

[참고] Google Assistant SDK

<https://developers.google.com/assistant/sdk/guides/service/python/extend/custom-actions>

Register Custom Device Actions

action_power.ko.json

intent

```
"intent": {
  "name": "com.example.intents.OnOff",
  "parameters": [
    {
      "name": "onoff",
      "type": "OnOff"
    }
  ],
  "trigger": {
    "queryPatterns": [
      "전원(을)? $OnOff:onoff( )?주세요",
      "전원(을)? $OnOff:onoff( )?세요",
      "전원(을)? $OnOff:onoff"
    ]
  }
},
```

fulfillment

```
"fulfillment": {
  "staticFulfillment": {
    "templatedResponse": {
      "items": [
        {
          "simpleResponse": {
            "textToSpeech": "전원을 $onoff.raw했습니다."
          }
        },
        {
          "deviceExecution": {
            "command": "com.example.commands.OnOff",

```

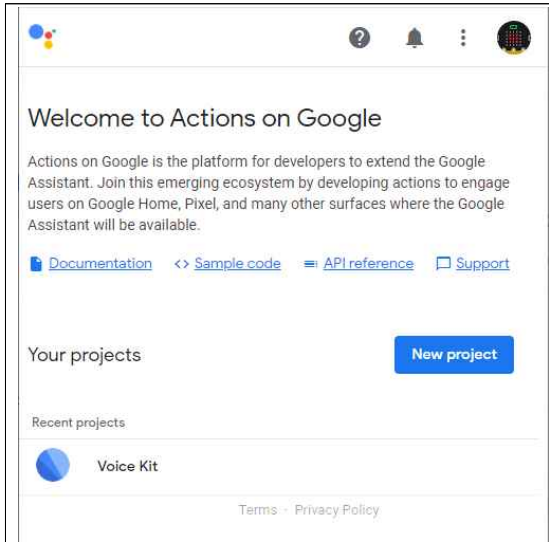
```
        "params": {
          "on": "$onoff"
        }
      }
    ]
  }
}
```

types

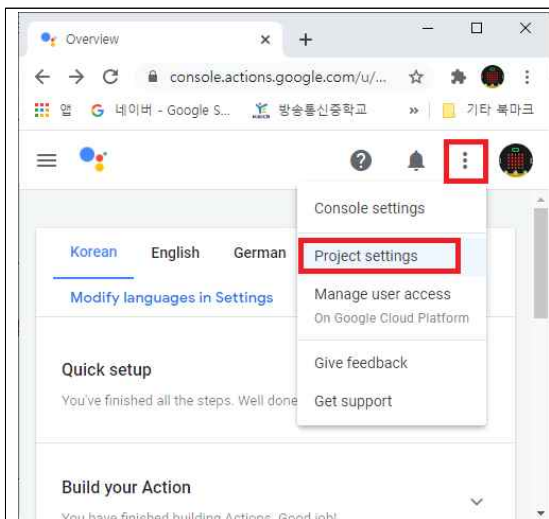
```
  "types": [
    {
      "name": "$OnOff",
      "entities": [
        {
          "key": "on",
          "synonyms": [
            "켜",
            "ㄱ"
          ]
        },
        {
          "key": "off",
          "synonyms": [
            "꺼",
            "ㄴ"
          ]
        }
      ]
    }
  ],
```

```
  "locale": "ko"
```

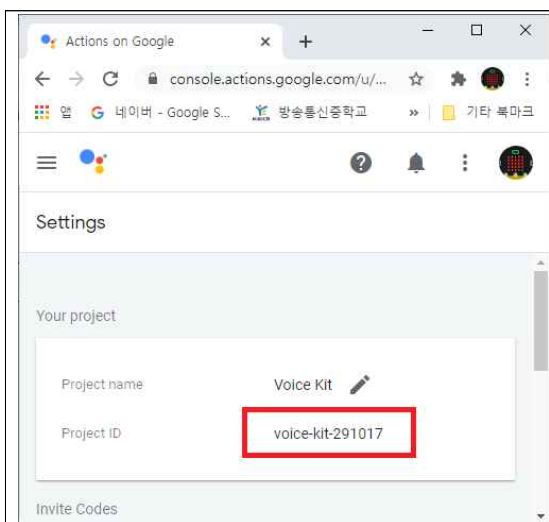
Project ID 확인하기



<https://console.actions.google.com/>
이 사이트에 접속하여 Voice Kit 프로젝트를 선택합니다.



