

라즈베리 파이 피코 W를 활용한 IoT 스마트 팜 만들기

목차

1. 이론

1) 라즈베리 파이 피코(Raspberry Pi Pico)

(1) 라즈베리 파이 피코 알아보기	4
(2) 라즈베리 파이 피코 W	5
(3) 사용 시 참고할 문서 및 예제	6

2) IoT 스마트 팜

(1) 스마트 팜 작동 구조	7
(2) 스마트 팜 부품 구성도	7
(3) 스마트 팜에 사용되는 부품	8

2. 실습

1) 파일 내려 받기

(1) 스마트 팜 소스 코드 내려 받기	10
(2) 라즈베리 파이 피코 W 펌웨어 설치	11

2) 통합개발환경(IDE) Thonny

(1) Thonny 설치	12
(2) Thonny 인터프리터 설정	13
(3) Thonny 화면 구성	14
(4) Thonny 사용 방법	14

목차

3) 회로 구성

(1) 확장 쉴드 결합	15
(2) 센서 모듈 연결	15
(3) 식물 생장 LED, OLED 연결	16
(4) 모터 연결	16
(5) 전원 연결	17

4) 소스 코드

(1) 사용되는 소스 코드	18
(2) 소스 코드 수정	19
(3) 소스 코드 업로드	20

5) Firebase

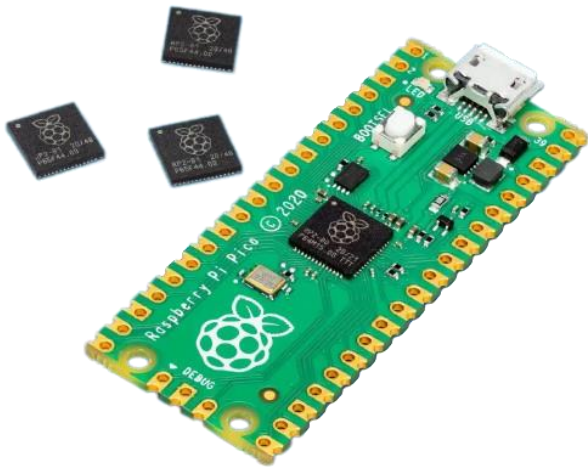
(1) Firebase 프로젝트 만들기	21
(2) 데이터베이스 만들기	22
(3) 데이터베이스 config 정보 얻기	24

6) GitHub Pages

(1) 컴퓨터에서 직접 웹페이지 서비스하기	25
(2) GitHub repository 만들기	26
(3) GitHub Pages 사용 설정하기	27

7) 최종 작동 확인

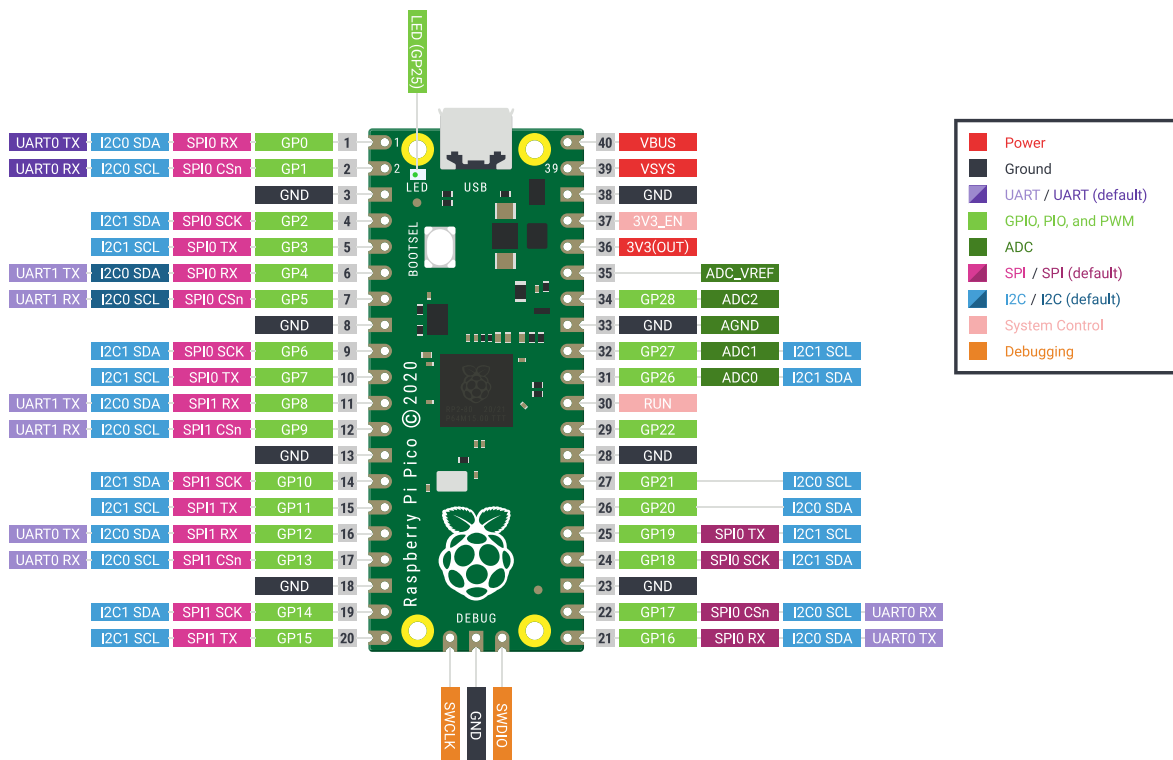
라즈베리 파이 피코 알아보기



라즈베리 파이 피코는 영국의 라즈베리 파이 재단에서 개발한 RP2040 칩을 탑재해 C언어나 마이크로파이썬(MicroPython) 등의 프로그래밍 언어로 동작하도록 한 가정·산업용 마이크로컨트롤러 보드이다.

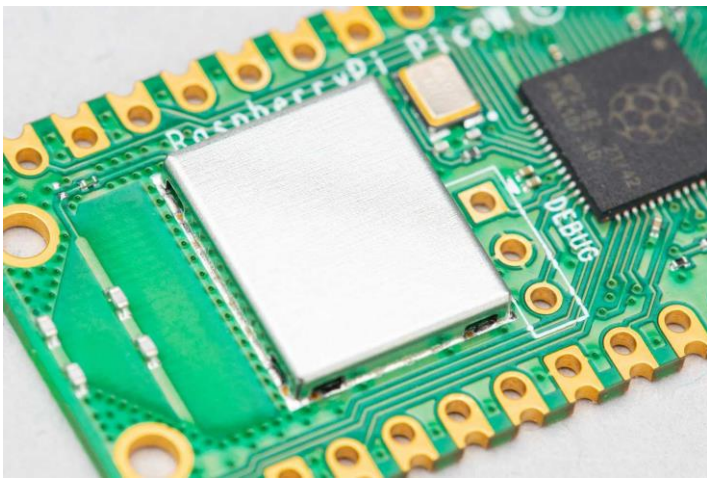
* 마이크로컨트롤러: 마이크로프로세서와 입출력 장치를 하나의 칩으로 만들어 정해진 기능을 수행하도록 한 컴퓨터


여러 소프트웨어를 구동하도록 설계된 운영체제(OS)가 들어가는 라즈베리 파이와는 달리, 라즈베리 파이 피코에는 펌웨어가 들어간다. 이 때문에 성능은 상용 PC에 비해 낮으나 단순하고 반복적인 동작을 수행하는 피지컬 컴퓨팅에 최적화되어 있다.



전원 핀(3V3, GND)으로 센서와 액추에이터에 전원을 공급하고 GPIO(다용도 입출력) 핀으로 센서나 액추에이터와 전기 신호를 주고 받을 수 있다. 프로그래밍 언어로 작성된 소스 코드를 업로드하면 센서의 측정 값을 읽거나 LED 점등, 모터 회전 등의 동작을 수행할 수 있다.

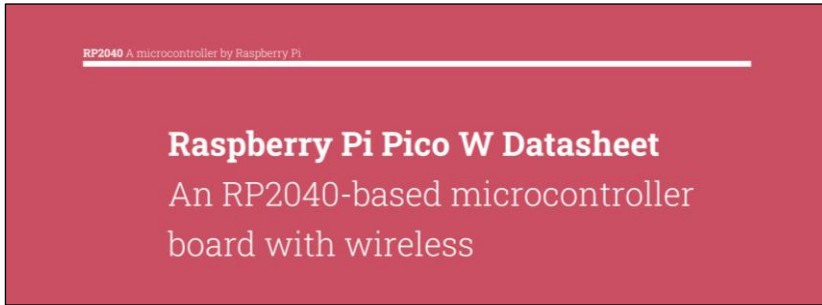
라즈베리 파이 피코 W



라즈베리 파이 피코 W는 무선 통신이 필요한 IoT 애플리케이션 및 프로젝트를 위해 라즈베리 파이 피코에 2.4 GHz 802.11n 무선 LAN을 탑재한 버전이다. 무선 LAN이 탑재되어 있어 Wi-Fi 연결(2.4 GHz)이 가능하고 인터넷에도 접속하여 세계 어디에서나 라즈베리 파이 피코 W를 통해 제어하도록 할 수도 있다. 

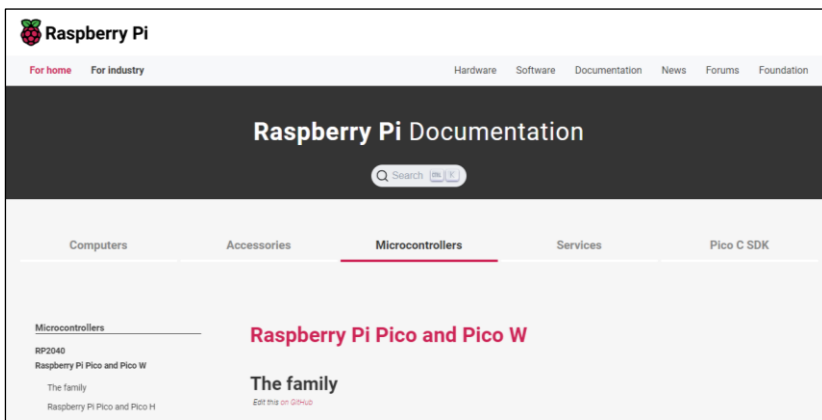
라즈베리 파이 피코와는 무선 LAN이 탑재된 것을 제외하고는 마이크로컨트롤러(RP2040), 플래시 메모리 용량 등의 사양이 모두 동일하고 GPIO 핀의 개수와 구성도 똑같다. 따라서 라즈베리 파이 피코에서 작동하는 소스 코드를 피코 W에 그대로 붙여 넣고 Wi-Fi를 사용하는 기능을 추가할 수도 있다.

사용 시 참고할 문서 및 예제



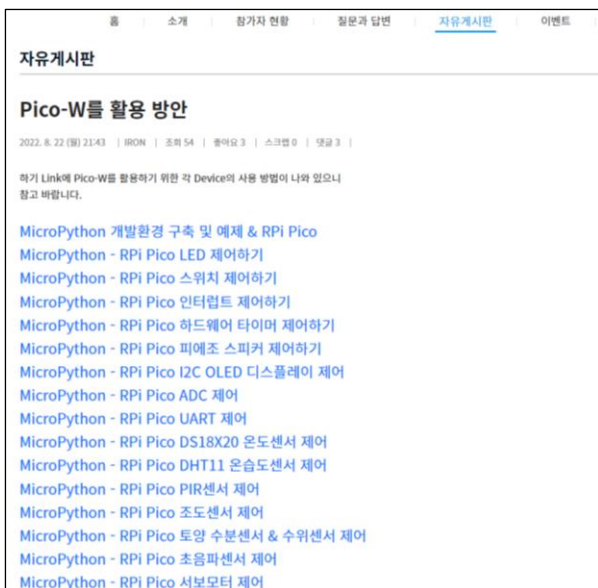
<https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf>

