figure 26. Pull-Down 스위치 Close

반대로 OFF(=Close) 상태일 때는 VCC로부터 접지 방향으로, Digital Pin에 전류가 흐르며 "Digital Read" 값은 "1"을 발생한다.

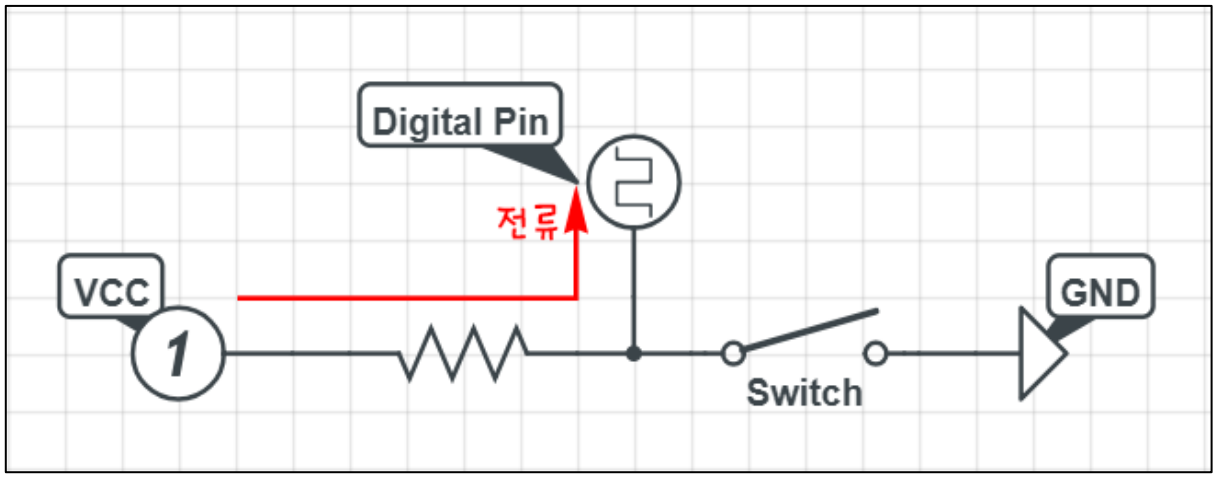

Figure 27. Pull-Up 스위치 Open

반대 경우로 Pull-Up 스위치를 살펴보면, 스위치 OFF(=Open) 상태일때는 GND 방향으로 전류가 흐르지 않고, Digital Pin을 Ground로 전류가 통해서 해당 Digital Pin에 "1" 값을 출력하게 된다. 반대로

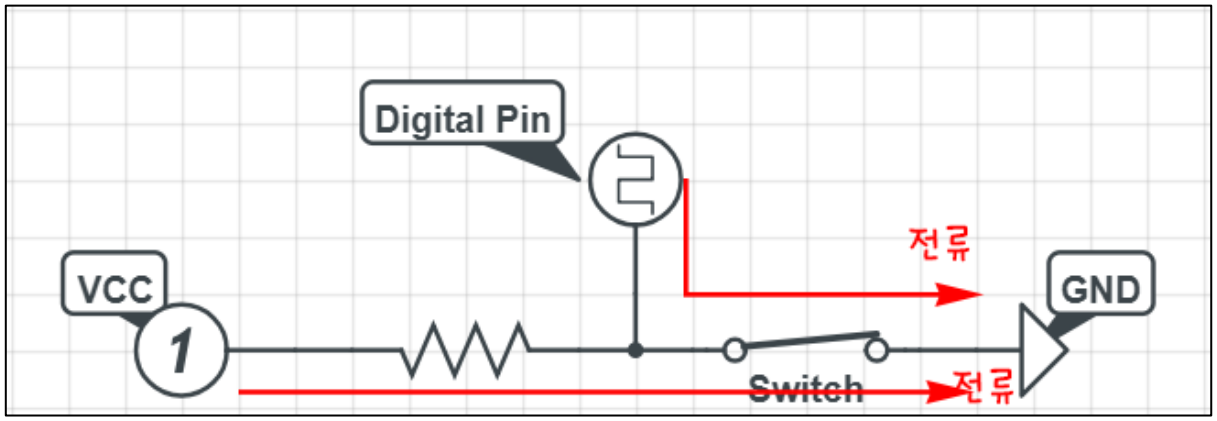

Figure 28. Pull-Up 스위치 Close

Figure 28에서 보면 스위치가 ON(=Close)되면서, VCC와 Digital Pin 모두 전류가 접지(GND) 쪽으로 통하게 되며, Digital Pin에는 전류가 흐르지 않고 "0"을 출력한다. 따라서 스위치 관련된 회로를 구성 할 때 해당 개념들 (플로팅, Pull-Up, Pull-Down)을 잘 기억하여 회로를 작성하도록 하고, 가장 직관적으로 "누른다"와 "ON"이 바로 생각 날 수 있는 "Pull-Down" 회로도를 기초로 실험을 하도록 한다.

