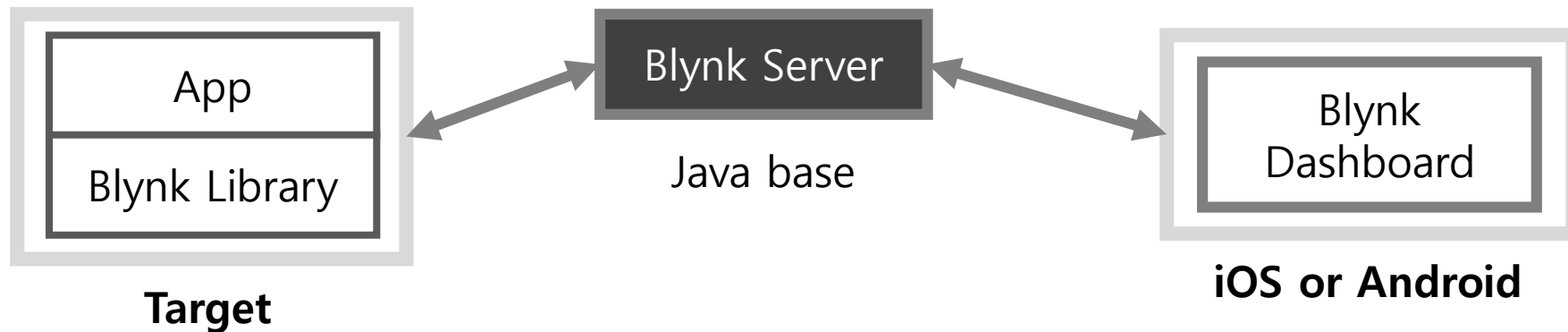


# Blynk

- Blynk는 상용 IoT 프레임워크
  - Blynk Server를 통해 타겟의 하드웨어를 스마트폰에서 원격 제어
    - 타겟은 라즈베리파이, 아틱, 아두이노, ESP8266 등 다양한 하드웨어 지원
    - 모바일 앱은 안드로이드폰과 아이폰 동시 지원
    - Blynk Server는 MQTT 기반 오픈소스로 타겟에 설치 가능
      - 인터넷에 공개 서버인 'blynk-cloud.com' 제공

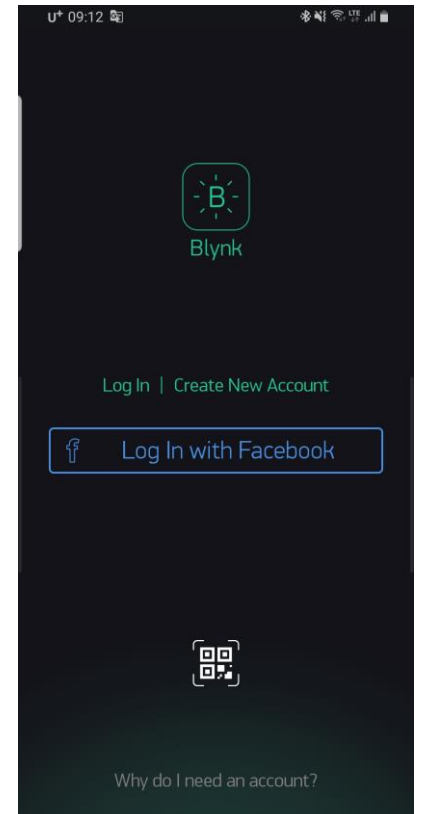


# Blynk

- 스마트폰에서 프로젝트를 생성한 후 타킷에서 서비스 구현
  - 스마트폰
    - 구글 또는 애플 마켓을 통해 Blynk 설치
      - 계정을 등록하면 위젯 사용에 필요한 2000 에너지 무료 제공
    - Blynk 앱 대시보드에서 프로젝트 생성
      - 생성된 인증 토큰은 타킷에서 서비스를 구현할 때 사용
      - Blynk Server 선택. 기본값은 인터넷 공개 서버
      - 센서 데이터 시각화와 GPIO 제어, 푸시 알림 구현은 '끌어다 놓기' 형식의 위젯 사용
        - 코딩 과정이 없으며 최종적으로 앱 스토어 또는 구글 플레이에 게시 가능
  - 타킷
    - 스마트폰에서 생성한 인증 토큰 필요
    - 타킷에서 서비스 구현
      - Blynk Server 선택. 기본값은 인터넷 공개 서버
      - 파이썬용 Blynk 하드웨어 라이브러리로 서비스 구현

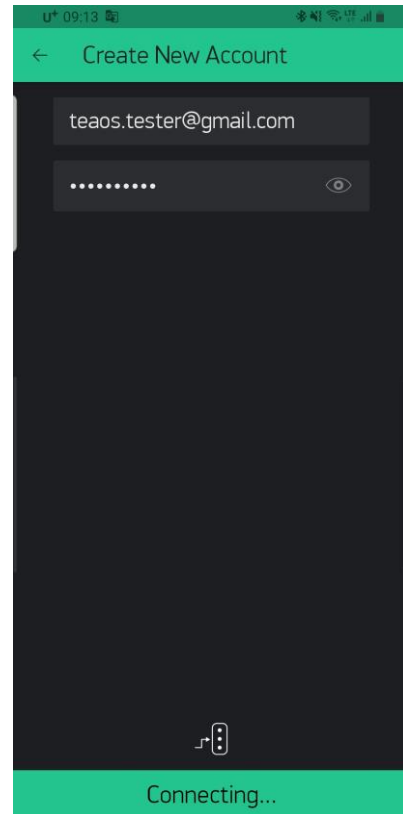
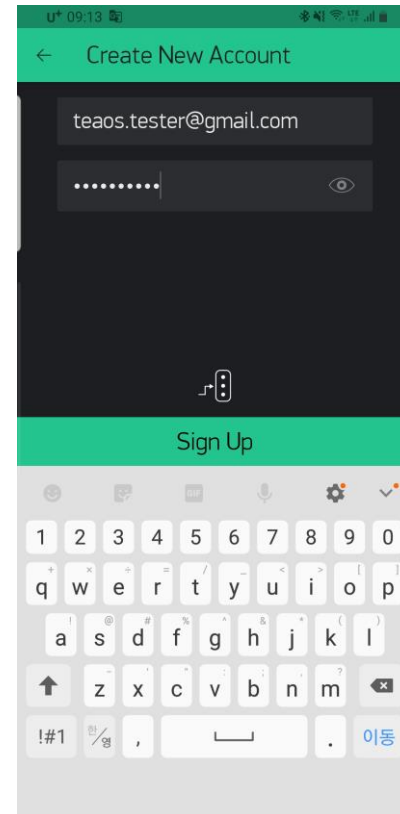
# Blynk 앱

- 계정 생성
  - 마켓에서 Blynk 앱을 설치한 후 실행
  - 로그인 창이 표시되면 'Create New Account' 선택
    - 이미 계정을 만들었다면 'Login' 선택