라즈베리 파이 피코 W의 설계 파일, 기계적 사양, 응용 프로그램 정보 등을 적은 데이터시트



<https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html>

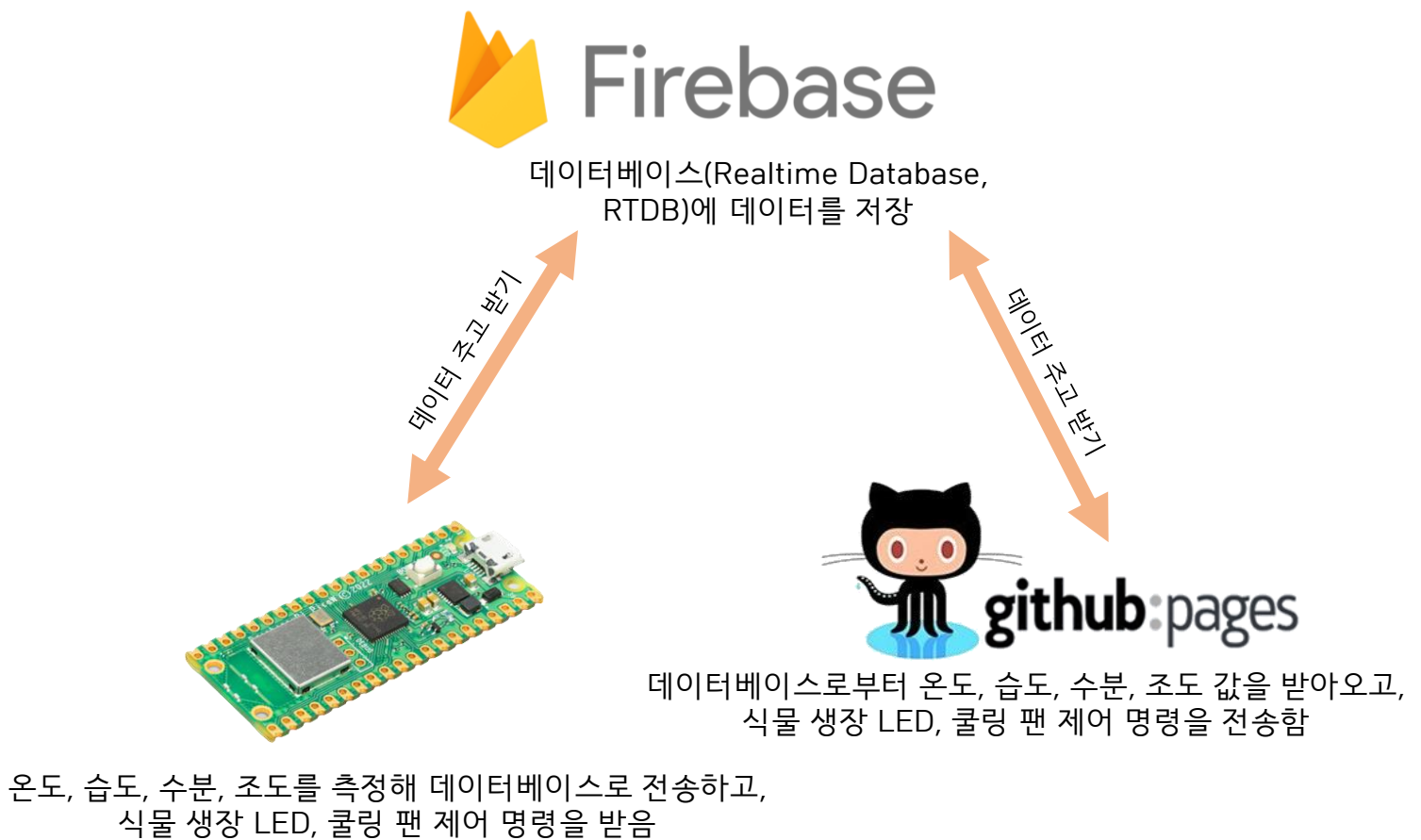
라즈베리 파이 피코와 피코 W의 사용 방법을 설명하는 라즈베리 파이 재단의 공식 문서



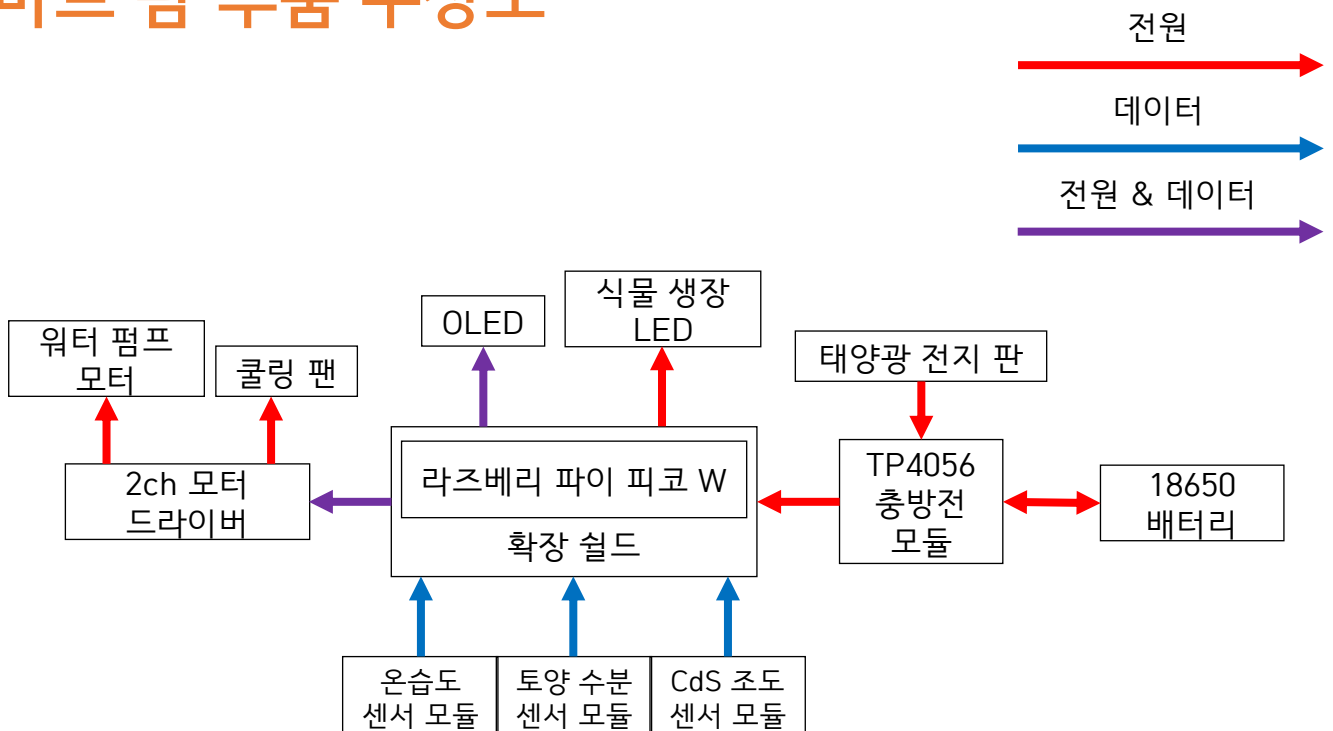
https://make.e4ds.com/contest/board_view.asp?idx=653&ctidx=8

라즈베리 파이 피코 W로 각종 센서와 액추에이터를 제어하는 방법을 설명하는 문서

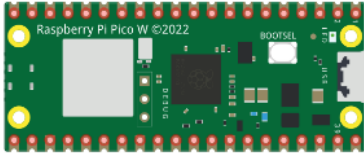
스마트 팜 작동 구조



스마트 팜 부품 구성도

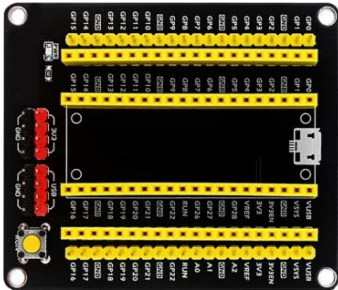


스마트 팜에 사용되는 부품



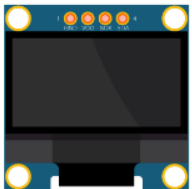
라즈베리 파이 피코 W

Python 프로그램을 통해 각종 센서와 액추에이터를 GPIO 핀에 연결하여 제어하고 인터넷에 연결하여 데이터베이스와 데이터를 주고 받는다.



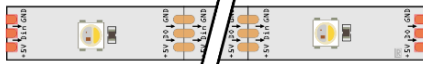
피코 확장 쉴드

라즈베리 파이 피코를 결합해 GPIO 핀을 확장한다.



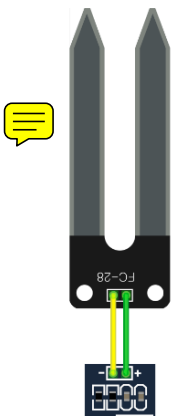
OLED(I2C 타입)

I2C 통신으로 피코 W로부터 데이터를 받아 화면에 표시한다.



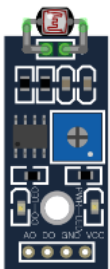
식물 생장 LED

실내에서 식물을 재배할 때 부족한 햇빛을 대체한다.



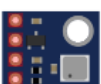
토양 수분 센서 모듈

흙에 꽂아 저항 값을 측정해 흙에 포함된 수분이 얼마나 많은지를 측정한다.



CdS 조도 센서 모듈

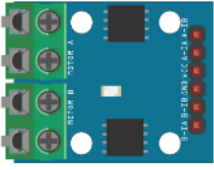
광자 양이 많으면 저항 값이 낮아지고 광자 양이 적으면 저항 값이 높아지는 황화 카드뮴(CdS)의 특성을 이용해 조도를 측정한다.



온습도 센서 모듈(AHT10)

측정된 온도와 습도 값을 I2C 통신을 통해 전송한다.

스마트 팜에 사용되는 부품



2ch 모터 드라이버

모터의 회전 방향, 회전 속도 등을 입력 받아 모터를 제어한다.



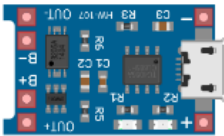
워터 펌프 모터

날개를 회전시켜 액체에 회전력을 주어 액체의 위치를 바꾼다.



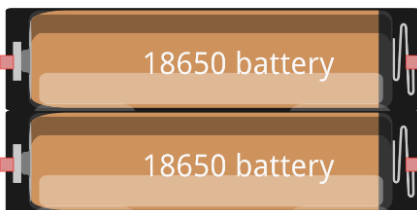
쿨링 팬

날개를 회전시켜 바람을 일으킨다.



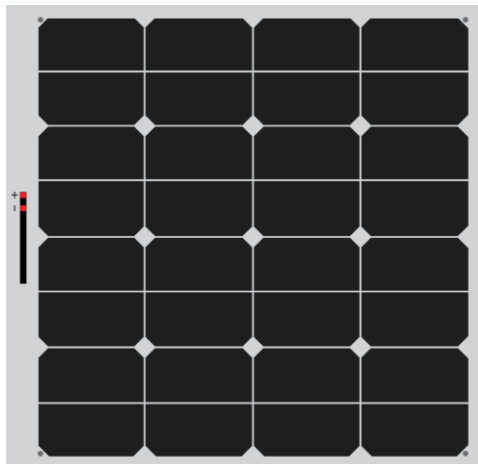
충방전 모듈(TP4056)

USB 포트를 통해 리튬 이온 배터리를 충전하고 리튬 이온 배터리의 전기 에너지를 외부에 연결된 장치로 출력한다.



18650 배터리(+홀더)

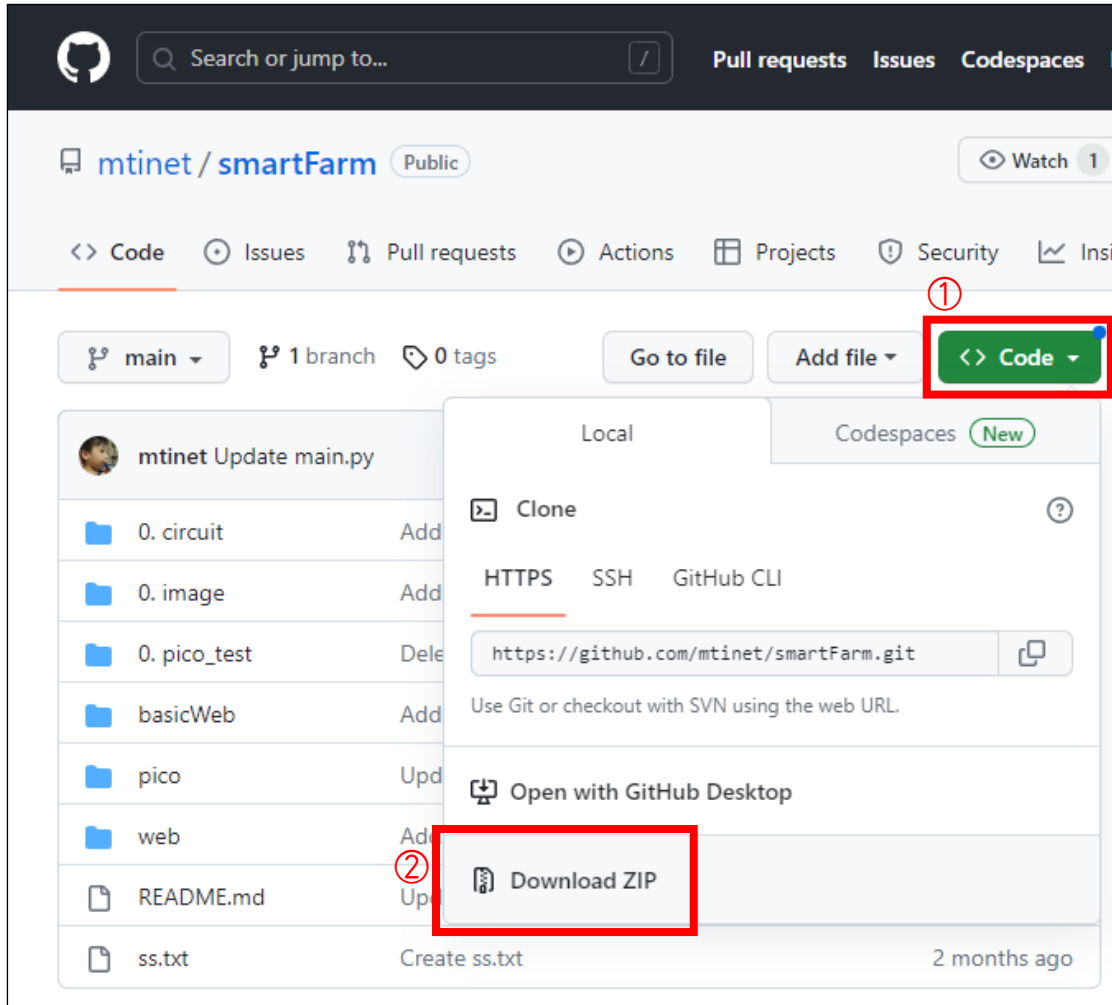
리튬 이온 배터리로, 전기 에너지를 모아 두었다가 필요할 때 출력한다.



