

라즈베리파이 IOT 키트 V2.0

사용자 설명서



목차

라즈베리파이 IOT 키트 V2.0 사용자 설명서	1
가) 예제코드 다운로드 및 실행	3
1. IOT 키트 예제 다운로드 및 설치	3
2. 라즈베리파이 5 사용 시 lgpio 설치	4
나) 라즈베리파이 제어	5
1. GPIO 제어 하기	5
① 라즈베리파이에 LED, 스위치 연결	5
② GPIO 출력 제어하기(LED)	8
③ 스위치 입력 테스트	9
2. 릴레이 보드 제어하기	10
① 릴레이 보드 연결하기	10
② 릴레이 동작 테스트하기	12
3. PMS7003 먼지센서 사용하기	13
① 먼지센서 연결하기	13
② 먼지센서 동작시켜 데이터 받아보기	14
다) Blynk 로 원격 제어하기	17
1. Blynk 소개	17
2. Blynk 설치	17
① 라즈베리파이에 Blynk - python 버전 설치	17
② 스마트폰에 Blynk 설치	18
3. Blynk 위젯 설정하기	18
① GPIO 제어용 템플릿 구성	20
② 먼지센서 데이터 수신용 템플릿 구성	23
③ 디바이스 추가하기	25
4. 디바이스 토큰 정보 확인하기	26
5. Blynk 로 GPIO 제어하기	27
① 하드웨어 구성	27
② Blynk GPIO 디바이스 선택	27
③ 예제코드 실행	27
6. Blynk 로 먼지센서 데이터 받아오기	29
① 하드웨어 구성	29
② Blynk PMS 디바이스 선택	29
③ 예제코드 실행	29
라) 통합하여 IOT 환경 구축하기	31
1. 하드웨어 및 Blynk 위젯 구성	31
① GPIO 구성	31
② Blynk 위젯 구성	31
2. 예제 실행	32

가) 예제코드 다운로드 및 실행

라즈베리파이 IOT 키트 V2.0 를 사용하기 위해서는 라즈베리파이 OS 가 설치된 라즈베리파이와 라즈베리파이에 접속할 수 있는 환경이 갖춰져야 합니다. (모니터/VNC/SSH 등)

라즈베리파이 IOT 키트 V2.0 의 예제코드는 파이썬을 이용해 작성되었습니다.
소스 코드는 엘레파츠 GitHub 에서 다운받으실 수 있습니다.

> 엘레파츠 IOT KIT GitHub 페이지
<https://github.com/eleparts/iotkit>

아래에서는 라즈베리파이의 터미널 환경에서 예제코드를 다운로드 받아 실행하는 과정을 자세히 진행하도록 하겠습니다.

1. IOT 키트 예제 다운로드 및 설치

라즈베리파이의 터미널 창을 열고 위의 예제코드를 다운받는 명령어를 실행해 줍니다.
GitHub 의 저장소에 업로드 되어있는 프로그램을 다운로드(복사)해 주는 명령어는 **git clone https://github.com/eleparts/iotkit** 입니다.

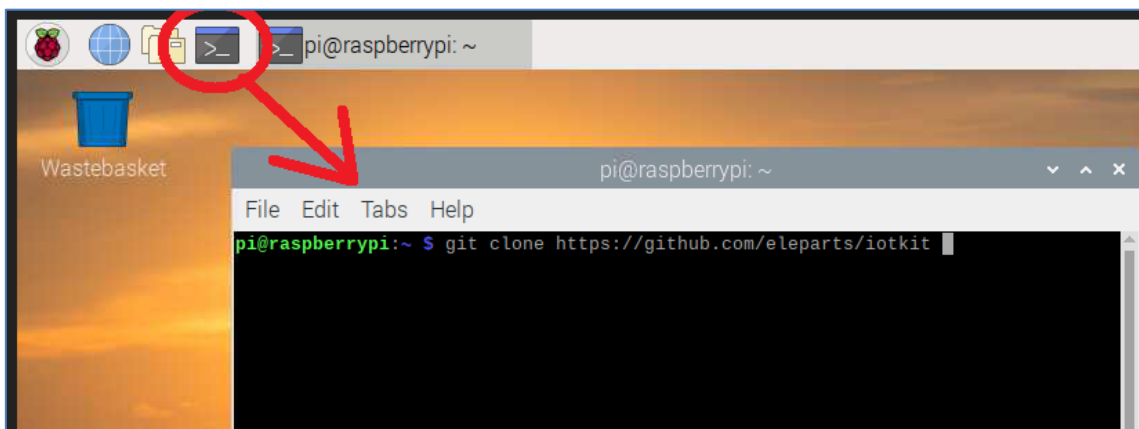


그림 1 GitHub 저장소 복사

기본 예제파일은 iotkit/example 경로에 있으나 추가 예제코드의 경우 start.sh 스크립트를 이용하여 다운로드 해 주어야 합니다.

먼저 **cd iotkit** 를 입력해 다운로드 한 디렉터리로 이동해 줍니다.
※ 디렉터리나 파일명 입력 시 tab 키를 눌러 자동완성을 이용하면 편리합니다.

그리고 **chmod +x start.sh** 명령어로 스크립트 실행 권한을 추가해 준 뒤 **./start.sh** 명령어로 스크립트 파일을 실행해 줍니다.

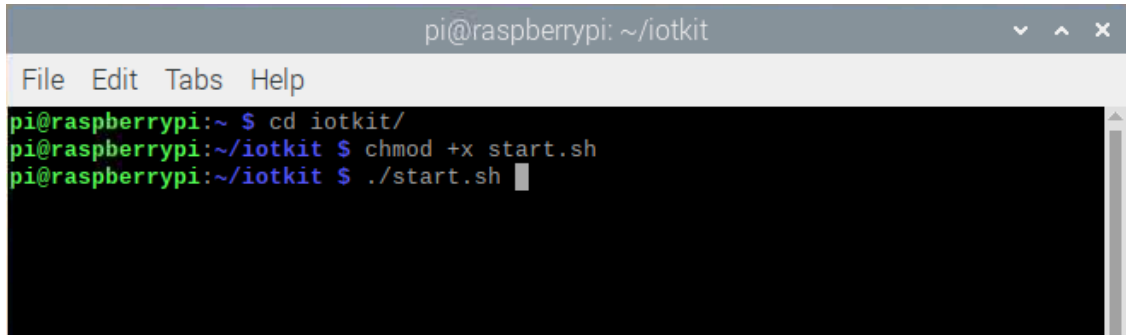


그림 2 다운로드 스크립트 실행

스크립트 프로그램이 실행되면 나머지 예제 코드도 자동으로 다운로드 됩니다.

다운로드가 완료되면 키트 구성 부품을 라즈베리파이에 연결하고 예제 프로그램을 테스트 해 볼 수 있습니다.

예제 코드의 동작 설명은 GitHub 에서 직접, 혹은 라즈베리파이에서 예제코드 파일을 열어 확인 가능합니다.

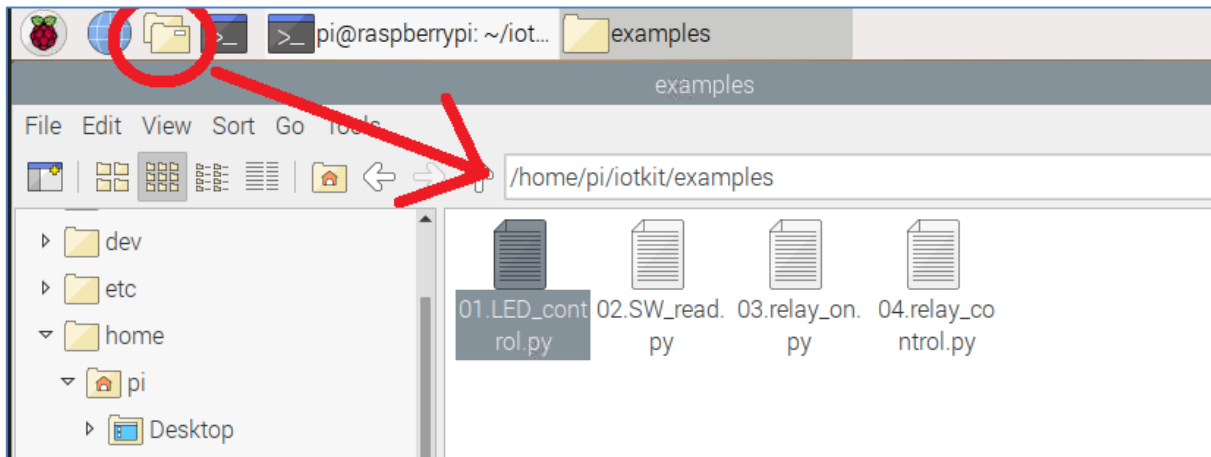


그림 3 GUI 환경에서 파일 확인

위 경로에서 .py 파일을 열 경우 예제 코드를 직접 동작 가능한 파이썬용 Thonny IDE 로 파일이 열리게 되어, 예제 코드를 바로 실행해 볼 수 있습니다.

※ 한글이 정상적으로 표시되지 않는 경우 터미널 창에 `sudo apt install fonts-unfonts-core` 를 입력 해 한글 폰트 설치 후 **재 부팅**해 주면 됩니다.

2. 라즈베리파이 5 사용 시 lgpio 설치

만약 **라즈베리파이 5** 를 사용한다면 기존 RPI.GPIO 라이브러리를 사용하기 위해 pi5 대응 GPIO 라이브러리인 lgpio 를 추가로 설치해 주어야 합니다.

터미널 창에 아래 명령어를 입력해 주시면 자동으로 설치가 진행됩니다.

```
pip install --break-system-packages rpi-lgpio
```

나) 라즈베리파이 제어

1. GPIO 제어 하기

① 라즈베리파이에 LED, 스위치 연결

하드웨어에서 가장 기본적인 테스트는 LED 점멸(깜빡이기) 테스트입니다.

라즈베리파이에 회로를 연결하기 위해 브레드보드에 LED 및 스위치 등을 꽂아 회로를 구성해 주도록 하겠습니다.

브레드보드는 아래의 검은 선을 따라 5 개의 홀이 서로 연결되어 있어 각 부품의 핀을 서로 연결시켜 줍니다. 또한, 세로줄인 빨간색, 파란색 선은 길게 이어져 있어 전원을 연결해 전원 공급용 선으로 사용하면 매우 편리합니다.



그림 4 브레드보드

점퍼케이블로 핀을 연결하실 때에는 색을 정해 (+ (VCC)는 빨간색, - (GND)는 검은색, GPIO(데이터통신) 핀은 초록, 노랑, 흰색 등등) 연결해 주면 추후 회로 확인에 큰 도움이 됩니다.

※ 주의: 핀 연결 시 절대 + (VCC)와 - (GND)가 직접 연결되어서는 안됩니다. (쇼트/합선으로 파손 발생)

브레드보드에 꽂는 소자 중에서 **LED 는 극성이, 스위치(다리 4 개)는 연결 방향이 있는 소자**이므로 꽂을 때 주의해 사용해 주셔야 합니다.

LED 는 다리가 휘어진/긴 다리가 + 극성이며, 짧은 쪽이 - 극 입니다.

(다리를 절단한 경우 LED 원통 부분 하단의 깎인 방향(-), 내부 금속 모양 등으로도 구별이 가능합니다.)

포함된 **스위치**는 제품 하단에 막대가 그려져 있으며, 해당 하단 막대 그림을 기준으로 같은 방향의 핀은 항상 연결되어 있습니다. (4 개 다리 중 각각 2 개는 서로 연결되어있고, 스위치를 누르면 전부 연결됩니다.)

라즈베리파이에 부품 연결은 아래 사진을 따라 똑같이 해 주시면 됩니다.