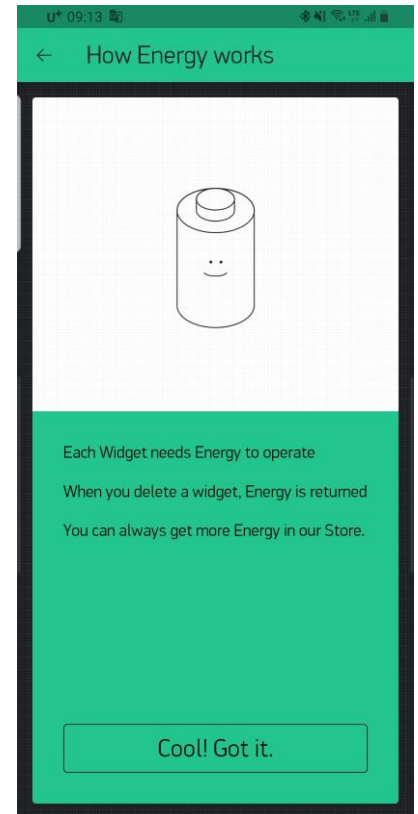
# Blynk 앱

- 계정 생성 (계속)
  - ID에 자신의 메일 계정 입력
  - 패스워드를 입력한 후 'Sign UP' 선택
    - 프로젝트를 생성하면 메일 계정으로 인증 토큰 발송



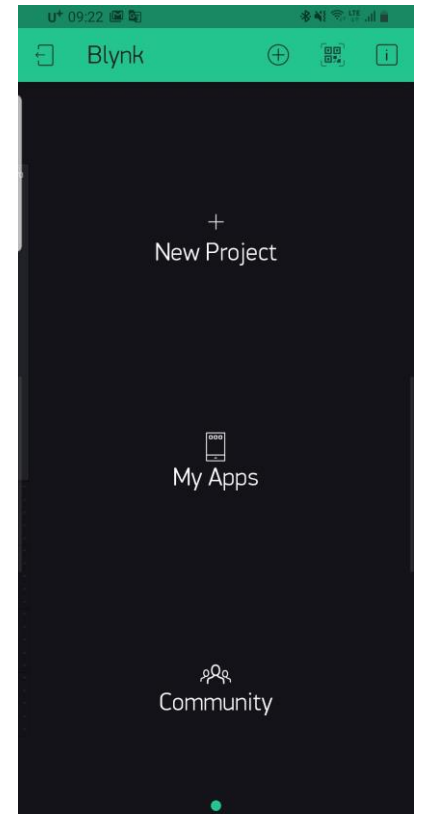
# Blynk 앱

- 계정 생성 (계속)
  - 위젯마다 사용 에너지(금액)가 있음
    - 일정 분량의 에너지 무료 제공
    - 위젯을 사용하면 그만큼 에너지 감소
      - 에너지가 부족해 충전이 필요하면 마켓에서 구매
      - 테스트용으로는 한 두개의 위젯만 추가, 제거하며 사용
    - 위젯을 제거하면 그만큼 에너지 증가
  - 에너지 충전 안내 화면에서 'Cool! Got it.' 선택



# Blynk 앱

- 계정 생성 (계속)
  - 계정 생성이 완료되면 '새 프로젝트 생성' 화면 표시
  - 에너지 충전 안내 화면에서 'Coo! Got it.' 선택

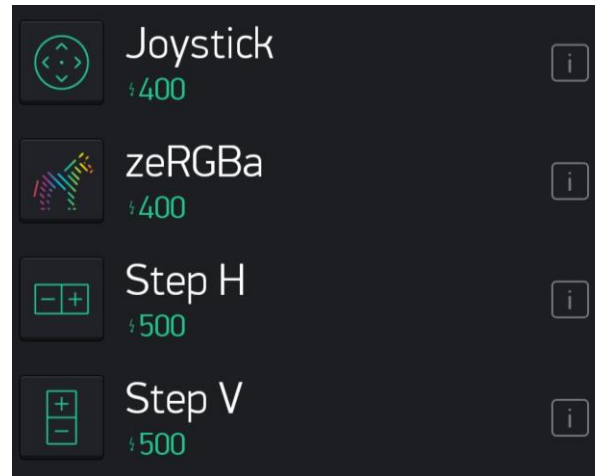
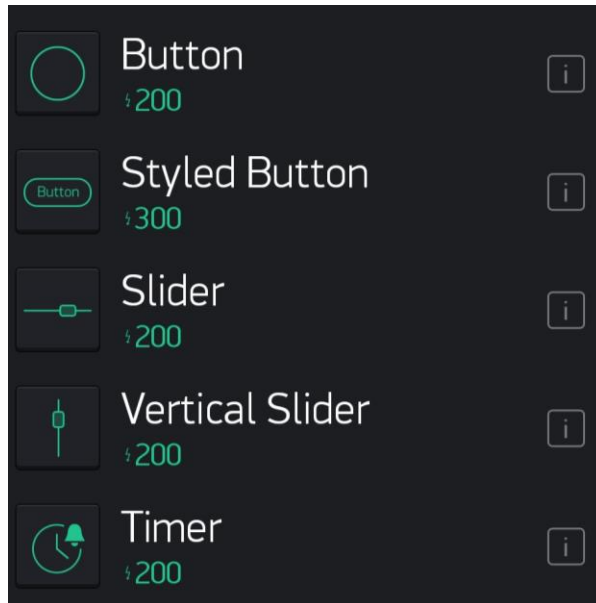


# Blynk 앱

- 위젯
  - 위젯으로 타깃 하드웨어 제어 코딩 배제
    - 대시보드에 '끓어다 놓기'로 필요한 위젯을 배치한 후 설정 변경
  - 타깃 제어에 직접 관여하는 것은 CONTROLES와 DISPLAYS 위젯
    - CONTROLERS: 타깃 기준 입력 장치로 타깃에 데이터 전송
    - DISPLAYS: 타깃 기준 출력 장치로 타깃에서 전송한 데이터 표시
    - NOTIFICATIONS: 트위터, 알림, 메일 연동
    - DEVICE MANAGEMENT: 장치 선택
    - OTHER: 블루투스, 뮤직 플레이 등 기타 기능
    - INTERFACE: 탭, 메뉴, 입력 상자 등 화면 구성
    - SMARTPHONE SENSORS: 스마트폰 내장 센서

# Blynk 앱

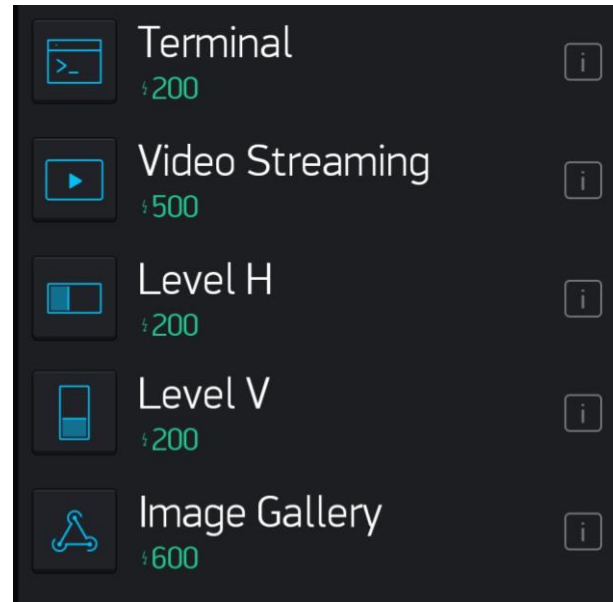
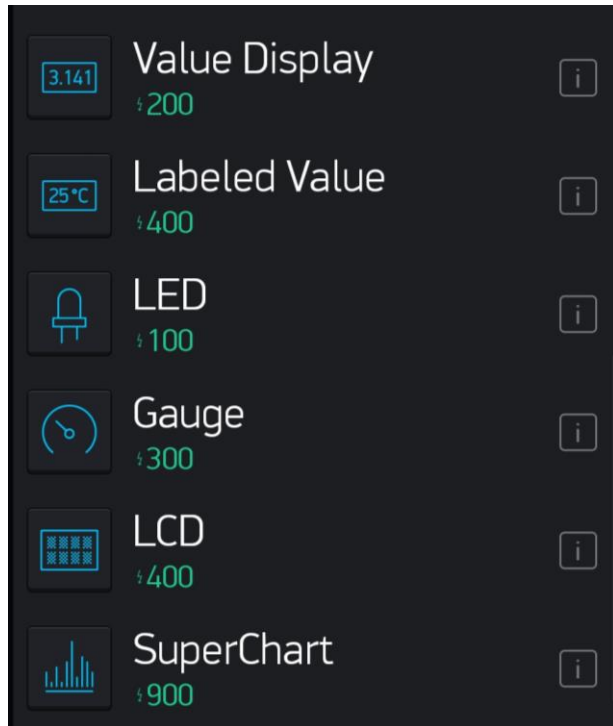
- CONTROLLERS 위젯