태양광 전지 판

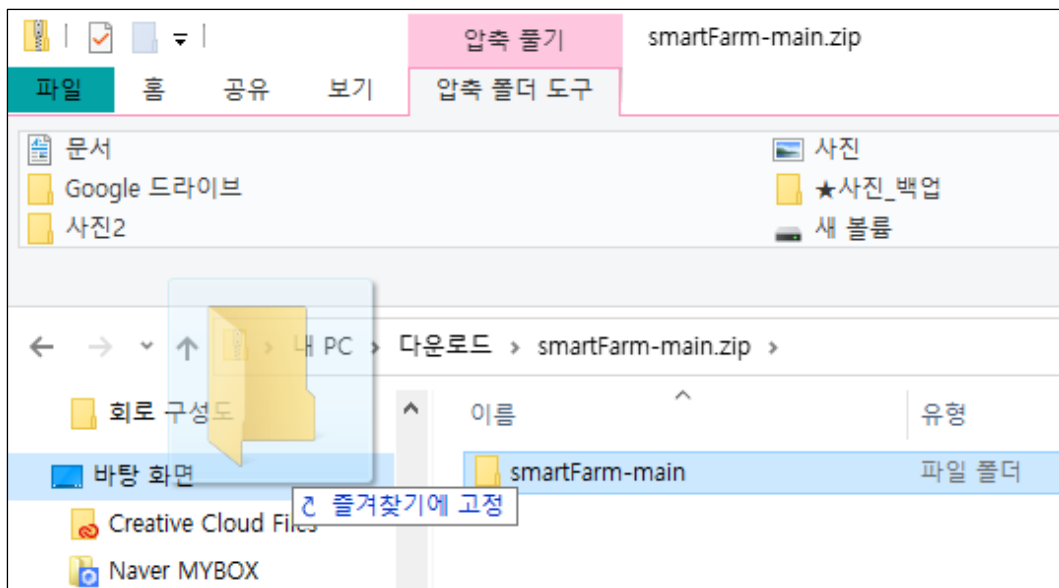
반도체의 광전효과를 이용해 태양빛을 전기 에너지로 변환한다.

스마트 팜 소스 코드 내려 받기



<https://github.com/mtinet/smartFarm>

‘Code’ 버튼을 클릭하고 ‘Download ZIP’ 버튼을 클릭해 압축 파일을 내려 받는다.



파일을 압축 해제하여 smartFarm-main 폴더를 바탕화면에 놓는다.

라즈베리 파이 피코 W 펌웨어 설치

Firmware

Releases

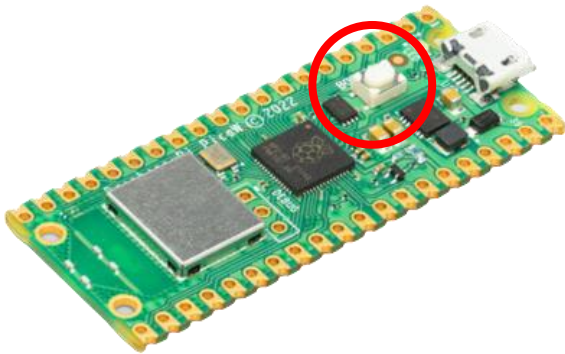
v1.20.0 (2023-04-26) .uf2 [Release notes] (latest)

Nightly builds

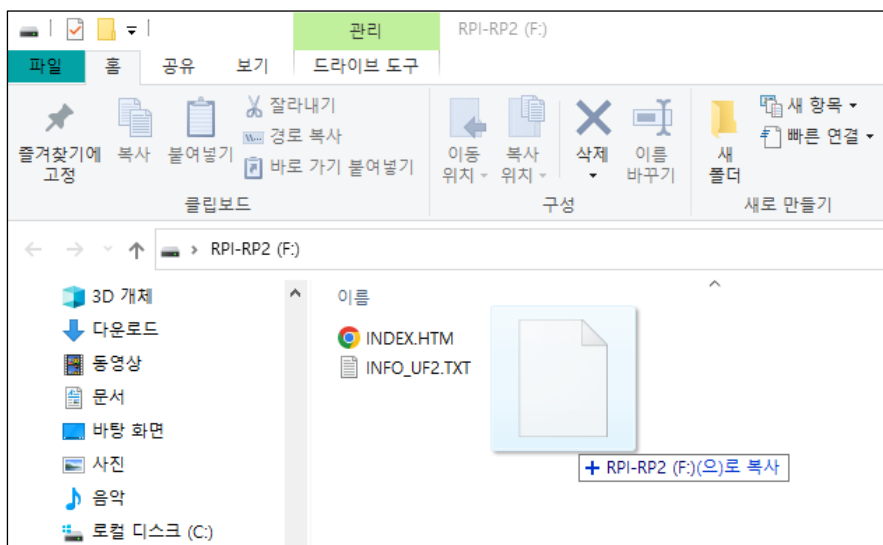
v1.20.0-189-gfd277704c (2023-06-10) .uf2
v1.20.0-188-gb4de697ad (2023-06-09) .uf2
v1.20.0-187-gf01d5fb65 (2023-06-08) .uf2
v1.20.0-183-ga1fbb1980 (2023-06-08) .uf2

<https://micropython.org/download/rp2-pico-w/>

라즈베리 파이 피코 W에 탑재되는 펌웨어(.uf2 파일)를 내려 받는다.



보드의 bootsel 버튼을 누른 채로 USB 케이블로 컴퓨터와 연결한다.



피코 W 폴더(내 PC → 'RPI-RP2' 드라이브)에 펌웨어 파일을 붙여 넣는다.

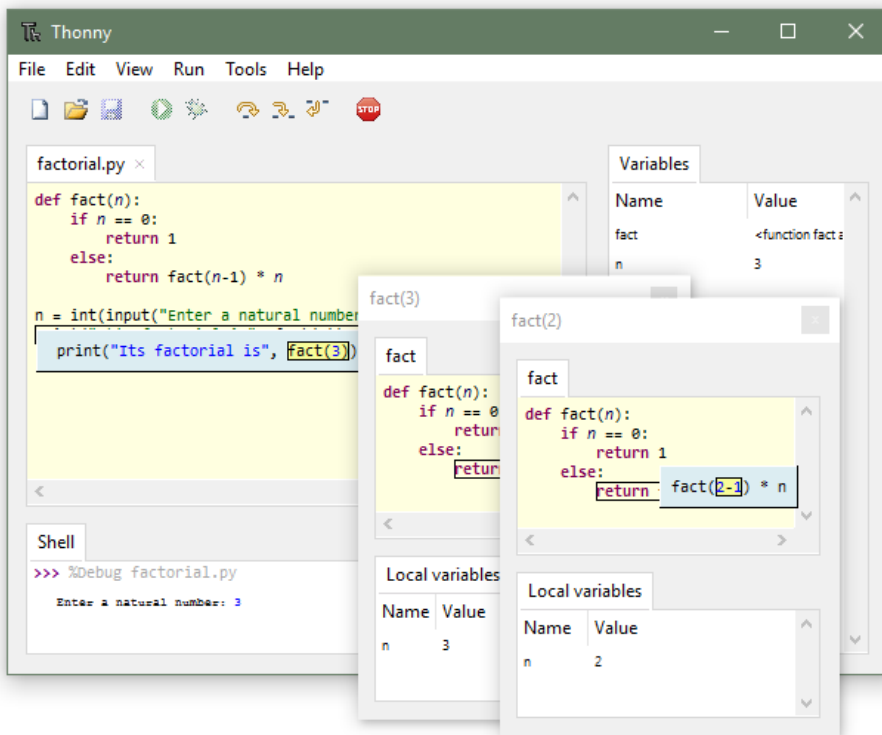
Thonny 설치

라즈베리 파이 피코에는 프로그래밍 언어로 작성된 소스 코드를 업로드해야 한다. Thonny는 마이크로파이썬으로 소스 코드를 작성하고 라즈베리 파이 피코로 실행 및 업로드할 수 있도록 하는 파이썬 용 통합 개발 환경(IDE)이다.

Thonny는 라즈베리 파이 피코 개발을 위한 마이크로파이썬 인터프리터와 모듈이 사전에 설치 되어 있어 Thonny 하나만 설치하면 피코 W 사용에 필요한 환경을 한 번에 구축할 수 있다.

Thonny
Python IDE for beginners

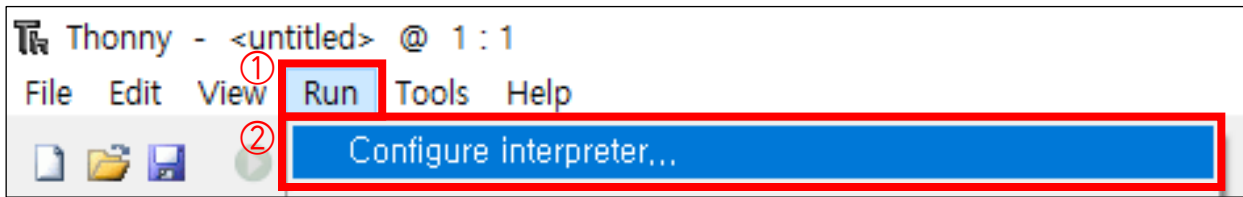
Download version 4.1.1 for
Windows • Mac • Linux



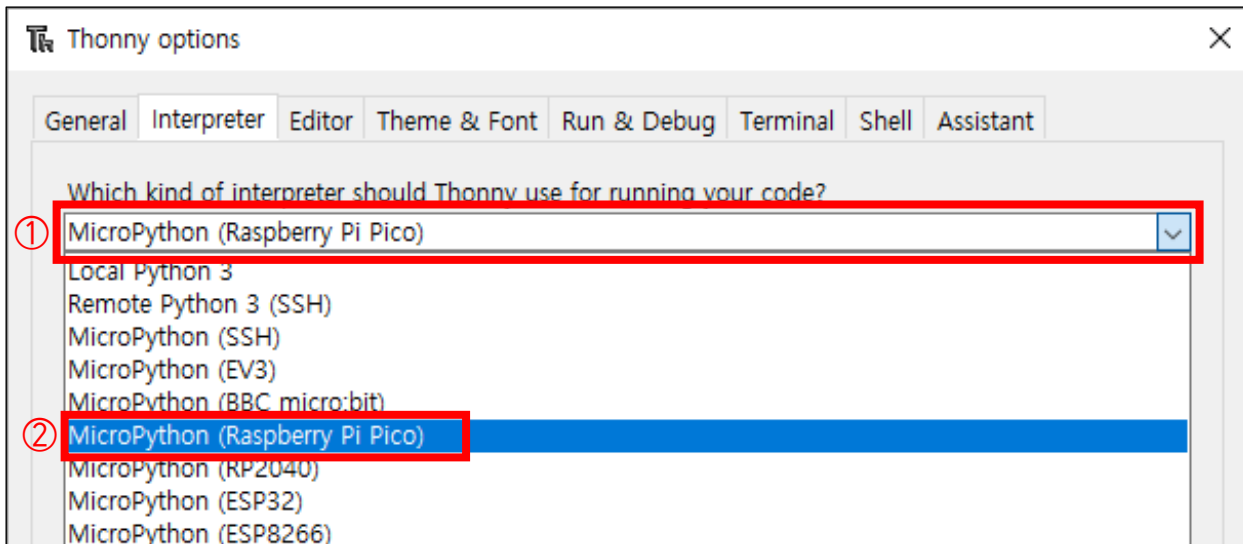
<https://thonny.org/>

Thonny의 최신 버전을 내려 받아 설치한다.

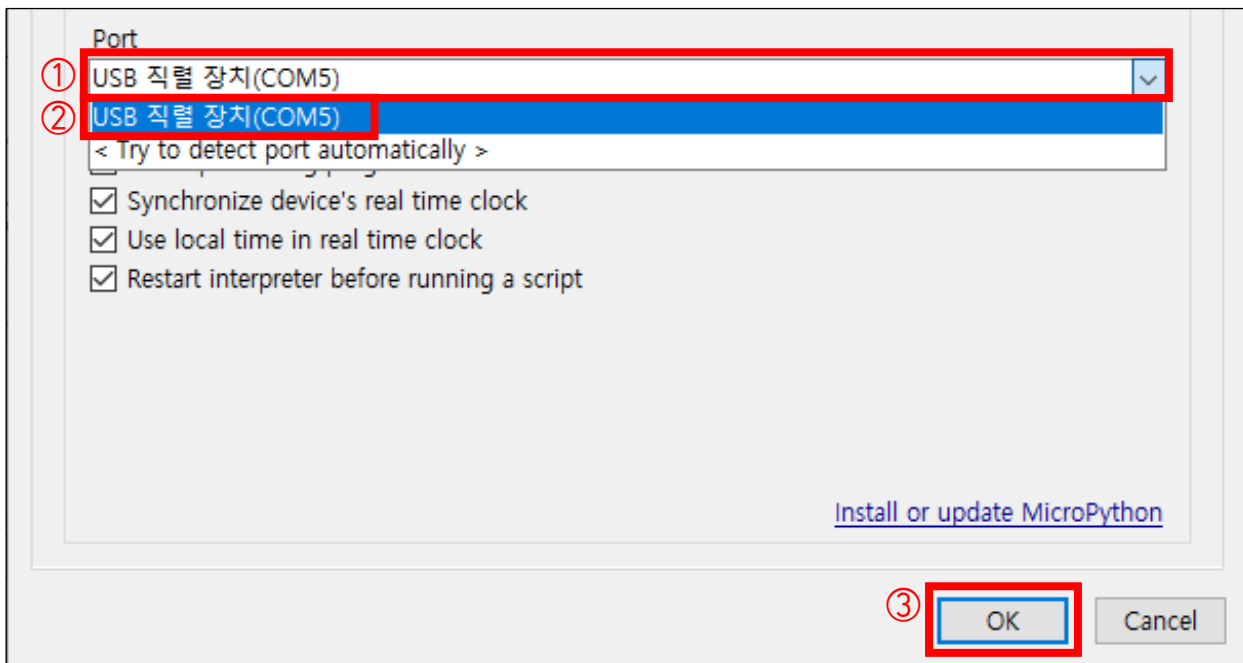
Thonny 인터프리터 설정



Thonny를 켜고 'Run → Configure interpreter...'로 진입한다.



