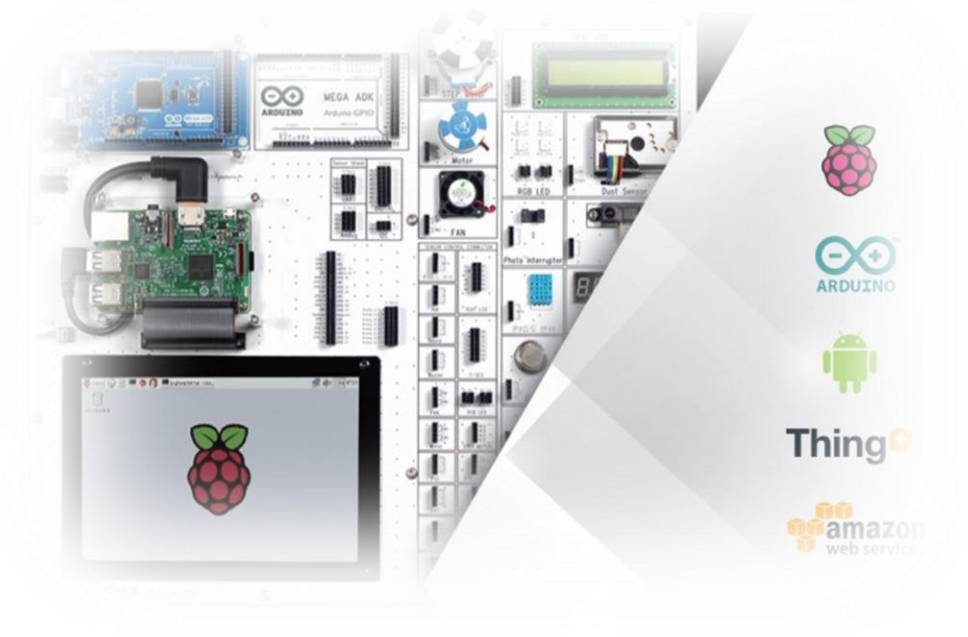
2021년도 2학기
휴먼스마트기기설계
강의자료 #2



교과목명 : 휴먼스마트기기설계 (01)

담당교수 : 이 수 형

E-mail : soohyong@uu.ac.kr

교재명 : 스마트기기 개발을 위한 사물인터넷, 이수형. (LINC+ 사업단 배포)

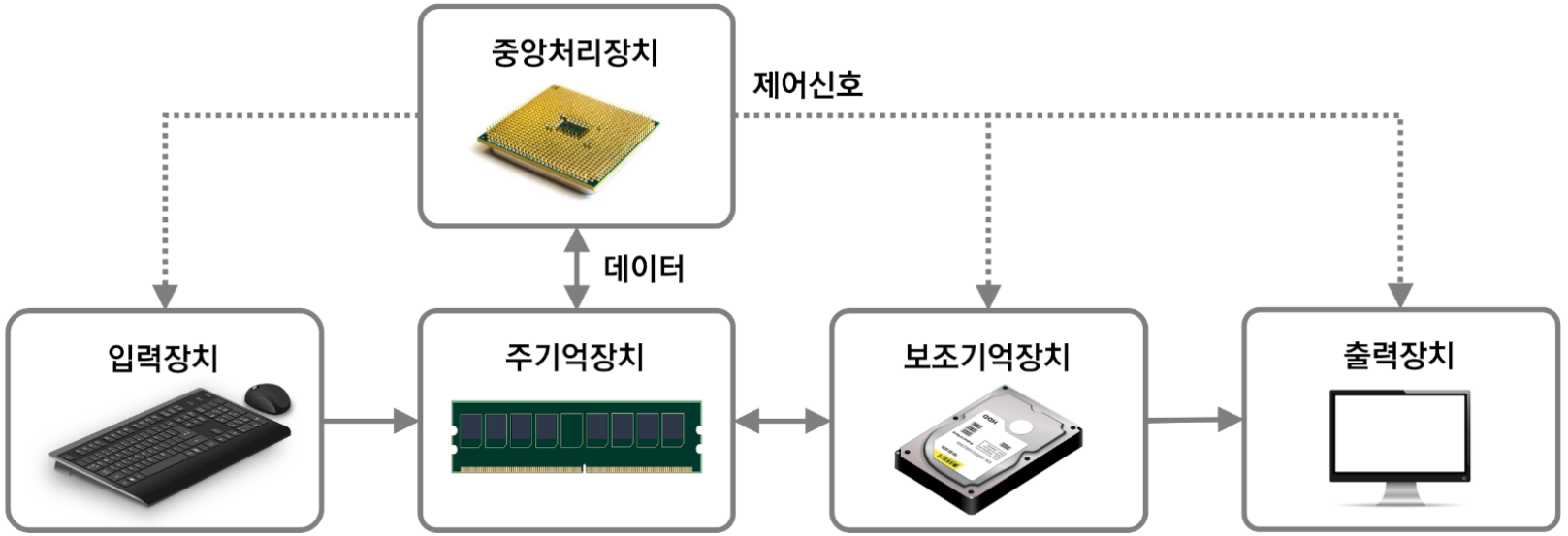
2. 사물인터넷용 제어 보드

단일보드 컴퓨터

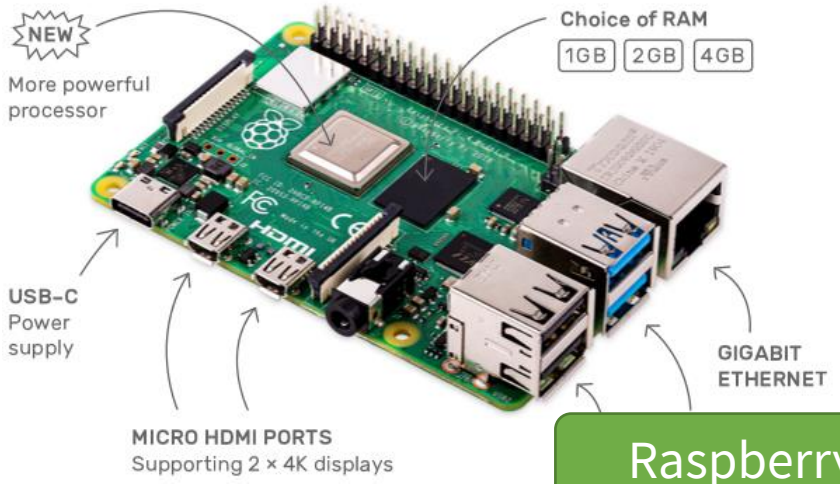
- 단일보드 컴퓨터 (SBC : Single Board Computer)
  - 하나의 회로기판에 마이크로프로세서, 메모리, 입출력 장치, 기타 기능들을 넣어 놓은 컴퓨터
  - 개발, 교육, 임베디드 시스템용으로 제작
  - 소형의 저전력 컴퓨터로 사용
  - 외부 장치 (센서)와는 직접적으로 연결 (데스크탑 : 확장 슬롯 등을 사용)

단일보드 컴퓨터

- 컴퓨터 시스템의 5대 구성 요소
  - 중앙처리장치(CPU: Central Processing Unit) : 인간의 두뇌, 연산/제어 기능
  - 입력장치 : 키보드/마우스 등과 같이 외부정보를 읽어들이는 장치.
  - 출력장치 : 모니터/프린터 등과 같이 정보를 보여주는 장치.
  - 주기억장치 : 컴퓨터가 수행중일때 사용하는 기억장치로 주로 RAM을 의미.
  - 보조기억장치 : HDD 등과 같이 컴퓨터의 전원이 없는 경우에도 저장 가능.

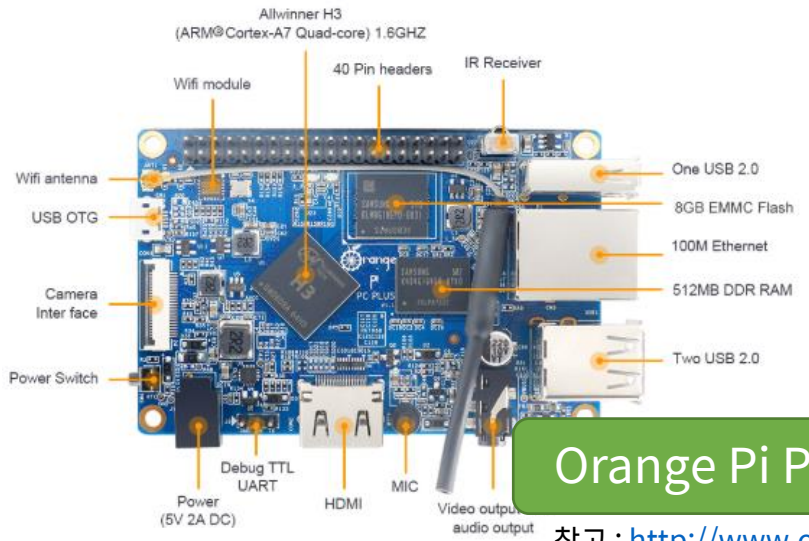


단일보드 컴퓨터의 예



Raspberry PI 4

참고 : https://www.raspberrypi.org/

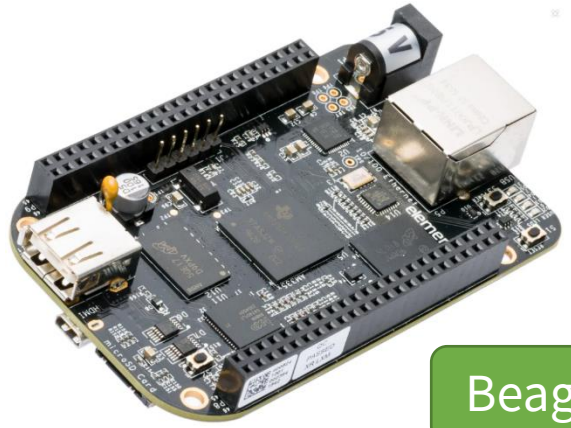


Orange Pi Pc Plus

참고 : http://www.orangepi.org/

ODROID-C2

참고 : https://www.hardkernel.com/ko/



BeagleBone Black

참고 : https://www.arrow.com/en/products/bb-black/beagleboardorg/

단일보드 마이크로컨트롤러

- 단일보드 마이크로컨트롤러
  - 단일보드 컴퓨터에 비해서 저사양, 소형화, 저가격
  - 단일보드 컴퓨터 : CPU 사용
    - 연산 및 제어회로만 가지고 있음
    - 예] Raspberry pi : ARM 기반의 CPU 사용
    - 외부 기억장치, 보조기억장치 등을 추가하여 사용
  - 단일보드 마이크로컨트롤러 : MPU
    - 연산/제어회로 + 기억장치(RAM) + 보조기억장치(Flash RAM) + 입/출력 포트
    - 예] 아두이노 : Atmel 사의 ATmega328 사용
  - 대표적인 제품 : 아두이노

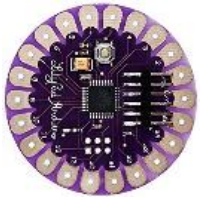
단일보드 마이크로컨트롤러 : 아두이노

- 아두이노란?
  - 아두이노(arduino)는 이탈리아의 IDII(Interaction Design Institutelvera)에서 2005년에 만든 초보자를 위한 하드웨어(hardware) 제작용 마이크로컨트롤러(microcontroller) 보드
  - 비전공자 및 초보자도 쉽게 하드웨어에 접근
  - 하드웨어 및 소프트웨어를 오픈소스로 공개 → 확산
  - 간단하게 센서, 스위치 등으로 제어 가능
  - C/C++ 기반 개발환경 제공
- 다양한 센서모듈
- 최근 모델
  - Nano 33 IoT, Nano BLE 등 사물인터넷용으로 개발

단일보드 마이크로컨트롤러 : 아두이노

	Uno	Leonardo	Due	Mega2560
외관				
용도	대표적인 보드	키보드/마우스 장치로 구동	32비트 MPU	Uno의 확장 및 대용량
MPU	ATmega328	ATmega32u4	AT91SAM3X8E	ATMega2560
동작전압	5V	5V	3.3V	5V
디지털 I/O핀	14	20	54	54
PWM핀	6	12	12	16
아날로그 입력	6	12	12	16
아날로그 출력	-	-	2	-
SRAM	2KB	2.5KB	96KB	8KB
Flash Memory	32KB	32KB	512KB	256KB
EEPROM	1KB	1KB	-	4KB
공급전원	USB/외부전원	USB/외부전원	USB/외부전원	USB/외부전원

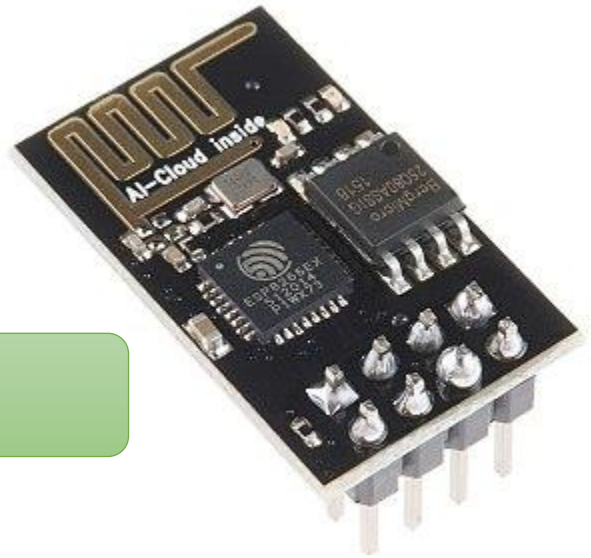
단일보드 마이크로컨트롤러 : 아두이노

	Nano	Micro	Pro Mini	LilyPad
외관				
용도	Uno의 소형 버전	Leonardo의 소형 버전	가장 작은 크기 실제 제품 제작용	옷에 부착가능/ 웨어러블 버전
MPU	ATmega328	ATmega32u4	ATmega168	ATMega168V
동작전압	5V	5V	3.3V or 5V	2.7V~5.5V
디지털 I/O핀	14	10	14	9
PWM핀	6	7	6	16
아날로그 입력	6	12	8	16
아날로그 출력	-	-	-	-
SRAM	2KB	2.5KB	2KB	1KB
Flash Memory	32KB	32KB	32KB	32KB
EEPROM	1KB	1KB	1KB	512B
공급전원	USB/외부전원	USB/외부전원	보드 직접연결	USB/외부전원

단일보드 마이크로컨트롤러 : ESP8266

- ESP8266 series
  - ESP8266 chip : Espressif사의 TCP/IP의 풀 스택의 무선인터넷 기능을 처리하는 마이크로컨트롤러
  - Ai-Thinker사에서 ESP8266 칩을 사용하는 ESP-01 모듈 출시
    ⇒ 아두이노 등과 연결 (시리얼 통신) 하여 무선인터넷 기능을 수행함
  - 저렴한 가격으로 빠른 속도로 성장함 (개당 1~2달러)