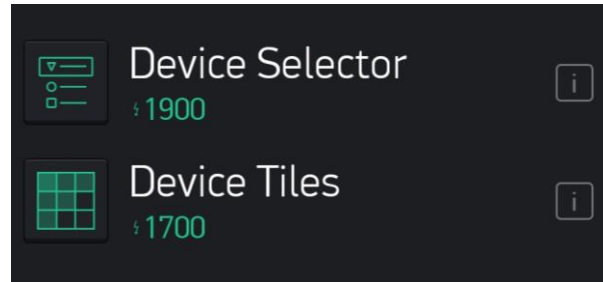
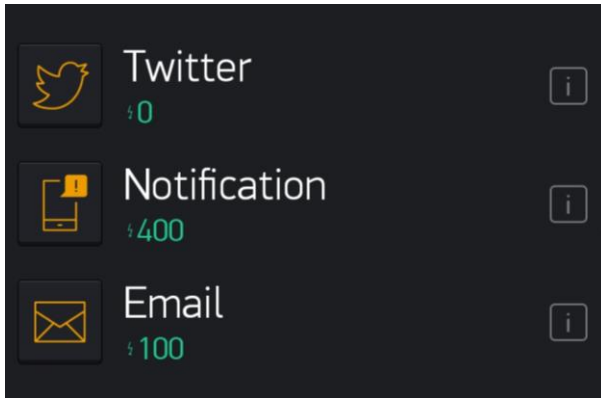
# Blynk 앱

- DISPLAYS 위젯



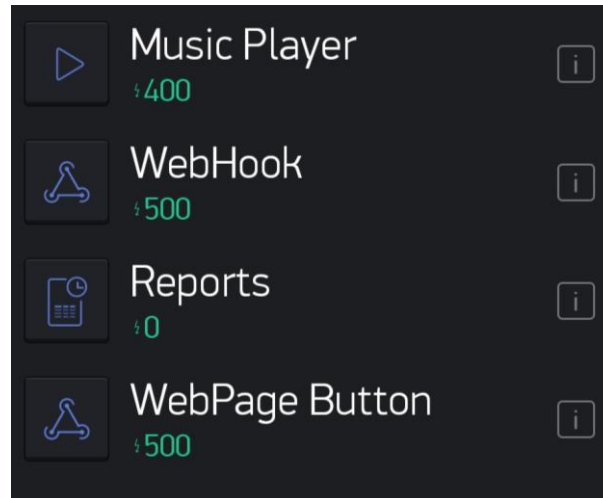
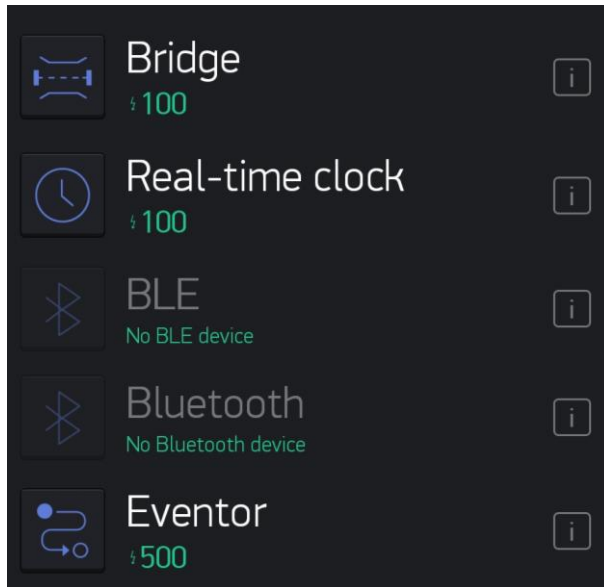
# Blynk 앱

- NOTIFICATIONS, DEVICE MANAGEMENT 위젯



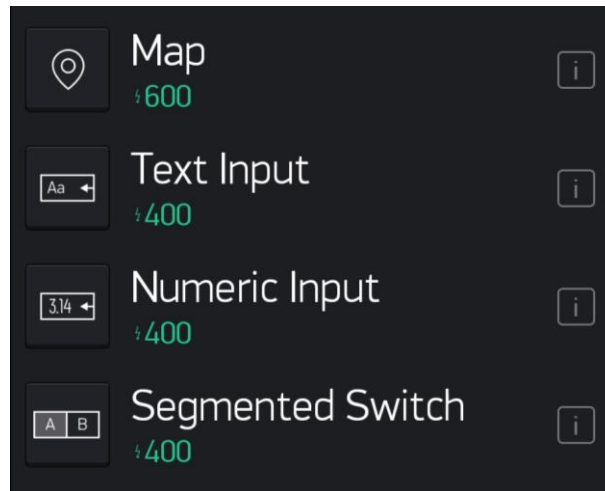
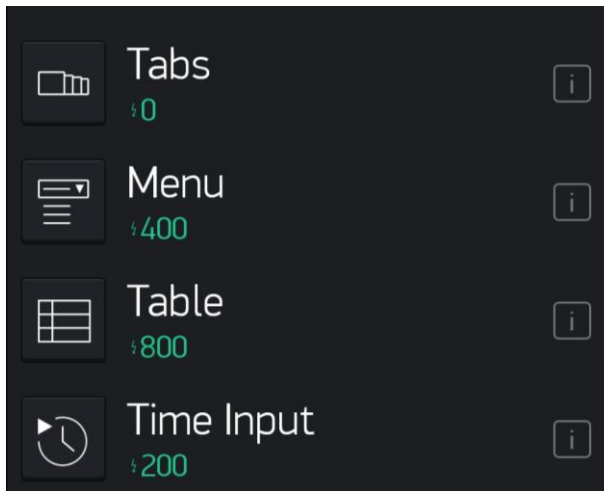
# Blynk 앱