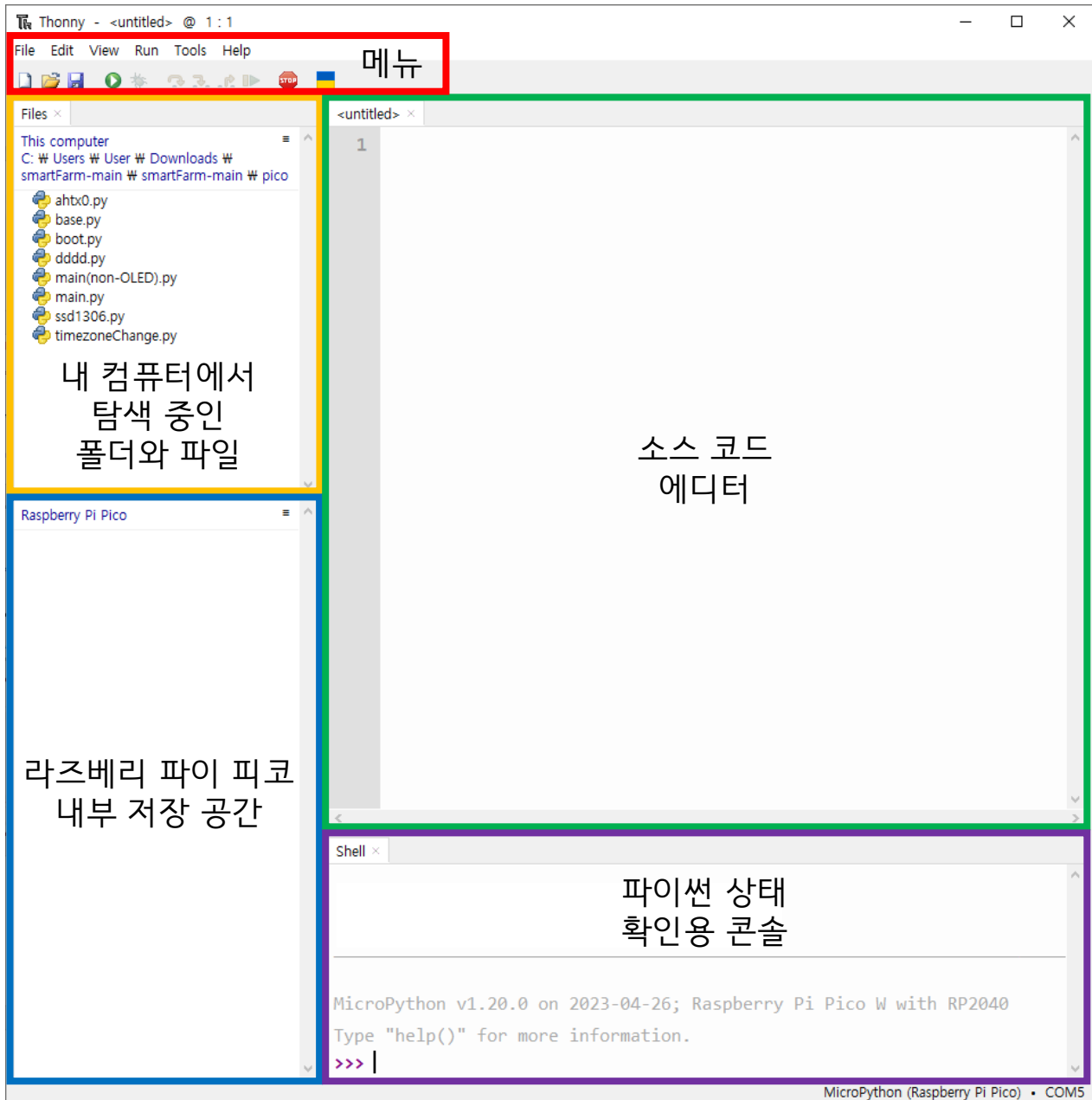
드롭 다운 메뉴에서 'MicroPython (Raspberry Pi Pico)'를 선택한다.



피코 W가 연결되어 있는 포트를 선택하고 'OK' 버튼을 클릭한다.

인터프리터 설정을 마치면 마이크로파이썬으로 소스 코드를 작성하고 피코 W에 업로드하거나 곧바로 실행 명령을 내려 동작을 수행하도록 할 수 있다.

Thonny 화면 구성 & 사용 방법

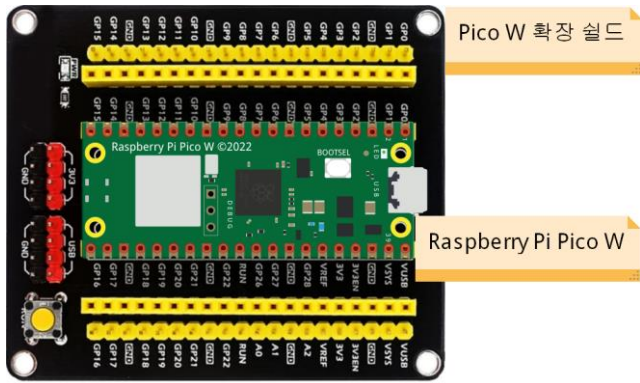


소스 코드 에디터에 소스 코드를 작성하고, 상단 메뉴에서 초록색 재생 버튼과 정지 버튼을 눌러서 라즈베리 파이 피코가 소스 코드 실행을 시작하거나 멈추게 할 수 있다.

라즈베리 파이 피코 내에서 소스 코드가 실행되면 피코의 파이썬 펌웨어가 연산한 결과를 파이썬 상태 확인용 콘솔에 보여주게 된다.

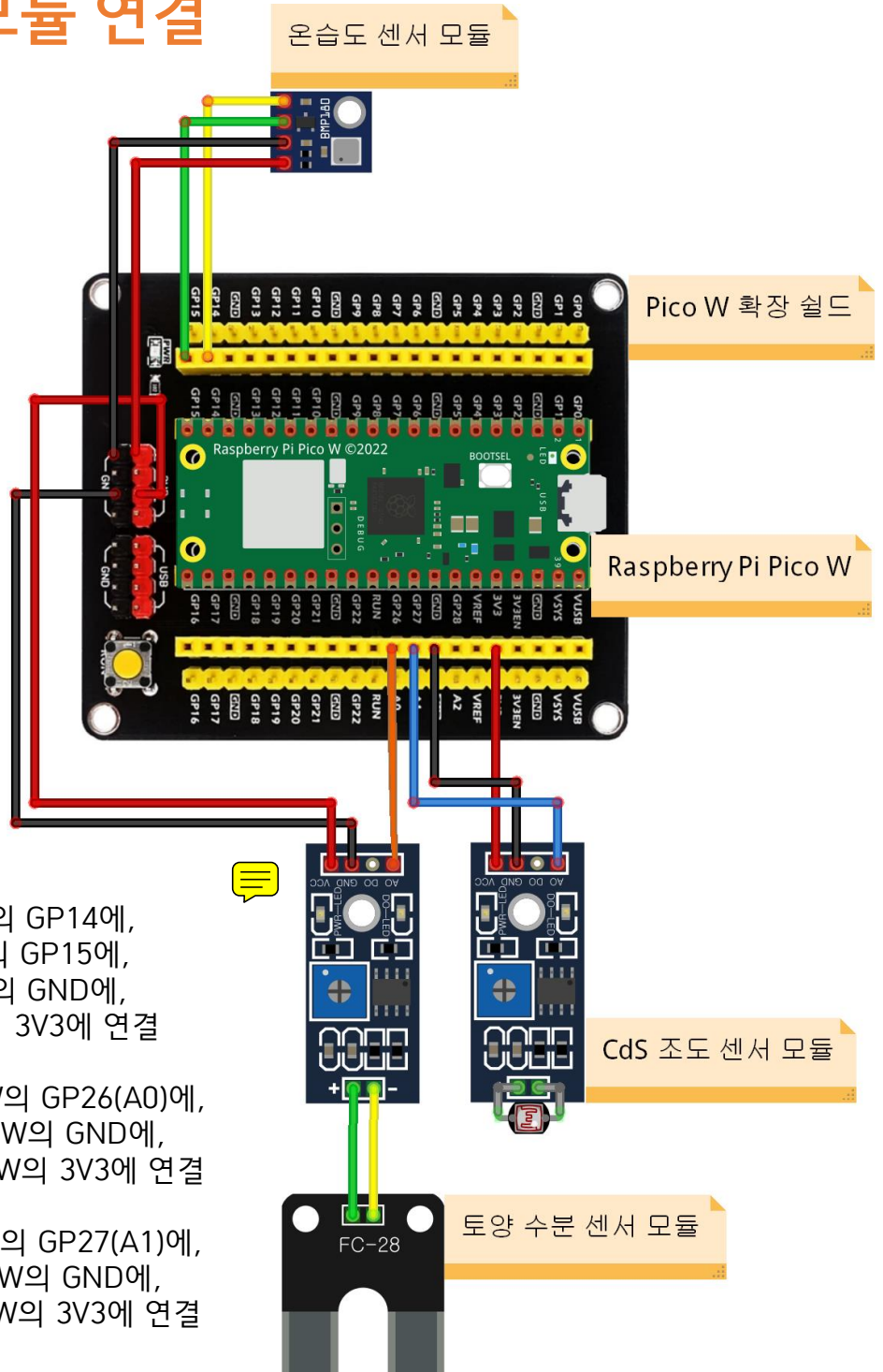
Thonny를 통해 소스 코드를 실행하지 않고, 라즈베리 파이 피코에 전원이 입력되면 소스 코드가 자동으로 실행되도록 하려면 상단 메뉴의 'File → Save as... → Raspberry Pi Pico'로 진입해 피코 내부 저장 공간에 'main.py'라는 이름으로 소스 코드를 저장해야 한다(20p 참고).

회로 구성 - 확장 쉴드 결합



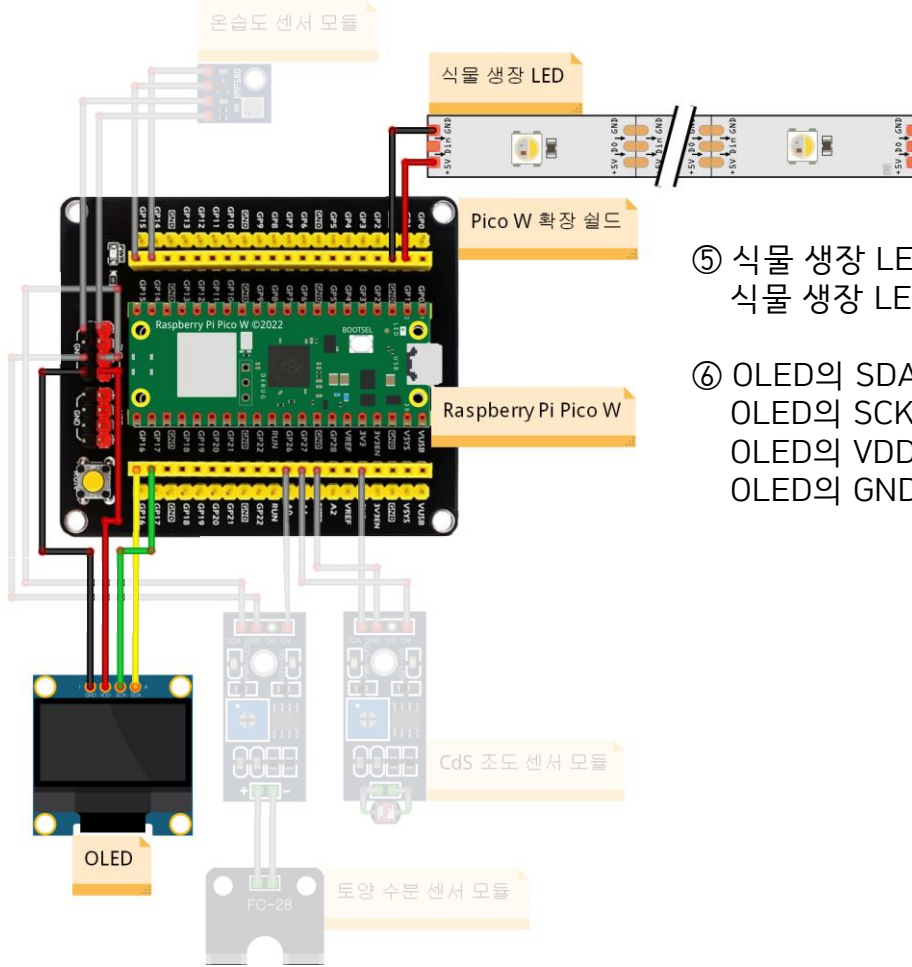
- ① 피코 W 확장 쉴드에 피코 W를 결합
※ 쉴드 뒷면 프린팅을 참고하여 피코 W의 방향 확인

회로 구성 - 센서 모듈 연결



- ② 온습도 센서 모듈의 SDA를 피코 W의 GP14에,
온습도 센서 모듈의 SCL을 피코 W의 GP15에,
온습도 센서 모듈의 GND를 피코 W의 GND에,
온습도 센서 모듈의 VIN을 피코 W의 3V3에 연결
- ③ 토양 수분 센서 모듈의 A0을 피코 W의 GP26(A0)에,
토양 수분 센서 모듈의 GND를 피코 W의 GND에,
토양 수분 센서 모듈의 VCC를 피코 W의 3V3에 연결
- ④ CdS 조도 센서 모듈의 A0을 피코 W의 GP27(A1)에,
CdS 조도 센서 모듈의 GND를 피코 W의 GND에,
CdS 조도 센서 모듈의 VCC를 피코 W의 3V3에 연결