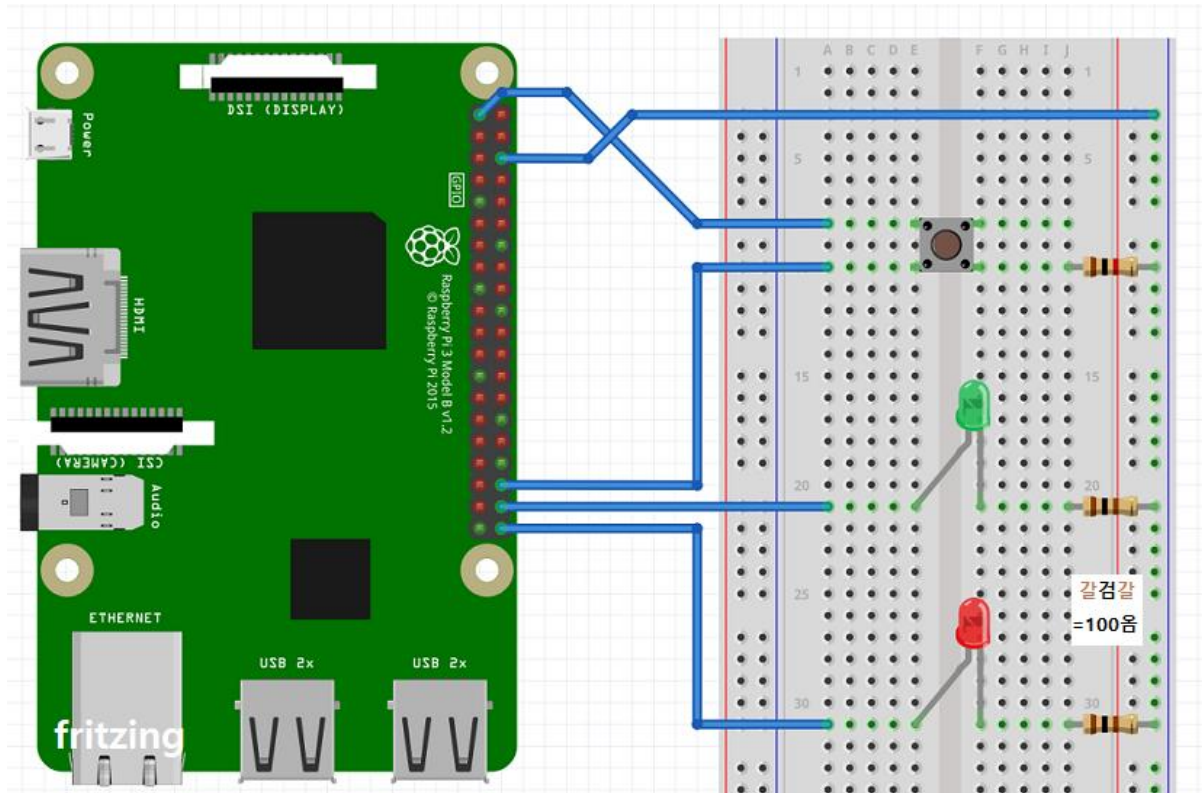


그림 5 라즈베리파이에 LED 및 스위치 연결

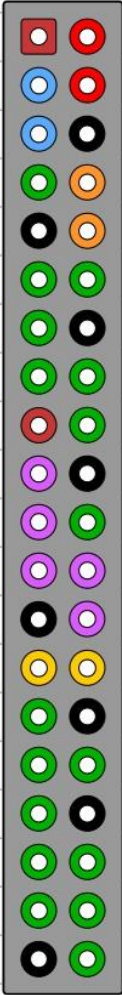
저항은 방향이 없으며, 저항 값은 띠의 색상으로 구분 가능합니다.
LED 에는 갈색/검정/갈색/(금색)띠의 저항(100 옴)을 LED 옆에 연결해 줍니다.

스위치의 우측에 연결된 저항은 1k~10k 옴의 **풀다운 저항**으로, 스위치가 눌리지 않은 상태에서 0V 쪽에 연결되어 GPIO의 핀의 **전압을 안정시켜**(풀 다운 = GND/ 풀 업 = VCC)주는 역할을 해 줍니다.
라즈베리파이는 프로그램에서 자체 내부 풀 업/ 풀 다운을 지원하기 때문에 꼭 연결하지 않아도 되지만 처음 공부하시는 경우 연결해 주시는 것을 추천합니다.

라즈베리파이의 핀 중에서 맨 위의 좌측이 3.3V 핀이며, 위에서 세 번째 줄 우측에 연결된 핀은 GND 핀입니다.

핀 번호는 아래 라즈베리파이 GPIO 핀 배치도를 참고해 주시면 됩니다.

※ 참고: 라즈베리파이 GPIO 핀 배치도

Raspberry Pi 3 GPIO Header			
Pin#	NAME		NAME Pin#
01	3.3v DC Power		DC Power 5v 02
03	GPIO02 (SDA1 , I²C)		DC Power 5v 04
05	GPIO03 (SCL1 , I²C)		Ground 06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14 08
09	Ground		(RXD0) GPIO15 10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18 12
13	GPIO27 (GPIO_GEN2)		Ground 14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23 16
17	3.3v DC Power		(GPIO_GEN5) GPIO24 18
19	GPIO10 (SPI_MOSI)		Ground 20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25 22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08 24
25	Ground		(SPI_CE1_N) GPIO07 26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC 28
29	GPIO05		Ground 30
31	GPIO06		GPIO12 32
33	GPIO13		Ground 34
35	GPIO19		GPIO16 36
37	GPIO26		GPIO20 38
39	Ground		GPIO21 40

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

그림 6 라즈베리파이 핀 배치도

라즈베리파이의 끝부분이 1,2 번 핀 방향이며 안쪽이 1 번, 바깥쪽이 2 번 핀입니다.

핀 번호는 물리적인 핀 순서이며 프로그램 내부에서는 GPIOxx 번호(BCM 번호)를 사용합니다.

② GPIO 출력 제어하기(LED)

라즈베리파이의 GPIO의 출력은 3.3V로 HIGH / LOW (3.3V / 0V)로 제어됩니다.
다만, 전압/전류는 눈으로 바로 확인이 불가능하기 때문에 LED를 연결하여 동작을 확인할 수 있습니다.

위에서 다운받은 예제 코드를 이용하여 연결된 LED를 제어해 보도록 하겠습니다.

cd examples 명령어로 예제코드가 있는 디렉터리로 이동합니다.

ls 명령어로 현재 디렉터리에 있는 파일 목록을 확인 가능합니다. (-l은 상세보기 옵션)

그리고 **python 01.LED_control.py** 명령어로 실행해 주면 LED가 동작하게 됩니다.

```

pi@raspberrypi: ~/iotkit/examples
File Edit Tabs Help
pi@raspberrypi:~/iotkit $ cd examples/
pi@raspberrypi:~/iotkit/examples $ ls -l
total 16
-rw-r--r-- 1 pi pi 782 Jul 17 04:55 01.LED_control.py
-rw-r--r-- 1 pi pi 1406 Jul 17 04:55 02.SW_read.py
-rw-r--r-- 1 pi pi 559 Jul 17 04:55 03.relay_on.py
-rw-r--r-- 1 pi pi 1998 Jul 17 04:55 04.relay_control.py
pi@raspberrypi:~/iotkit/examples $ python3 01.LED_control.py
  
```

그림 7 예제 01.LED_control.py

예제를 실행시켜 주면 아래와 같이 LED가 번갈아 가며 켜지게 됩니다. (5회 반복 후 종료)

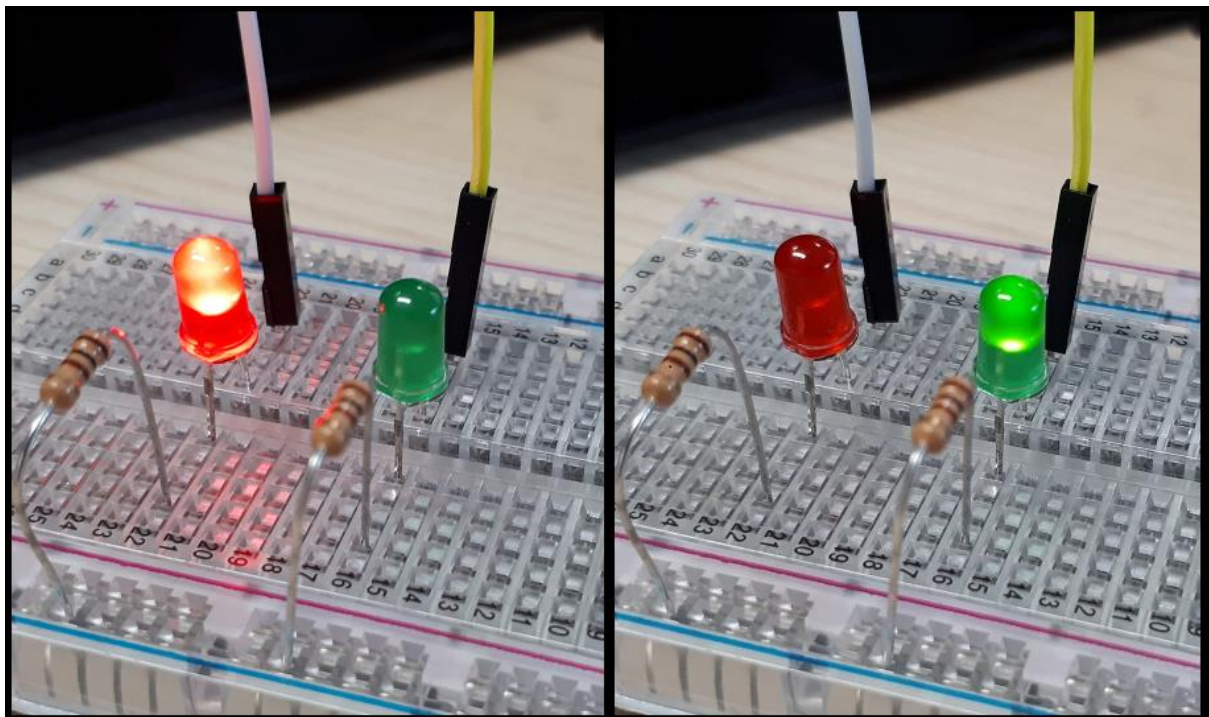
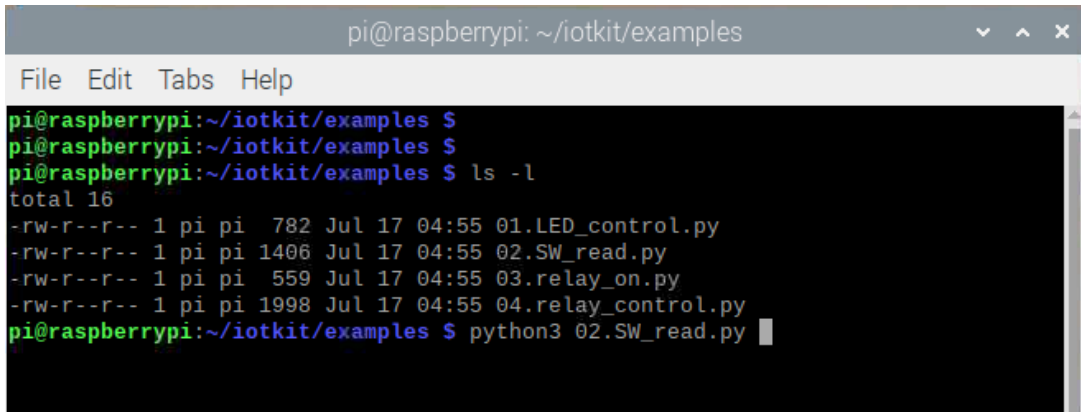


그림 8 01.LED_control.py 실행결과

③ 스위치 입력 테스트

스위치 입력 예제도 동일 디렉터리에서 `python 02.SW_read.py` 명령어를 입력해 실행해 주시면 됩니다



```

pi@raspberrypi: ~/iotkit/examples
File Edit Tabs Help
pi@raspberrypi:~/iotkit/examples $
pi@raspberrypi:~/iotkit/examples $
pi@raspberrypi:~/iotkit/examples $ ls -l
total 16
-rw-r--r-- 1 pi pi 782 Jul 17 04:55 01.LED_control.py
-rw-r--r-- 1 pi pi 1406 Jul 17 04:55 02.SW_read.py
-rw-r--r-- 1 pi pi 559 Jul 17 04:55 03.relay_on.py
-rw-r--r-- 1 pi pi 1998 Jul 17 04:55 04.relay_control.py
pi@raspberrypi:~/iotkit/examples $ python3 02.SW_read.py
  
```

그림 9 예제 02.SW_read.py

예제 실행 시 빨간색 LED가 켜지며, 스위치를 누르면 3초간 녹색 LED와 빨간색 LED가 같이 켜졌다가 꺼지며 종료됩니다.