- OTHER 위젯



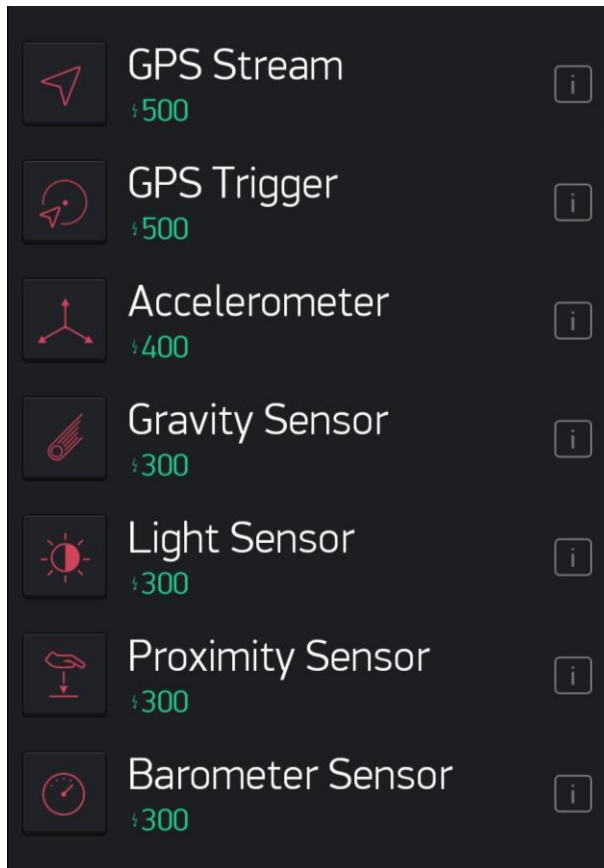
# Blynk 앱

- INTERFACE 위젯



# Blynk 앱

- SMARTPHONE SENSORS 위젯



# 타킷 Blynk 라이브러리

- BlynkLib 모듈의 Blynk Class
  - `__init__(token, server="blynk-cloud.com", port=None, connect=True, ssl=False)`
    - token: 모바일 앱에서 생성한 프로젝트 인증키
    - server: blynk 서버. 기본값은 인터넷 공개서버
    - port: blynk 서버 포트. 기본값은 None
      - ssl 인자가 True이면 8441, False이면 80
    - connect: 작업 핀. 기본값은 None
    - value: `get_val()`의 반환 값. 기본값은 None
  - `connect()`
    - 서버에 연결
  - `disconnect()`
    - 서버와 연결 해제

# 타킷 Blynk 라이브러리

- run()
  - while 무한 루프에서 서버와 통신
  - 메시지 교환을 해제한 후 다시 설정하면 통신을 다시 시작함
- on\_connect(func)
  - 서버에 연결되면 호출할 콜백 등록
  - func: 콜백으로 호출할 사용자 함수
- set\_user\_task(func, msec)
  - 일정 시간마다 호출할 콜백 등록
  - func: 콜백으로 호출할 사용자 함수
  - msec: 밀리초 단위 호출 시간

# 타킷 Blynk 라이브러리

- `virtual_write(port, value)`
  - 모바일 앱으로 데이터 전송
  - `port`: 가상 통신 채널
  - `value`: 전달할 값
    - 타입은 모바일 앱에서 해당 `port`를 사용하는 위젯에 따라 다름
- `@blynk.VIRTUAL_READ(port)`
  - 모바일 앱에서 읽기 요청이 있을 때 호출할 사용자 함수 데커레이터
    - 타킷은 `virtual_write()`로 데이터 전송
  - `port`: 가상 통신 채널
- `@blynk.VIRTUAL_WRITE(port)`
  - 모바일 앱에서 쓴 데이터가 있을 때 호출할 사용자 함수 데커레이터
    - 모바일 앱에서 전달한 데이터는 사용자 함수의 `value` 인자로 전달
  - `port`: 가상 통신 채널

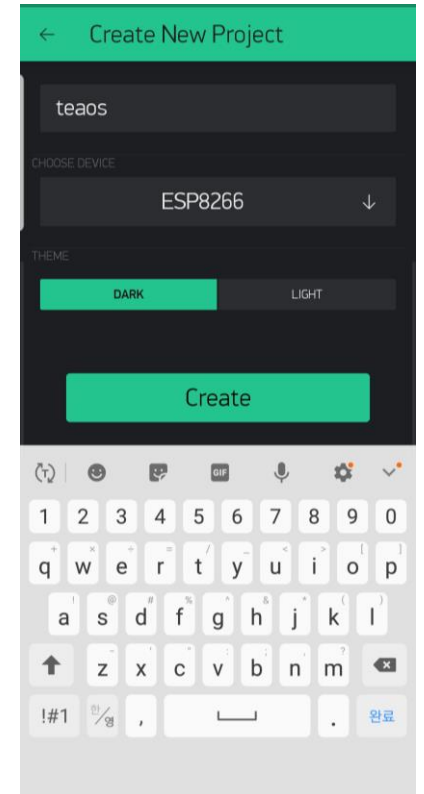
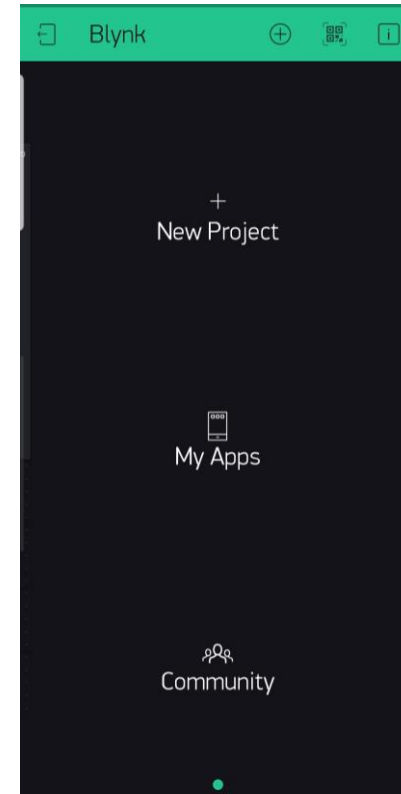


# 타킷 Blynk 라이브러리

- BlynkLib 모듈의 Blynk Class (계속)
  - email(to, subject, body)
    - 전자메일 전송
    - to: 수신자 주소
    - subject: 제목
    - body: 내용

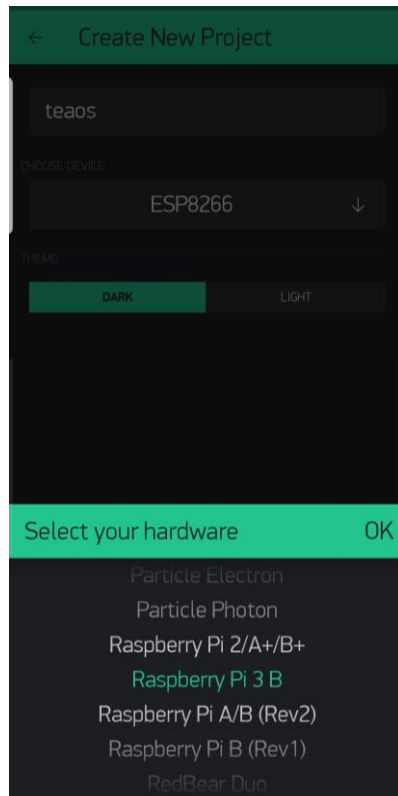
# 타킷의 CPU 온도 얻기

- 모바일 앱
  - Blynk에서 'New Project' 선택
    - 프로젝트 생성 화면 표시
    - 프로젝트 이름에 'SODA' 입력