회로 구성 - 식물 성장 LED, OLED 연결



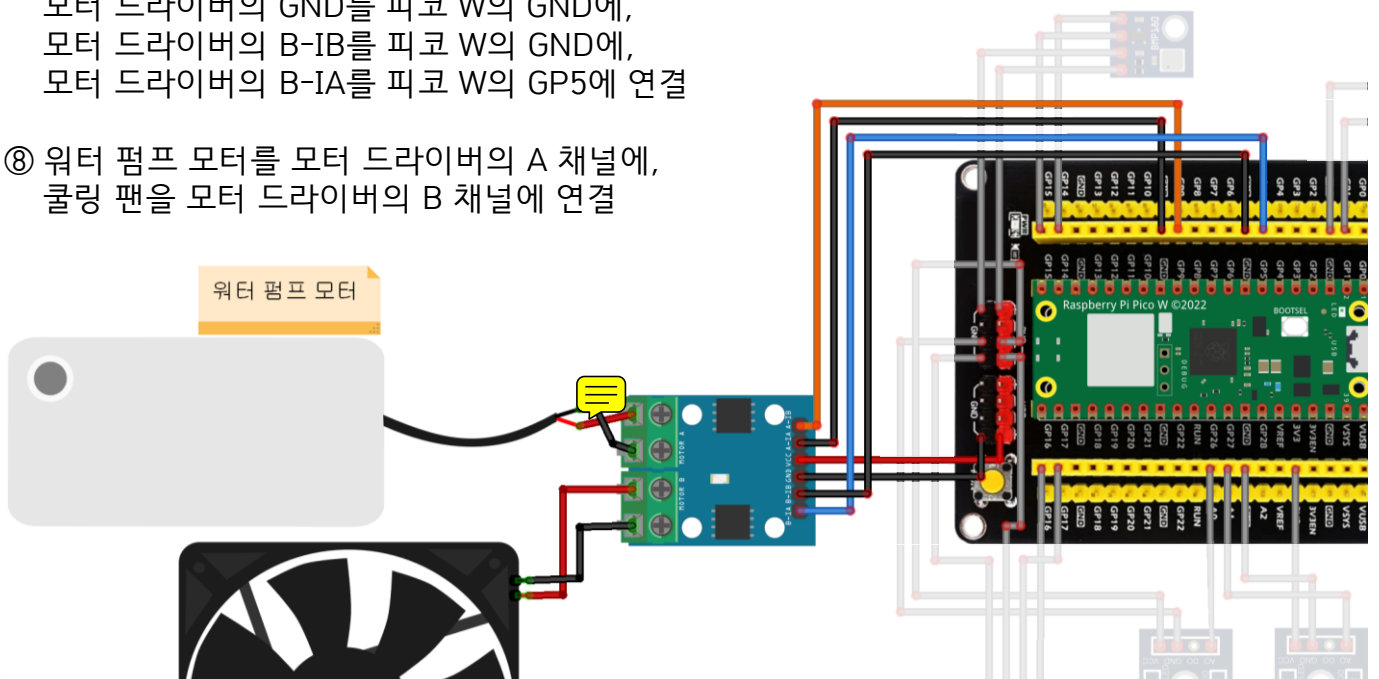
⑤ 식물 성장 LED의 (+)를 피코 W의 3V3에
식물 성장 LED의 (-)를 피코 W의 GND에 연결

⑥ OLED의 SDA를 피코 W의 GP16에,
OLED의 SCK를 피코 W의 GP17에,
OLED의 VDD를 피코 W의 3V3에,
OLED의 GND를 피코 W의 GND에 연결

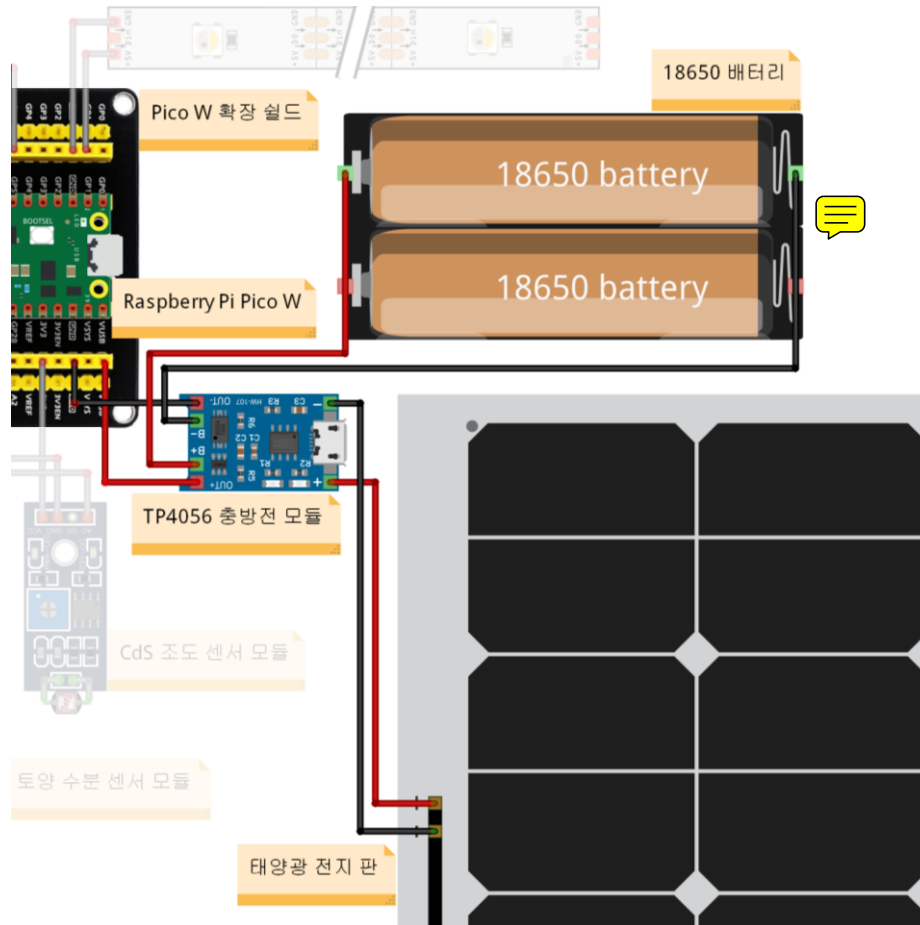
회로 구성 - 모터 연결

⑦ 모터 드라이버의 A-IB를 피코 W의 GP9에,
모터 드라이버의 A-IA를 피코 W의 GND에,
모터 드라이버의 VCC를 피코 W의 USB에,
모터 드라이버의 GND를 피코 W의 GND에,
모터 드라이버의 B-IB를 피코 W의 GND에,
모터 드라이버의 B-IA를 피코 W의 GP5에 연결

⑧ 워터 펌프 모터를 모터 드라이버의 A 채널에,
쿨링 팬을 모터 드라이버의 B 채널에 연결

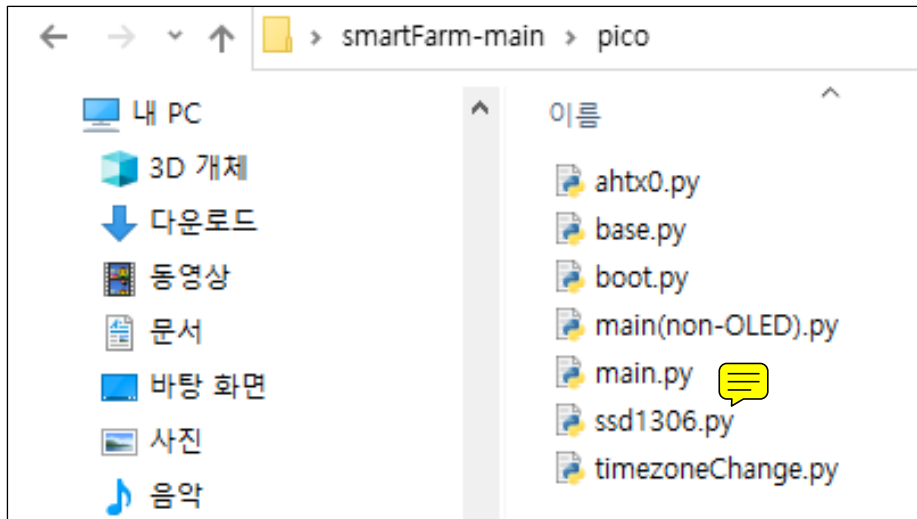


회로 구성 - 전원 연결



- ⑨ 충전전 모듈의 OUT-를 피코 W의 GND에,
충방전 모듈의 OUT+를 피코 W의 VUSB에 연결
- ⑩ 18650 배터리 홀더의 (+)를 충전전 모듈의 B+에,
18650 배터리 홀더의 (-)를 충전전 모듈의 B-에 연결
- ⑪ 태양광 전지 판의 (+)를 충전전 모듈의 (+)에,
태양광 전지 판의 (-)를 충전전 모듈의 (-)에 연결

사용되는 소스 코드



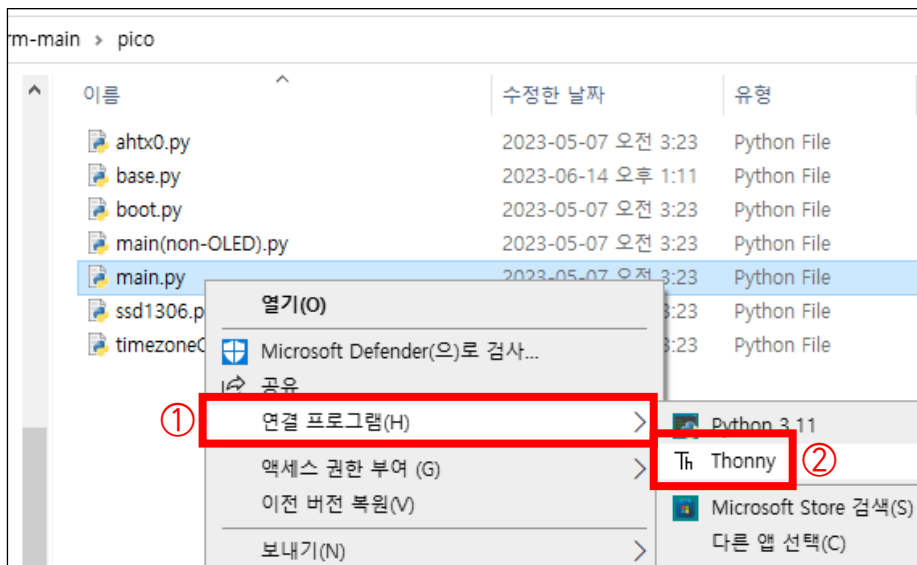
바탕화면에 내려 받았던(10p 참고) smartFarm-main 폴더의 'pico' 폴더에 진입한다.



- base.py: 데이터베이스(Firebase RTDB)에 데이터를 랜덤 값으로 업로드하는 코드
- boot.py: 전원을 켰을 때 가장 먼저 실행되는 보드 설정 코드
- main(non-OLED).py: OLED를 사용하지 않는 스마트 팜의 메인 실행 코드
- main.py: OLED를 사용하는 스마트 팜의 메인 실행 코드
- ssd1306.py: OLED를 사용하기 위한 모듈 파일(OLED 사용 시 같이 업로드)

OLED를 사용하는 경우: boot.py, main.py, ssd1306.py가 필요

OLED를 사용하지 않는 경우: boot.py, main(non-OLED).py가 필요



파일을 열 때는 파일을 우클릭한 후 '연결 프로그램(H) → Thonny'를 선택한다.

소스 코드 수정

base.py, main(non-OLED).py, main.py가 보드에 업로드 되면, 피코 W에 전원이 입력되었을 때 자동으로 소스 코드가 실행된다. 그 전에 피코 W가 연결할 Wi-Fi 공유기의 SSID(이름)와 패스워드를 수정해야 하며, Firebase RTDB의 주소도 수정해야 한다.