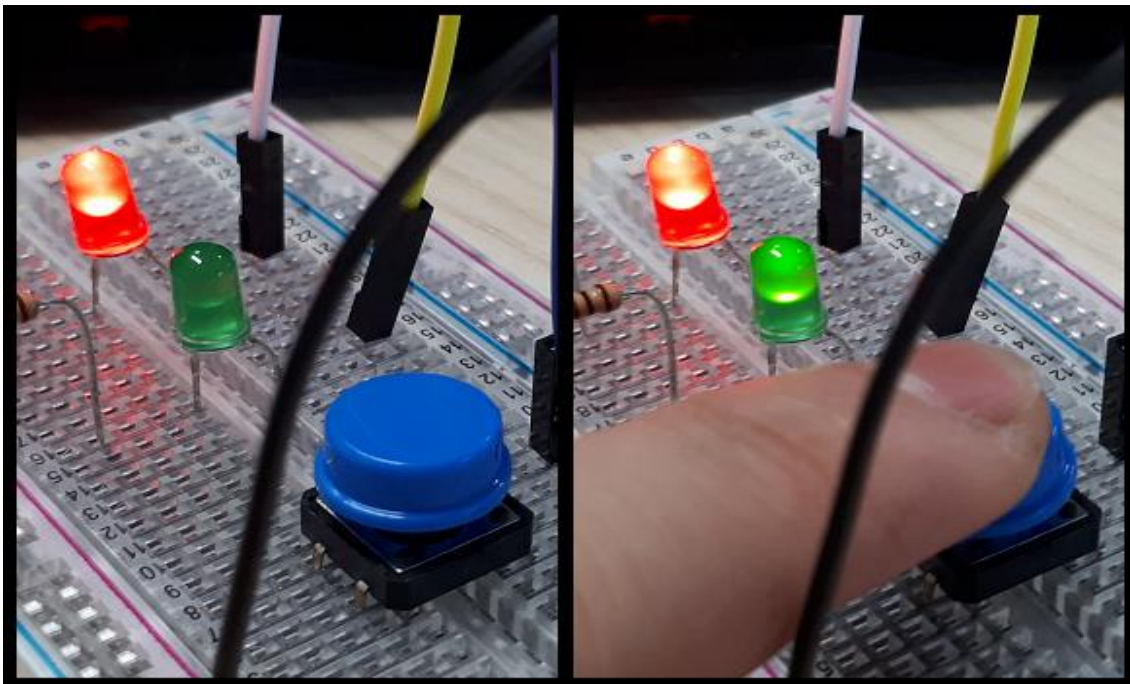


그림 10 02.SW_read.py 실행결과

2. 릴레이 보드 제어하기

① 릴레이 보드 연결하기

릴레이 보드는 브레드보드에 연결하지 않고 점퍼 케이블로 직접 연결이 가능합니다.
또한, 서로 다른 핀을 사용하도록 하여 스위치와 LED 를 제거하지 않고 같이 연결이 가능합니다.

보드의 사양은 제어신호 전압 3.3V, 전원 3~5V 사양이므로 전원은 5V 에 연결해 주고 컨트롤 신호는 라즈베리파이의 GPIO(3.3V)에 연결해 주시면 됩니다.

릴레이 보드의 GPIO 는 18 번 핀을 사용합니다.

아래 회로도과 동일하게 연결해 주시면 됩니다.
(각 모듈 핀 부분에 적힌 핀 이름을 참고해 주세요)

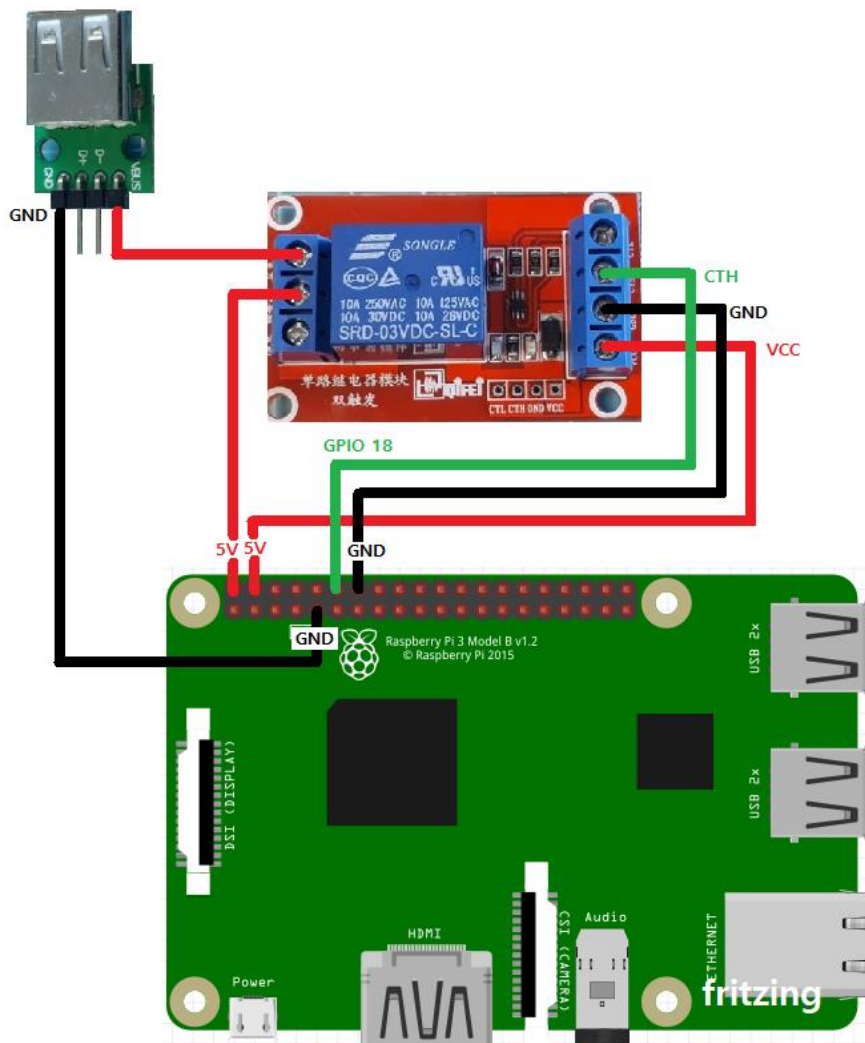


그림 11 릴레이보드 연결

릴레이 보드에 점퍼 케이블을 연결할 때는 터미널블록의 나사를 적당히 풀어 준 뒤 점퍼 케이블의 핀 부분을 터미널 블록에 꽂아 나사를 돌려서 고정시켜 주시면 됩니다.

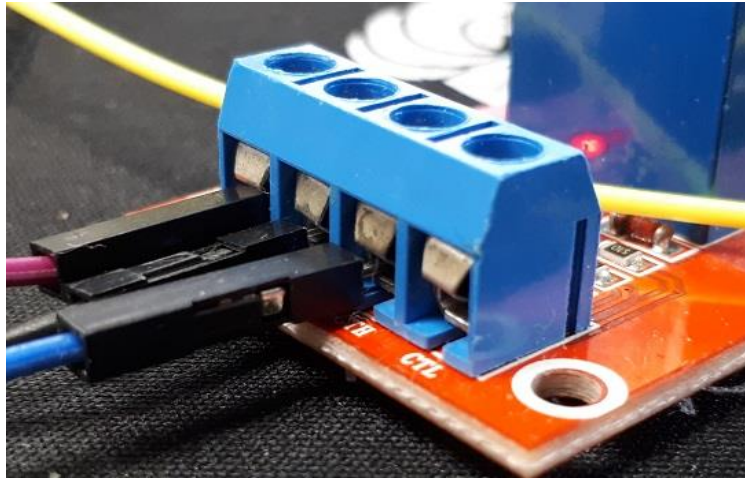


그림 12 릴레이 보드에 점퍼케이블 연결

GPIO 가 입력되는 부분의 CTL / CTH 는 각각 LOW 일 때/HIGH 일 때 동작(컨트롤)되는 것을 뜻하며, USB 가 연결되는 쪽의 핀은 가운데를 기준으로 좌 우가 반대되는 동작을 합니다.

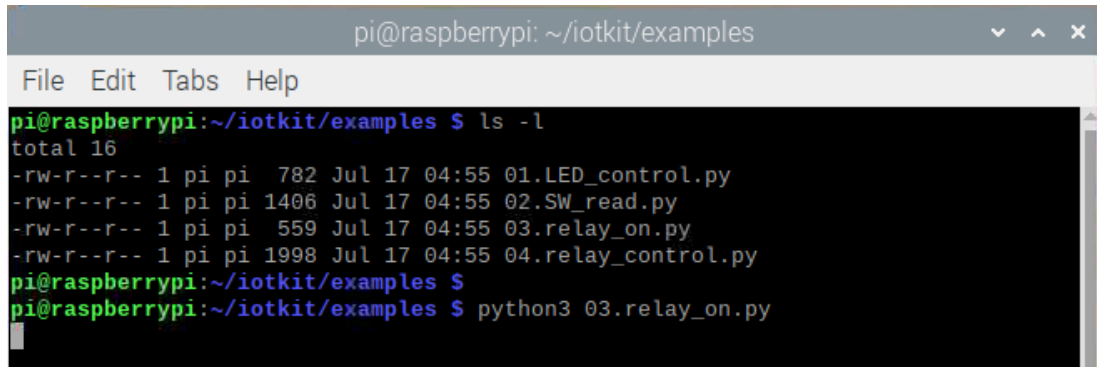
USB 에는 구성품인 미니 선풍기를 연결해 릴레이의 동작을 확인할 수 있습니다.
(릴레이 동작은 소리로도 확인 가능하며 동작 시 ‘딸깍’하는 소리가 나게 됩니다.)

※ 라즈베리파이의 5V 전원을 이용하므로 5W (5V 1A)이상의 제품을 연결할 경우 라즈베리파이의 전원이 부족해 질 수 있습니다.

② 릴레이 동작 테스트하기

릴레이를 사용하는 예제는 03.relay_on.py 및 04.relay_control.py 이며, 04.relay_control.py 는 LED 및 스위치가 연결되어 있어야 합니다.

첫 번째 릴레이 예제는 `python 03.relay_on.py` 를 입력해 실행시켜 주면 됩니다.



```

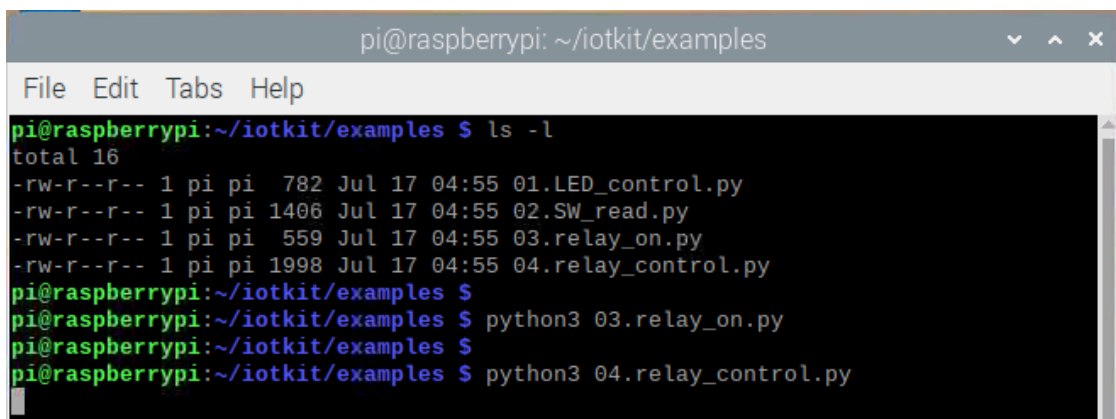
pi@raspberrypi: ~/iotkit/examples
File Edit Tabs Help
pi@raspberrypi:~/iotkit/examples $ ls -l
total 16
-rw-r--r-- 1 pi pi 782 Jul 17 04:55 01.LED_control.py
-rw-r--r-- 1 pi pi 1406 Jul 17 04:55 02.SW_read.py
-rw-r--r-- 1 pi pi 559 Jul 17 04:55 03.relay_on.py
-rw-r--r-- 1 pi pi 1998 Jul 17 04:55 04.relay_control.py
pi@raspberrypi:~/iotkit/examples $
pi@raspberrypi:~/iotkit/examples $ python3 03.relay_on.py

```

그림 13 예제 03.relay_on.py

03.relay_on.py 예제를 실행시켜 주면 5 초 동안 릴레이가 작동 후 꺼진 뒤 프로그램이 종료됩니다.

두 번째 릴레이 예제는 `python 04.relay_control.py` 를 입력해 실행시켜 주면 됩니다.



```

pi@raspberrypi: ~/iotkit/examples
File Edit Tabs Help
pi@raspberrypi:~/iotkit/examples $ ls -l
total 16
-rw-r--r-- 1 pi pi 782 Jul 17 04:55 01.LED_control.py
-rw-r--r-- 1 pi pi 1406 Jul 17 04:55 02.SW_read.py
-rw-r--r-- 1 pi pi 559 Jul 17 04:55 03.relay_on.py
-rw-r--r-- 1 pi pi 1998 Jul 17 04:55 04.relay_control.py
pi@raspberrypi:~/iotkit/examples $
pi@raspberrypi:~/iotkit/examples $ python3 03.relay_on.py
pi@raspberrypi:~/iotkit/examples $
pi@raspberrypi:~/iotkit/examples $ python3 04.relay_control.py

```

그림 14 예제 04.relay_control.py

04.relay_control.py 예제는 실행시키면 빨간 LED 가 켜지고 대기상태가 되며, 스위치를 누르면 빨간 LED 가 꺼지고 녹색 LED 가 켜지면서 릴레이가 동작합니다.

그리고 2 초 후 다시 스위치를 누르면 녹색 LED 가 꺼지고 빨간 LED 가 켜지면서 3 초간 대기한 뒤 프로그램이 종료됩니다.

3. PMS7003 먼지센서 사용하기

① 먼지센서 연결하기

PMS7003 먼지센서는 USB to Serial(UART) 케이블과 인터페이스 보드를 이용해 라즈베리파이에 연결해 줄 수 있습니다.

UART 통신을 이용하기 때문에 전원(+, -)외의 통신단자 Tx(송신), Rx(수신)을 연결해 주어야 합니다.