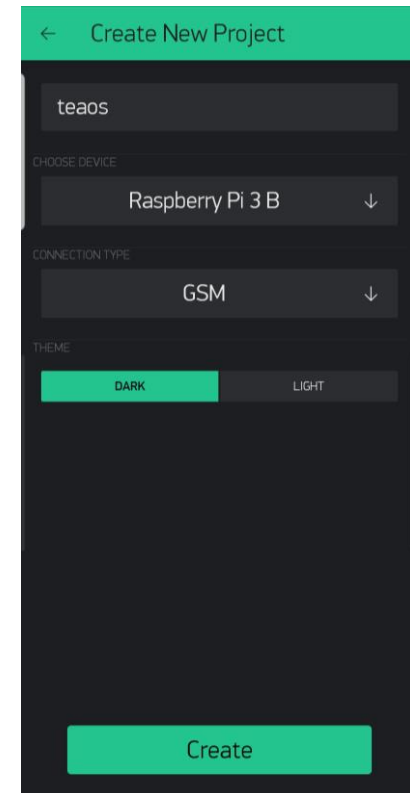
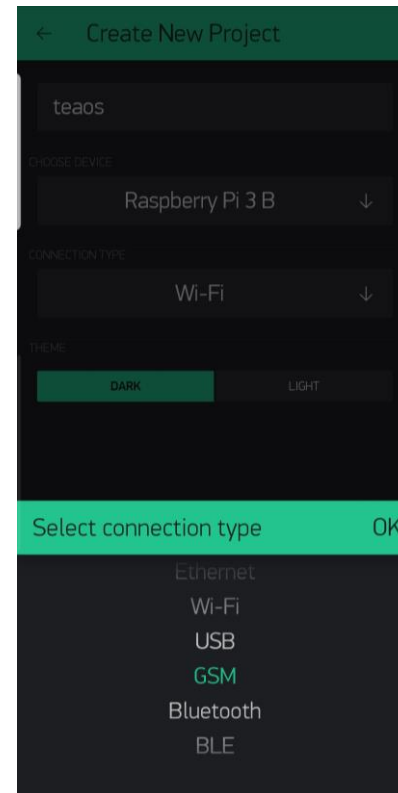
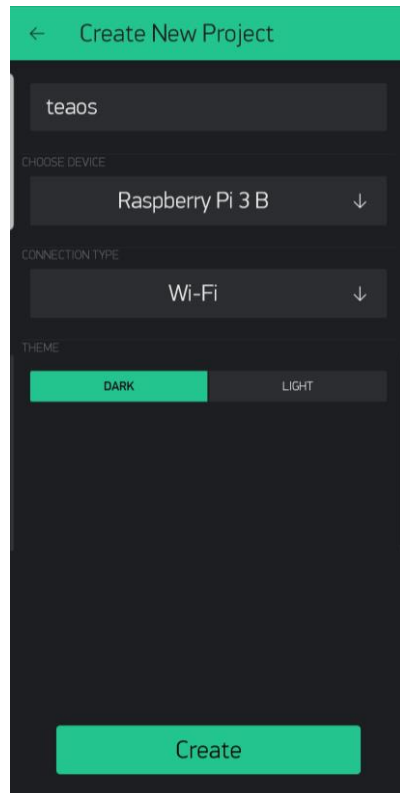
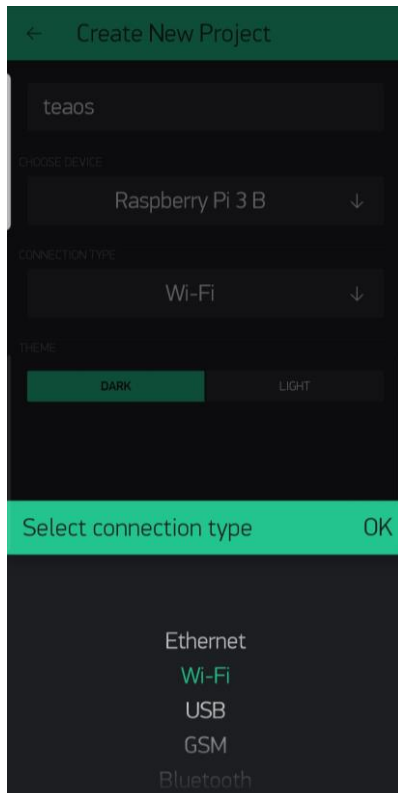
# 타깃의 CPU 온도 얻기