ESP-01



참고 : https://en.wikipedia.org/wiki/ESP8266

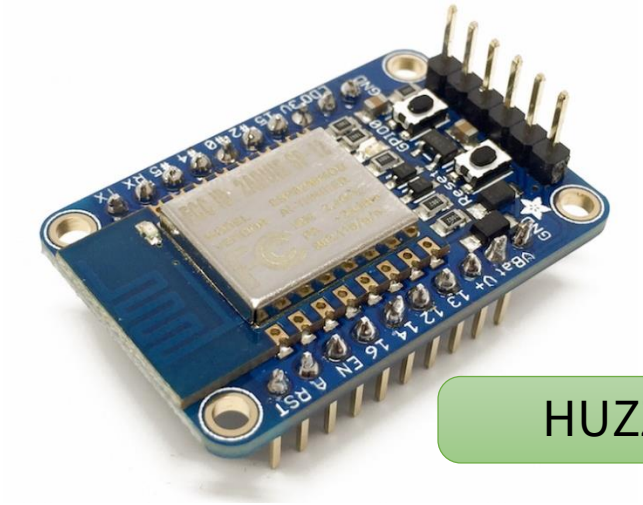
단일보드 마이크로컨트롤러 : ESP8266

- ESP8266 series
  - ESP-01 모듈 → ESP-01 외에 ESP-01S, ESP-01M, ESP-02, ESP-03, ESP-04, ESP-05, ESP-06, ESP-07, ESP-07S, ESP-08, ESP-09, ESP-10, ESP-11, ESP-12, ESP-12E, ESP-12F, ESP-13, ESP-14 모듈로 개선

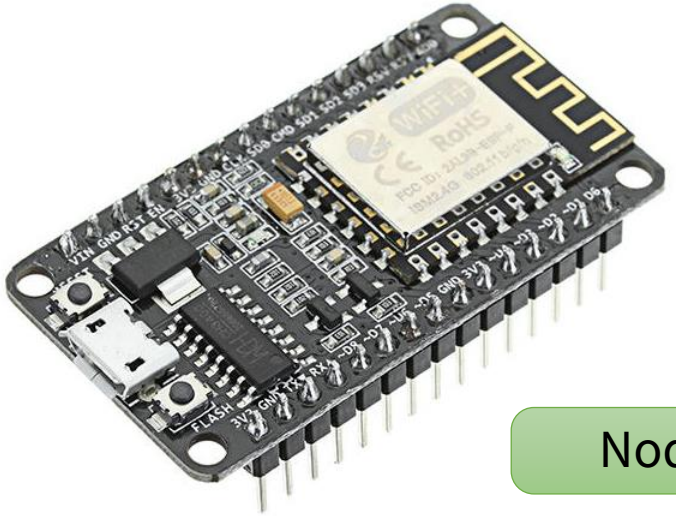
- 대부분의 모듈이 회로기판에 직접 납땜 등을 통하여 부착해야 함
- 전원장치 등의 추가 회로 필요
- 제품 제작시에는 유용하지만 실습시에는 불편함

단일보드 마이크로컨트롤러 : ESP8266

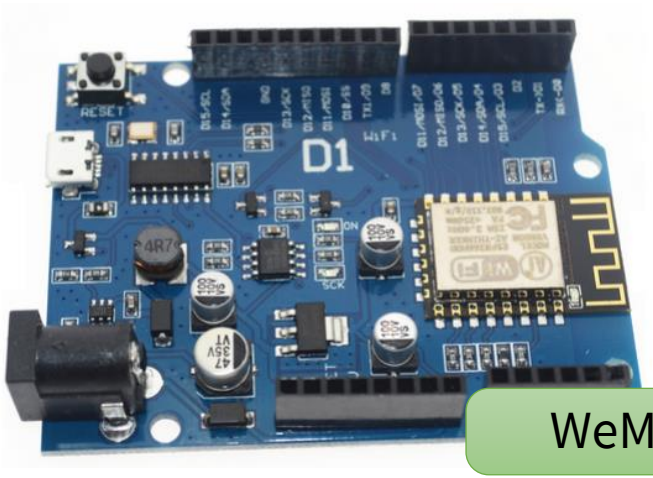
- 실습 및 개발용 보드 : ESP-12E 모듈 + 전원/USB + 알파



HUZZAH



NodeMCU



WeMos D1

WeMos D1 mini

3. 개발 환경 구축

NodeMCU

- ESP-12E/F 모듈을 사용한 단일보드 마이크로컨트롤러 보드
- ESP8266의 사양
  - 프로세서 : Tensilica 32비트 RISC CPU Xtensa LX106
  - 구동전압 : 3.3V
  - 구동 주파수 : 80~160MHz
  - 소모전류 : 10uA ~ 170mA
  - 메모리 (SRAM) : 32KB + 80KB
  - 플래시 메모리 : 최대 16MB까지 장착 가능
  - 외부 입/출력 핀 : 17 (다른 기능과 중복되어 사용함)
  - 아날로그 입력 : 1 (10비트 해상도)
  - 무선네트워크 : 802.11 b/g/n
  - 최대 TCP/IP 접속 : 5개

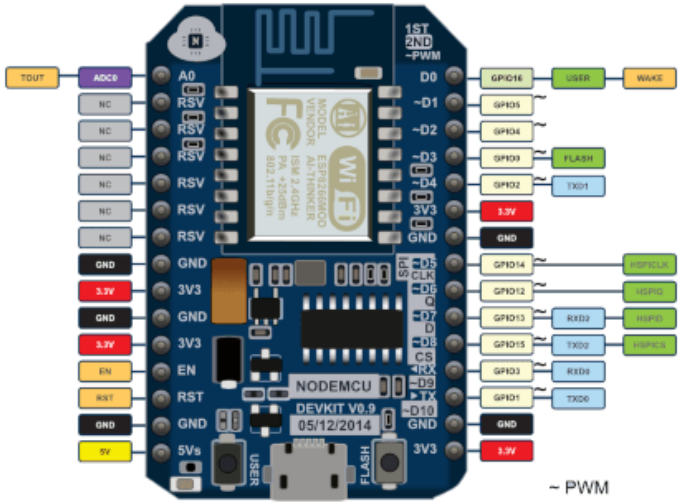
NodeMCU

- 구동 주파수 : 80MHz
- SRAM : 64KB
- 플래시 메모리 : 4MB
- UART : 1 (시리얼 통신용 포트)
- SPI : 1 (디지털 통신용)
- I2C : 1 (디지털 통신용)

NodeMCU 종류

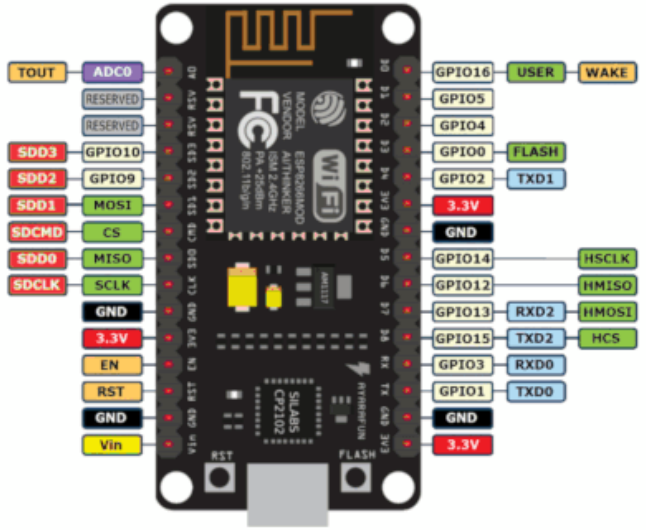
- 3가지 버전이 존재

NodeMCU Devkit V0.9
aka NodeMCU V1



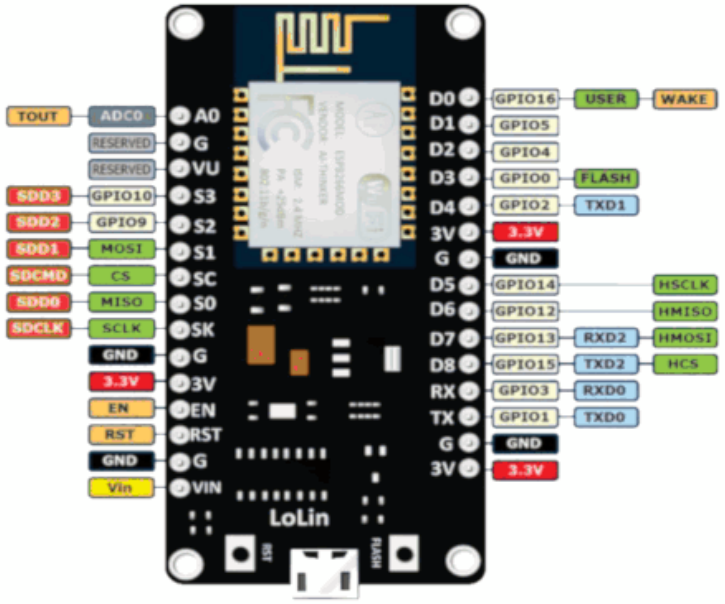
- 초기버전

NodeMCU Devkit V1.0
aka NodeMCU V2



- USB칩 : CP2102
- 브레드 보드에서 사용 가능

Lolin NodeMCU
aka "NodeMCU V3"