USB to Serial(UART)케이블에서는 **녹색이 Tx, 흰색이 Rx** 이며 인터페이스 보드에 적힌 Rx, Tx 와 엇갈리도록 연결해 주시면 됩니다.

데이터 방향 (Tx -> Rx)	UART 케이블 (라즈베리파이)	인터페이스보드 (먼지센서)
라즈베리파이 -> 먼지센서	녹색(Tx)	Rx 핀
라즈베리파이 <- 먼지센서	흰색(Rx)	Tx 핀

표 1 UART 케이블 연결방법 및 데이터 흐름

아래 사진을 참고해서 연결합니다.



그림 15 먼지센서 연결

그리고 인터페이스 보드에 먼지센서를 연결, USB 는 라즈베리파이에 연결해 주시면 됩니다.

이때, 라즈베리파이에 연결한 USB 는 라즈베리파이에서 /dev 경로에 추가됩니다.

/dev 경로에서 ls 를 입력해 확인하는 것도 가능하지만 **ls /dev | grep ttyUSB** 명령어로 연결된 USB 리스트만 확인하는 것도 가능합니다.

위 명령어는 /dev 경로에 있는 ttyUSB~로 시작하는 파일/디렉터리의 목록(ls)을 전부 표시해 줍니다.

USB 를 연결하기 전/후로 `ls /dev | grep ttyUSB` 명령어를 입력해 확인해 보도록 합니다.

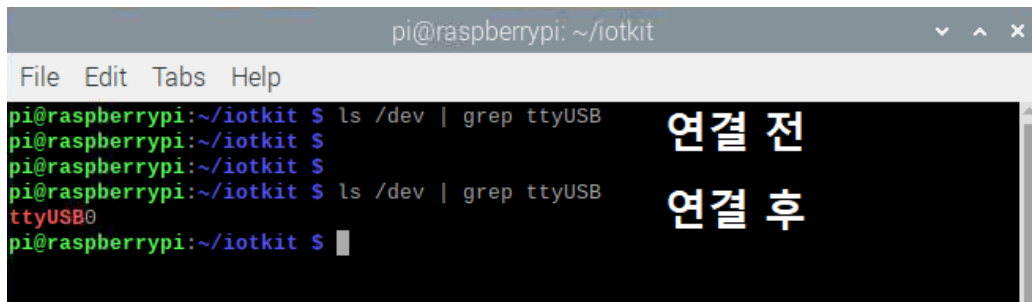


그림 16 USB 연결확인

/dev/ttyUSB0 로 먼지센서가 연결된 USB 케이블이 연결된 것을 확인할 수 있습니다.

② 먼지센서 동작시켜 데이터 받아보기

먼지센서 예제 코드는 start.sh 스크립트를 실행해서 추가로 다운로드 해야 합니다.
start.sh 스크립트를 실행시켜 준 경우 examples 디렉터리에 05.PMS7003 디렉터리가 추가됩니다.

만일 위에서 실행시켜 주지 않은 경우 아래와 같이 실행해 주시면 됩니다.

> iotkit 경로로 이동

`chmod +x start.sh`

`./start.sh`

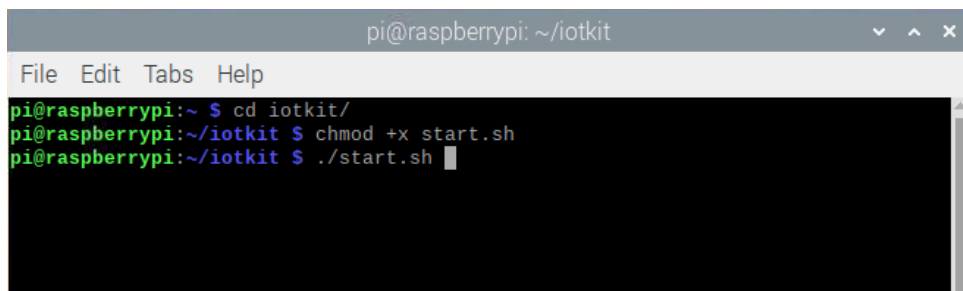


그림 17 start.sh 스크립트 실행

위 스크립트를 실행하면 아래 경로의 PMS7003 예제를 다운로드 해 주게 됩니다.

<https://github.com/eleparts/PMS7003>

examples 경로의 05.PMS7003 으로 다운로드 되므로 해당 경로로 이동해 줍니다.

`cd example/05.PMS7003`

PMS7003.py 는 라이브러리 및 먼지센서 테스트용 프로그램이 내장되어 있습니다.

dust_chk.py 는 PMS7003.py 를 import 하여 사용하는 예제 프로그램 입니다.


```

pi@raspberrypi: ~/iotkit/examples/05.PMS7003
File Edit Tabs Help
pi@raspberrypi:~/iotkit $ cd examples/05.PMS7003/
pi@raspberrypi:~/iotkit/examples/05.PMS7003 $ ls -l
total 20
drwxr-xr-x 2 pi pi 4096 Jul 18 01:57 'data sheet'
-rw-r--r-- 1 pi pi 2388 Jul 18 01:57 dust_chk.py
-rw-r--r-- 1 pi pi 5907 Jul 18 01:57 PMS7003.py
-rw-r--r-- 1 pi pi 1799 Jul 18 01:57 README.md
pi@raspberrypi:~/iotkit/examples/05.PMS7003 $

```

그림 18 PMS7003 라이브러리

05.PMS7003 디렉터리로 이동 후 프로그램 실행에 앞서 연결되어 있는 포트의 설정을 해 주어야 합니다. GUI 환경에서 해당 파일을 열거나, 터미널 창에서 nano 혹은 vi를 사용해 수정 후 저장해 주시면 됩니다.

여기서는 터미널 창에서 수정할 수 있도록 nano 에디터를 사용, **nano PMS7003.py** 를 입력해 파일을 열어준 뒤 화살표키를 이용, 스크롤을 아래로 쪽 내려 #USE PORT 부분을 찾아 UART 를 **USB0** 로 변경합니다.

```

pi@raspberrypi: ~/iotkit/examples/05.PMS7003
File Edit Tabs Help
GNU nano 7.2 PMS7003.py
# UART / USB Serial : 'dmesg | grep ttyUSB'
USB0 = '/dev/ttyUSB0'
UART = '/dev/ttyAMA0'
# USE PORT
SERIAL_PORT = UART
# Baud Rate
Speed = 9600

pi@raspberrypi: ~/iotkit/examples/05.PMS7003
File Edit Tabs Help
GNU nano 7.2 PMS7003.py *
# UART / USB Serial : 'dmesg | grep ttyUSB'
USB0 = '/dev/ttyUSB0'
UART = '/dev/ttyAMA0'
# USE PORT
SERIAL_PORT = USB0
# Baud Rate
Speed = 9600
# example
if __name__ == '__main__':

```

그림 19 PORT 변경

빨간 사각형 부분의 UART를 USB0로 변경해 주시면 되며, 이는 바로 위 주황색 원 부분에 정의되어 있습니다. 만약 위에서 /dev/ttyUSB0 가 아닌 /dev/ttyUSB1 등에 연결된 경우 주황색 원 뒤의 '/dev/ttyUSB0' 부분을 동일하게 변경해 주어야 합니다.

[컨트롤 + O] ⇒ [엔터]로 저장, [컨트롤 + X]로 닫아주신 뒤 해당 프로그램을 하면 됩니다.

프로그램 실행은 **python PMS7003.py** 로 실행시켜 줍니다.

실행에 문제가 발생하는 경우 앞에 sudo 를 붙여 실행해 주면 됩니다.

```

pi@raspberrypi: ~/iotkit/examples/05.PMS7003
File Edit Tabs Help
0.5um in 0.1L of air : 270
1.0um in 0.1L of air : 60
2.5um in 0.1L of air : 17
5.0um in 0.1L of air : 3
10.0um in 0.1L of air : 0
Reserved F : 151 | Reserved B : 0
CHKSUM : 664 | read CHKSUM : 664 | CHKSUM result : True
=====
DATA read success
=====
Header : B M | Frame length : 28
PM 1.0 (CF=1) : 5 | PM 1.0 : 5
PM 2.5 (CF=1) : 14 | PM 2.5 : 14
PM 10.0 (CF=1) : 16 | PM 10.0 : 16
0.3um in 0.1L of air : 972
0.5um in 0.1L of air : 282
1.0um in 0.1L of air : 64
2.5um in 0.1L of air : 18
5.0um in 0.1L of air : 3
10.0um in 0.1L of air : 0
Reserved F : 151 | Reserved B : 0
CHKSUM : 711 | read CHKSUM : 711 | CHKSUM result : True
=====

```

그림 20 PMS7003 동작 테스트

PMS7003.py 예제 코드를 동작시키면 위와 같이 1 초 간격으로 먼지센서의 데이터를 다운로드 하게 됩니다.

dust_chk.py 프로그램은 PMS7003.py 라이브러리를 이용해 먼지센서를 사용하는 예제 코드로 PMS7003.py 와 동일하게 USB0 로 변경 후 실행시켜 주시면 데이터를 1 회 출력합니다.

다) Blynk 로 원격 제어하기

1. Blynk 소개

Blynk 는 <https://blynk.io/> 에서 제작한 IOT 플랫폼입니다

Android 및 IOS 환경에서 앱을 다운로드 하여 아두이노, ESP 시리즈, 라즈베리파이 등의 하드웨어를 원격 제어하거나 데이터를 수신하는 동작 등을 매우 손쉽게 구성할 수 있습니다.

앱의 UI 구성도 간단하여 필요한 블록을 배치하고 각 블록에 가상 포트를 연결하는 것으로 상당히 쉽게 시스템을 구성할 수 있습니다.

아두이노 및 ESP32 환경에서 사용을 주력으로 제작하고 있지만 라즈베리파이에서도 사용 가능한 Python 버전이 존재합니다.

여기서는 Blynk 앱을 다운로드 하고 라즈베리파이에 Blynk Python 버전을 설치하여 두 기기를 연동하고 실시간으로 제어하는 동작을 진행하도록 하겠습니다.

다만, 무료 버전의 경우 사용할 수 있는 위젯 및 가상 핀의 수량 제한 등의 제약이 있습니다.

본 키트에서는 무료 버전 사용을 기준으로 예제를 구성해 보도록 하겠습니다.

2. Blynk 설치

Blynk 를 이용하기 위해서는 스마트폰과 라즈베리파이에 각각 Blynk 를 다운로드 해 주어야 합니다.

① 라즈베리파이에 Blynk - python 버전 설치

Blynk 를 사용하기 위해서는 Blynk 라이브러리를 설치해 주어야 합니다.

간단 설치 명령(pip 등)은 구 버전이 설치되는 문제가 있어 라이브러리가 저장된 저장소를 다운로드 한 뒤 설치해 주면 됩니다.

git clone 으로 저장소를 다운로드하고 설치된 경로로 이동합니다.

```
git clone https://github.com/eleparts/blynk-library-python
cd Blynk-library-python
```

그 다음 아래 명령어로 설치를 해 주시면 됩니다.

```
sudo python setup.py install
```

```
pi@raspberrypi: ~/blynk-library-python
File Edit Tabs Help
pi@raspberrypi:~$ git clone https://github.com/eleparts/blynk-library-python
Cloning into 'blynk-library-python'...
remote: Enumerating objects: 384, done.
remote: Counting objects: 100% (56/56), done.
remote: Compressing objects: 100% (38/38), done.
remote: Total 384 (delta 27), reused 42 (delta 18), pack-reused 328
Receiving objects: 100% (384/384), 81.41 KiB | 4.79 MiB/s, done.
Resolving deltas: 100% (233/233), done.
pi@raspberrypi:~$ cd blynk-library-python/
pi@raspberrypi:~/blynk-library-python$ sudo python setup.py install
```

그림 21 Blynk python 버전 설치

② 스마트폰에 Blynk 설치

스마트폰에서는 Android/ IOS 에 따라 각각의 스토어를 통해 간단하게 다운로드 하실 수 있습니다.
각 스토어에서 Blynk 를 검색하면 Blynk 앱을 확인할 수 있습니다.

Android 는 플레이 스토어

<https://play.google.com/store/search?q=blynk>

IOS 에서는 앱 스토어에서 다운로드가 가능합니다.

<https://apps.apple.com/kr/app/blynk-iot/id1559317868>

스토어에서 설치를 해 주신 뒤 실행해준 뒤 기존 계정이 있다면 로그인, 아니라면 새롭게 가입해 주시면 됩니다.

가입 시에는 이메일을 입력 후 인증해 주시면 되며, 받은 메일에서 인증 링크를 눌러 비밀번호를 설정해 준 뒤 앱에서 로그인 해 줍니다.

3. Blynk 위젯 설정하기

기존 Blynk 에 비해 2.0 버전으로 변경된 Blynk 는 사용법이 약간 더 복잡해졌습니다.
설정 절차는 아래와 같습니다.