- 모바일 앱 (계속)
  - CHOOSE DEVICE는 'Raspberry Pi 3B' 선택



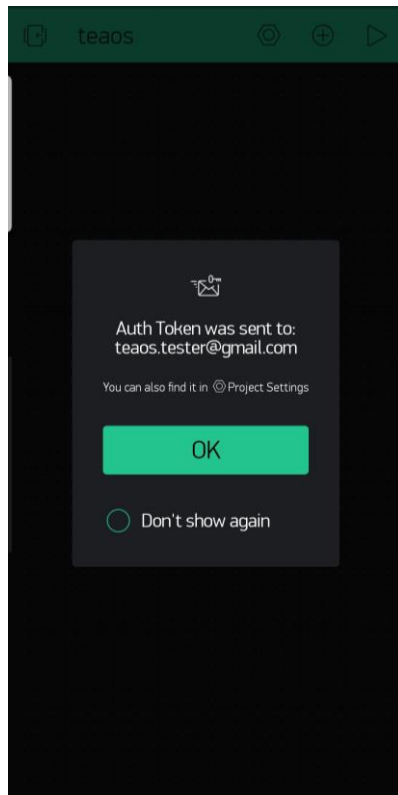
# 타킷의 CPU 온도 얻기

- 모바일 앱 (계속)
  - 인터넷 연결 유형인 CONNECTION TYPE은 'Wi-Fi' 또는 'GSM' 선택







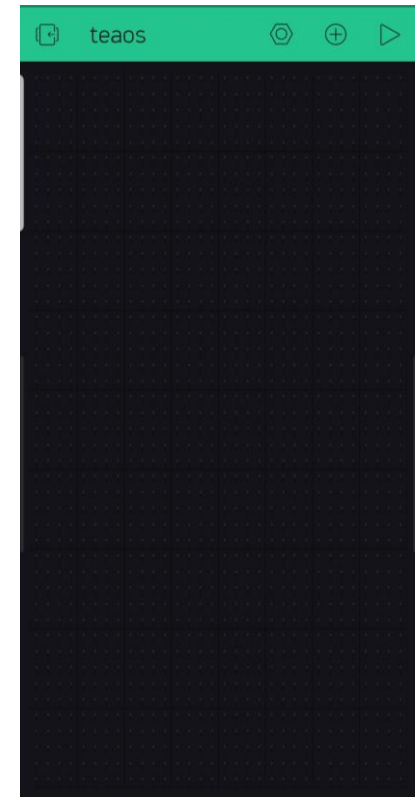
# 타킷의 CPU 온도 얻기

- 모바일 앱 (계속)
  - 프로젝트 생성이 완료되면 인증 토큰을 등록된 메일로 전송해 줌



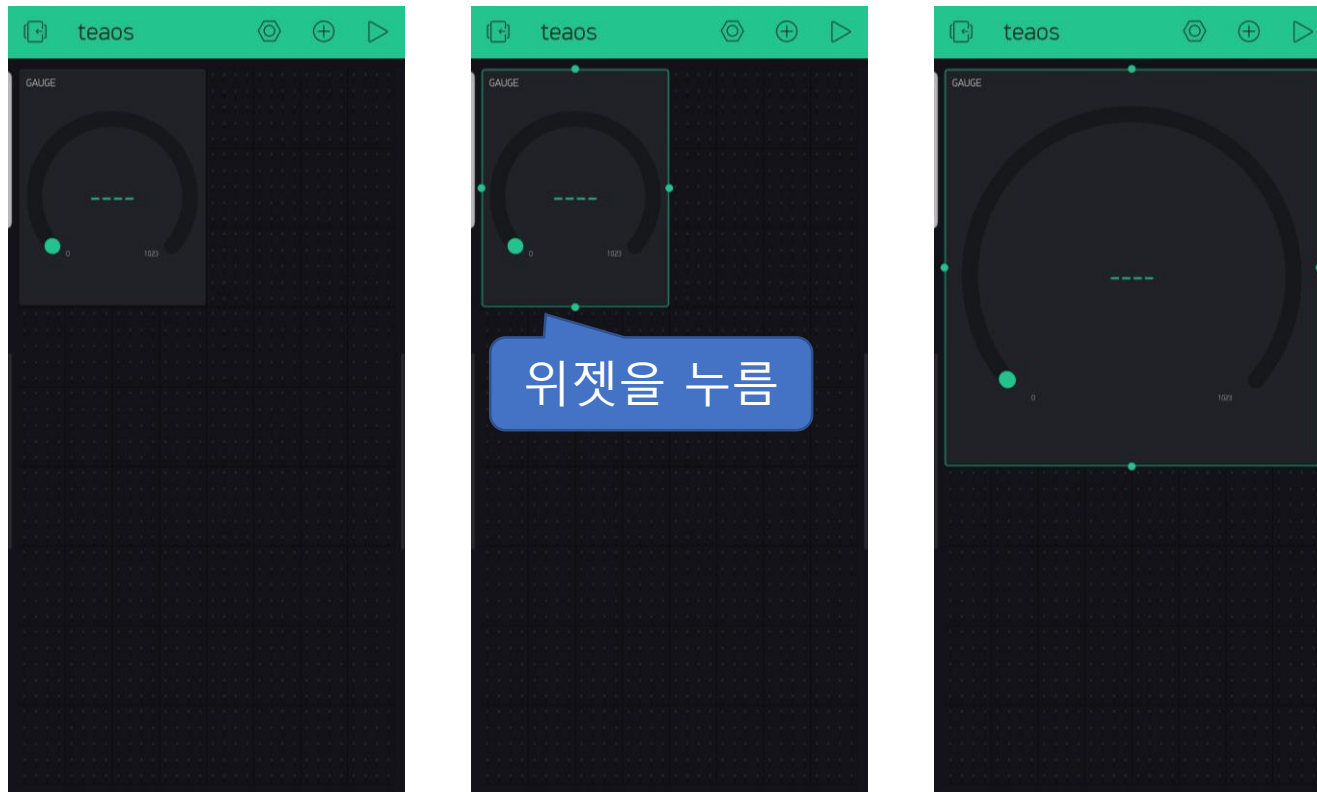
# 타킷의 CPU 온도 얻기

- 모바일 앱 (계속)
  - 빈 대시보드에 위젯을 추가해 UI 구현
  -  : 뒤로
    - 로그 아웃 또는 다른 프로젝트 선택
  -  : 현재 프로젝트 설정
  -  : 위젯 추가
    - 오른쪽에서 왼쪽으로 '스와이프' 해도 됨
  -  : 실행



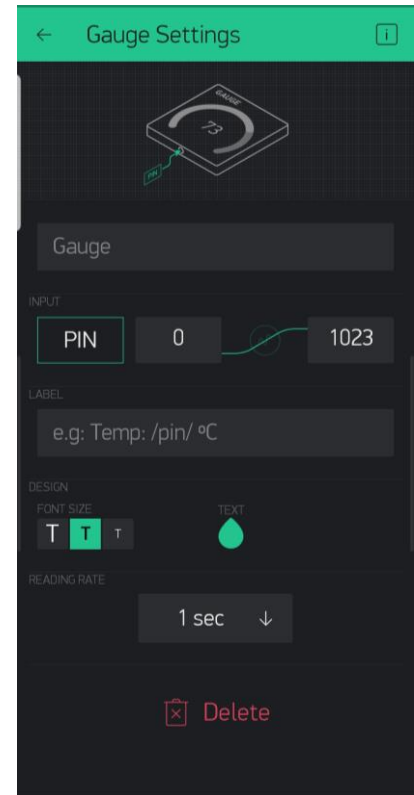
# 타킷의 CPU 온도 얻기

- 모바일 앱 (계속)
  - 타킷에서 수신한 온도를 표시할 GAUGE 위젯 추가



# 타킷의 CPU 온도 얻기

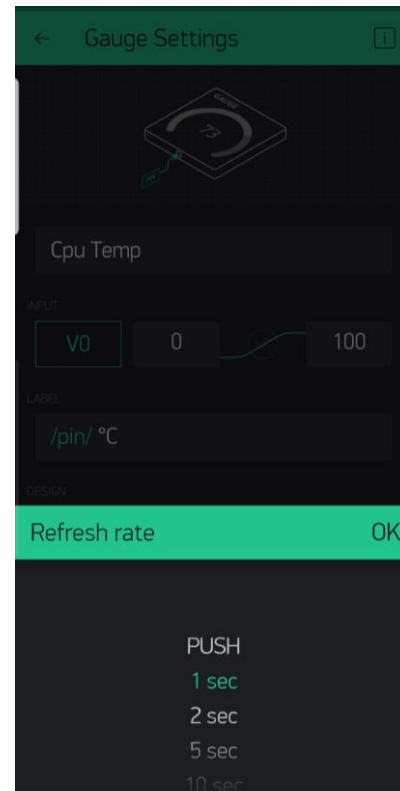
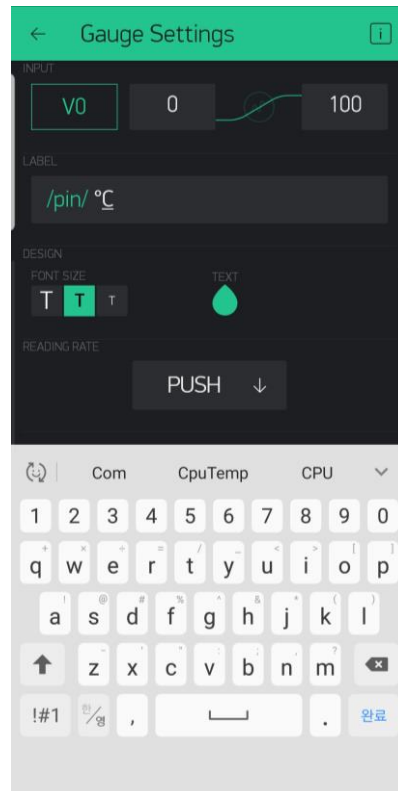
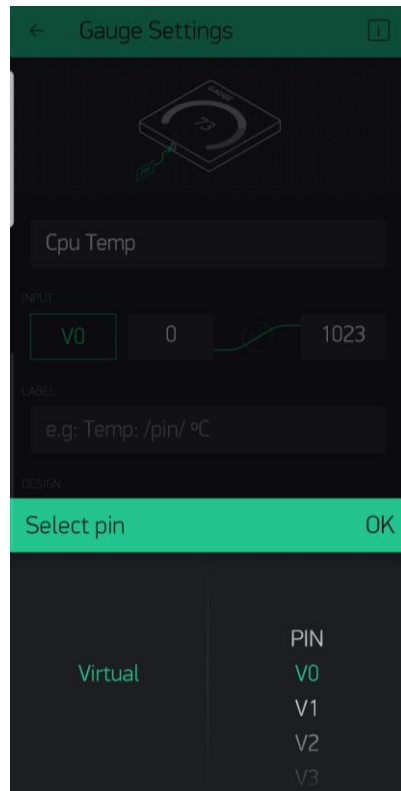
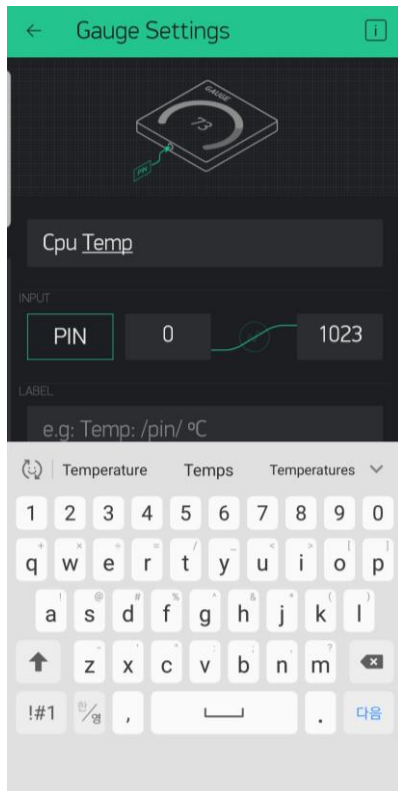
- 모바일 앱 (계속)
  - 위젯을 터치해 Settings 화면을 표시한 후 설정
    - Title: CPU Temp
    - INPUT: V0
    - LABEL: /pin/ °C
    - READING RATE: 1sec