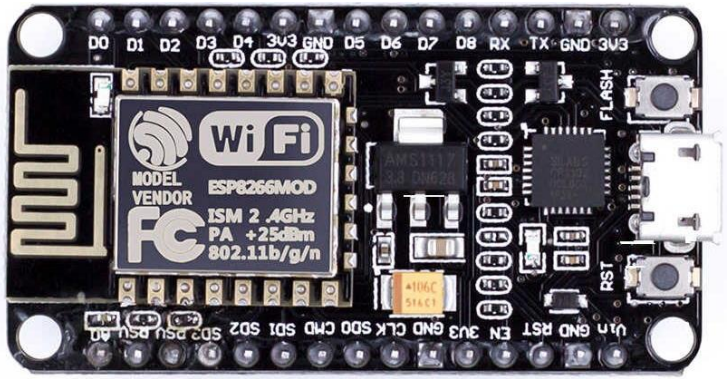


- USB칩 : CH340G
- 보드의 폭이 커서 일반적인 브레드 보드에서 사용이 불가능

NodeMCU 비교

NA225

CP2102

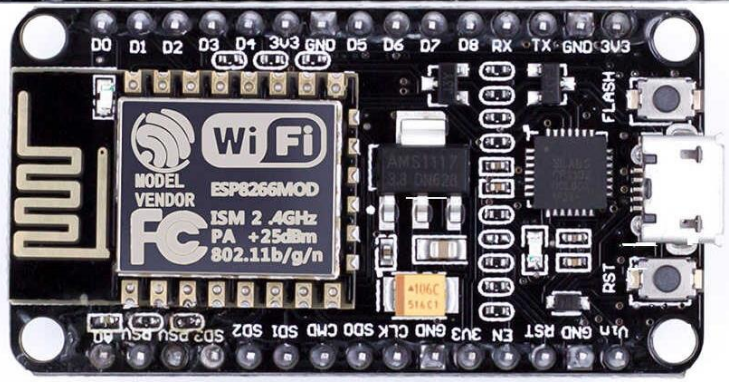


NA226

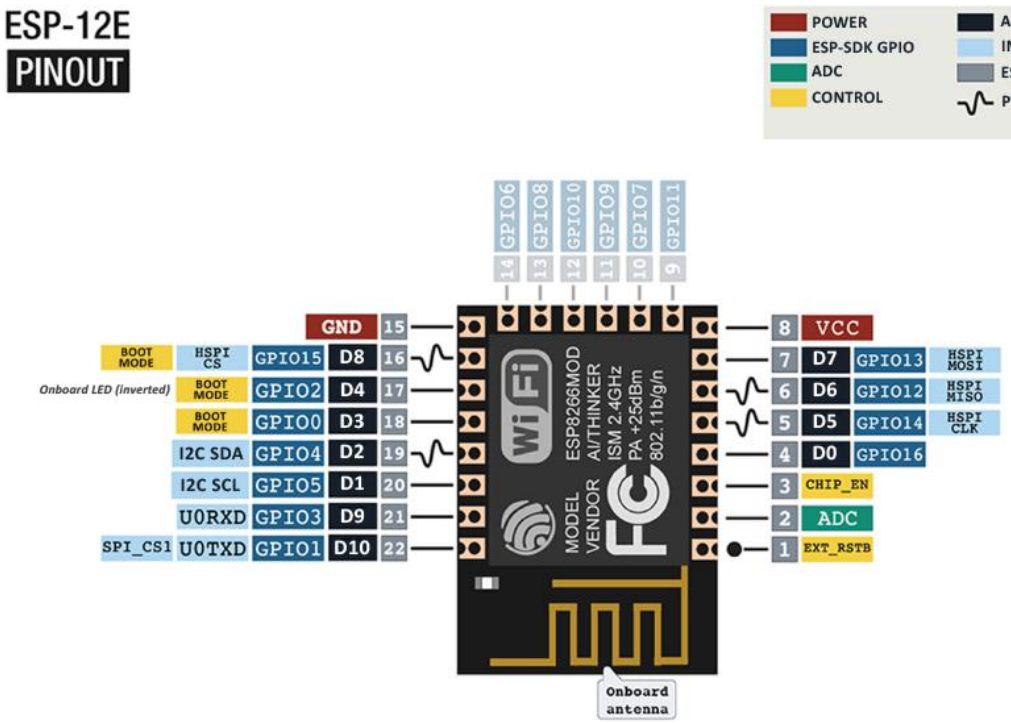
CH340

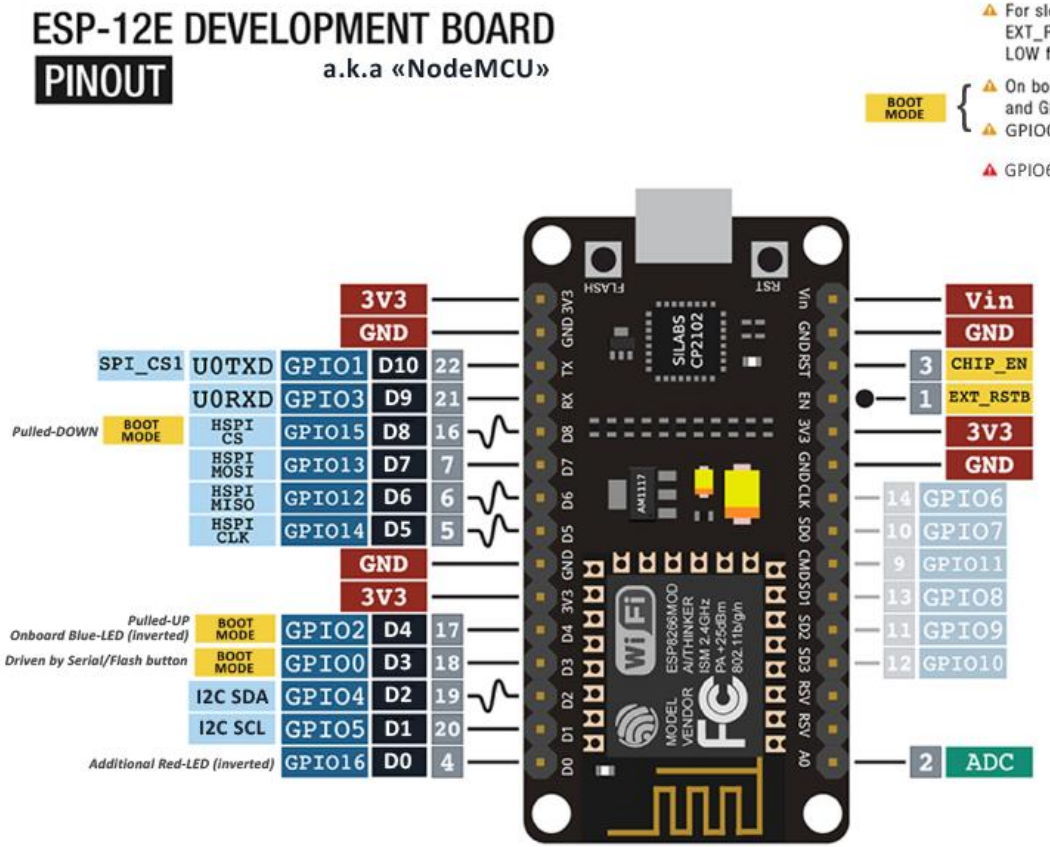
NA225

CP2102



NodeMCU pin map





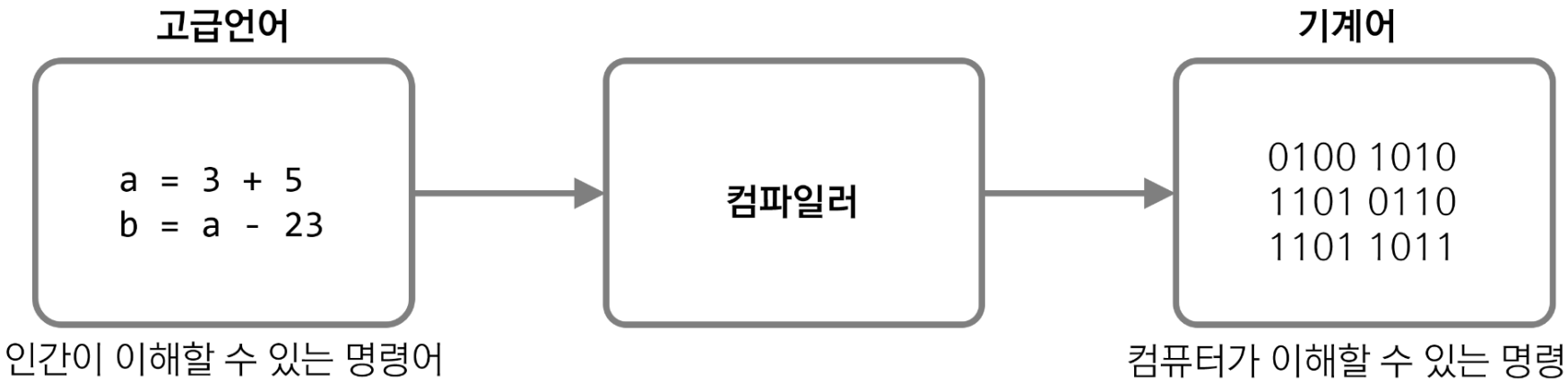
GPIO (General Purpose Input/Output)

개발환경

- ESP8266 개발 방법
  - Espressif사에서 제공한 SDK를 사용
    - C/C++ 언어를 이용하여 개발
    - 가장 기본이 되는 개발 방법
  - Sming framework
    - SDK를 쓰기 쉽게 만들어놓은 개발 환경, 아두이노와 비슷한 스타일로 작성
  - Arduino IDE (Integrated Development Environment) 활용
    - C/C++기반의 간단한 구조
    - 아두이노와 동일한 방법으로 개발, 아두이노 개발 경험이 있으면 쉽게 적응 가능
  - NodeMCU : Lua
    - Lua 스크립트를 사용하는 방법
  - 기타
    - Javascript, Python, BASIC 등의 언어로도 개발이 가능

프로그래밍

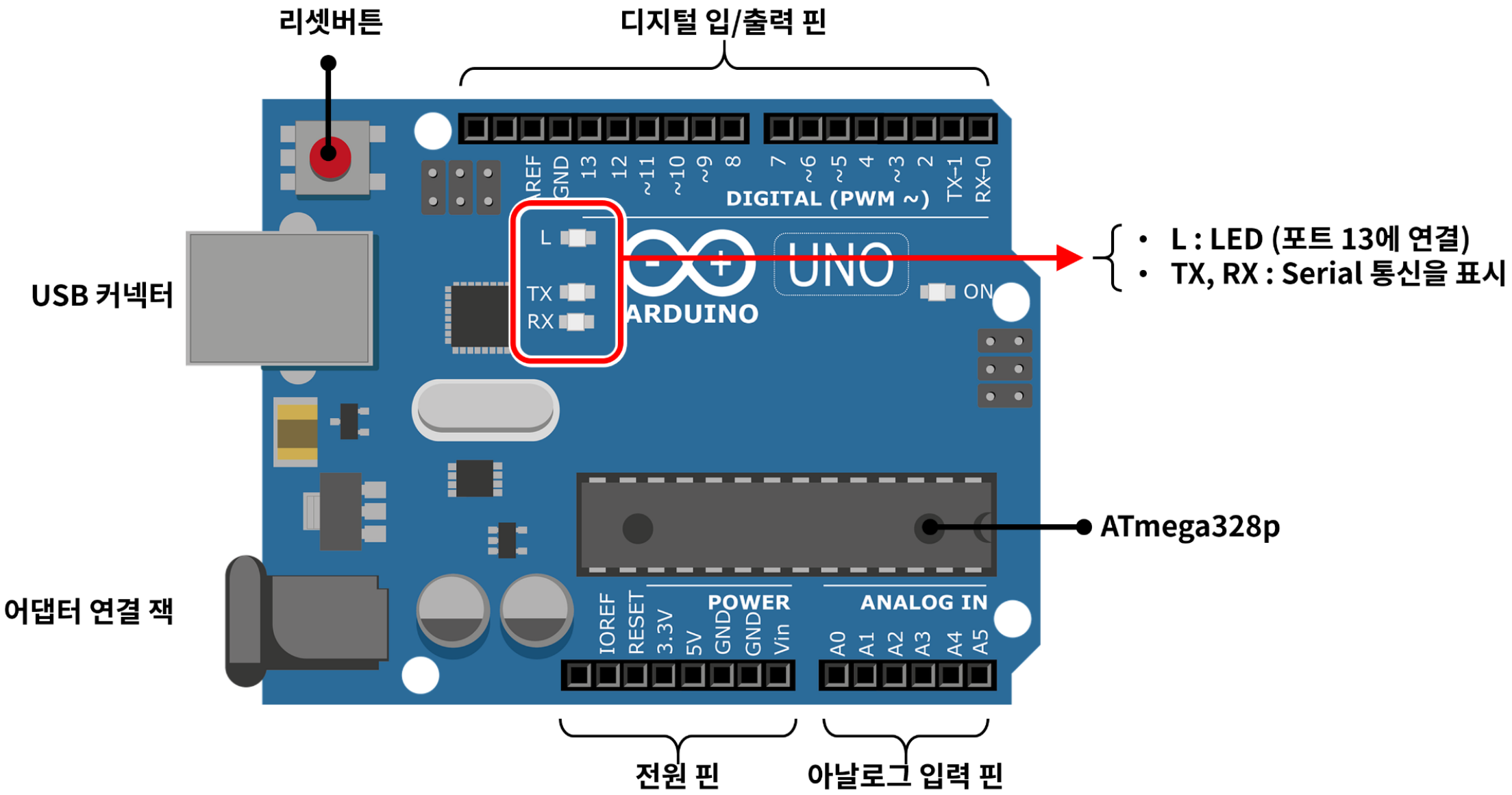
- 프로그래밍
  - 프로그램 : 업무수행을 위한 계획 (컴퓨터가 작동하는 순서를 미리 만들어놓은것)
  - 프로그래밍 : 프로그램을 작성하는 것
- 컴퓨터 프로그램
  - 컴퓨터 : 스위치의 제어로 조작하는 전기기기
  - 복잡한 기능 : 많은 스위치 필요 → 기계어 (machine language) : 이해하기 어려움
  - 인간이 이해하기 쉬운 언어 : 고급 언어 (high level language)



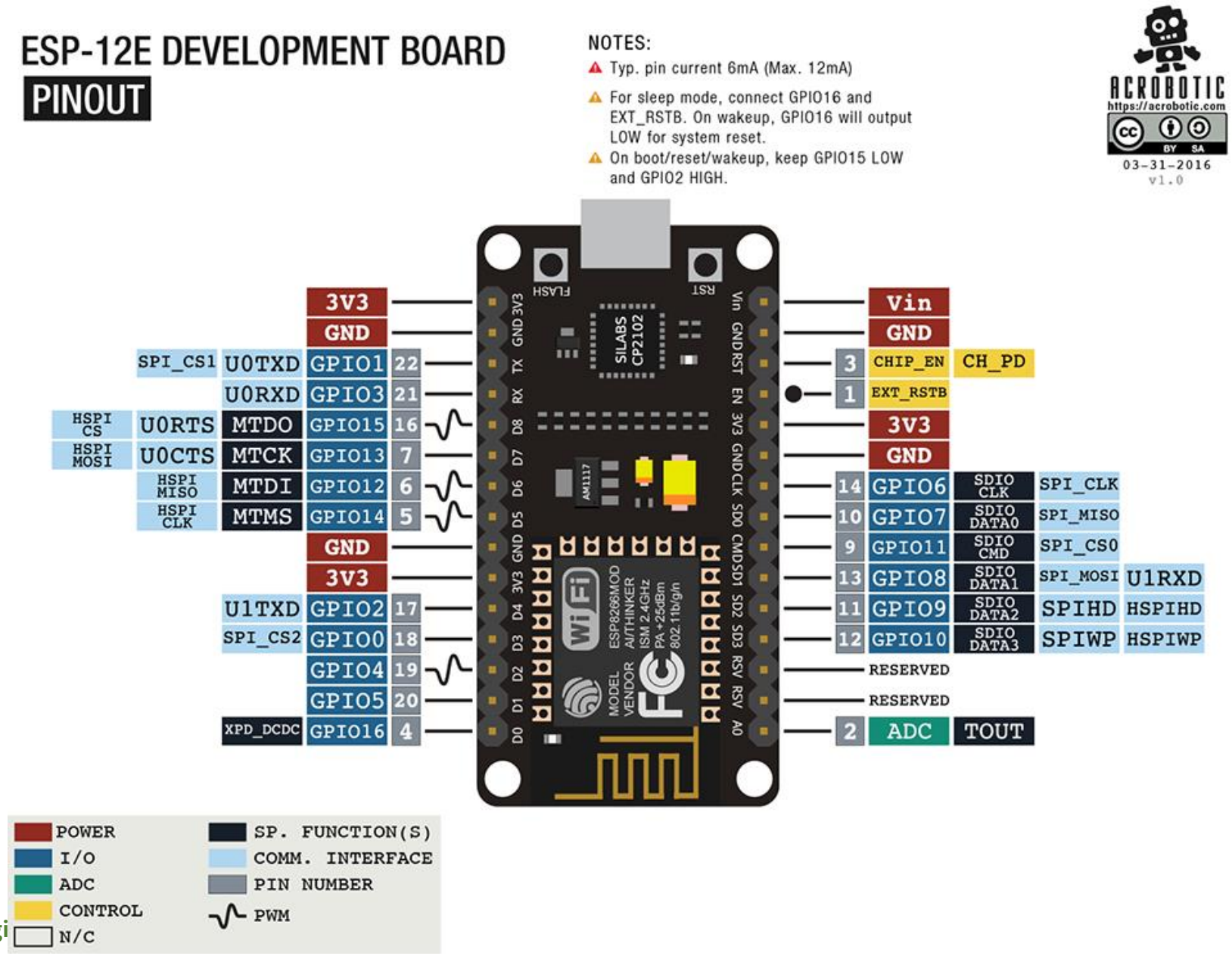
실습에 사용되는 키트

- 아두이노 키트 사용 + NodeMCU + 추가 부품

아두이노 UNO

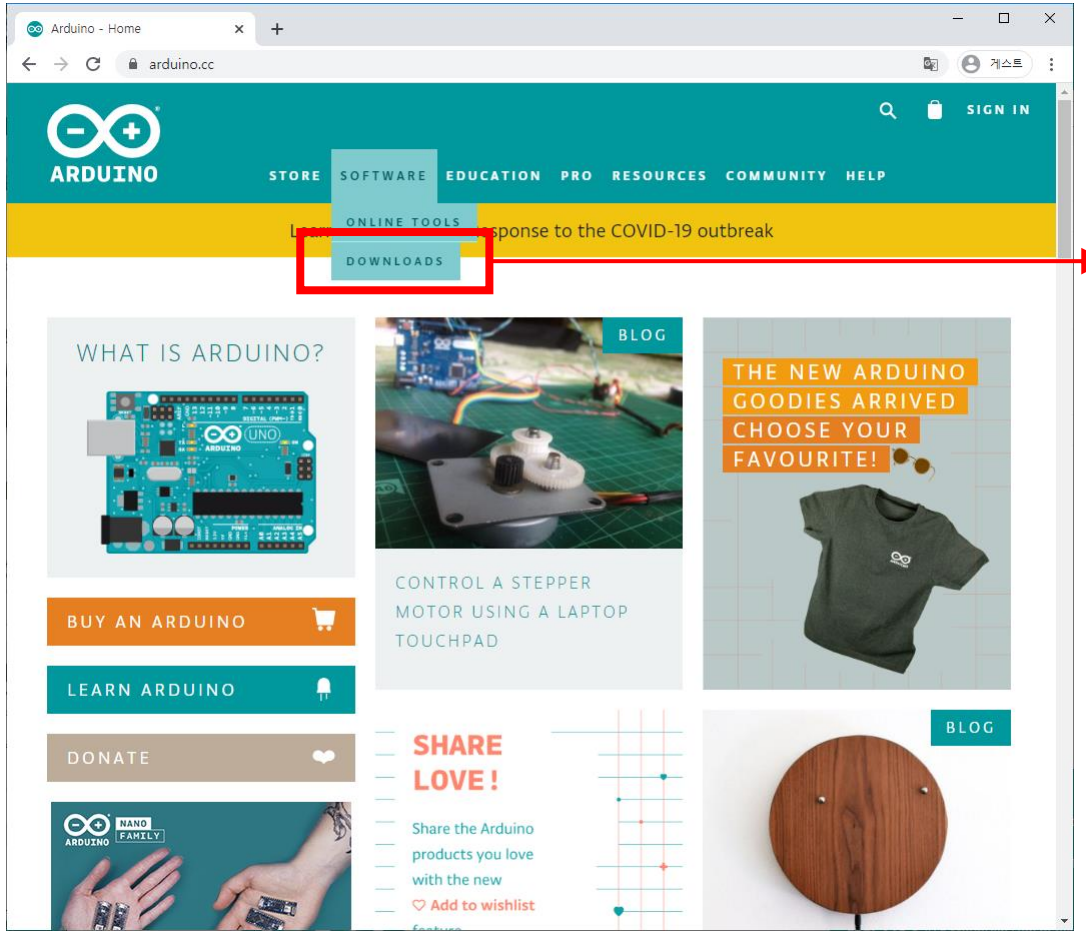

리셋버튼
디지털 입/출력 핀
USB 커넥터
어댑터 연결 잭
• L : LED (포트 13에 연결)
• TX, RX : Serial 통신을 표시
ATmega328p
전원 핀
아날로그 입력 핀
DIGITAL (PWM ~)
POWER
ANALOG IN
UNO
ARDUINO
ON
L
TX
RX