1. 템플릿(Template) 생성
2. 템플릿에서 위젯(Widget) 및 데이터스트림(DataStream) 추가
3. 위젯 상세 설정 및 데이터스트림 연결
4. 생성된 템플릿으로 디바이스 추가

무료 버전의 데이터스트림(원격 통신용 Virtual 핀) 수량 제한 및 템플릿, 기기 수 제한이 있기 때문에 여기서는 2 개의 템플릿을 생성하고 GPIO 용 디바이스와 PMS7003 센서 구동용 디바이스로 각각 나누어 구성해 주겠습니다.

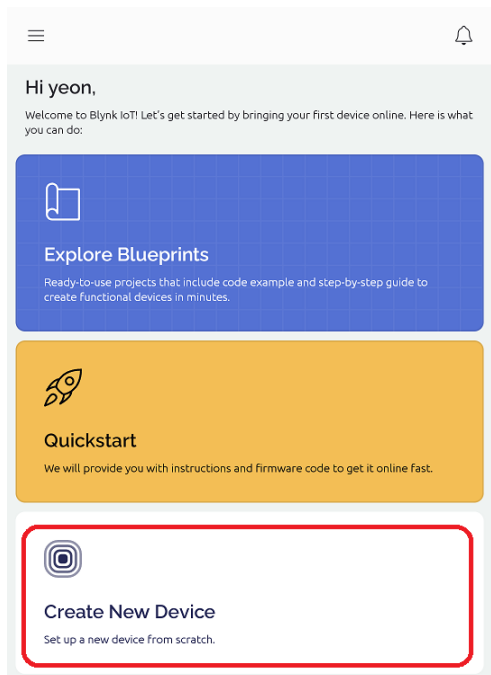


그림 22 첫 페이지

[그림 22 첫 페이지]는 앱에 로그인 후 보이는 첫 페이지(메인 페이지)입니다.

이 페이지는 템플릿 및 디바이스가 추가되면 이에 맞게 변경되므로 적당히 넘겨 주시면 됩니다.

맨 아래에 있는 Create New Device 를 눌러주면 먼저 템플릿 (Template)을 추가하게 됩니다.

이제 차례대로 PIGPIO 및 PIPMS 두 개의 페이지를 만들도록 하겠습니다.

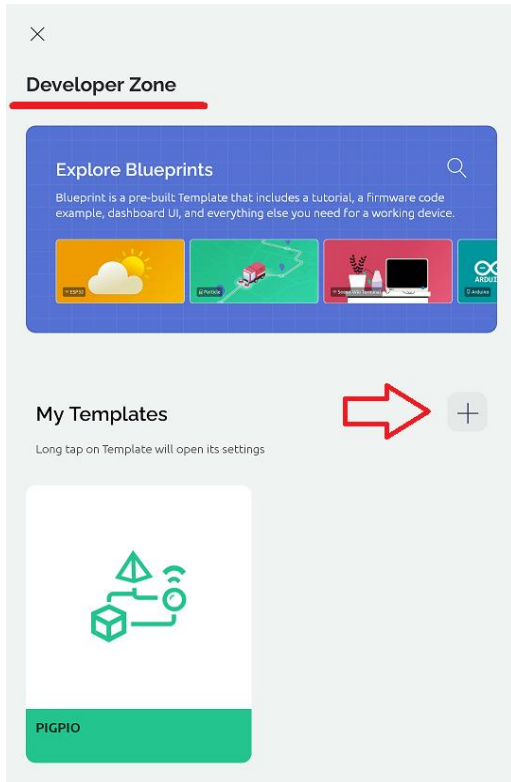


그림 23 템플릿 작성 1

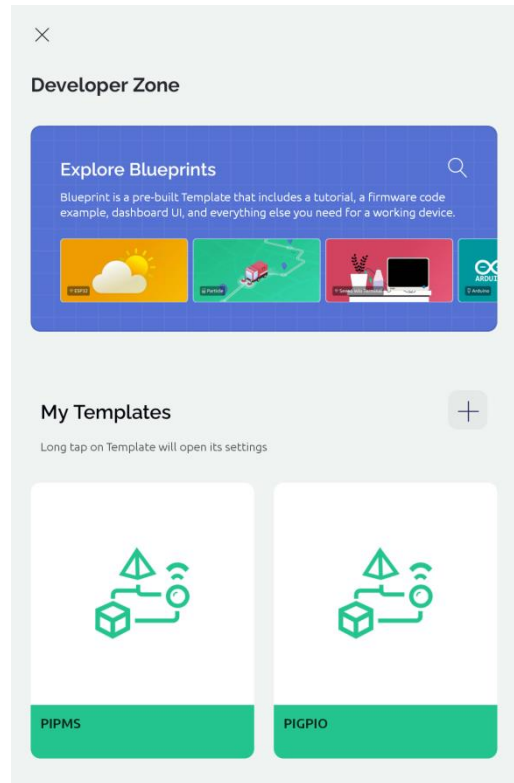


그림 24 템플릿 작성 2

첫 번째 템플릿을 만들면 Developer Zone 으로 이동하며, 여기서 템플릿을 관리할 수 있습니다.

[+] 버튼을 눌러 추가 템플릿을 미리 만들어 둘 수 있으므로 두 종류의 템플릿을 미리 작성해 주도록 합니다.

만약 [그림 25 템플릿 작성 후 메인 페이지]과 같은 페이지로 이동되었다면, 하단의 Developer Zone 버튼을 눌러 다시 진입해 주시면 됩니다.

이후 템플릿 제작이 완료되면 이 메인 페이지로 이동해 디바이스를 추가해 주게 됩니다.

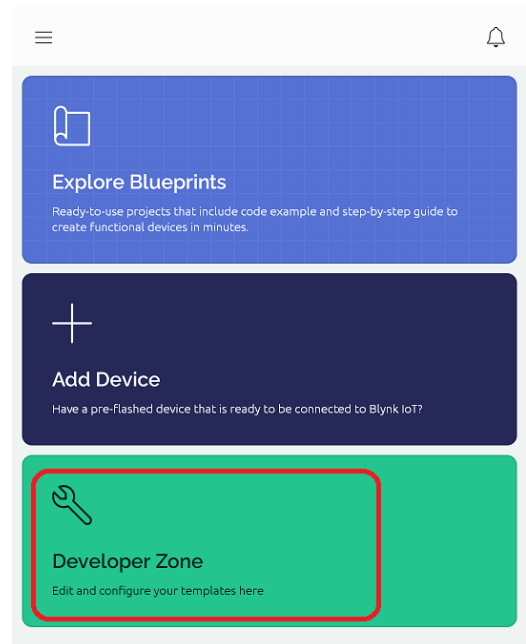


그림 25 템플릿 작성 후 메인 페이지

두 개의 템플릿을 모두 작성하였다면 GPIO 제어용 템플릿을 눌러 열어 줍니다.

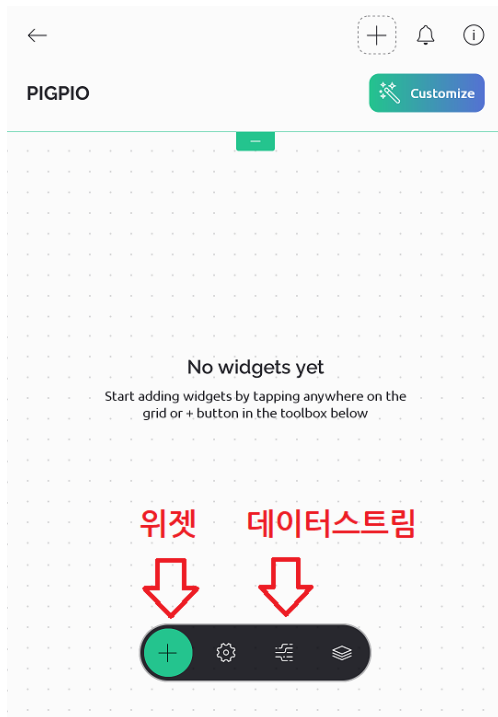


그림 26 템플릿 편집 페이지

좌측 템플릿 편집 페이지에서 템플릿에 위젯을 추가하거나 데이터스트림을 편집할 수 있습니다.

배경 혹은 녹색 (+) 버튼을 누르면 위젯 추가가 가능하며, 하단 데이터스트림 메뉴를 누르면 데이터스트림 편집 페이지로 이동됩니다.

다만, 무료 버전은 사용할 수 있는 위젯의 종류가 한정적이며 데이터스트림은 앱과 디바이스가 통신하는 데이터 슬롯으로 템플릿당 5 개까지 사용이 가능합니다.

이제 두 가지 템플릿을 각각 설정해 주도록 하겠습니다.

① GPIO 제어용 템플릿 구성

GPIO 템플릿은 아래 표와 같은 구성으로 작성해 줍니다.

가상 핀	가상 핀 설정	위젯	위젯 모드
V0	integer 0/1	Button	MODE : SWITCH
V1	integer 0/1	Button	MODE : SWITCH
V2	string	Value Display	-
V3	integer 0/1	Button	MODE : SWITCH
V4	integer 0/255	LED	-

표 2 GPIO 템플릿 구성

가상 핀은 데이터스트림 페이지에서 추가하면 되며, 위젯은 배경 부분 혹은 녹색 + 버튼을 눌러 추가해 주면 됩니다.

먼저 위젯부터 추가해 보겠습니다.

빈 부분을 눌러 위젯 추가 페이지를 열어준 뒤 스크롤을 내려 위 표에 맞는 위젯을 각각 추가해 줍니다.

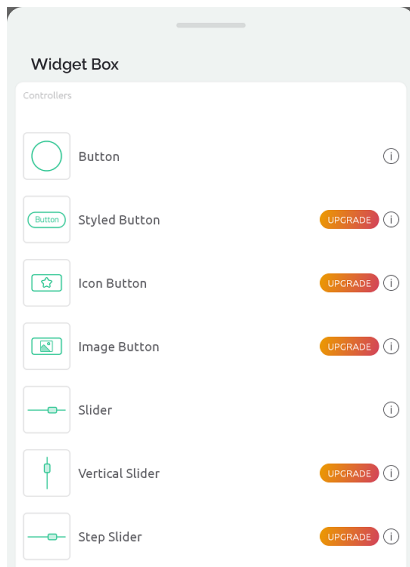


그림 27 위젯 추가 페이지

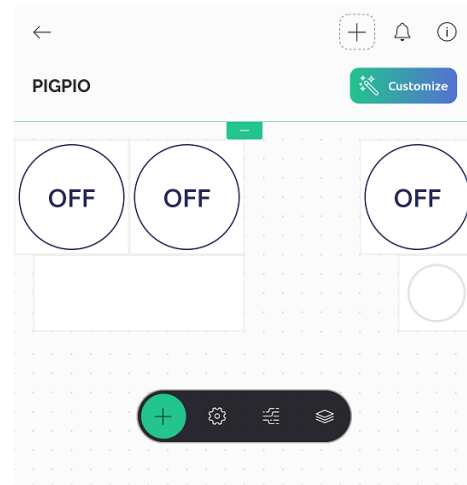


그림 28 GPIO 템플릿 위젯

위 사진처럼 적당히 배치해 주도록 합니다.

버튼 총 3 개, 디스플레이 1 개, LED 1 개를 추가해 주면 됩니다.

위젯은 길게 누르면 드래그해 옮길 수 있고, 다시 내려놓으면 크기를 조절할 수 있습니다.

그리고 데이터스트림을 추가해 줍니다. 하단 메뉴의 3 번째 버튼을 누르면 됩니다.

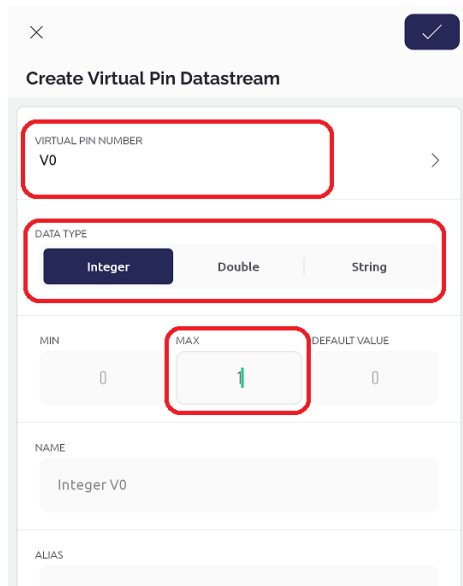


그림 29 가상 핀 추가

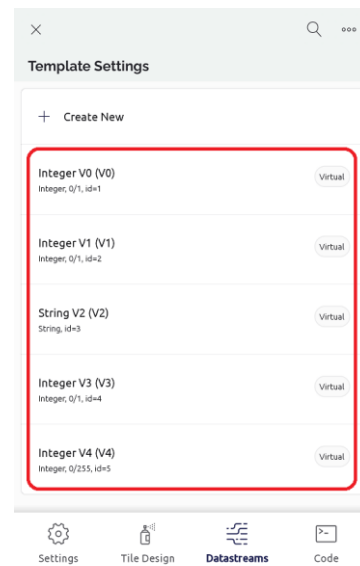


그림 30 GPIO 템플릿 가상 핀

데이터스트림 편집 페이지에서 Create New 버튼을 눌러 가상 핀을 각각 추가해 주면 됩니다. VIRTUAL PIN NUMBER 번호에 주의하여 DATA TYPE 과 MAX 값을 설정해 줍니다.