※ Firebase RTDB의 주소: 23p '데이터베이스 만들기'에서 확인

```
14
15 # 와이파이 연결하기
16 wlan = network.WLAN(network.STA_IF)
17 wlan.active(True)
18 if not wlan.isconnected():
19     wlan.connect("KT_GiGA_DC1E", "027612688m")
20     print("waiting for Wi-Fi connection", end= "...")
21     while not wlan.isconnected():
22         print(".", end="")
23         time.sleep(1)
24 else:
25     print(wlan.ifconfig())
26     print("WiFi is Connected")
27
28 # RTDB주소
29 url = "https://smartfarm-f867f-default-rtdb.firebaseio.com/"
```

- base.py: 19, 29번째 줄 수정

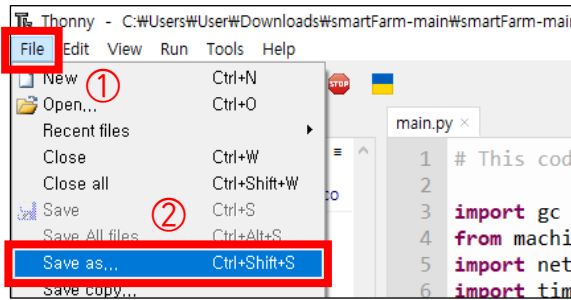
```
9
10 # 이름, 위도, 경도 표시하기(자신의 스마트팜 위치를 검색에서 넣어주세요.)
11 nickname = 'mtinet' # 닉네임 변수를 자신만의 닉네임으로 수정하세요.
12 lat = 37.49836 # 위도 변수를 자신의 위도 좌표로 수정하세요.
13 long = 126.9253 #경도 변수를 자신의 경도 좌표로 수정하세요.
14 SSID = "U+Net454C" # 공유기의 SSID를 따옴표 안에 넣으세요.
15 password = "DDAE014478" # 공유기의 password를 따옴표 안에 넣으세요.
16
17 # RTDB주소
18 url = "https://smartfarm-f867f-default-rtdb.firebaseio.com/"
19 mapor1 = "https://smartfarmlocation-default-rtdb.firebaseio.com/"
20
```

- main(non-OLED).py: 14~15, 18번째 줄 수정

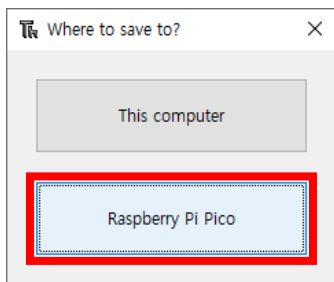
```
14
15 # 이름, 위도, 경도 표시하기(자신의 스마트팜 위치를 검색에서 넣어주세요.)
16 SSID = "KT_GiGA_DC1E" # 공유기의 SSID를 따옴표 안에 넣으세요.
17 password = "027612688m" # 공유기의 password를 따옴표 안에 넣으세요.
18 nickname = 'mtinet' # 닉네임 변수를 자신만의 닉네임으로 수정하세요.
19 lat = 37.49836 # 위도 변수를 자신의 위도 좌표로 수정하세요.
20 long = 126.9253 #경도 변수를 자신의 경도 좌표로 수정하세요.
21 moistureStandardValue = 10 # 수분센서 기준값을 설정하세요. 수분 센서 측정값이
22 temperatureStandardValue = 20 # 온도센서 기준값을 설정하세요. 온도 센서 측정값이
23
24 # RTDB주소
25 url = "https://smartfarm-f867f-default-rtdb.firebaseio.com/"
26 mapor1 = "https://smartfarmlocation-default-rtdb.firebaseio.com/"
27
```

- main.py: 15~16, 24번째 줄 수정

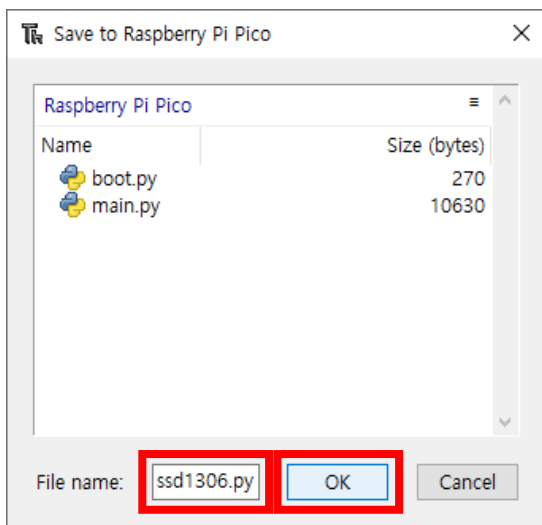
소스 코드 업로드



파이썬 파일(.py)을 연 상태에서 'File → Save as...'로 진입한다.



'Raspberry Pi Pico'를 선택한다.



File name에 파일 이름을 확장자명(.py)까지 적고 'OK' 버튼을 클릭한다.

※ main(non-OLED).py 파일은 저장할 때 파일 이름을 'main.py'로 저장하여야 함



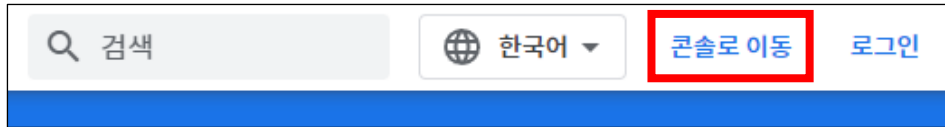
피코 W에 업로드 할 각각의 파일에 대하여 위 과정을 수행한다.

OLED를 사용하는 경우: boot.py, main.py, ssd1306.py가 필요

OLED를 사용하지 않는 경우: boot.py, main(non-OLED).py가 필요

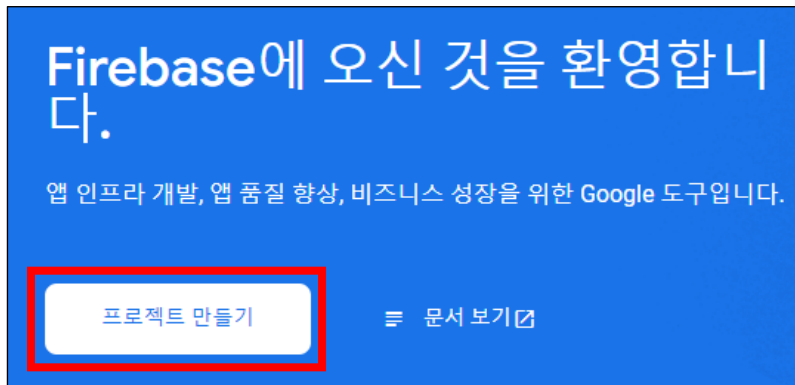
피코 W에 업로드된 파이썬 소스 코드는 피코 W에 전원이 입력됐을 때 자동으로 실행된다.

Firebase 프로젝트 만들기

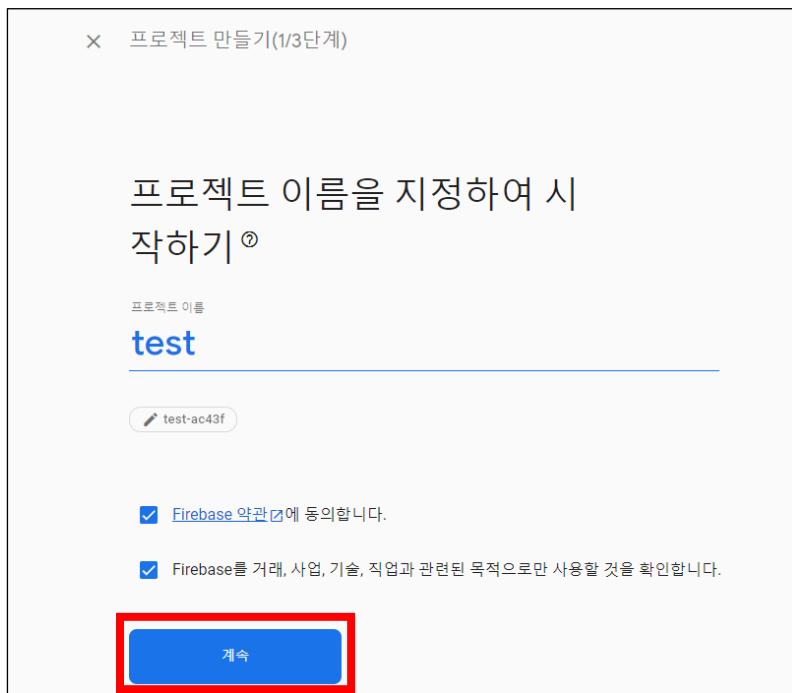


<https://firebase.google.com/>

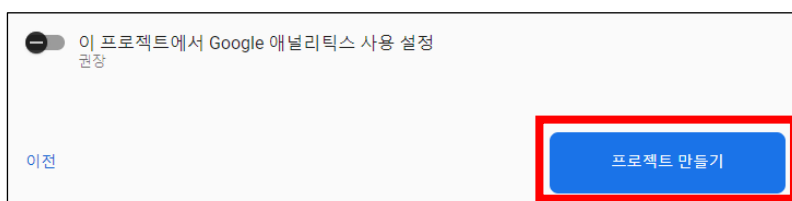
구글 계정으로 로그인한 뒤 우측 상단 '콘솔로 이동' 버튼을 클릭한다.



'프로젝트 만들기' 버튼을 클릭한다.

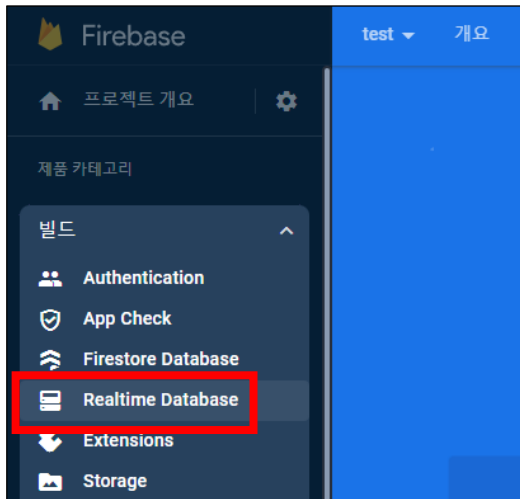


프로젝트 이름을 지정하고 체크박스에 모두 체크한 뒤 '계속' 버튼을 클릭한다.

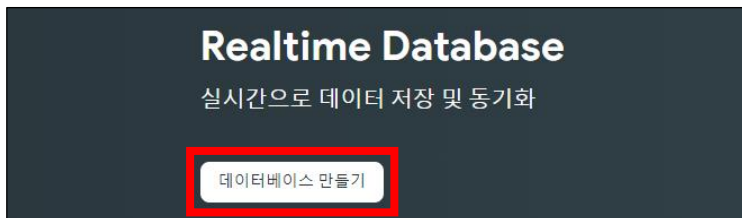


Google 애널리틱스 사용은 해제하고 '프로젝트 만들기' 버튼을 클릭한다.

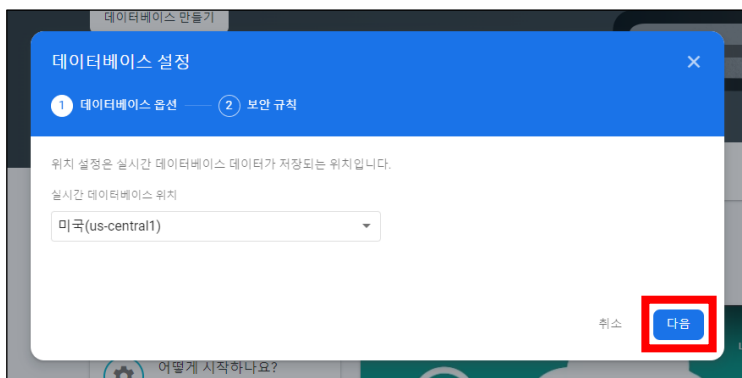
데이터베이스 만들기



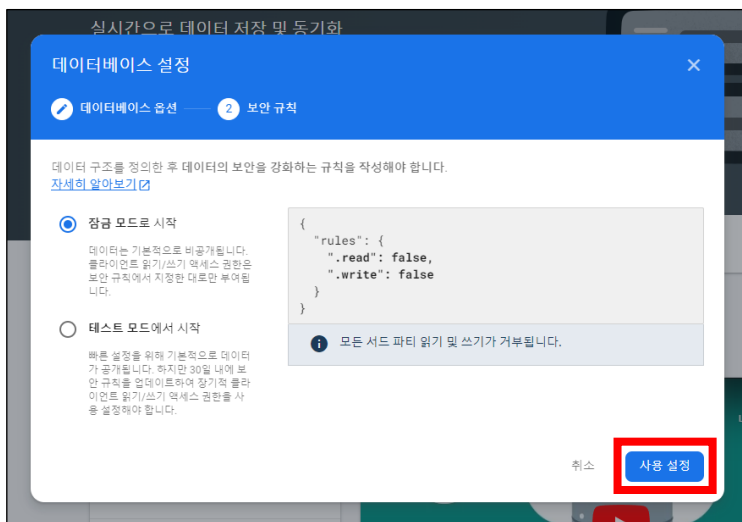
프로젝트 화면 좌측 메뉴에서 'Realtime Database'로 진입한다.



'데이터베이스 만들기' 버튼을 클릭한다.



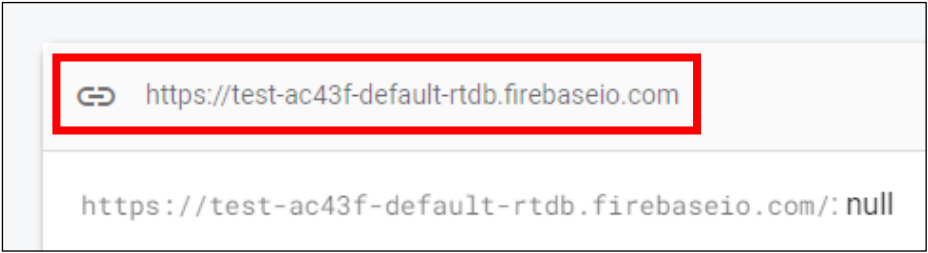
데이터베이스의 물리적 위치를 선택하고 '다음' 버튼을 클릭한다.



'잠금 모드로 시작'으로 놔두고 '사용 설정' 버튼을 클릭한다.

※ 설정은 추후 언제든지 변경 가능

데이터베이스 만들기



이후 나타나는 링크가 데이터베이스의 접속 주소로, 소스 코드 수정(19p 참고) 시 필요하다.

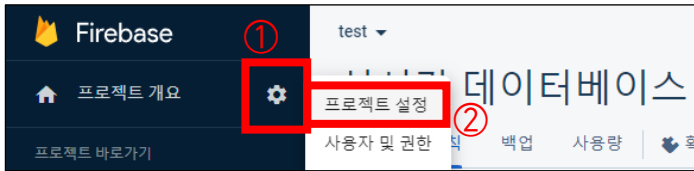


규칙 탭에서 데이터베이스의 읽기(“.read”) 권한과 쓰기(“.write”) 권한을 설정해줄 수 있으며, 기본값은 ‘false’로 설정되어 있다. ‘false’를 ‘true’로 바꾼 뒤 ‘게시’ 버튼을 누르면 누구나 데이터베이스에 접근해서 데이터를 읽고 쓸 수 있다.