NodeMCU pin map



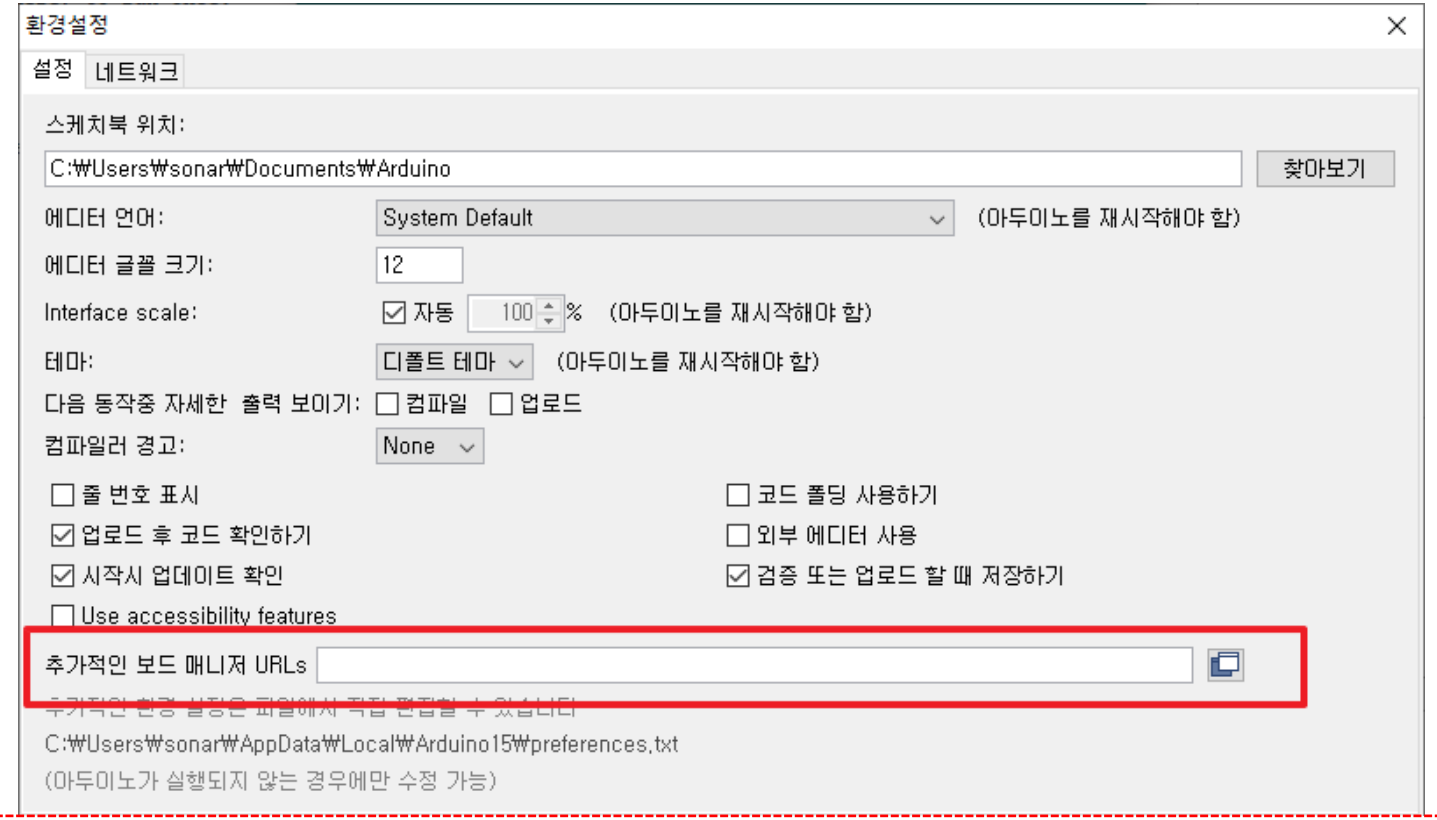
아두이노 IDE 구축

- 통합개발환경의 다운로드 및 설치



ESP8266 설정

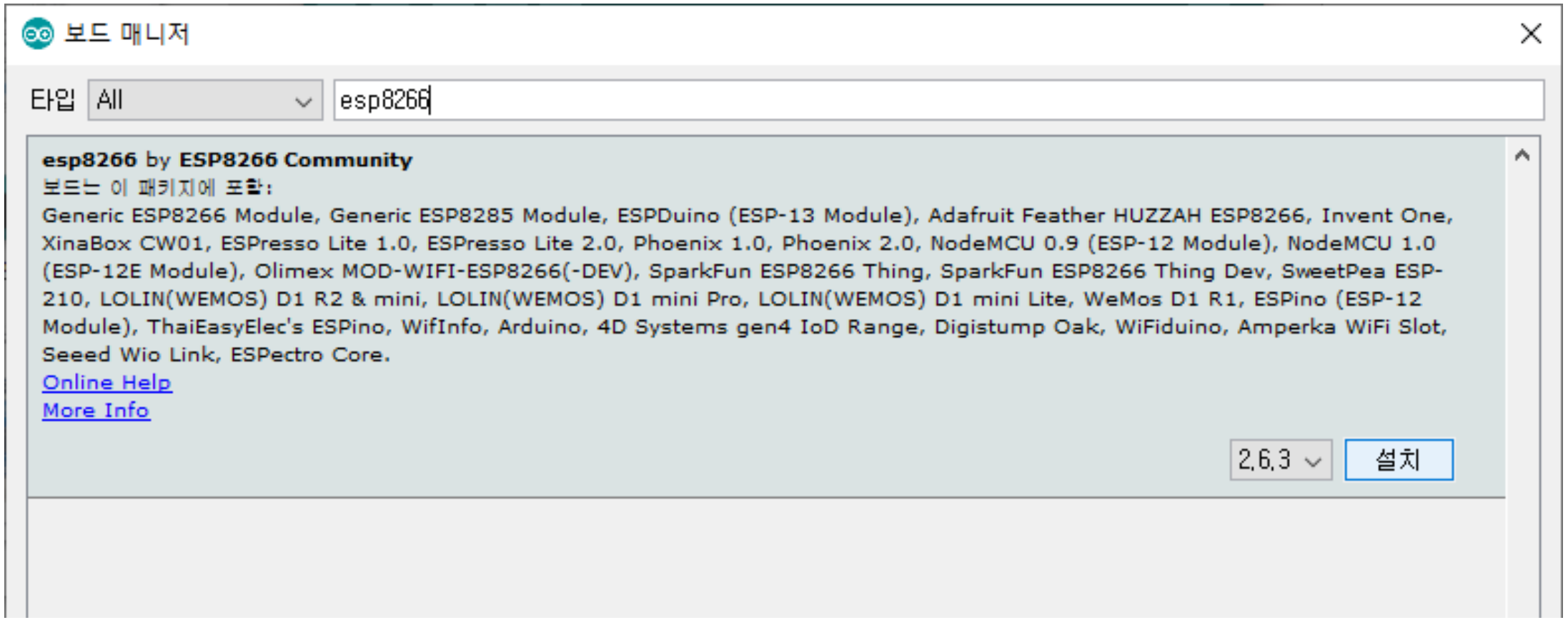
- 아두이노 IDE + ESP8266 추가 설정 #1



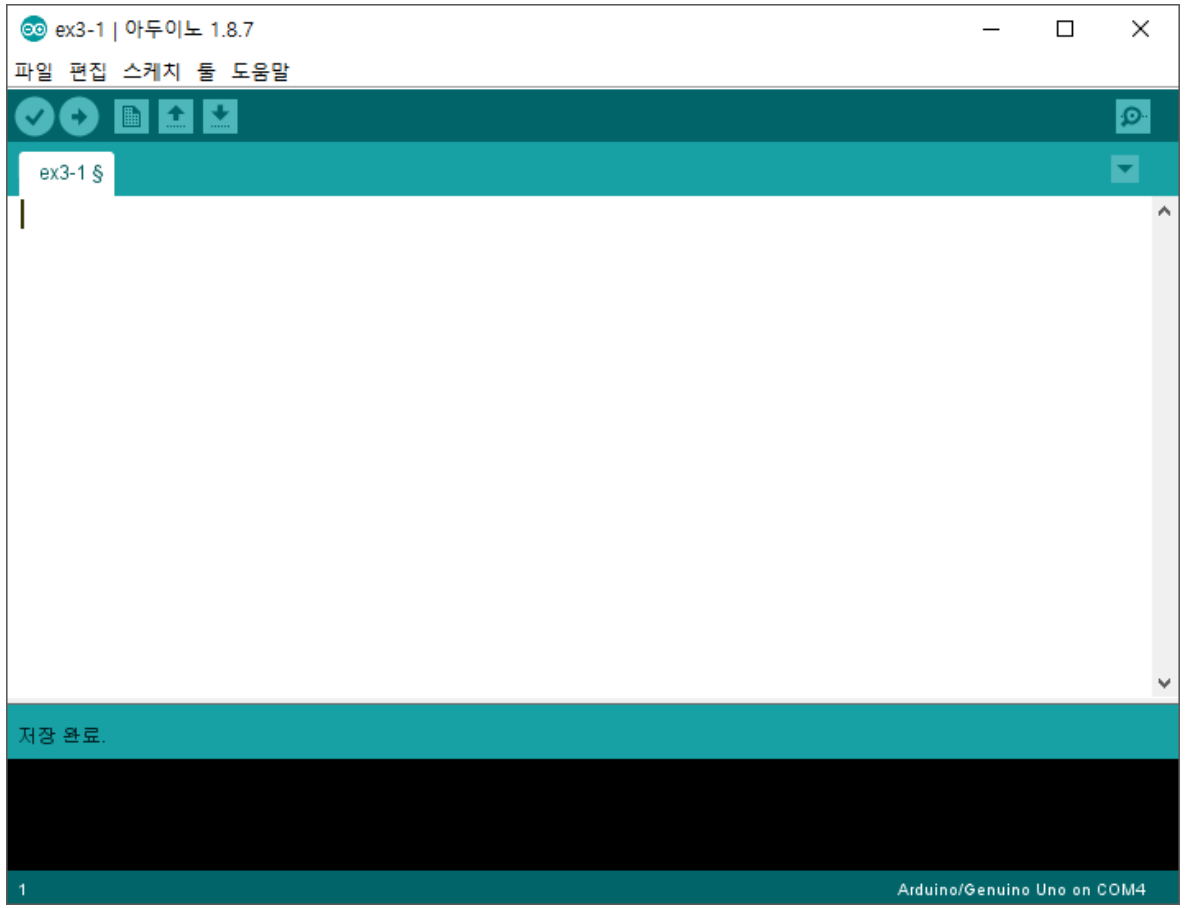
https://arduino.esp8266.com/stable/package_esp8266com_index.json

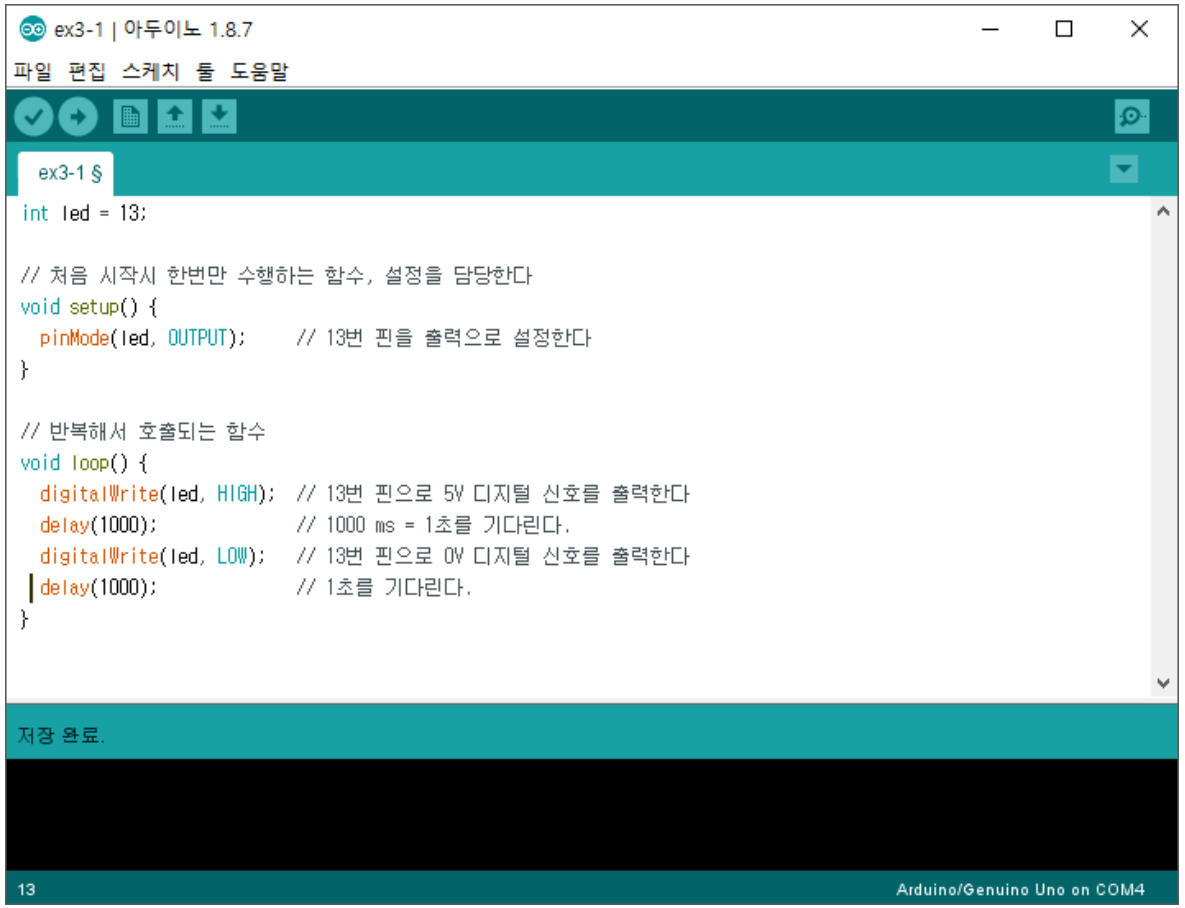
ESP8266 설정

- 아두이노 IDE + ESP8266 추가 설정 #2
  - 메뉴 [툴 → 보드 → 보드 매니저…] 수행 후 esp8266 설치



IDE 실행 결과


ex3-1 | 아두이노 1.8.7
파일 편집 스케치 툴 도움말
ex3-1 §
저장 완료.
1
Arduino/Genuino Uno on COM4


ex3-1 | 아두이노 1.8.7
파일 편집 스케치 툴 도움말
ex3-1 §
int led = 13;
// 처음 시작시 한번만 수행하는 함수, 설정을 담당한다
void setup() {
pinMode(led, OUTPUT); // 13번 핀을 출력으로 설정한다
}
// 반복해서 호출되는 함수
void loop() {
digitalWrite(led, HIGH); // 13번 핀으로 5V 디지털 신호를 출력한다
delay(1000); // 1000 ms = 1초를 기다린다.
digitalWrite(led, LOW); // 13번 핀으로 0V 디지털 신호를 출력한다
delay(1000); // 1초를 기다린다.
}
저장 완료.
13
Arduino/Genuino Uno on COM4