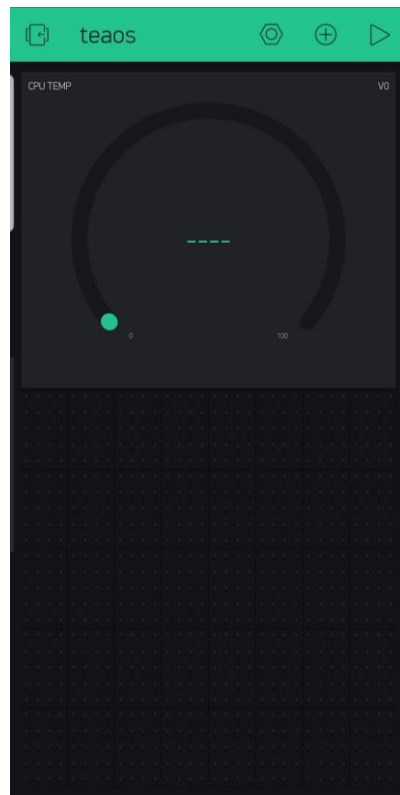
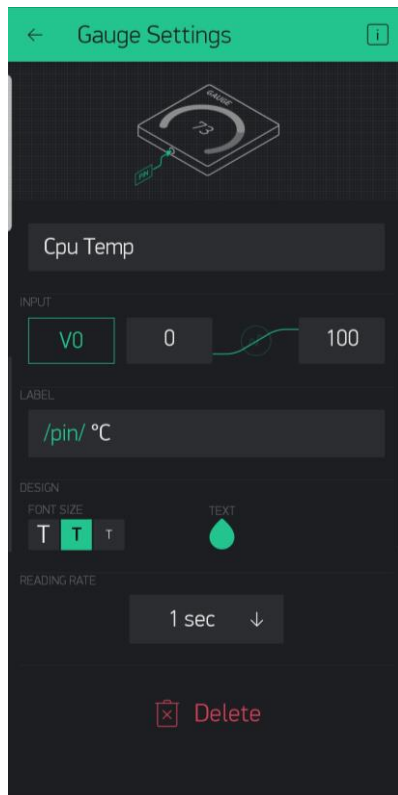
# 타킷의 CPU 온도 얻기

- 모바일 앱 (계속)
  - 위젯을 터치해 Settings 화면을 표시한 후 설정



# 타킷의 CPU 온도 얻기

- 모바일 앱 (계속)
  - 설정이 완료되면 왼쪽 상단의 를 눌러 대시보드로 복귀



# 타킷의 CPU 온도 얻기

- 타킷 서비스

- 모바일 앱에서 요청할 때마다 CPU 온도를 전송하는 코드 구현
  - 모바일 앱에서 프로젝트를 생성한 후 메일로 받은 인증 코드 필요

```
import BlynkLib
from subprocess import Popen, PIPE

blynk = BlynkLib.Blynk('f0b5d2417e4f42199c7ae84e8c15aa3f') #프로젝트를 생성할 때 메일로 받은 인증 코드

def get_cpu_temp():
    with Popen(['cat', '/sys/class/thermal/thermal_zone0/temp'], stdout=PIPE) as p:
        out, err = p.communicate() #out = "55687"
        return float("%.2f"%(int(out)/1000))


@blynk.VIRTUAL_READ(0) #모바일 앱의 게이지 위젯은 가상 포트0 사용
def on_cpu_temp():
    temp = get_cpu_temp()
    blynk.virtual_write(0, temp)

blynk.run()
```

# 타킷의 CPU 온도 얻기

- 타킷에서 서비스를 실행한 후 스마트폰에서 Blynk 대시보드 실행
  - 타킷과 스마트폰 모두 인터넷에 연결되어 있어야 함

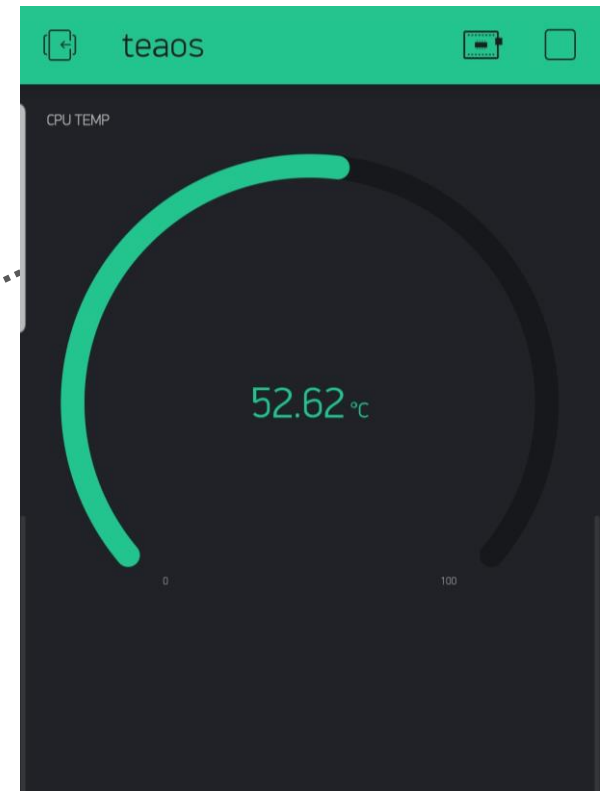
```
> ~ /Project/python/work
python3 blynk_cpu_temp.py
```