아래 NAME 항목은 0 번부터 순서대로 매겨지며, 중복되지만 않으면 됩니다.

최종적으로 [그림 30 GPIO 템플릿 가상 핀]처럼 설정해 주었다면 잘 설정된 것입니다.

마지막으로 가상 핀과 버튼을 각각 매칭시켜 주도록 하겠습니다.

위젯 편집 모드에서 각 위젯을 누르면 위젯 설정 페이지로 이동됩니다.
여기서 데이터스트림/가상 핀을 각 위젯에 맞게 추가해 주면 됩니다.

이때, 특정 위젯은 가상 핀의 설정을 사용할지 버튼의 설정을 사용할 지 아래처럼 선택할 수 있습니다.

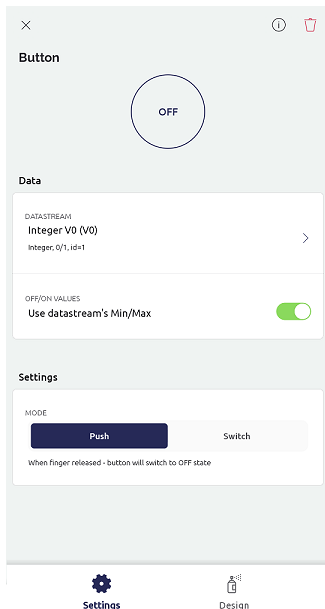


그림 31 스위치 설정
(데이터스트림 설정 사용)

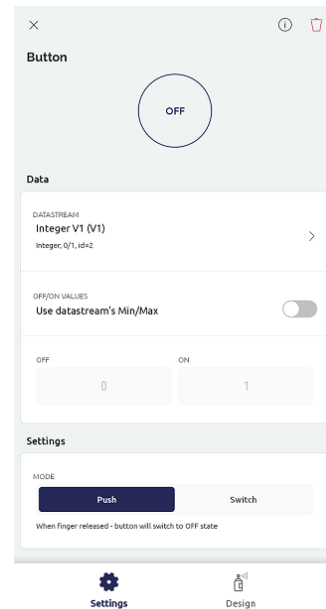


그림 32 스위치 설정
(데이터스트림 설정 미사용)

데이터스트림 설정에서 맞춰두었다면 데이터스트림 설정을 사용하면 되며, 그렇지 않다면 버튼에서 맞게 설정해 줍니다.

해당 값은 라즈베리파이와 주고받는 데이터 값 범위 설정이므로 [표 2 GPIO 템플릿 구성]의 설정과 동일하게 맞춰 주어야 합니다.

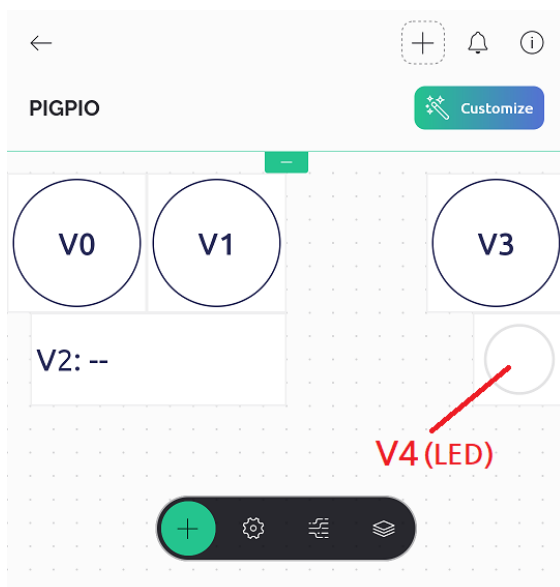


그림 33 GPIO 템플릿 설정 완료

모든 설정이 완료되면 [그림 33 GPIO 템플릿 설정 완료]처럼 5 개의 위젯과 가상 핀이 연결됩니다.

V4/LED 는 따로 가상 핀 번호가 표시되지 않으므로 적당하게 배치를 완료해 주시면 됩니다.

② 먼지센서 데이터 수신용 템플릿 구성

PMS7003 템플릿은 아래 표와 같은 구성으로 작성해 줍니다.

가상 핀	가상 핀 설정	위젯	위젯 모드
V4	integer 0/255	LED	-
V5	string	Value Display	-
V6	string	Value Display	-
V7	string	Value Display	-

표 3 PMS 템플릿 구성

가상 핀은 데이터스트림 페이지에서 추가하면 되며, 위젯은 배경 부분 혹은 녹색 + 버튼을 눌러 추가해 주면 됩니다.

먼저 위젯부터 추가해 보겠습니다.

빈 부분을 눌러 위젯 추가 페이지를 열어준 뒤 스크롤을 내려 위 표를 참고해 위젯을 각각 추가해 줍니다.

추가된 위젯은 꺾(길게) 눌렀다 떼면 크기를 조절하거나 이동할 수 있습니다.

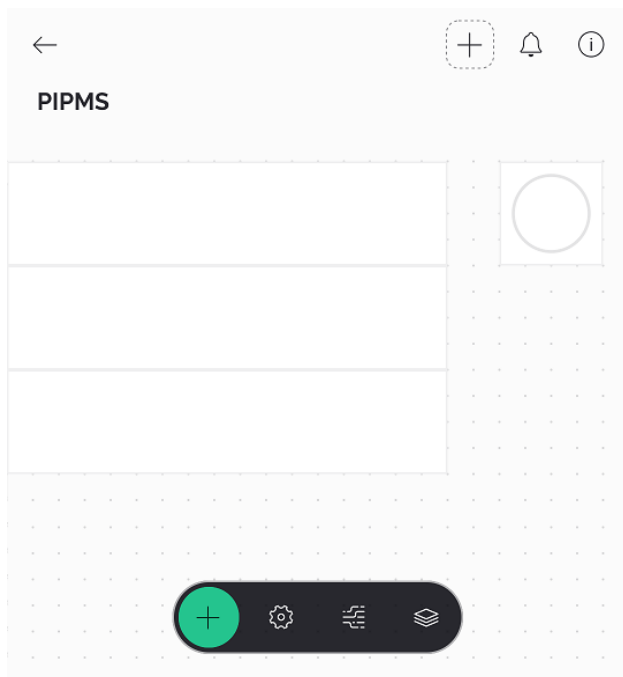


그림 34 PMS 템플릿 위젯

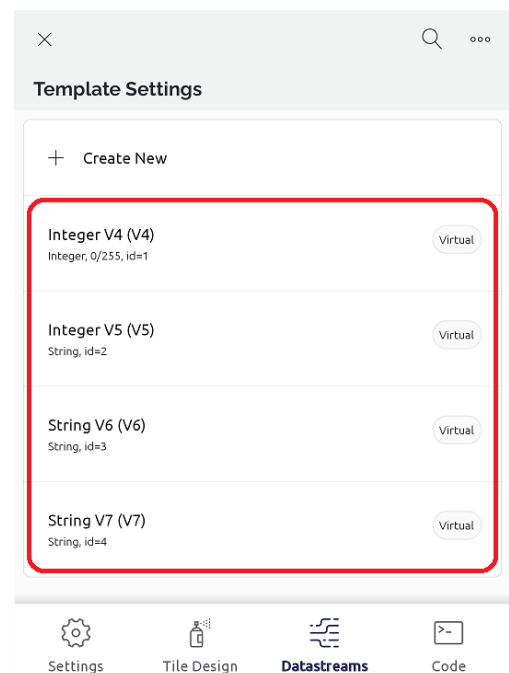


그림 35 PMS 템플릿 가상 핀

위젯을 배치해준 뒤 마찬가지로 데이터스트림 편집 페이지(하단 메뉴 3 번째 아이콘)로 들어가 Create New 을 눌러 가상 핀을 추가해 줍니다.

VIRTUAL PIN NUMBER 번호가 4 번부터 시작하므로 번호에 주의하면서 DATA TYPE 을 설정해 줍니다.

그 외에는 추가로 변경할 항목은 없으나 NAME 항목이 자동으로 입력되지 않아 중복되는 이름 오류가 뜰 수 있으니 적당히 변경해 주시면 됩니다.

위젯과 가상 핀 설정을 마쳤다면 위젯과 가상 핀을 연결시켜 줍니다.

이번에 사용하는 위젯은 가상 핀 외에 추가로 설정할 부분은 없으므로 Display 및 LED 번호를 잘 맞춰 연결해 주면 됩니다.

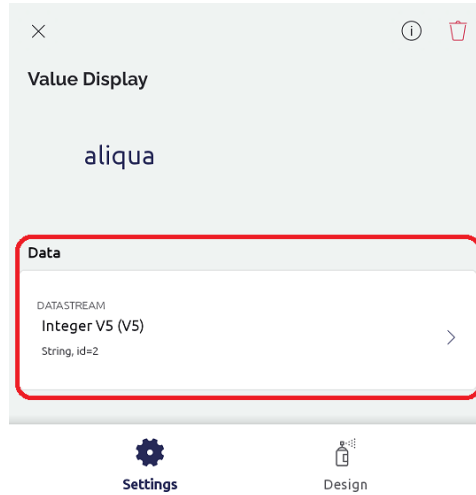


그림 36 가상 핀 연결

설정이 완료된 PMS7003 먼지센서용 템플릿 구성입니다.

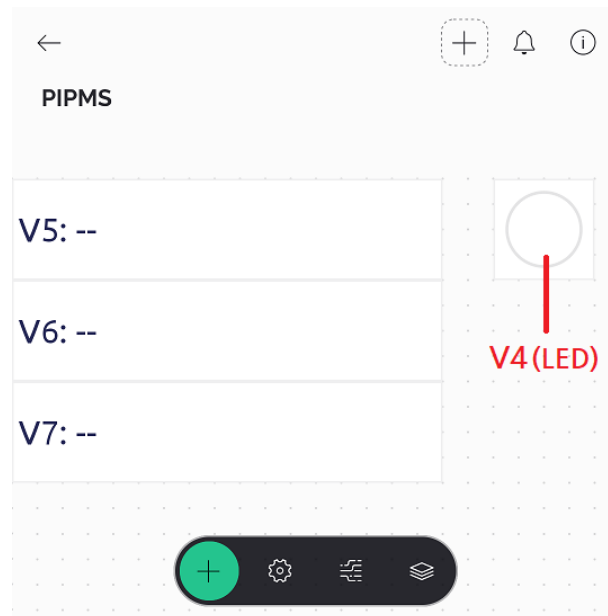


그림 37 PMS 템플릿 설정 완료

가상 핀 5,6,7 은 Display 에 순서대로 배치해 주시고,
LED 는 따로 가상 핀 번호가 표시되지 않지만 4 번 가상 핀과 연결해 주면 됩니다.

③ 디바이스 추가하기

템플릿 작성이 완료되었으면 이제 디바이스를 추가해 템플릿을 등록해 주어야 합니다.

뒤 버튼을 눌러 상위 메뉴로/메인 페이지로 이동해 줍니다.

메인 페이지에서 Add Device 메뉴를 누른 뒤 Manually from template 항목을 눌러 템플릿을 불러와 디바이스를 만들어 주면 간단하게 디바이스를 추가할 수 있습니다.

아래 [그림 39 템플릿을 선택해 디바이스 추가]와 같이 제작해 둔 템플릿이 목록에 나오므로 선택 후 이름을 작성해 디바이스를 추가하면 됩니다.

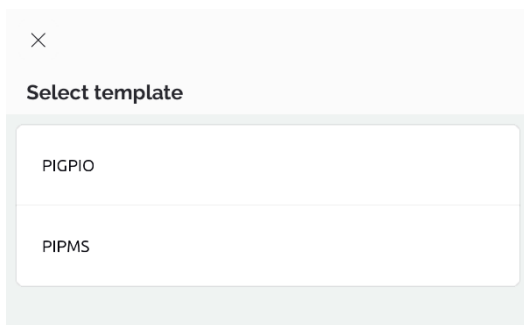


그림 39 템플릿을 선택해 디바이스 추가

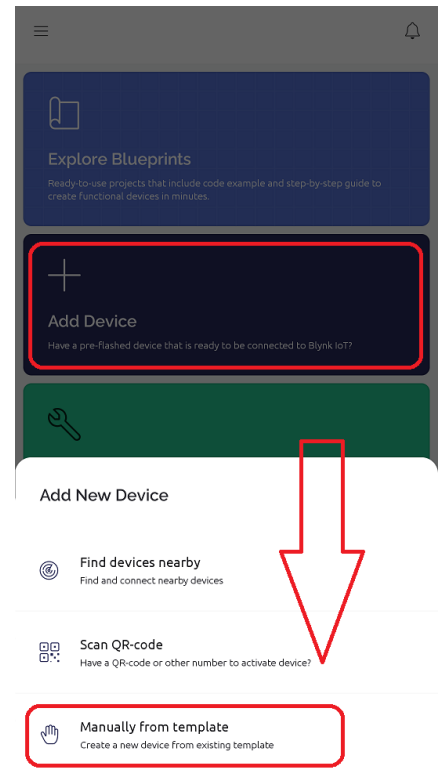


그림 38 디바이스 추가하기

두 번째 디바이스는 우측 상단 (+) 버튼으로 추가해 주시면 됩니다.