프로그램의 구조

- C/C++을 이용하지만, 일반적인 프로그램과 다른 구조
  - setup() 함수와 loop()함수로 이루어짐
  - setup() : 처음 한번만 수행, loop() : 무한히 반복해서 수행

ex1.m

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

첫번째 스케치

- 보드 연결 : USB를 이용하여 컴퓨터에 연결

Example 3-1

```
int led = LED_BUILTIN;

// 처음 시작시 한번만 수행하는 함수, 설정을 담당한다
void setup() {
    pinMode(led, OUTPUT);      // 13번 핀을 출력으로 설정한다
}

// 반복해서 호출되는 함수
void loop() {
    digitalWrite(led, HIGH);   // 13번 핀으로 5V 디지털 신호를 출력한다
    delay(1000);               // 1000 ms = 1초를 기다린다.
    digitalWrite(led, LOW);    // 13번 핀으로 0V 디지털 신호를 출력한다
    delay(1000);               // 1초를 기다린다.
}
```

첫번째 스케치 (아두이노)

- pinMode() 함수
  - 디지털 입/출력 핀의 사용 방식 설정 (입력/출력)

```
pinMode(13, OUTPUT);        pinMode(12, INPUT);
```

  - 보통 초기에 한번만 수행하면 됨 ⇒ setup() 함수
- digitalWrite() 함수
  - 디지털 출력핀으로 출력값을 지정 (HIGH, LOW 또는 1, 0)
  - HIGH : 5V, LOW : 0V

```
digitalWrite(13, HIGH);
```

- delay() 함수
  - 시간지연 함수, 단위 mili-second.

```
delay(1000);
```

첫번째 스케치 (NodeMCU)

- pinMode() 함수
  - 디지털 입/출력 핀의 사용 방식 설정 (입력/출력)

```
pinMode(D13, OUTPUT);        pinMode(D12, INPUT);
```

  - 보통 초기에 한번만 수행하면 됨 ⇒ setup() 함수
- digitalWrite() 함수
  - 디지털 출력핀으로 출력값을 지정 (HIGH, LOW 또는 1, 0)
  - HIGH : 5V, LOW : 0V

```
digitalWrite(D13, HIGH);
```

- delay() 함수
  - 시간지연 함수, 단위 mili-second.

```
delay(1000);
```

ESP8266의 입/출력 포트 번호

- %AppData%\Local\Arduino15\packages\esp8266\hardware\esp8266\2.6.3\variants\nodemcu\pins_arduino.h

```
#define PIN_WIRE_SDA (4)
#define PIN_WIRE_SCL (5)
static const uint8_t SDA = PIN_WIRE_SDA;
static const uint8_t SCL = PIN_WIRE_SCL;
#ifndef LED_BUILTIN
#define LED_BUILTIN 2
#endif
#define LED_BUILTIN_AUX 16
static const uint8_t D0   = 16;
static const uint8_t D1   = 5;
static const uint8_t D2   = 4;
static const uint8_t D3   = 0;
static const uint8_t D4   = 2;
static const uint8_t D5   = 14;
static const uint8_t D6   = 12;
static const uint8_t D7   = 13;
static const uint8_t D8   = 15;
static const uint8_t D9   = 3;
static const uint8_t D10  = 1;
```

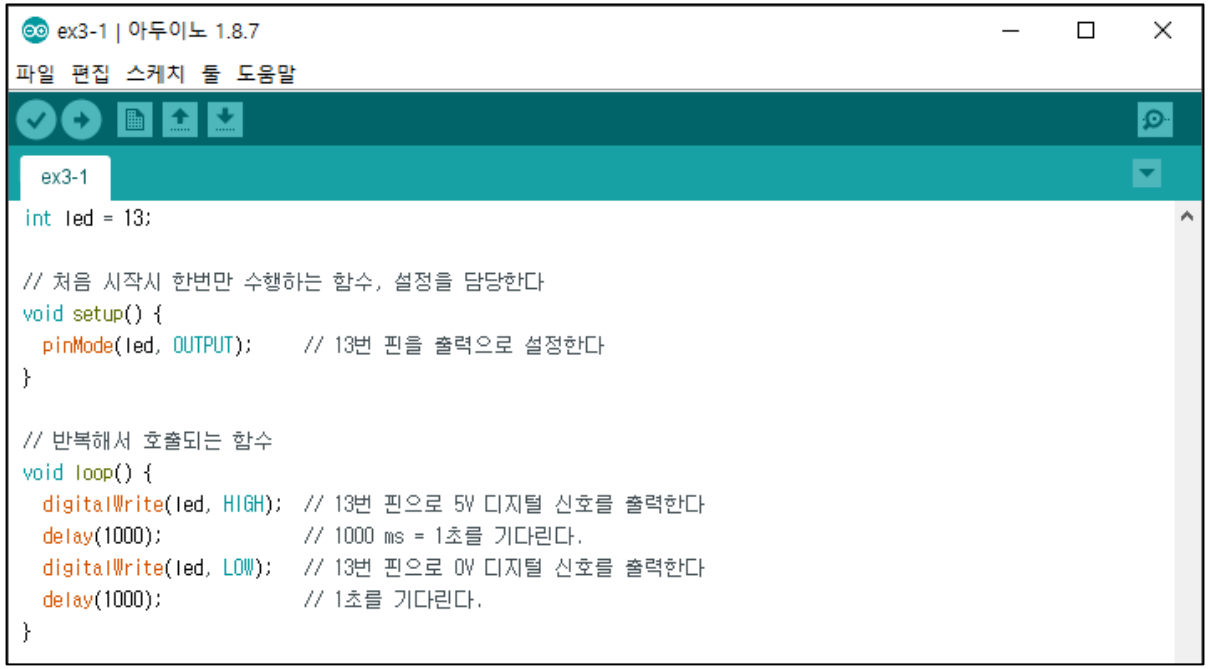
- D0 = 16, D1 = 5, … 으로 Arduino와 다른 방식으로 연결됨
- LED_BUILTIN : 2번 포트 (D4)에 연결
  Arduino에서는 13번 포트에 연결
- LED_BUILTIN_AUX : 2번째 보드내장 LED (빨간색)

Arduino vs. ESP8266

- 구동 전압이 다르다
  - 아두이노 : 5V ↔ ESP8266 : 3.3V
    - ➢ 많은 센서 모듈들이 아두이노용으로 개발되었으며, 5V에 맞추어져 있음
    - ➢ 상당 수의 모듈들은 3.3V~5V에서 구동되나 일부 다른 모듈들이 있으므로 주의
  - 입/출력 핀을 지정하는 방법이 다름
    - ➢ 하드웨어가 다르기 때문에 핀 번호가 다름
    - ➢ 아두이노의 2번 포트가 ESP8266의 2번 포트가 아님
    - ➢ 호환성을 유지하기 위해서 : 숫자 nn 대신 Dnn으로 사용

```
DigitalWrite(3, HIGH); ⇒ DigitalWrite(D3, HIGH);
```

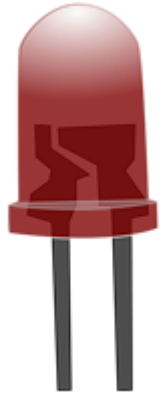
4. 디지털 입출력의 기초
(1학기 마이크로프로세서 : Arduino)

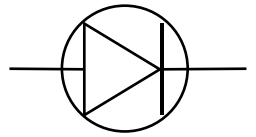


4.1 LED를 이용한 디지털 신호 출력

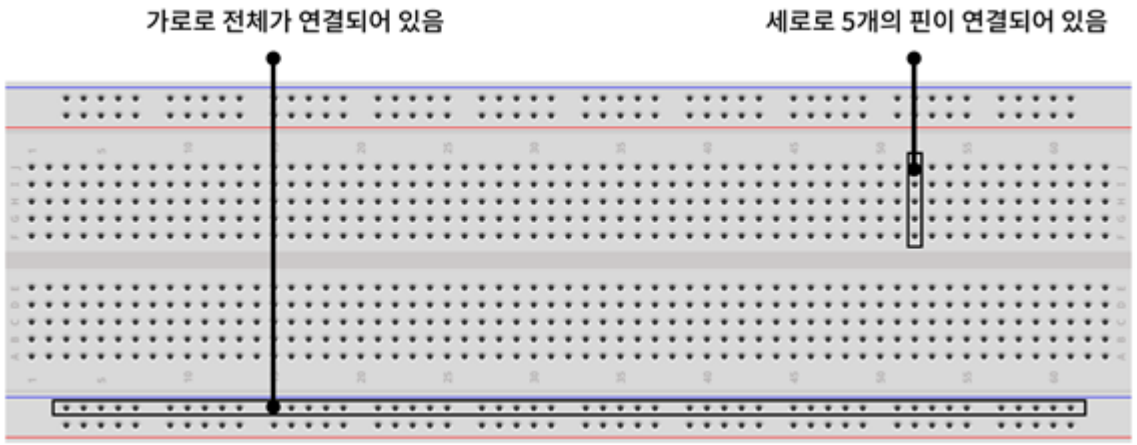
- LED (Light Emitting Diode)
  - 1962 닉 홀로니악이 발명
  - 전원이 방향에 따라 빛을 방출
  - Anode (양극), Cathode (음극)
  - 초기 : 빨간색, 초록색
  - 최근 : 노란색, 파란색, 흰색, 3색
  - 내부저항이 적으며 20~30mA의 전류 사용

  → 5V연결시 200옴 정도의 저항을 직렬로 연결

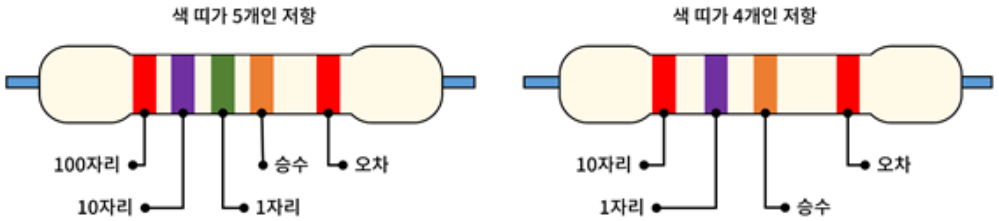




- 브레드보드
  - 전원선으로 사용되는 가로선 (빨간색 : +, 파란색 : -)
  - 구멍 사이의 간격은 0.1인치 (2.54mm) : DIP타입의 IC 칩의 간격과 동일

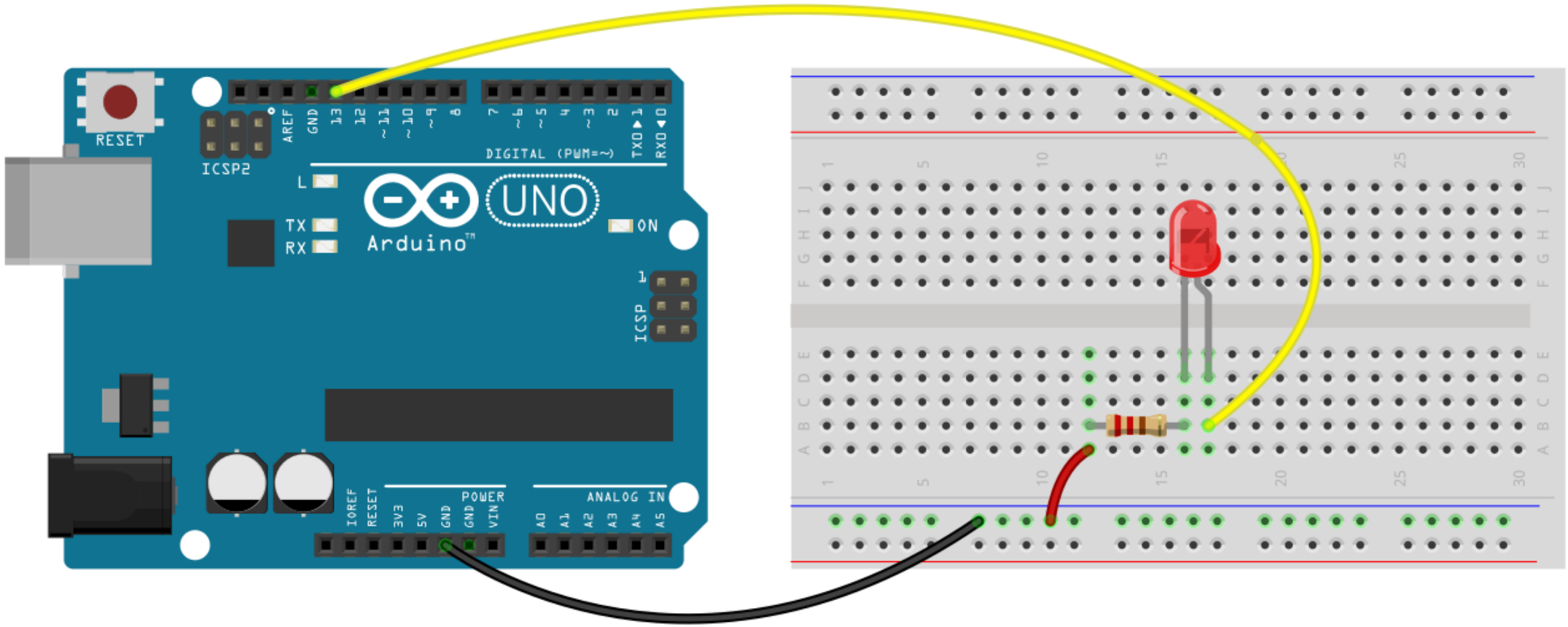


- 저항
  - 전류의 흐름을 방해하는 소자
  - 저항값 : 색 띠로 계산



색상 \ 구분		첫번째 띠	두번째 띠	세번째 띠	내번째 띠	다섯번째 띠
		숫자 1	숫자 2	숫자 3	승수	오차
	검은색	0	0	0	$10^0 = 1$	
	갈색	1	1	1	$10^1 = 10$	±1%
	빨간색	2	2	2	$10^2 = 100$	±2%
	주황색	3	3	3	$10^3 = 1000$	
	노란색	4	4	4	$10^4 = 10000$	
	초록색	5	5	5	$10^5 = 100000$	±0.5%
	파란색	6	6	6	$10^6 = 1000000$	±0.25%
	보라색	7	7	7	$10^7 = 10000000$	±0.1%
	회색	8	8	8	$10^8 = 100000000$	±0.05%
	흰색	9	9	9		
	황금색					±5%
	은색					±10%