Give Blynk a Github star! => <https://github.com/>

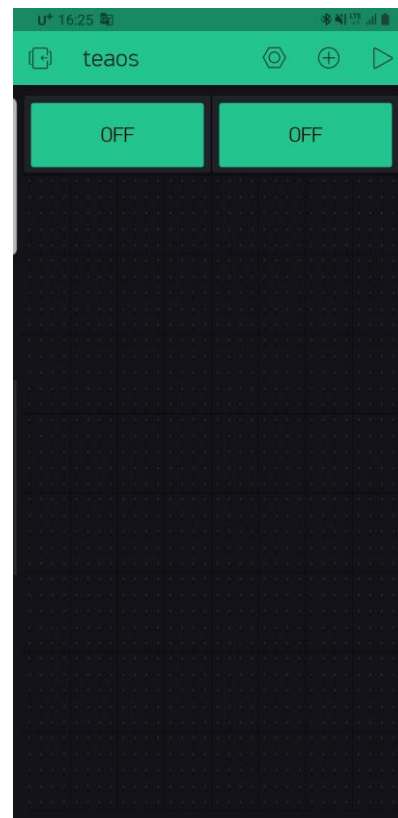
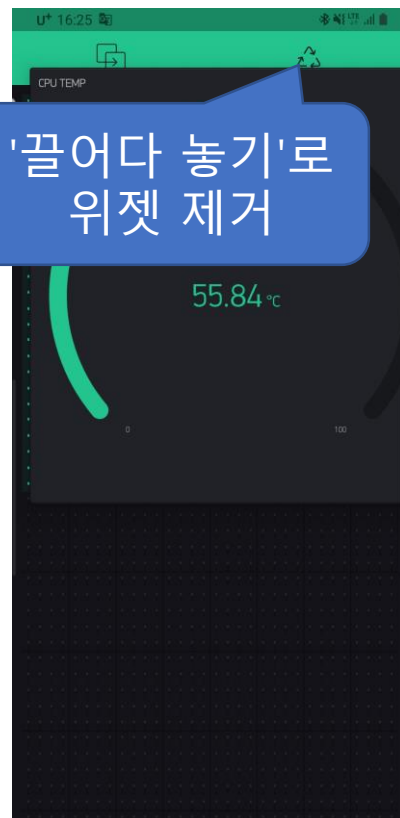
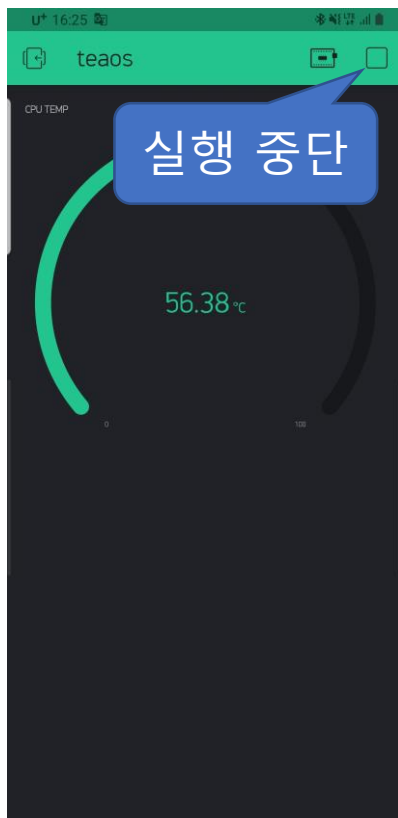
```
TCP: Connecting to blynk-cloud.com:80
Blynk connection successful, authenticating...
Access granted, happy Blynking!
```

blynk-cloud.com



# 타킷의 LED 켜고 끄기

- 모바일 앱
  - 기존 프로젝트에서 위젯을 제거한 후 Styled Button 2개 추가

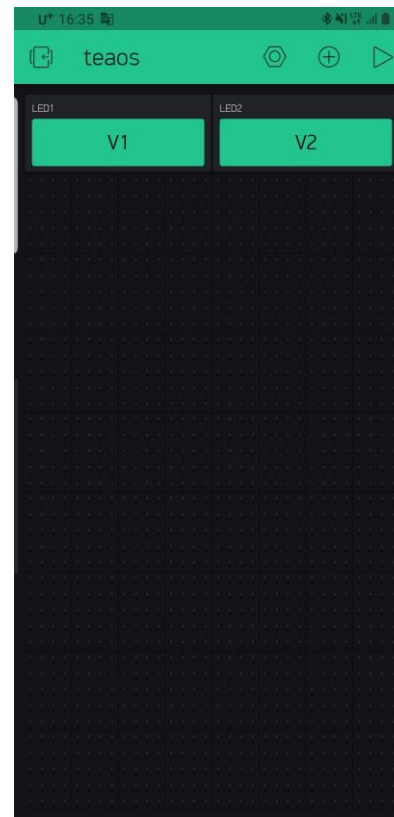
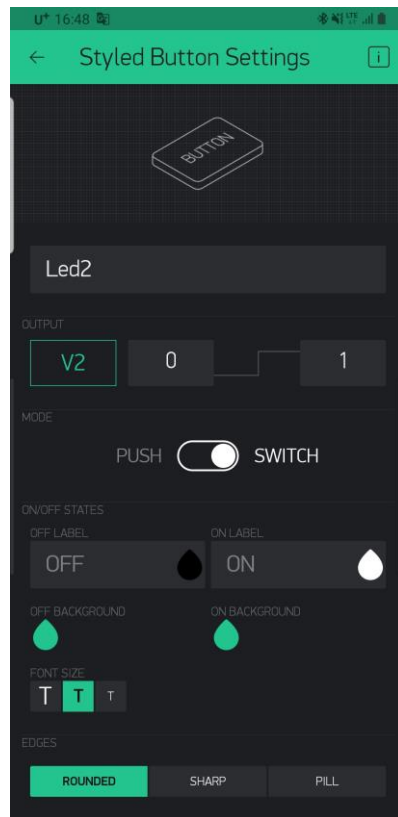
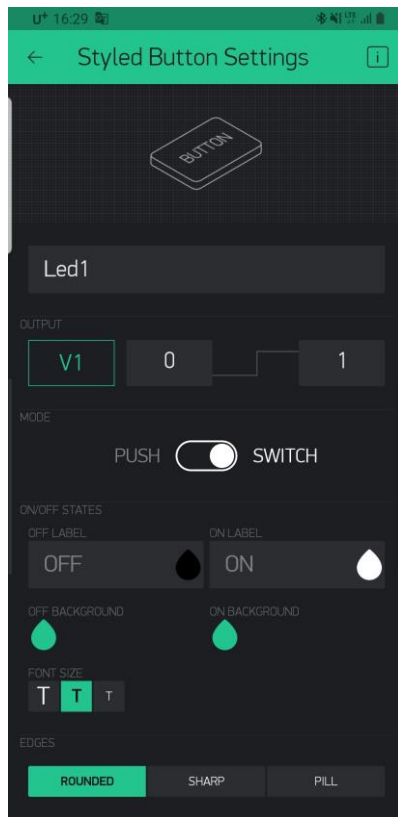


# 타킷의 LED 켜고 끄기

- 모바일 앱 (계속)
  - 위젯을 터치해 Settings 화면을 표시한 후 설정
    - Styled Button first
      - Title: Led1
      - OUTPUT: V1
      - MODE: SWITCH
      - OFF/ON LABEL: OFF, ON (default)
    - Styled Button second
      - Title: Led2
      - OUTPUT: V2
      - MODE: SWITCH
      - OFF/ON LABEL: OFF, ON (default)

# 타킷의 LED 켜고 끄기

- 모바일 앱 (계속)
  - 위젯을 터치해 Settings 화면을 표시한 후 설정



# 타킷의 LED 켜고 끄기

- 타킷 서비스
  - 모바일 앱에서 LED 제어 데이터를 보내주면 이를 처리하는 코드 구현

```
import BlynkLib
from subprocess import Popen

blynk = BlynkLib.Blynk('f0b5d2417e4f42199c7ae84e8c15aa3f') #자신의 프로젝트 인증 코드 사용

def led_ctl(num, on=True):
    Popen("echo %d > /sys/class/gpio/gpio%d/value"%(on, num + 13), shell=True)

@blynk.VIRTUAL_WRITE(1)    #첫 번째 Styled Button의 가상 포트
def on_led1_ctl(on):        #Styled Button이 ON이면 "1", OFF이면 "0" 수신
    led_ctl(1, bool(int(on)))

@blynk.VIRTUAL_WRITE(2)    #두 번째 Styled Button의 가상 포트
def on_led2_ctl(on):
    led_ctl(2, bool(int(on)))


blynk.run()
```



# 타킷의 LED 켜고 끄기

- 타킷에서 서비스를 실행한 후 스마트폰에서 Blynk 대시보드 실행
  - 타킷과 스마트폰 모두 인터넷에 연결되어 있어야 함

```
x > ~ /Project/python/work
python3 blynk_led.py
```



Give Blynk a Github star! => <https://github.com>

TCP: Connecting to blynk-cloud.com:80  
Blynk connection successful, authenticating...  
Access granted, happy Blynking!

blynk-cloud.com