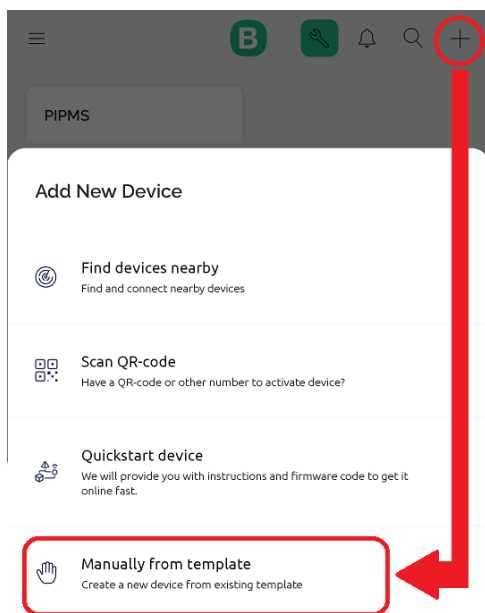


그림 40 두번째 디바이스 추가

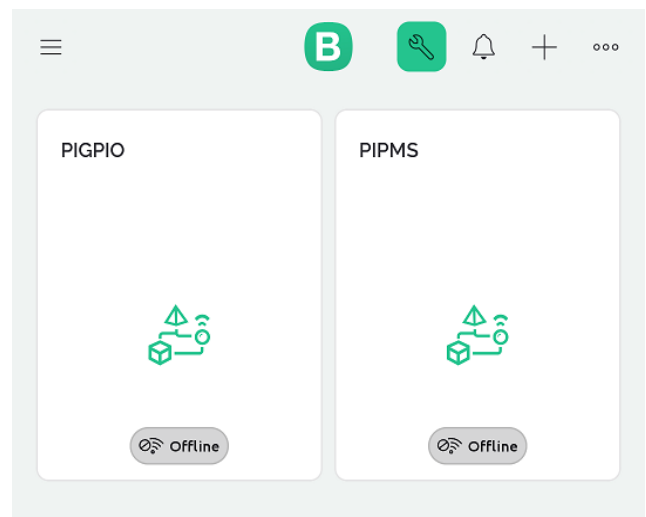


그림 41 디바이스 추가 완료

여기까지 하였다면 PI-GPIO와 PI-PMS 예제의 설정이 모두 완료된 것 입니다.

이제 디바이스의 토큰 정보를 가져와 라즈베리파이에서 예제를 실행해 주시면 됩니다.

4. 디바이스 토큰 정보 확인하기

라즈베리파이에서 Blynk 에 연결하기 위해서는 Blynk 에서 생성한 디바이스에 해당하는 토큰이 필요합니다. 이 토큰은 Blynk 대시보드 페이지에서 확인할 수 있습니다.

아래 Blynk 대시보드 페이지 링크로 이동해 줍니다.

<https://blynk.cloud/dashboard/>

대시보드 페이지로 이동하면 로그인 후 디바이스 메뉴에서 생성한 디바이스 목록을 확인할 수 있습니다. Auth Token 값도 바로 적혀있지만 뒤 글자가 잘려있으므로 눌러 해당 디바이스 상세 페이지로 이동해 줍니다.

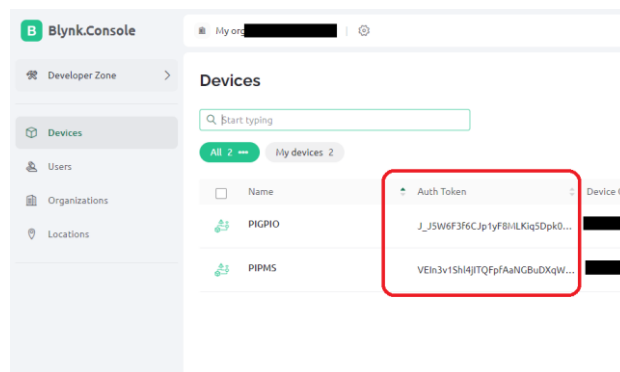


그림 42 Blynk 대시보드 페이지

디바이스 상세 페이지에서 Devices info 탭을 눌러 주시면 디바이스 정보 확인이 가능합니다. 여기서 하단의 블러 처리된 Auth Token 부분에 마우스를 올리면 복사 버튼이 생성됩니다.

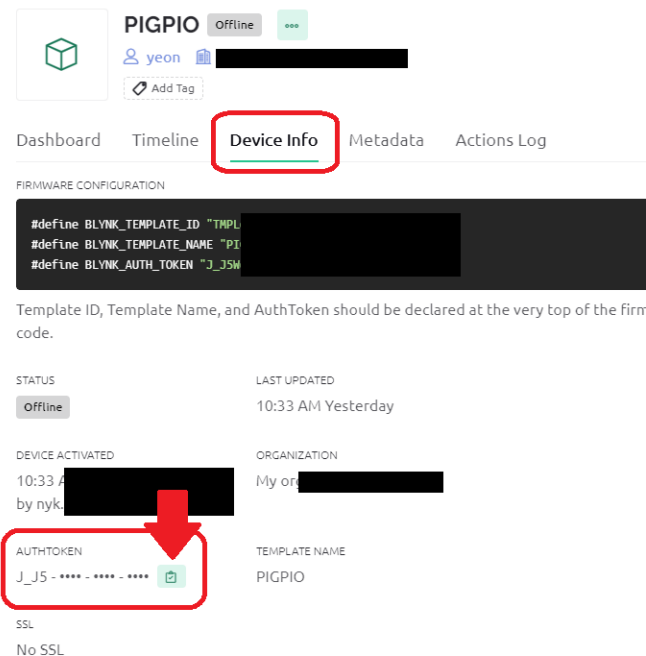


그림 43 Blynk 디바이스 상세 정보 페이지

복사 버튼을 눌러 토큰 값을 복사 해 라즈베리파이에서 사용할 수 있게 저장해 주도록 합니다. 디바이스마다 토큰이 할당되므로 다른 디바이스도 동일하게 토큰 값을 찾아 주시면 됩니다.

5. Blynk 로 GPIO 제어하기

① 하드웨어 구성

스위치 및 릴레이 회로 구성은 위 [나)라즈베리파이 제어]와 동일하게 해 주시면 됩니다.

※ 참고 1 - 그림 5 라즈베리파이에 LED 및 스위치 연결

※ 참고 2 - 그림 11 릴레이보드 연결

② Blynk GPIO 디바이스 선택

Blynk 앱에서 GPIO 디바이스를 열어 줍니다.

아래 예제코드 수정 시 GPIO 디바이스 토큰을 사용하시면 됩니다.

③ 예제코드 실행

Blynk 앱을 동작시켜 주었다면 라즈베리파이의 동작을 제어해 줄 예제 코드를 실행해 주어야 합니다.

먼저 blynk_example/blynk_python_GPIO 디렉터리로 이동해 줍니다.

아래 디렉터리에서 blynk_python_GPIO.py 및 blynk_python_GPIO_V2.py 가 예제 코드 파일입니다.



```

pi@raspberrypi: ~/iotkit/blynk_examples/blynk_python_GPIO
File Edit Tabs Help
pi@raspberrypi:~/iotkit/blynk_examples $ cd blynk_python_GPIO/
pi@raspberrypi:~/iotkit/blynk_examples/blynk_python_GPIO $ ls -l
total 208
drwxr-xr-x 2 pi pi 4096 Mar 8 13:01 소스 코드
-rw-r--r-- 1 pi pi 4716 Mar 8 13:01 blynk_GPIO.fzz
-rw-r--r-- 1 pi pi 180188 Mar 8 13:01 blynk_GPIO.png
-rw-r--r-- 1 pi pi 3425 Mar 13 10:39 blynk_python_GPIO.py
-rw-r--r-- 1 pi pi 3878 Mar 8 13:01 blynk_python_GPIO_V2.py
-rw-r--r-- 1 pi pi 1065 Mar 8 13:01 LICENSE
drwxr-xr-x 4 pi pi 4096 Mar 8 13:01 old_version
-rw-r--r-- 1 pi pi 2715 Mar 8 13:01 README.md
pi@raspberrypi:~/iotkit/blynk_examples/blynk_python_GPIO $
  
```

그림 44 Blynk GPIO 예제

blynk_python_GPIO.py 는 릴레이 동작이 없는 예제이며, V2 버전이 릴레이 기능이 추가된 예제 코드입니다.

해당 예제 코드를 동작하기 전에 위에서 저장해 둔 GPIO 디바이스용 토큰을 입력해 주어야 합니다.

※참조: [디바이스 토큰 정보 확인하기]

토큰은 편집기를 이용해 예제코드를 열어 토큰이 정의된 부분에 값을 넣어 주면 됩니다.

터미널 환경에서 nano 를 이용한다면 `nano blynk_python_GPIO.py` 명령어로 예제 파일을 열고 방향키로 토큰 항목을 찾아 토큰을 입력해준 뒤 **컨트롤+O**, **컨트롤+X** 로 저장 후 종료해 주시면 됩니다.

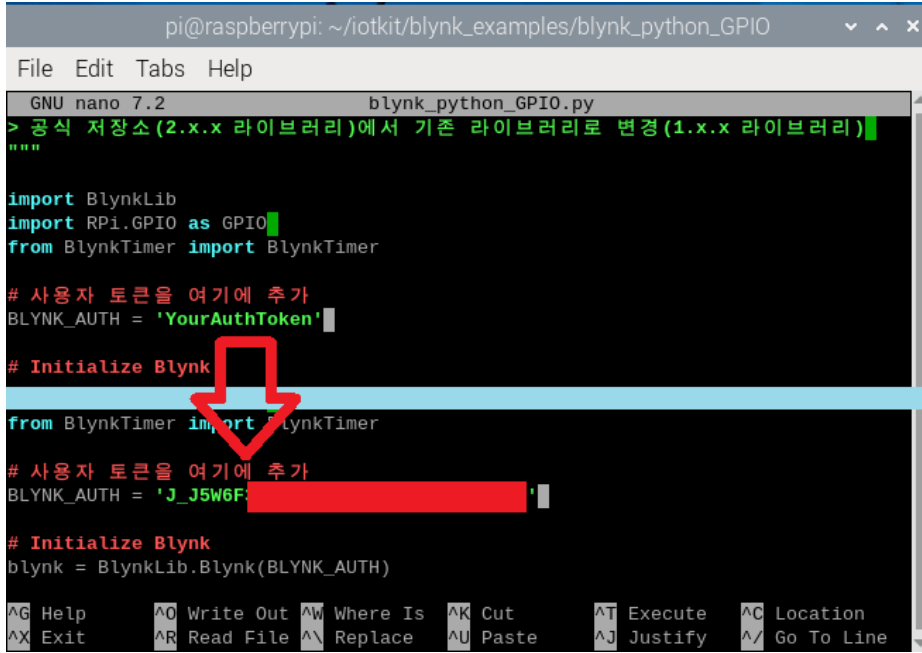


그림 45 GPIO 토큰 입력

마찬가지로 릴레이가 포함된 V2 예제도 동일하게 `python blynk_python_GPIO_V2.py` 로 열고 수정 및 저장해 주시면 됩니다.

그리고 `python blynk_python_GPIO.py` 명령어로 프로그램을 실행시켜 줍니다.

릴레이를 연결하였다면 위 프로그램 대신 `python blynk_python_GPIO_V2.py` 로 프로그램을 실행시켜 주시면 릴레이 구동 버튼을 사용할 수 있습니다.

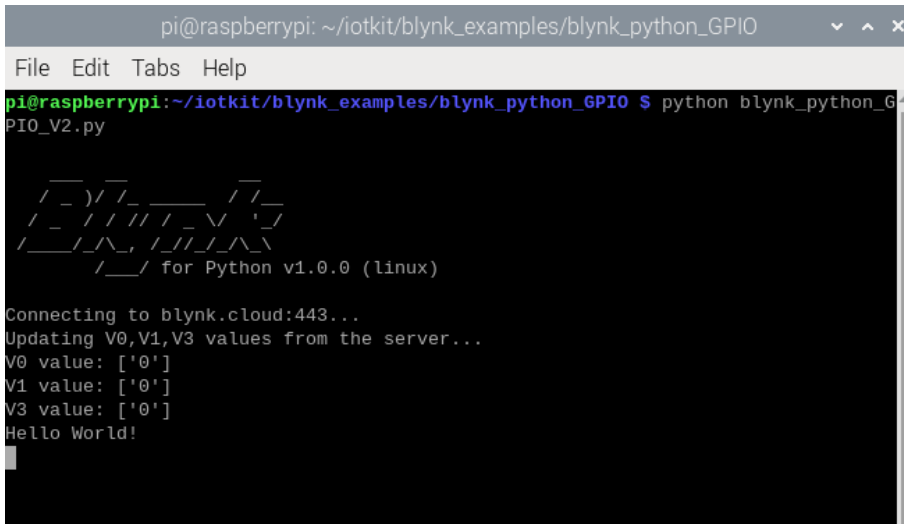


그림 46 Blynk GPIO 동작

프로그램을 실행시켜 준 뒤 스마트 폰 Blynk 앱의 GPIO 디바이스에서 버튼을 누르거나 라즈베리파이에 연결된 스위치를 눌러보시면 이벤트 로그가 표시되면서 각 버튼에 연동되어 있는 LED 및 릴레이가 구동되는 것을 확인하실 수 있습니다.