- 회로 구성


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN

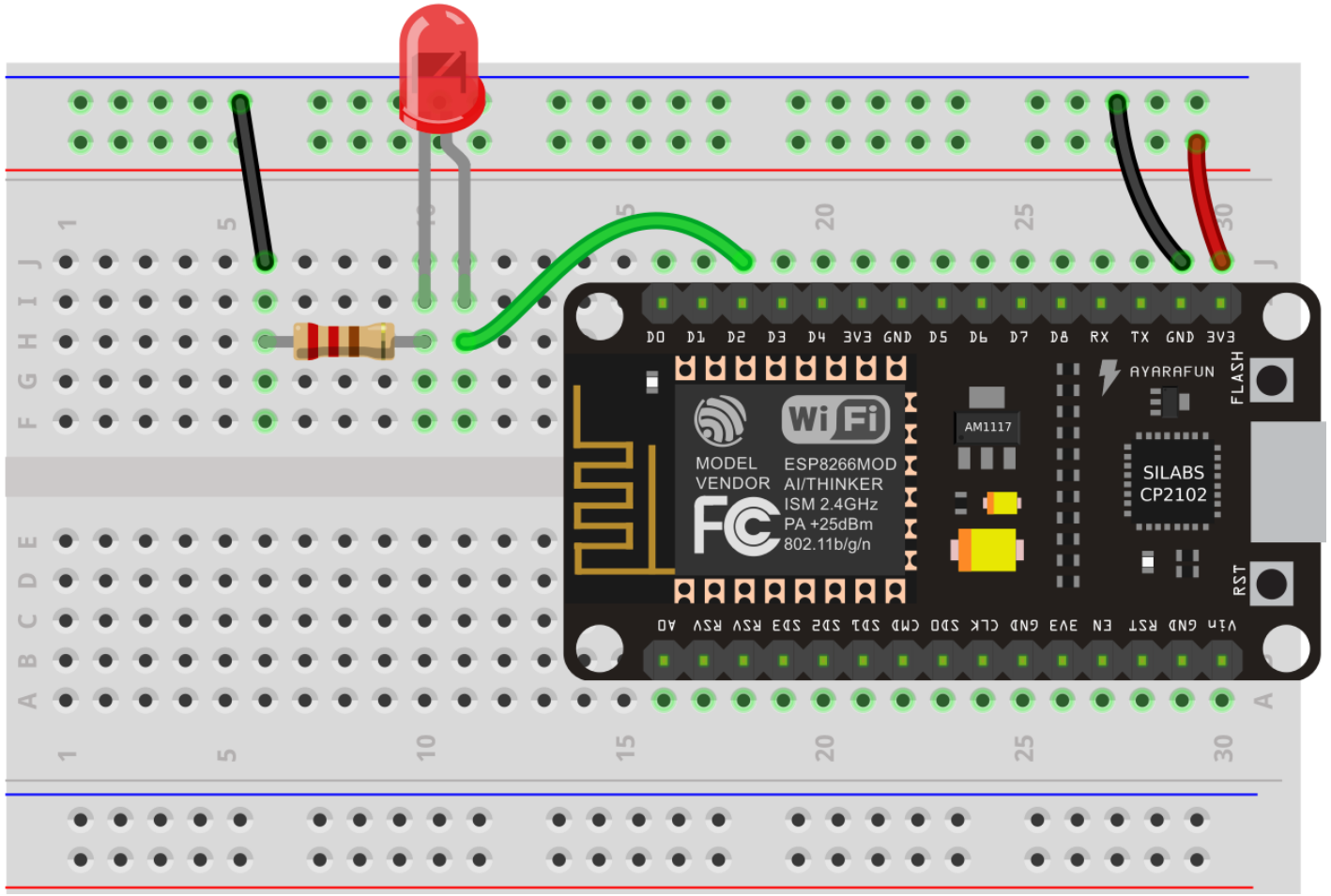
- 스케치

예제 4-1. 첫번째 스케치

```
int led = 13;

// 처음 시작시 한번만 수행하는 함수, 설정을 담당한다
void setup() {
    pinMode(led, OUTPUT);      // 13번 핀을 출력으로 설정한다
}

// 반복해서 호출되는 함수
void loop() {
    digitalWrite(led, HIGH);  // 13번 핀으로 5V 디지털 신호를 출력한다
    delay(1000);              // 1000 ms = 1초를 기다린다.
    digitalWrite(led, LOW);    // 13번 핀으로 0V 디지털 신호를 출력한다
    delay(1000);              // 1초를 기다린다.
}
```

4.1 LED를 이용한 디지털 신호 출력

- 회로 구성



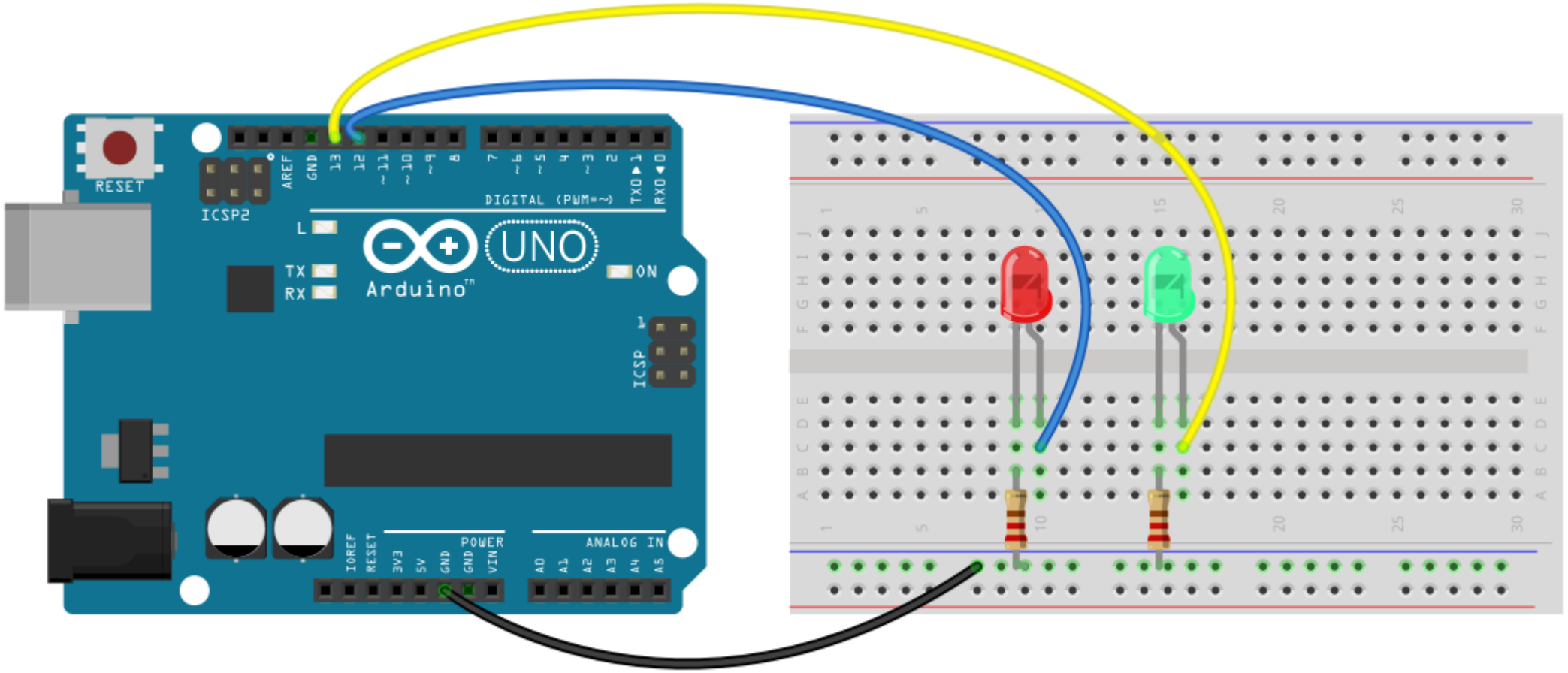
- 스케치

예제 4-1. 첫번째 스케치

```
// LED 깜박이기

// 처음에 한번 수행하는 함수
void setup() {
    pinMode(D2, OUTPUT);      // D2 핀을 출력용으로 설정
}

// setup() 함수 수행 후에 무한히 반복하는 함수
void loop() {
    digitalWrite(D2, HIGH);   // D2핀으로 3.3V 출력, LED On
    delay(1000);              // 1초 기다리기
    digitalWrite(D2, LOW);    // D2핀으로 0V 출력, LED Off
    delay(2000);              // 2초 기다리기
}
```

4.2 두개의 LED를 제어하기

- 회로 구성



예제 4-2. 두 개의 LED 순차 점등

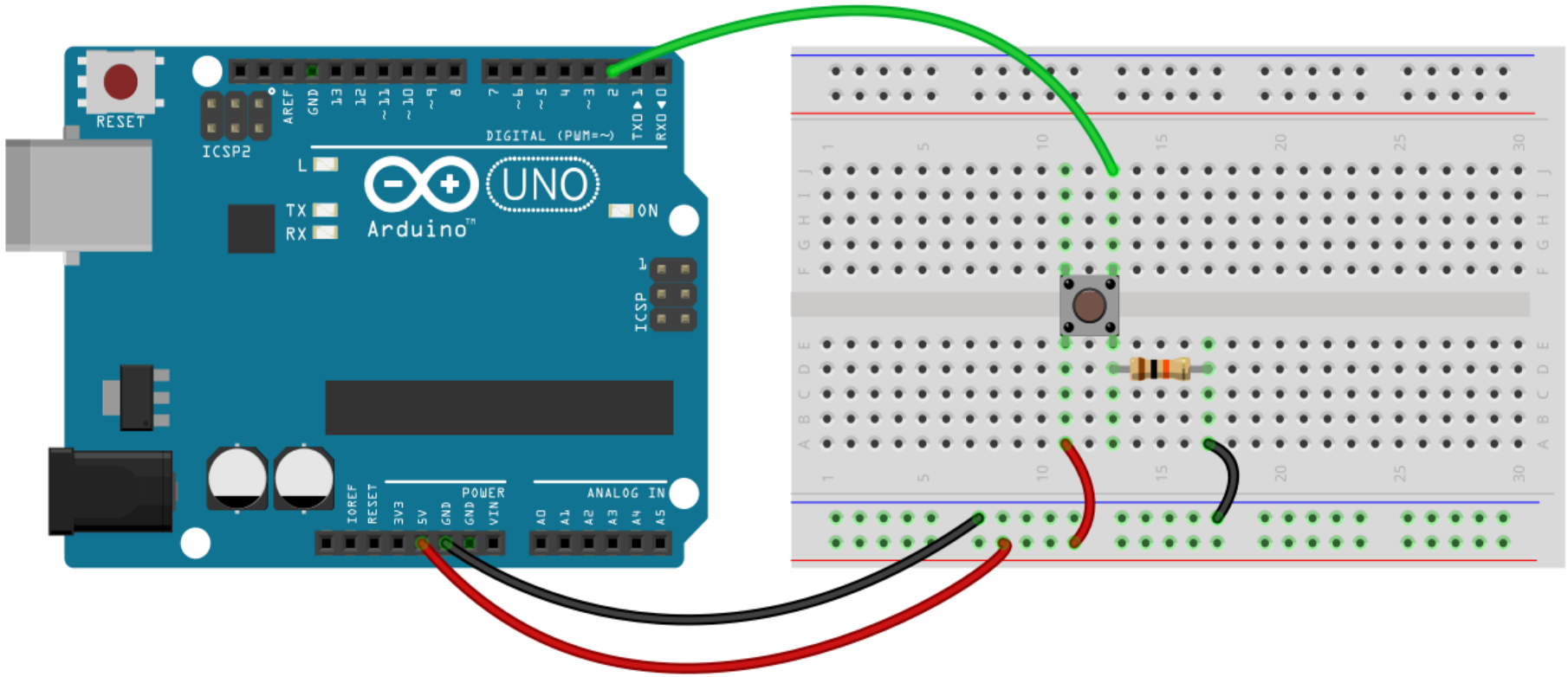
```
int ledA = 13;
int ledB = 12;

// 처음 시작시 한번만 수행하는 함수, 설정을 담당한다
void setup() {
    pinMode(ledA, OUTPUT);      // 13번 핀을 출력으로 설정한다
    pinMode(ledB, OUTPUT);      // 12번 핀을 출력으로 설정한다
}

// 반복해서 호출되는 함수
void loop() {
    // LED 1만 켜기
    digitalWrite(ledA, HIGH);   // LED 1 켜기
    digitalWrite(ledB, LOW);    // LED 2 끄기
    delay(1000);                // 1초를 기다린다.
    // LED 2만 켜기
    digitalWrite(ledA, LOW);    // LED 1 끄기
    digitalWrite(ledB, HIGH);   // LED 2 켜기
    delay(1000);                // 1초를 기다린다.
}
```

추가실습

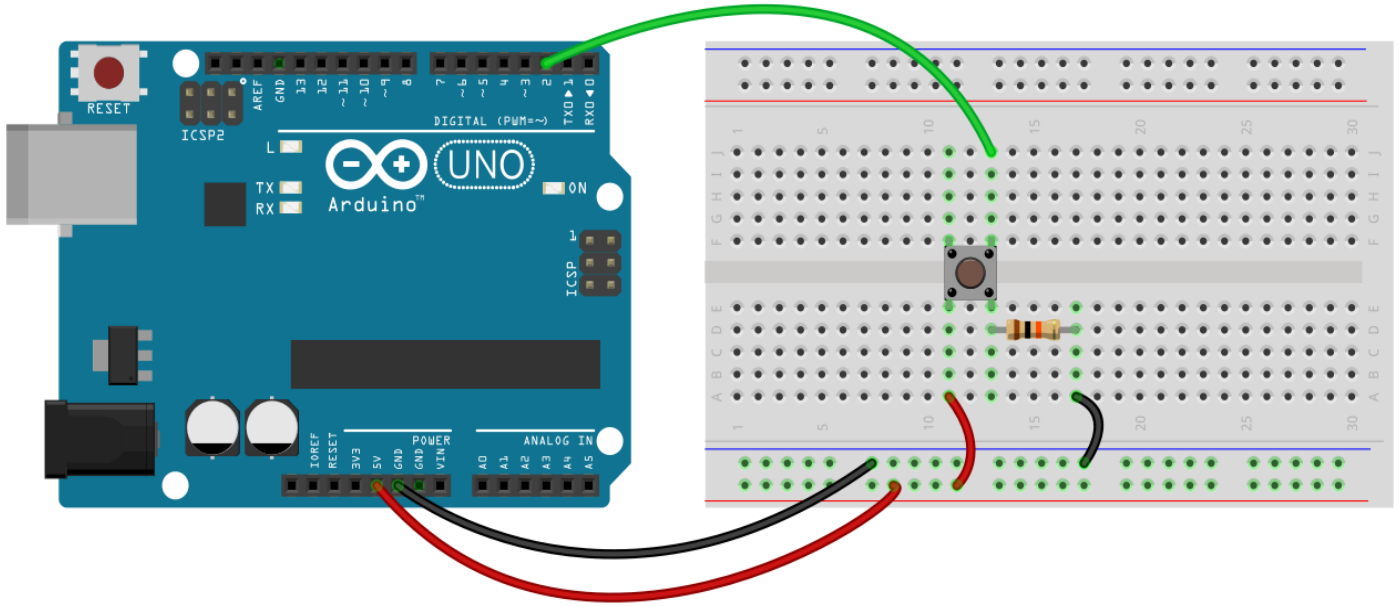
- 아두이노의 예제와 동일하게 LED를 하나 더 추가해서 2개의 LED를 연결하는 회로를 구성하고, 2개의 LED가 교대로 켜지도록 스케치를 작성하라.