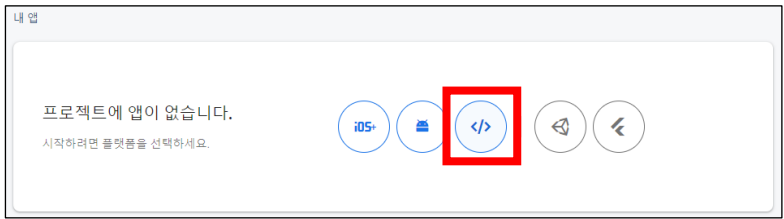
보안 및 트래픽에 따른 결제 비용 증가에 영향을 미칠 수 있으니 테스트를 할 때만 ‘true’로 설정하고 실제 사용할 때는 ‘false’로 바꾸어 접근한 사람의 사용자 권한에 따라 사용하게 하거나, 액세스 토큰을 부여 받아 사용하게 해야 한다.

이렇게 만들어진 데이터베이스에 실제로 데이터를 쓰거나 읽으려면
데이터베이스의 config 정보를 얻어야 한다.

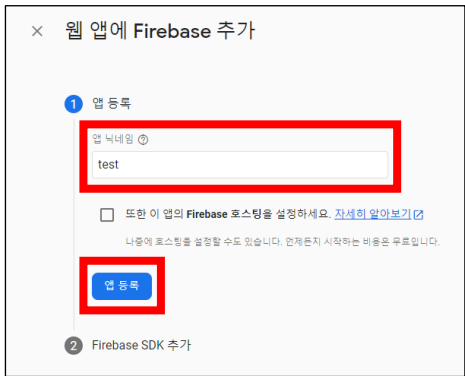
데이터베이스 config 정보 얻기



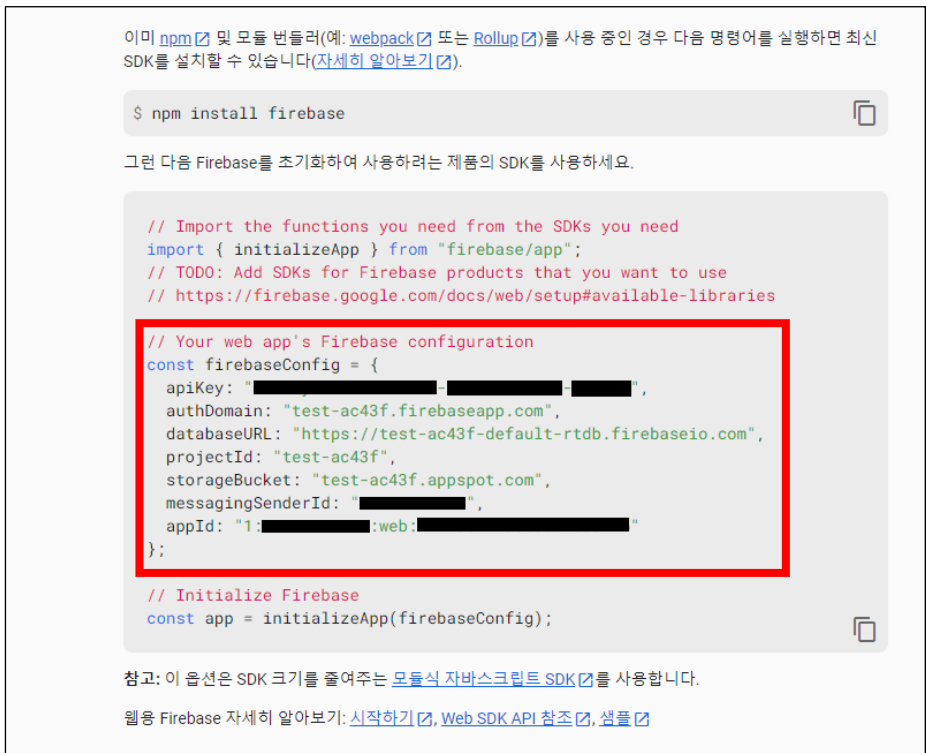
‘프로젝트 개요’ 옆 기어 모양 설정 버튼을 누르고 ‘프로젝트 설정’으로 진입한다.



플랫폼은 WEB(‘</>’ 아이콘)을 선택하여 웹 앱 추가

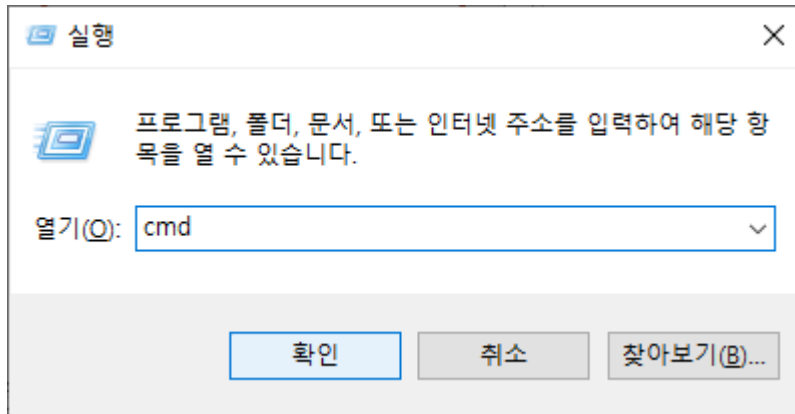


앱 닉네임 기재하고 ‘앱 등록’ 버튼 클릭

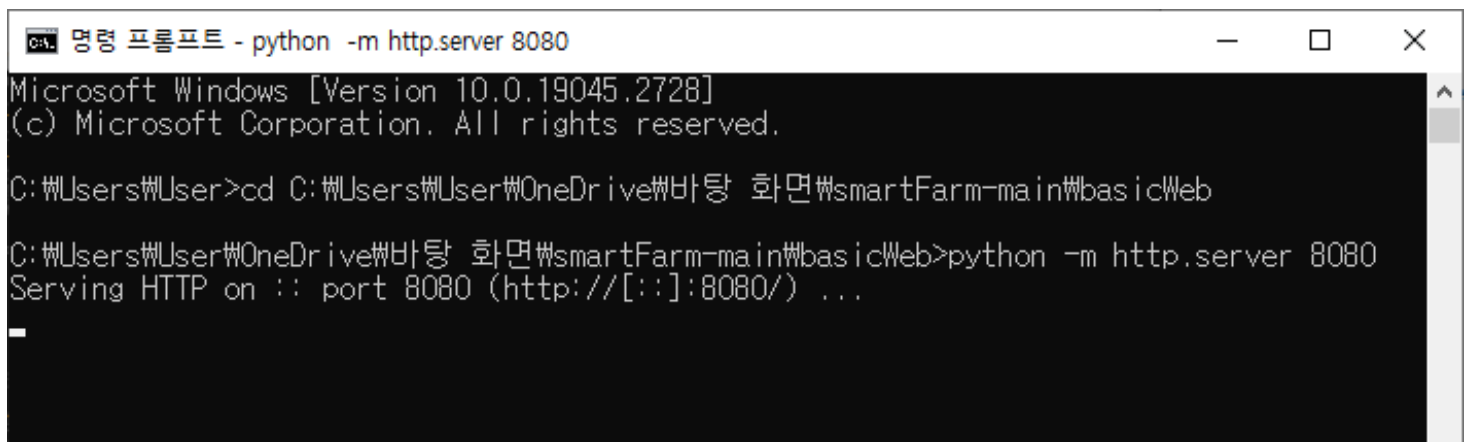


firebaseConfig 내부의 config 정보를 복사해서, 내려 받았던 smartFarm-main 폴더의 web/public/js/firebase.js 파일에 붙여 넣으면 자신의 RTDB에 접근해서 온도, 습도, 조도 측정 및 LED, Fan 제어를 위한 smartFarm 제어용 웹 서비스를 바로 이용할 수 있다.

컴퓨터에서 직접 웹페이지 서비스하기

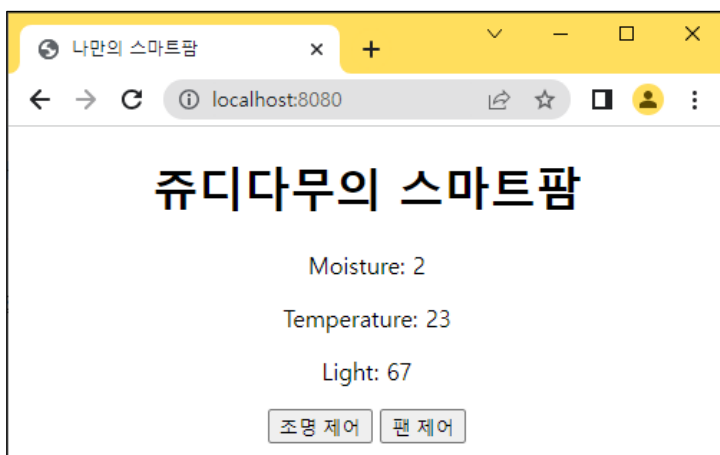


키보드의 Win 키와 'R' 키 동시 입력 후 'cmd'를 입력하고 '확인' 버튼을 클릭한다.



“cd (폴더 경로)” 명령어를 입력하여, 바탕화면에 내려 받았던 smartFarm-main 폴더 안의 'basicWeb' 폴더로 이동한다.

폴더 이동이 완료되었으면 “python -m http.server 8080” 명령어를 입력한다.



웹 브라우저 주소창에 'localhost:8080'를 입력해 접속하여 정상 작동되고 있는지 확인한다.

이런 식으로 컴퓨터로 직접 웹페이지를 서비스하려면 컴퓨터를 계속 켜놓아야 하고, 외부 네트워크에서 접속하기 위해서는 각종 네트워크 설정이 필요하기 때문에 GitHub Pages를 사용해 웹 페이지를 서비스하려는 것이다.

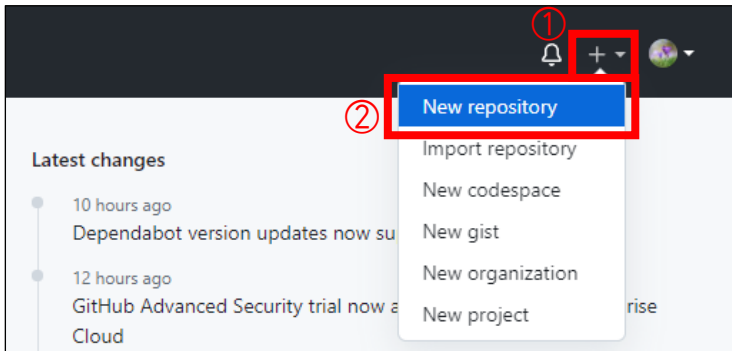
GitHub Repository 만들기



GitHub Pages는 HTML, CSS, JavaScript로 구현되는 정적 웹페이지를 호스트하는 서비스를 무료로 제공하고 있다.

<https://github.com>

GitHub 사이트에 접속하여 회원가입(Sign up)을 진행한다.



GitHub 메인 페이지 우측 상단의 '+' 버튼을 클릭하고 'New repository' 버튼을 클릭한다.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner *

Crocus726

Repository name *

smartFarm

smartFarm is available.

Great repository names are short and memorable. Need inspiration? How about [improved-waffle?](#)

Description (optional)

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

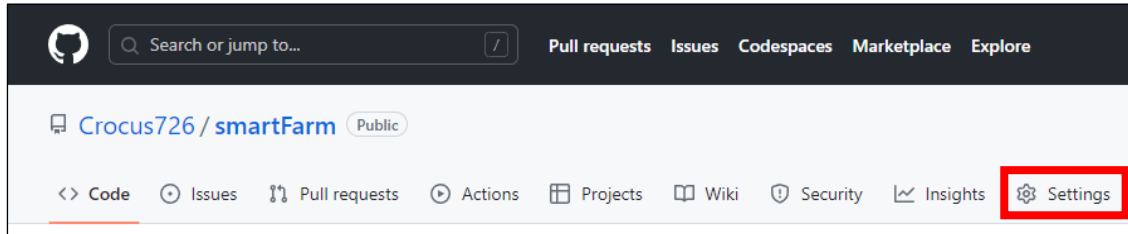
This will set `main` as the default branch. Change the default name in your [settings](#).

③ You are creating a public repository in your personal account.

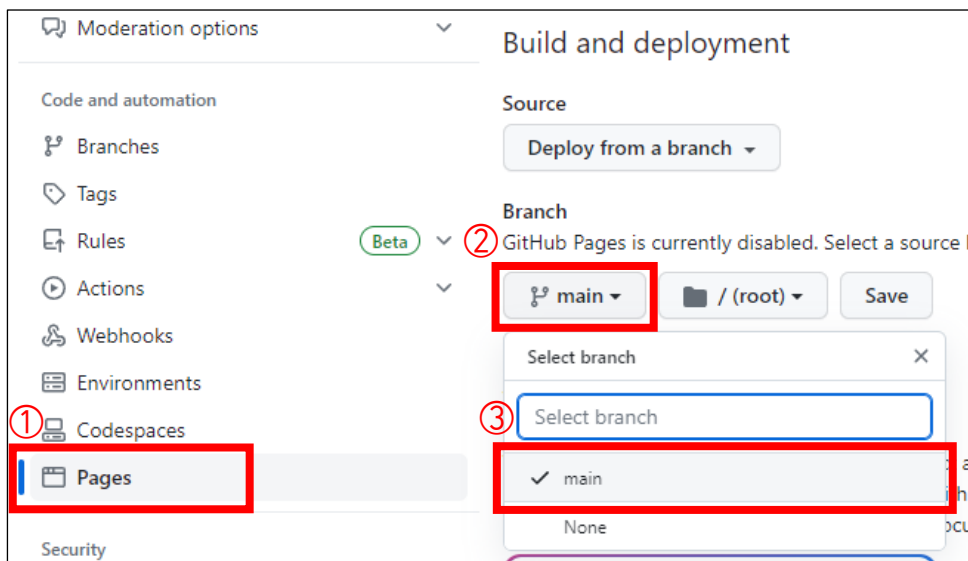
Create repository

Repository name을 입력하고, 기본 해제 되어 있는 'Add a README file' 체크박스는 체크하여 'Create repository' 버튼을 클릭한다.