6. Blynk 로 먼지센서 데이터 받아오기

① 하드웨어 구성

위 PMS 센서 테스트와 동일하게 PMS7003 먼지센서를 인터페이스 보드와 USB to UART 케이블에 연결하여 라즈베리파이의 USB 에 연결해 주시면 됩니다.



그림 47 PMS7003 먼지센서 연결

② Blynk PMS 디바이스 선택

Blynk 앱에서 PMS 디바이스를 열어 줍니다.
아래 예제코드 수정 시 PMS 디바이스 토큰을 사용하시면 됩니다.

③ 예제코드 실행

Blynk 예제는 blynk_example/blynk_python_PMS7003 에 다운로드 되어 있습니다.
해당 경로로 이동하여 예제 코드에 토큰을 입력해 주어야 합니다.

```
pi@raspberrypi: ~/iotkit/blynk_examples/blynk_python_PMS7003
File Edit Tabs Help
pi@raspberrypi:~/iotkit/blynk_examples/blynk_python_PMS7003 $ ls -l
total 28
-rw-r--r-- 1 pi pi 2492 Mar  8 13:01 bly_PMS7003.py
-rw-r--r-- 1 pi pi 1065 Mar  8 13:01 LICENSE
drwxr-xr-x 4 pi pi 4096 Mar  8 13:01 old_version
-rw-r--r-- 1 pi pi 6340 Mar  8 13:01 PMS7003.py
-rw-r--r-- 1 pi pi 3138 Mar  8 13:01 README.md
-rwxr-xr-x 1 pi pi 125 Mar  8 13:01 start.sh
pi@raspberrypi:~/iotkit/blynk_examples/blynk_python_PMS7003 $
```

그림 48 Blynk PMS7003 예제

blynk_python_PMS7003 경로로 이동한 뒤 **nano bly_PMS7003.py** 명령어로 예제 파일을 열어 줍니다.
그리고 BLYNK_AUTH = 'YourAuthToken' 항목을 찾아 PMS 디바이스 토큰을 입력해 줍니다.

```

pi@raspberrypi: ~/iotkit/blynk_examples/blynk_python_PMS7003
File Edit Tabs Help
GNU nano 7.2 bly_PMS7003.py *
import serial
import RPi.GPIO as GPIO
from PMS7003 import PMS7003
from BlynkTimer import BlynkTimer

BLYNK_AUTH = 'YourAuthToken'

#=====
# Baud Rate
Speed = 9600

# UART / USB Serial
USB0 = '/dev/ttyUSB0'
UART = '/dev/ttyAMA0'

# USE PORT
SERIAL_PORT = USB0 #기본값 USB0, 연결방식에 맞춰 변경

#serial setting
ser = serial.Serial(SERIAL_PORT, Speed, timeout = 1)

⌘ Help ⌘ Write Out ⌘ Where Is ⌘ Cut ⌘ Execute ⌘ Location
⌘ Exit ⌘ Read File ⌘ Replace ⌘ Paste ⌘ Justify ⌘ Go To Line

```

그림 49 PMS 토큰 입력

그리고 사용한 연결 방식에 맞춰 위 사진의 아래 USE PORT를 확인해 주시면 됩니다.

※ 기본값 **USB0**, 상단 [먼지센서 연결하기] 항목 참고.

저장은 **컨트롤+O**, 종료는 **컨트롤+X** 입니다.

차례대로 눌러 저장한 뒤 nano 를 닫아 줍니다.

설정이 완료되면 프로그램을 실행해 주시면 됩니다.

명령어는 `python bly_PMS7003.py` 입니다.

```
pi@raspberrypi: ~/iotkit/blynk_examples/blynk_python_PMS7003
File Edit Tabs Help
pi@raspberrypi:~/iotkit/blynk_examples/blynk_python_PMS7003 $ python bly_PMS7003.py

  _ _ _ _ _
 / _ ) / _ _ _ _ _ / _ _ _
 / _ _ / _ _ / _ _ \ \ ' _ \
/_ _ _/_ _ \_/_ _/_/_/_/_/_
  _ _ _/_ _ _ _ _ _ _ _ _ _
    for Python v1.0.0 (linux)

Connecting to blynk.cloud:443...
send - PM1.0: 63 | PM2.5: 91 | PM10: 95
send - PM1.0: 63 | PM2.5: 92 | PM10: 95
```

그림 50 Blynk 먼지센서 예제 실행.

라즈베리파이의 터미널 창에서는 위와 같이 값을 전송하게 되며, Blynk 앱에서는 Display 위젯에 먼지 측정 값이 출력됩니다.

사용 중 프로토콜 에러 혹은 먼지센서와 연결이 끊어지는 경우 LED 위젯이 켜지는 것을 확인하실 수 있습니다.

※ 먼지센서 연결 해제/통신 오류는 먼지센서 인터페이스 보드에 연결된 흰색 핀(먼지센서 Tx, 라즈베리파이 Rx)을 빼 간단히 테스트 해볼 수 있습니다.

라) 통합하여 IOT 환경 구축하기

1. 하드웨어 및 Blynk 위젯 구성

① GPIO 구성

GPIO 연결은 위에서 한 것과 동일하게 구성해 주시면 됩니다.
간단히 정리하면 아래와 같습니다.

GPIO 번호	연결 항목
20	LED
21	LED
18	릴레이
16	스위치

표 4 iotkit GPIO 구성

PMS7003 먼지센서도 동일하게 연결해 둔 상태로 두시면 됩니다.

② Blynk 위젯 구성

Blynk 위젯은 기존 작성한 PMS 디바이스를 사용하되, 스위치를 하나 추가해 줍니다.
위젯 구성은 아래와 같습니다.

가상 핀	가상 핀 설정	위젯	위젯 모드
V3	integer 0/1	Button	MODE : SWITCH
V4	integer 0/255	LED	-
V5	string	Value Display	-
V6	string	Value Display	-
V7	string	Value Display	-

그림 51 iotkit 위젯 구성

V3 가상 핀과 스위치가 하나 추가된 것 외에는 동일한 구성입니다.
위와 같이 구성을 해 주시고 적당히 위치에 맞게 배치해 주시면 됩니다.

우측 [그림 52 iotkit 위젯 배치]는 최종 구성 화면입니다.
위젯의 위치는 자유롭게 배치해 줍니다.

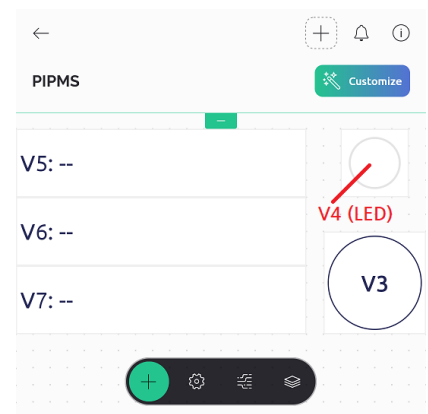


그림 52 iotkit 위젯 배치

2. 예제 실행

예제 실행 방법은 마찬가지로 토큰을 입력 후 실행해 주면 됩니다.

```

pi@raspberrypi: ~/iotkit
File Edit Tabs Help
pi@raspberrypi:~/iotkit $ ls -la
total 88
drwxr-xr-x  8 pi pi  4096 Mar 13 16:09 .
drwx----- 18 pi pi  4096 Mar 13 17:00 ..
drwxr-xr-x  4 pi pi  4096 Mar  8 13:01 blynk_examples
drwxr-xr-x  3 pi pi  4096 Mar  8 13:02 examples
drwxr-xr-x  8 pi pi  4096 Mar 13 10:38 .git
drwxr-xr-x  2 pi pi  4096 Mar  8 13:01 hardware
-rw-r--r--  1 pi pi  4138 Mar 13 16:22 iotkit.py
prw-rw-r--  1 pi pi    0 Mar 13 16:00 log-spy0
-rw-r--r--  1 pi pi 26526 Mar  8 13:01 LICENSE
-rw-r--r--  1 pi pi  6340 Mar  8 13:01 PMS7003.py
drwxr-xr-x  2 pi pi  4096 Mar 13 15:56 __pycache__
-rw-r--r--  1 pi pi  4467 Mar  8 13:01 README.md
-rwxr-xr-x  1 pi pi  1034 Mar  8 13:01 start.sh
drwxr-xr-x  2 pi pi  4096 Mar  8 13:01 usermanual
pi@raspberrypi:~/iotkit $ nano iotkit.py
  
```

그림 53 iotkit 예제

nano iotkit.py 명령어로 예제 코드 파일을 열어 수정해 주시면 됩니다.
예제 파일을 열고 토큰을 넣어 줍니다.

```

pi@raspberrypi: ~/iotkit
File Edit Tabs Help
GNU nano 7.2 iotkit.py
import serial
import RPi.GPIO as GPIO
from PMS7003 import PMS7003
from BlynkTimer import BlynkTimer

# 토큰을 여기에 추가
BLYNK_AUTH = 'YourAuthToken'

#=====
# Baud Rate
Speed = 9600

# UART / USB Serial
USB0 = '/dev/ttyUSB0'
UART = '/dev/ttyAMA0'

# USE PORT
SERIAL_PORT = USB0 #기본값 USB0, 연결방식에 맞춰 변경

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^E Replace ^U Paste ^J Justify ^_ Go To Line
  
```

그림 54 토큰 입력 및 포트 확인

먼지센서를 사용하기 위해 USB 포트도 확인을 해준 뒤 저장 후 종료합니다.

예제 코드 실행은 **python iotkit.py** 입니다.

※참고: 만약 실행 에러가 나는 경우 Blynk 앱에서 스위치 버튼(V3)을 눌러 줍니다.
V3 가상 핀이 사용된 적이 없는 경우 생성되지 않아 오류가 발생될 수 있습니다.


```

pi@raspberrypi: ~/iotkit
File Edit Tabs Help
pi@raspberrypi:~/iotkit $ python iotkit.py

  _ _ _ _ _
 / _ ) / _ _ _ _ _ / _ _
 / _ / / / / / _ _ \ ' _ \
 / _ _ / _ _ _ / / / _ _ \
  _ _ _ / _ _ _ _ _ _ _ _ _
    _ _ _ _ _ for Python v1.0.0 (linux)

Connecting to blynk.cloud:443...
Updating V3,V4... values from the server...
V3 value: ['0']
V4 value: ['0']
Hello World!
send - PM1.0: 60 | PM2.5: 95 | PM10: 100

```

그림 55 iotpit.py 실행

예제를 실행하면 [그림 55 iotpit.py 실행]과 같이 입력 데이터가 전달되면서 동작하게 됩니다.

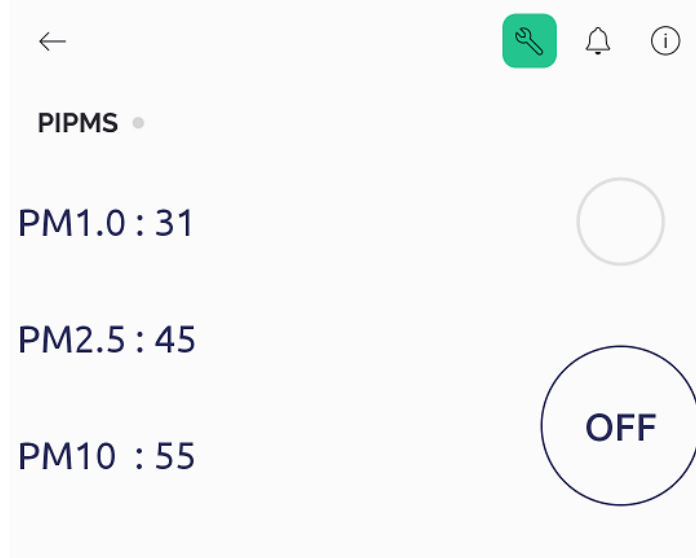


그림 56 Blynk 화면

앱에서는 센서로 측정한 데이터가 출력되며, 가상 스위치 혹은 물리 스위치를 누르는 경우 릴레이가 구동됩니다.

이것으로 제공되는 모든 예제 코드 확인이 완료되었습니다.
lotkit 를 구매해 주셔서 감사합니다.

본 문서는 네이버 나눔글꼴을 이용해 제작되었습니다.