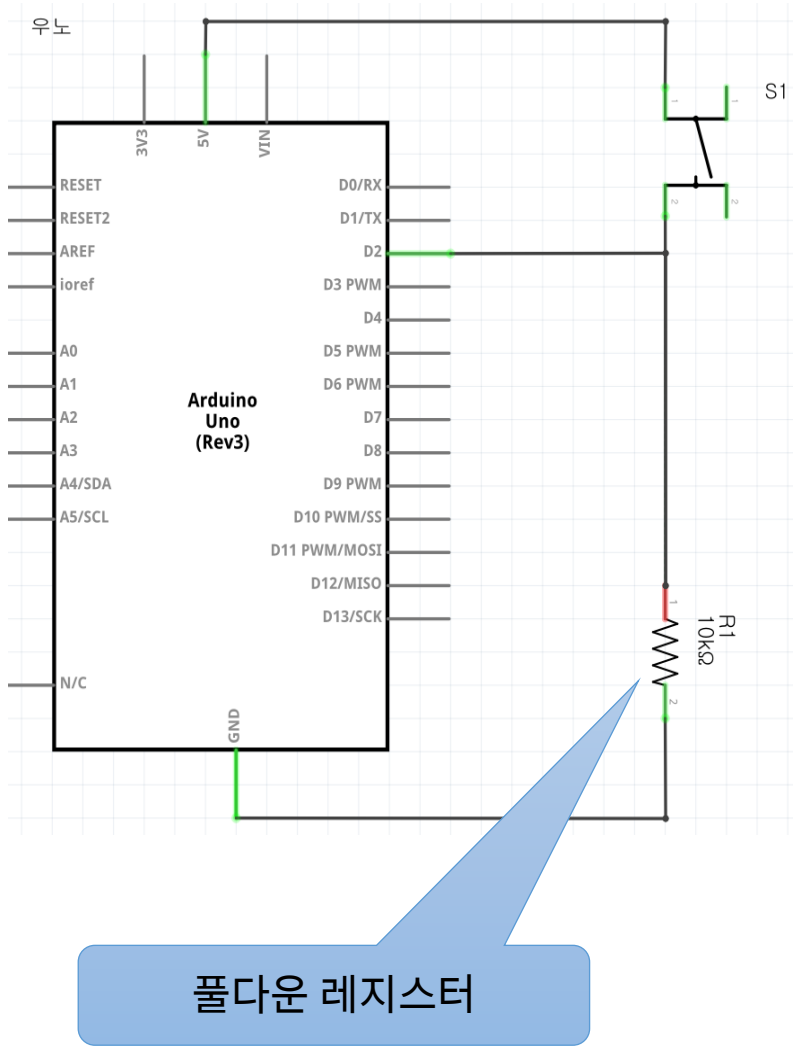
4.3 스위치를 이용한 디지털 입력

- 택트 스위치를 이용하여 입력받음



- 회로도
  - 저항의 역할 : 풀다운(pull-down) 저항
  - 스위치가 Off인 경우에 GND로 연결
  - 스위치 On인 경우에는 5V 인가





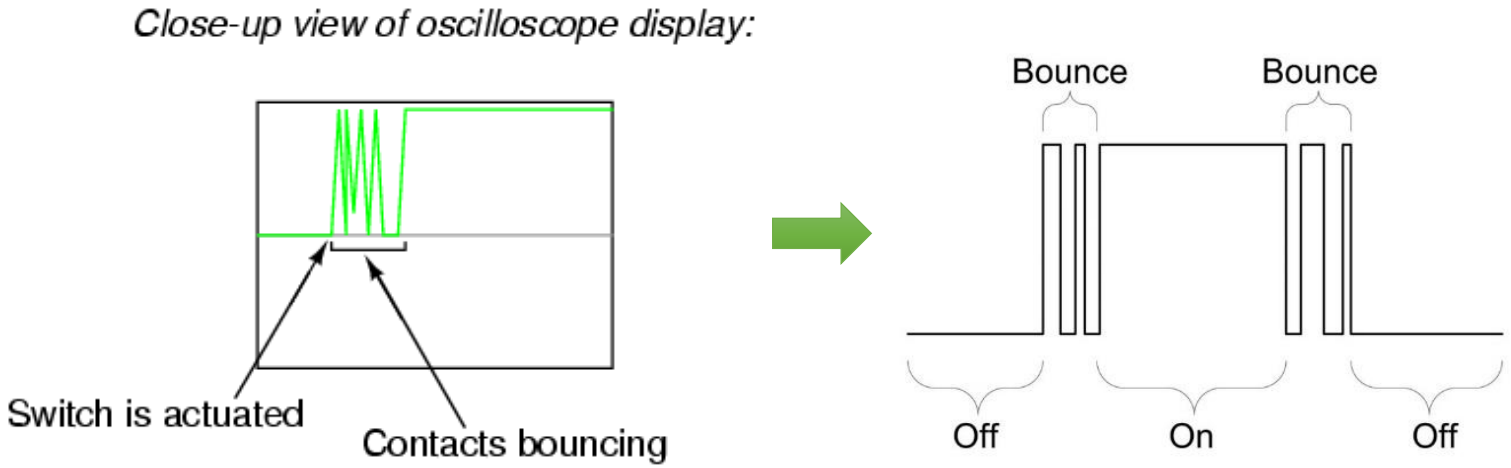
풀업/풀다운

- 풀업 (pull-up) / 풀다운 (pull-down)
  - 없는 경우 : floating 상태 (High impedance) ⇒ 전압이 결정되지 않음
  - 풀업 저항 : 스위치 오픈시에 5V로 입력 전압 유지
  - 풀다운 저항 : 스위치 오픈시에 0V(GND)로 입력 전압 유지

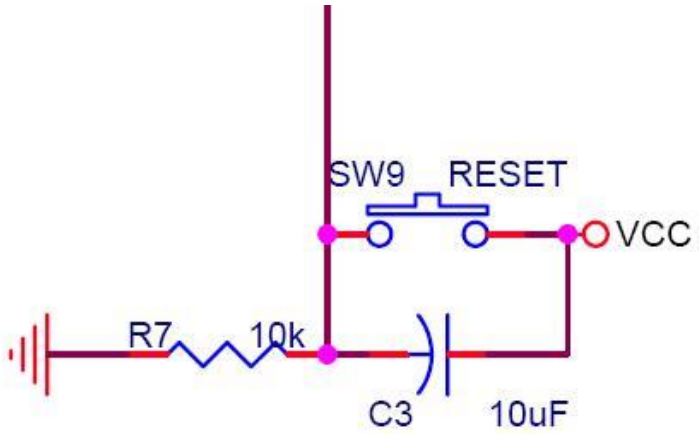
예제 4-3. 디지털 입력 프로그램

```
// 디지털 입/출력 핀 2번을 스위치의 입력으로 설정한다.
int pushButton = 2;

void setup() {
  // 시리얼 통신을 위하여 초기화하는 과정, 9600보레이트(baud-rate)로 설정한다.
  Serial.begin(9600);
  // 버튼 스위치가 연결된 2번 핀을 입력 모드로 설정한다.
  pinMode(pushButton, INPUT);
}

void loop() {
  // 현재 버튼이 연결되어 있는 핀의 값을 읽어들인다.
  int buttonState = digitalRead(pushButton);
  // 입력받은 값을 시리얼 통신을 통해서 PC로 전송한다.
  Serial.println(buttonState);
    // 입력 상태를 안정화하기 위하여 1미리초 동안 대기한다
  delay(1);
}
```

- 채터링(chattering)
  - 스위치의 물리적인 접촉이 일어나는 경우 잡음이 발생



  - 채터링 방지 회로의 예
    - ➢ Capacitor를 사용하여 상태가 천천히 변하도록 한다.
    - ➢ S/W 로 해결 : delay()함수 사용



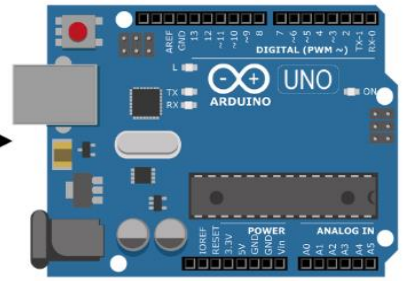
UART 통신

- 시리얼 통신
  - 아두이노와 PC와의 통신
  - UART(Universal Asynchronous Receiver and Transmittor)
  - RS-232, RS-422, RS-485 등의 표준방식이 존재
  - 아두이노에는 디지털 포트 0번과 1번이 준비되어 있음
  - 추가 시리얼 통신은 소프트웨어로 해결

아두이노 IDE & 시리얼 모니터

USB 연결

시리얼 통신

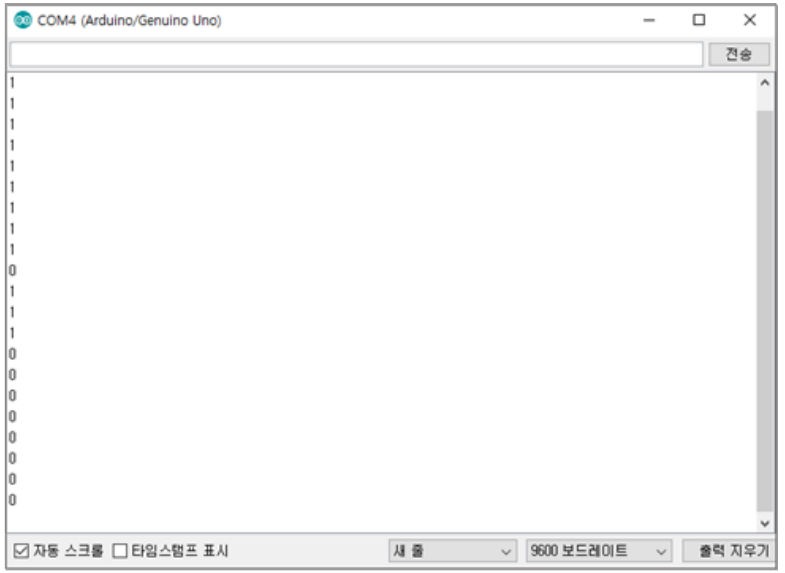


Serial 객체 사용

- Serial 객체
  - PC와의 시리얼통신을 위해 사용함
  - setup()함수에서 초기화
    - `Serial.begin(9600);`
    - 9600 보레이트(baud-rate)로 설정
      - ✓ Baud-rate : 300, 1200, 2400, 4800, 9600, 19200, 38400, …
  - Serial.println(값);
    - 주어진 값을 PC로 전송 print + line (줄넘김 포함)
      - ✓ `Serial.println("Hello");`
      - ✓ `Serial.print(3.14);`
      - ✓ `Serial.print('A');`
    - 읽기 : Serial.available() 함수와 Serial.read() 함수를 사용

시리얼 입력

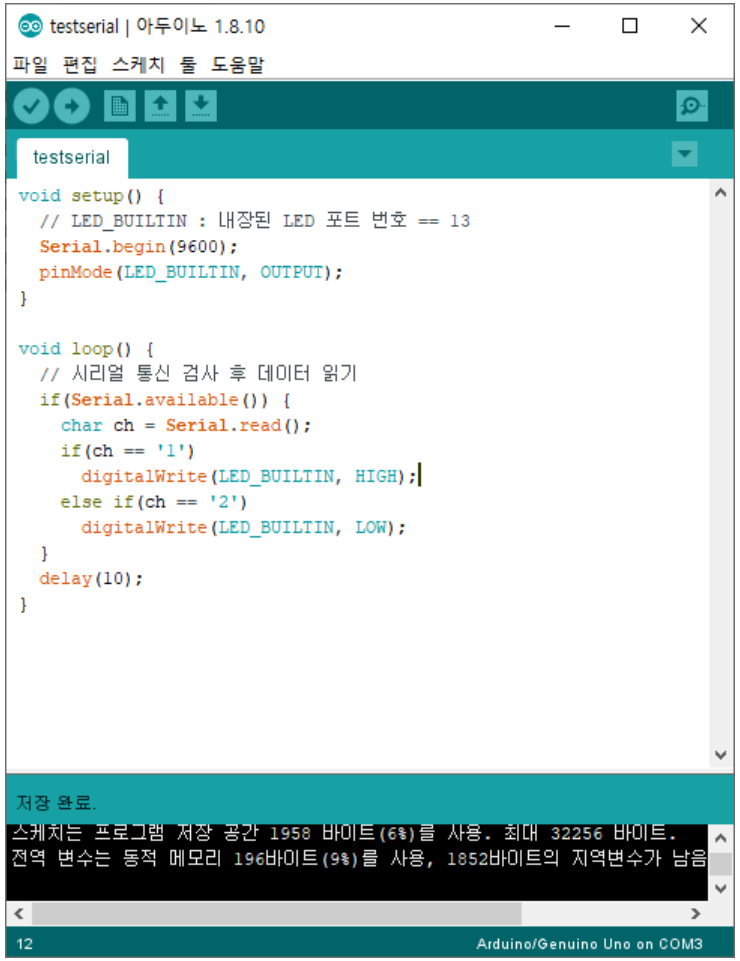
- Serial.available()
  - 현재 시리얼 포트에 새로운 값이 전송되었는지 판단하는 함수
  - 이 함수를 이용하여 새로운 값이 있는 경우만 읽어 들임
- Serial.read()
  - 시리얼 포트에 저장되어 있는 값을 읽어 들임