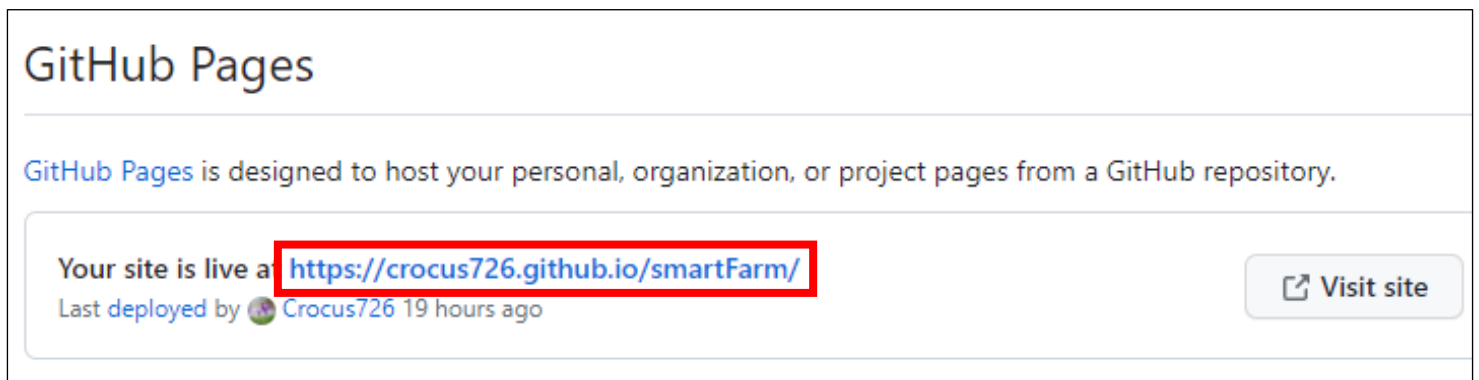
GitHub Pages 사용 설정하기



만든 repository 페이지에서 상단의 'Settings'로 진입한다.

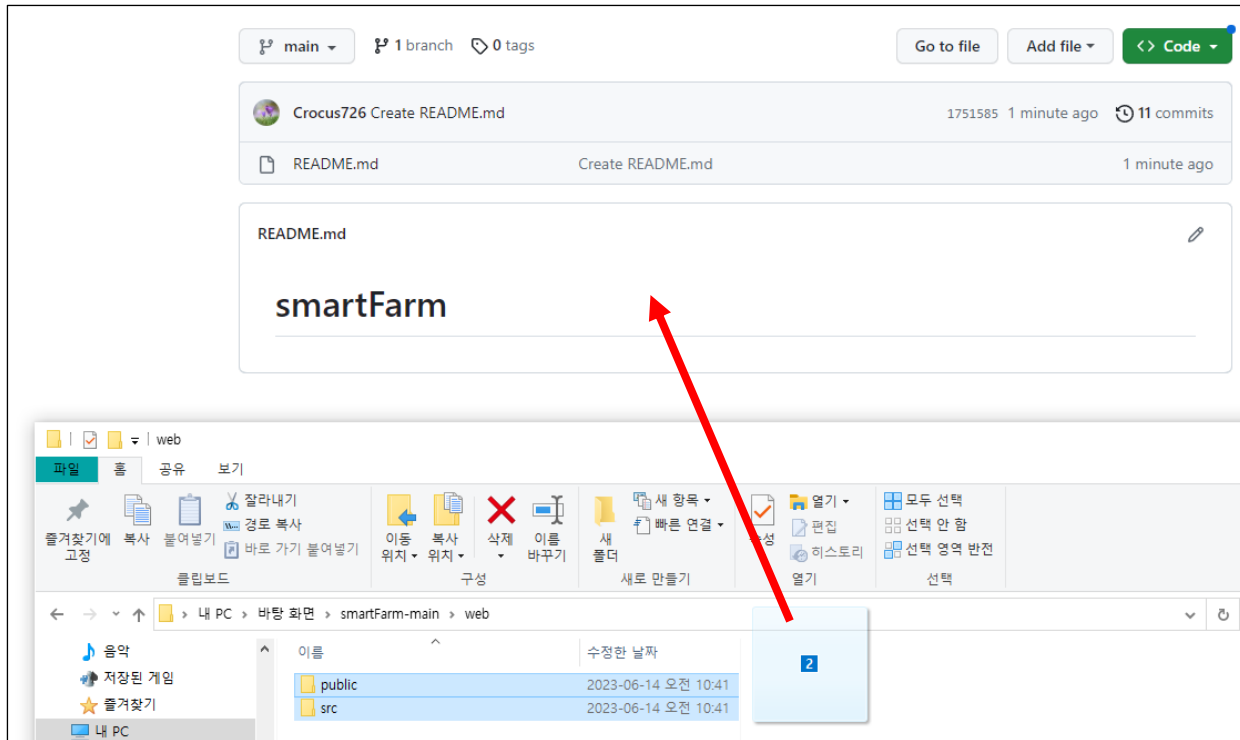


좌측 메뉴에서 'Pages'에 진입한 뒤 Branch를 'main'으로 선택한다.

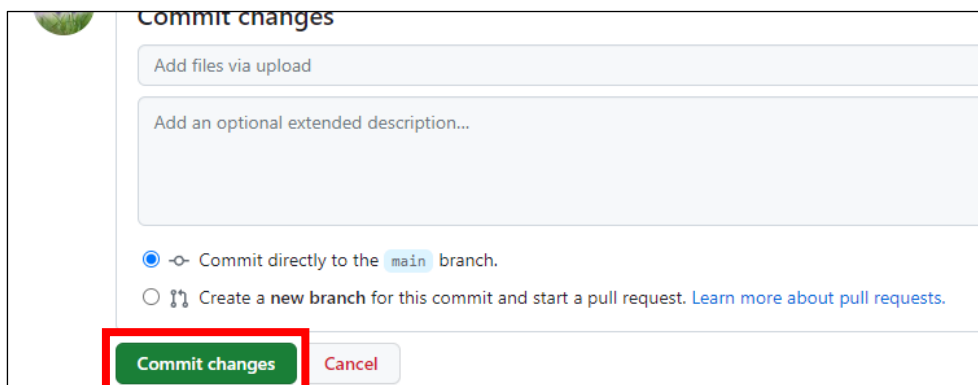


잠시 기다린 뒤(최대 5분) 새로 고침 하여 사이트가 서비스되고 있다는 문구와 접속 링크가 표시되는지 확인한다.

GitHub Pages 사용 설정하기



바탕화면에 내려 받았던 smartFarm-main 폴더의 'web' 폴더 안에 있는 'public' 폴더와 'src' 폴더를 모두 repository 페이지에 끌어다 놓는다(드래그&드랍).



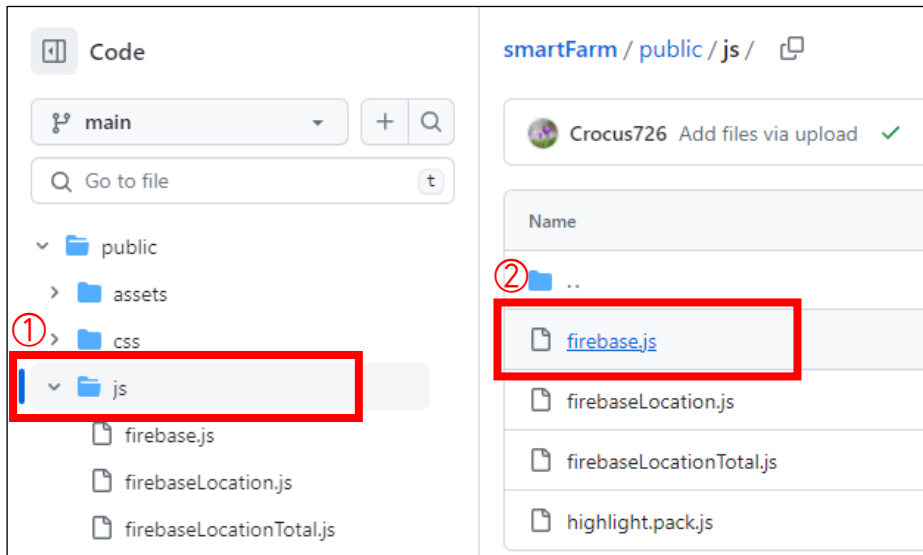
'Commit changes' 버튼을 클릭한다.



조금 기다린 뒤 [https://\(GitHub_ID\).github.io/smartFarm/public](https://(GitHub_ID).github.io/smartFarm/public)으로 접속해본다.

아직 Firebase RTDB 주소를 지정해주지 않았기 때문에 아무 값도 뜨지 않는다.

GitHub Pages 사용 설정하기



repository 페이지에서 'public/js' 폴더로 이동해 'firebase.js' 파일을 클릭한다.

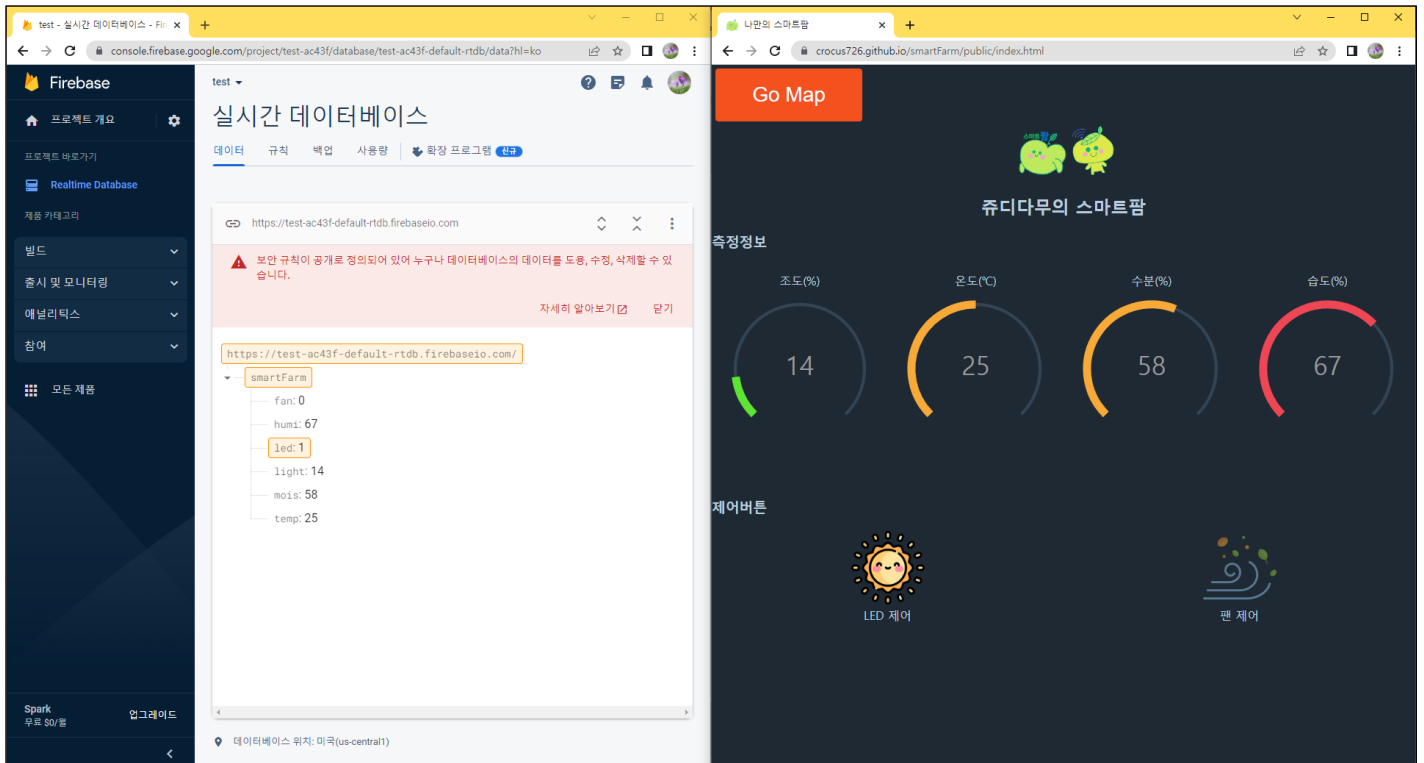


코드 우측 상단의 연필 모양 'Edit this file' 버튼을 클릭한다.



12~18번째 줄의 Firebase RTDB config 정보를 자신의 RTDB config 정보(24p 참고)로 수정하고 'Commit changes... → Commit changes'를 클릭한다.

최종 작동 확인



다음의 사항을 확인하여 본다.

- ① 라즈베리 파이 피코가 측정한 값이 Firebase RTDB에 제대로 업로드 되는지
- ② Firebase RTDB에 저장된 값과 웹페이지에 표시되는 값이 동일한지
- ③ 식물 생장 LED와 쿨링 팬을 제어할 때 Firebase RTDB에 저장된 값이 제대로 바뀌고 라즈베리 파이 피코가 LED와 팬을 제대로 제어하는지