✓ **회로 구성(Schematic) 및 핀 구성(Pin)**

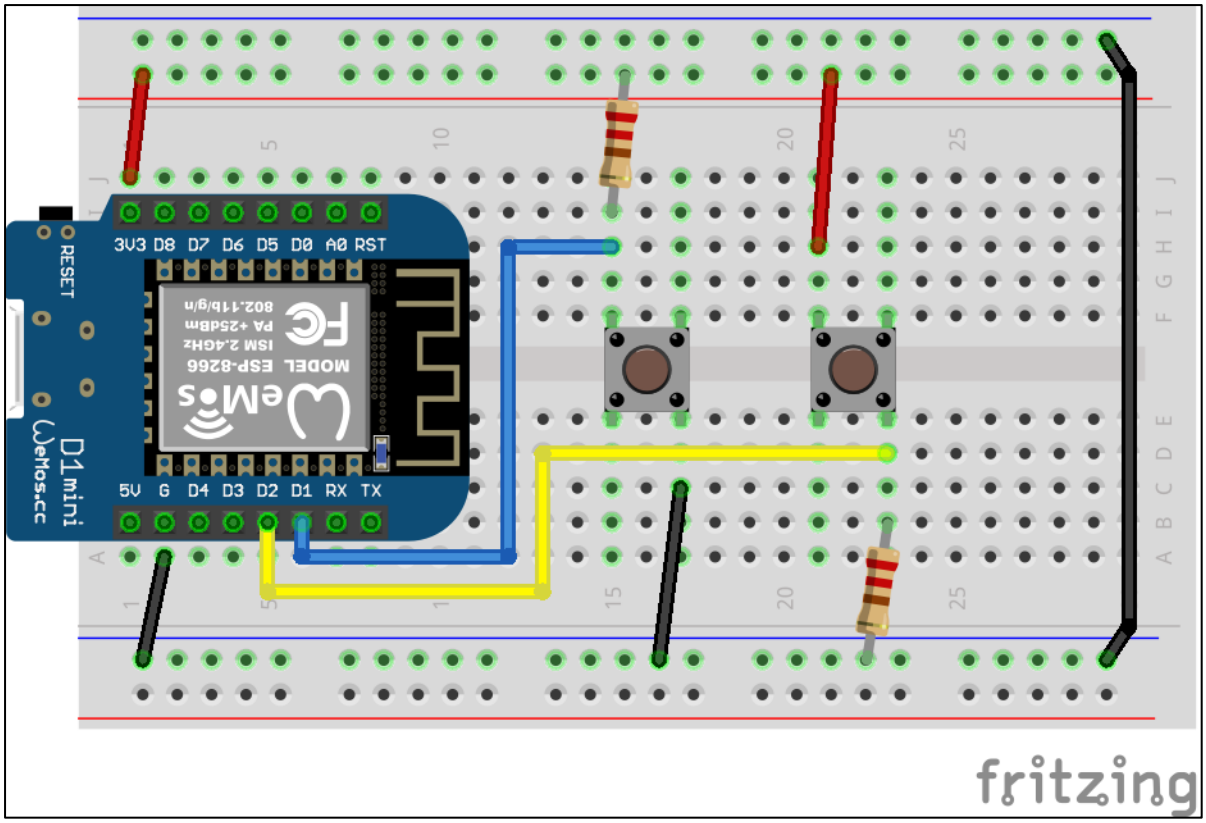

Figure 29. Switch 회로 구성

회로를 Figure 29와 같이 구성한다. 각각 왼쪽의 버튼은 Pull-Up 구성, 오른쪽의 버튼은 Pull-Down 구성으로 만든 스위치이다. 소스 작성 후, 테스트 과정에서 그 차이점을 확인하도록 한다.

Table 5. Pull-Up Switch 핀 구성

PULL-UP SWITCH	WEMOS BOARD
VCC	3.3V
GND	GND
DIGITAL INPUT	D1

Table 6. Pull-Down Switch 핀 구성

PULL-DOWN SWITCH	WEMOS BOARD
VCC	3.3V
GND	GND
DIGITAL INPUT	D2

각각 핀구성은 Table 5, 6을 참고한다.

✓ **프로그램 소스(Source Code)**

02_Switch.ino

```
const int Switch_1 = D1;
const int Switch_2 = D2;

void setup()
{
  Serial.begin(115200);

  pinMode(Switch_1, INPUT);
  pinMode(Switch_2, INPUT);
}

void loop()
{
  Serial.print("Pull-Up Switch :\t");
  Serial.println( digitalRead(Switch_1) );

  Serial.print("Pull-Down Switch :\t");
  Serial.println( digitalRead(Switch_2) );

  Serial.println("---------------------------------");
  delay(1000);
  }
```

프로그램 소스.

✓ **결과 확인(Result)**

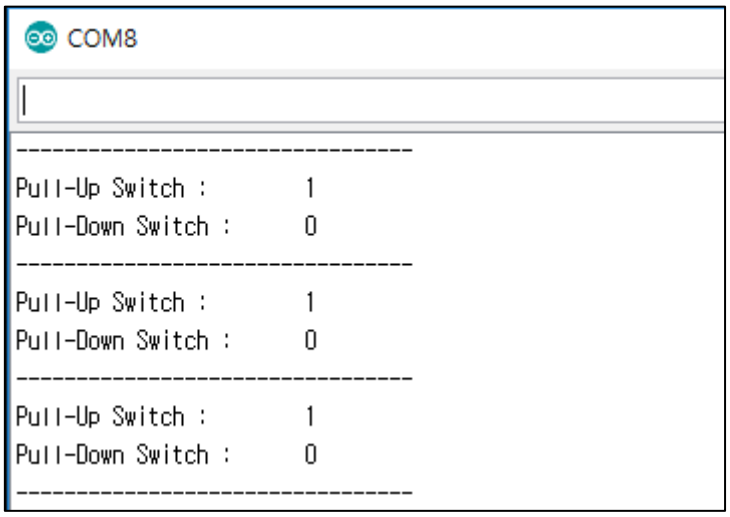

Figure 30. Switch 결과

시리얼 모니터를 통해 결과를 확인하며 스위치를 작동시키고 변화를 확인한다.

3.