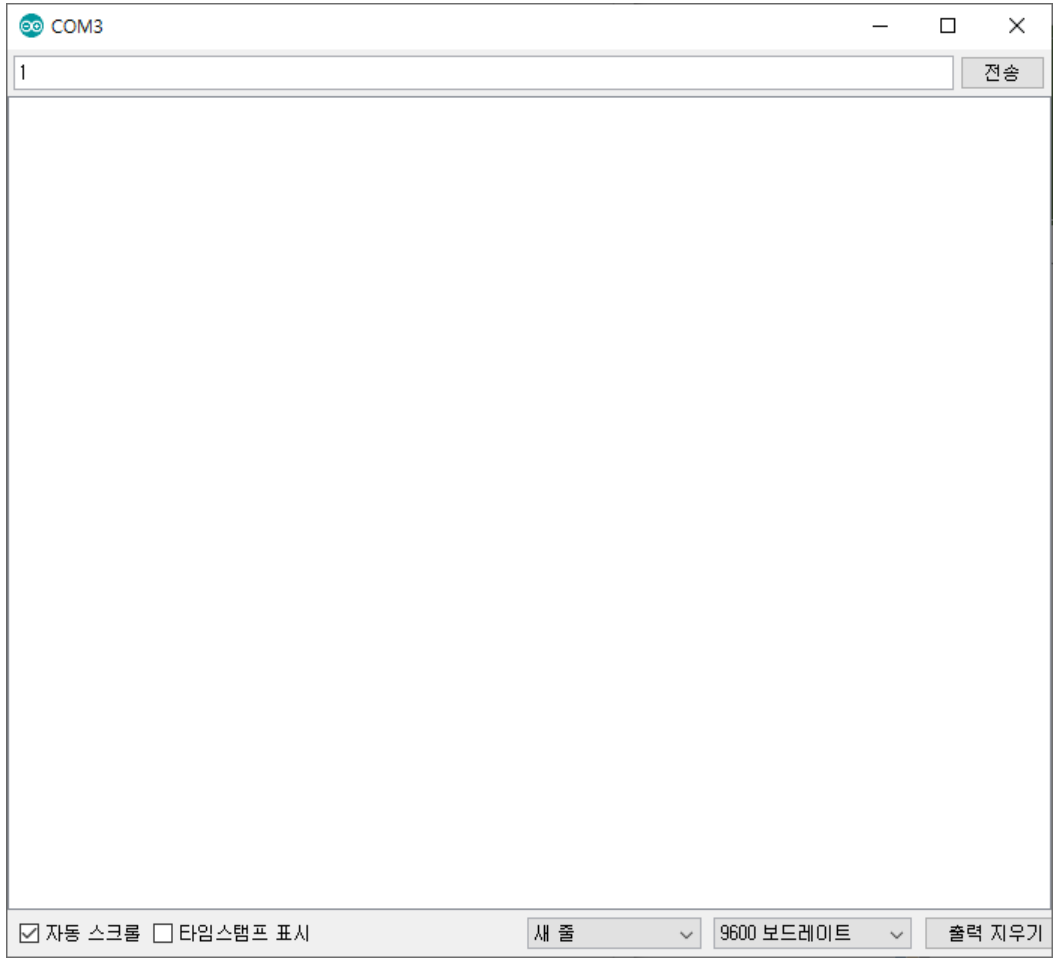


예제: 시리얼 입력을 통한 LED 제어

```
void setup() {
  // LED_BUILTIN : 내장된 LED 포트 번호 == 13
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  // 시리얼 통신 검사 후 데이터 읽기
  if(Serial.available()) {
    char ch = Serial.read();
    if(ch == '1')
      digitalWrite(LED_BUILTIN, HIGH);
    else if(ch == '2')
      digitalWrite(LED_BUILTIN, LOW);
  }
  delay(10);
}
```

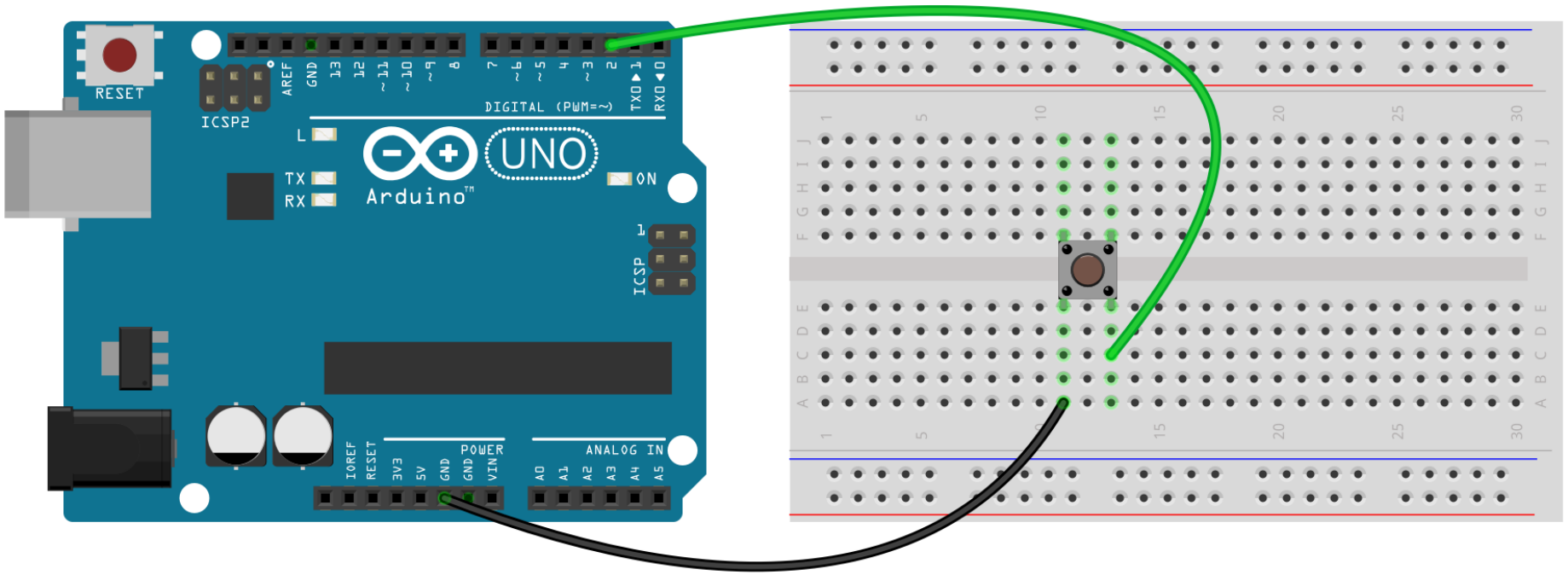
- 시리얼 모니터를 이용하여 전송





Internal pull-up register

- 아두이노 내부에 존재하는 풀업 저항
  - `pinMode(INPUT)` 대신 `pinMode(INPUT_PULLUP)` 사용
  - 외부에 별도의 풀업 저항이 필요 없음 ⇒ 회로가 단순해짐
  - 풀다운 → 풀업 : 스위치 On/Off시에 digitalRead()의 결과가 바뀜
    - ➢ On → LOW, Off → HIGH

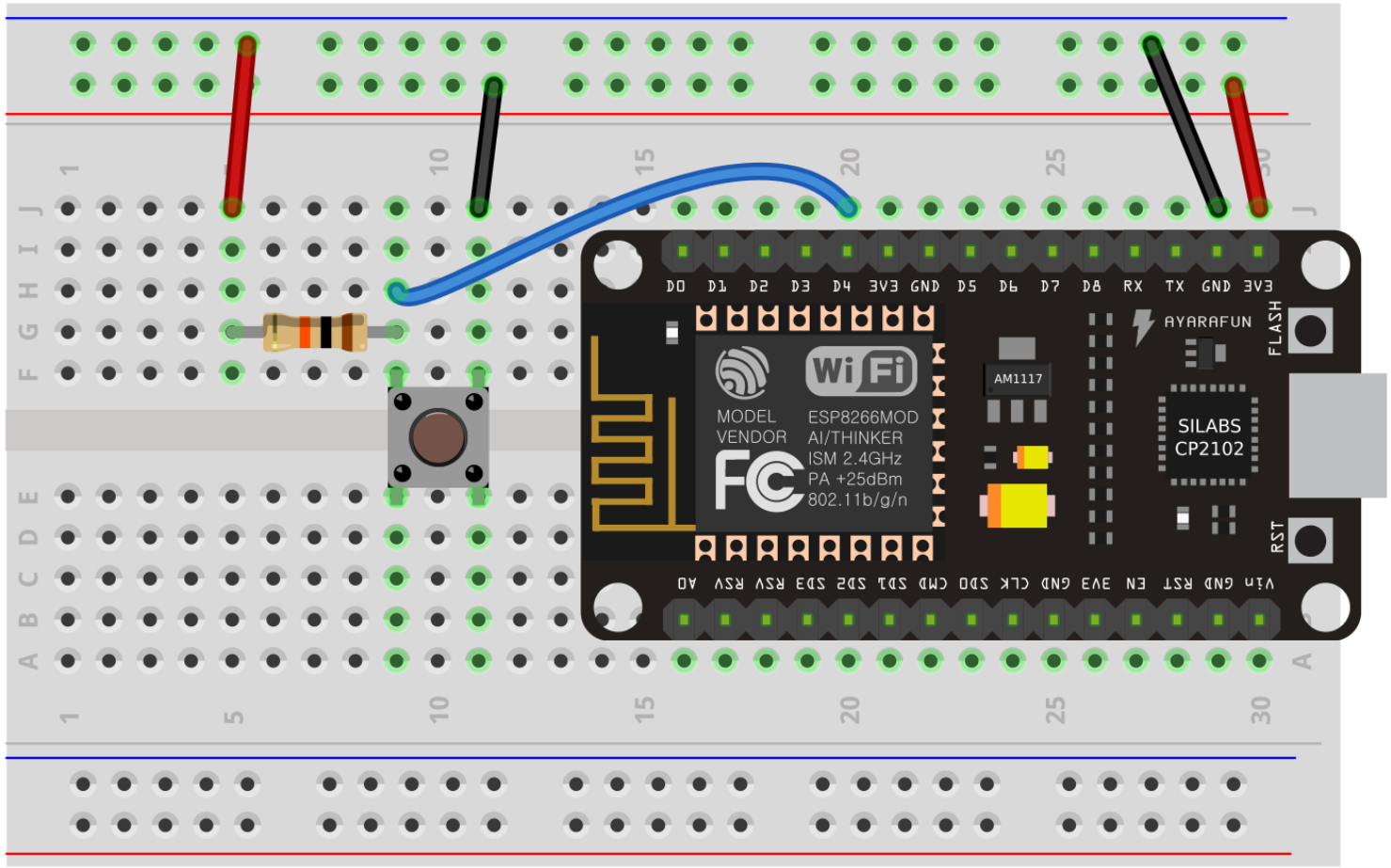


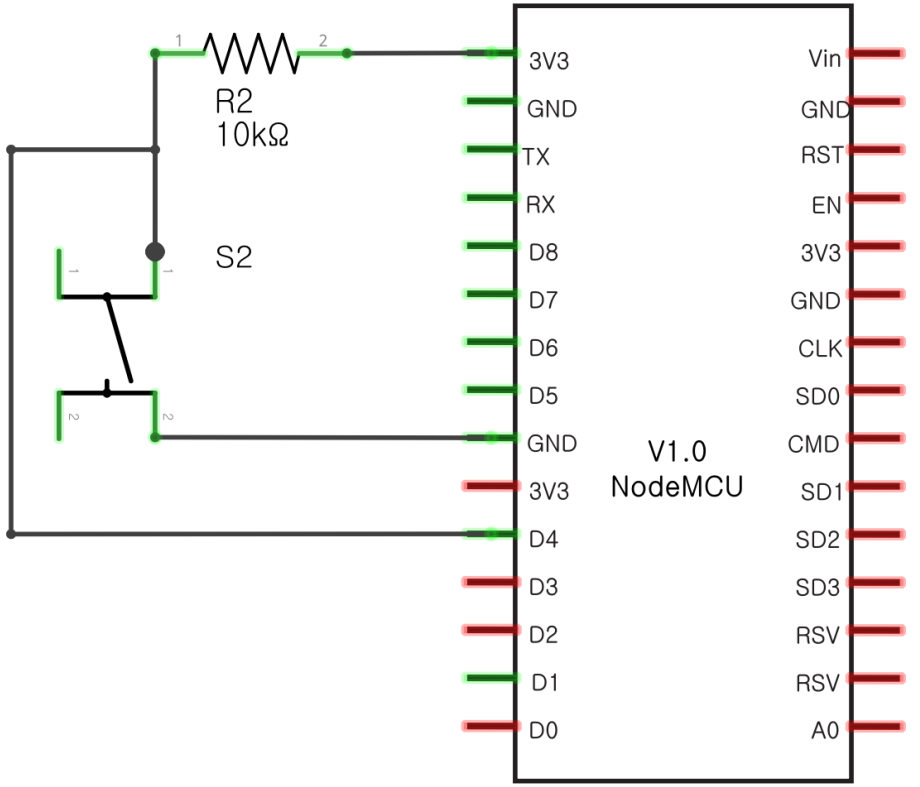
예제 4-3. 디지털 입력 프로그램 (pull-up)

```
// 디지털 입/출력 핀 2번을 스위치의 입력으로 설정한다.
int pushButton = 2;

void setup() {
  // 시리얼 통신을 위하여 초기화하는 과정, 9600보레이트(baud-rate)로 설정한다.
  Serial.begin(9600);
  // 버튼 스위치가 연결된 2번 핀을 입력 모드로 설정한다.
  pinMode(pushButton, INPUT_PULLUP);
}

void loop() {
  // 현재 버튼이 연결되어 있는 핀의 값을 읽어들인다.
  int buttonState = digitalRead(pushButton);
  // 입력받은 값을 시리얼 통신을 통해서 PC로 전송한다.
  Serial.println(buttonState);
    // 입력 상태를 안정화하기 위하여 1미리초 동안 대기한다
  delay(1);
}
```

4.2 스위치를 이용한 디지털 입력

- 회로 구성





예제 4-2. 디지털 입력 프로그램

```
// 스위치를 이용한 디지털 입력

// 처음에 한번 수행하는 함수
void setup() {
  // 시리얼 통신 초기화
  Serial.begin(9600);
  // 스위치가 연결되어 있는 D4핀을 입력용으로 설정
  pinMode(D4, INPUT);
}

// setup() 함수 수행 후에 무한히 반복하는 함수
void loop() {
  // 현재 버튼의 값을 읽어들인다.
  int button = digitalRead(D4);
  // 입력받은 값을 시리얼 통신을 통해서 PC로 전송한다.
  Serial.println(button);
  // 10ms 동안 기다린다.
  delay(500);
}
```

추가 실습

- 아두이노에서 본 것과 같이 내부 풀업저항을 이용하도록 회로를 구성하고 스케치를 작성하시오.