3개 점이 있는 아이콘을 누르고 Project settings를 선택합니다.



여기에서 Project ID를 확인합니다.

[참고] Register Custom Device Actions

https://developers.google.com/assistant/sdk/guides/service/python/extend/custom-actions#deploy_the_action_package

Action Package(actions.json) 배포

Google Assistant 가 action***.json 파일에 있는 동작을 인식하기 위해서

1. gactions 명령어와 동일한 디렉토리에서 기존 자격 증명을 제거합니다.

```
del creds.data
```

2. gactions update 명령어를 이용하여 action.json 파일을 Google 에 저장합니다. project_id 는 Actions Console에서 확인할 수 있습니다.

```
gactions update --action_package actions.json --project project_id
```

[참고] 메시지가 표시 Requested entity is not found되면 Actions Console에서 프로젝트를 가져와야합니다 .

3. 이 명령을 처음 실행하면 URL이 제공되고 로그인하라는 메시지가 표시됩니다. URL을 복사하여 브라우저에 붙여 넣으십시오 (모든 시스템에서 수행 할 수 있음). 페이지에서 Google 계정에 로그인하라는 메시지가 표시됩니다. 이전 단계 에서 프로젝트를 만든 Google 계정에 로그인 합니다 .

4. API의 권한 요청을 승인하면 "4 / XXXX"와 같은 코드가 브라우저에 나타납니다. 이 코드를 복사하여 터미널에 붙여 넣으십시오.

인증 코드 입력 :

승인에 성공하면 다음과 유사한 응답이 표시됩니다.

```
Your app for the Assistant for project my-devices-project was successfully updated with your actions.
```

5. gactionsCLI 를 사용하여 Action Package를 테스트 모드로 배포합니다 . 이 명령을 실행하기 전에 한 번 이상 Action Package를 Google에 저장해야합니다. 테스트 모드는 사용자 계정에서만 작업 패키지를 활성화합니다.

```
gactions test --action_package actions.json --project project_id
```

[참고] Handle Commands

https://developers.google.com/assistant/sdk/guides/service/python/extend/handle-device-commands#find_the_command_handler

명령 처리

Google 어시스턴트의 명령에 대한 응답으로 기기에서 코드를 실행하려면 다음 안내를 따르세요.

1. 샘플 실행

Google 어시스턴트가 적절한 질문에 대해 On/Off 명령을 전송하는지 확인하세요.

```
$ cd ~/aiyprojects-raspbian/src/examples/voice
$ python3 assistant_grpc_demo.py
...
INFO:root:Press button to start conversation...
```

노란 버튼을 누르고 “전원 켜” 라고 말해보세요.

콘솔 출력에 다음 문이 표시되어야합니다.

```
INFO:root:Press button to start conversation...
INFO:root:Conversation started!
INFO:aiy.assistant.grpc:Recording started.
...
INFO:aiy.assistant.grpc:End of audio request detected.
INFO:aiy.assistant.grpc:You said: "전원을 켜 주세요".
INFO:aiy.assistant.grpc:Playing started.
INFO:root:Power on
DEBUG:aiy.assistant.grpc:Updating conversation state.
INFO:aiy.assistant.grpc:Assistant said: "전원을 켜겠습니다."
INFO:aiy.assistant.grpc:Not expecting follow-on query from user.
INFO:root:Waiting for device executions to complete.
INFO:aiy.assistant.grpc:Recording stopped.
INFO:aiy.assistant.grpc:Playing stopped.
```

2. 샘플 소스 코드 탐색

소스 코드에서 명령문이 출력되는 위치를 찾을 수 있습니다.

```
$ grep -nr "Power on"
./assistant_grpc_demo_snowboy.py:58:         logging.info('Power on')
./assistant_grpc_demo.py:55:         logging.info('Power on')
```

nano 명령어로 소스코드를 엽니다.

```
$ nano assistant_grpc_demo.py
```

Ctrl + W를 눌러 OnOff를 검색하세요. 아니면 Alt + G를 누를 후 라인 번호 55를 입력하고 Enter를 눌러보세요.

```
50     device_handler = action_helpers.DeviceRequestHandler(device_id)
51
52     @device_handler.command('com.example.commands.OnOff')
53     def onoff(on):
54         if on == "on":
55             logging.info('Power on')
56         elif on == "off":
57             logging.info('Power off')
```

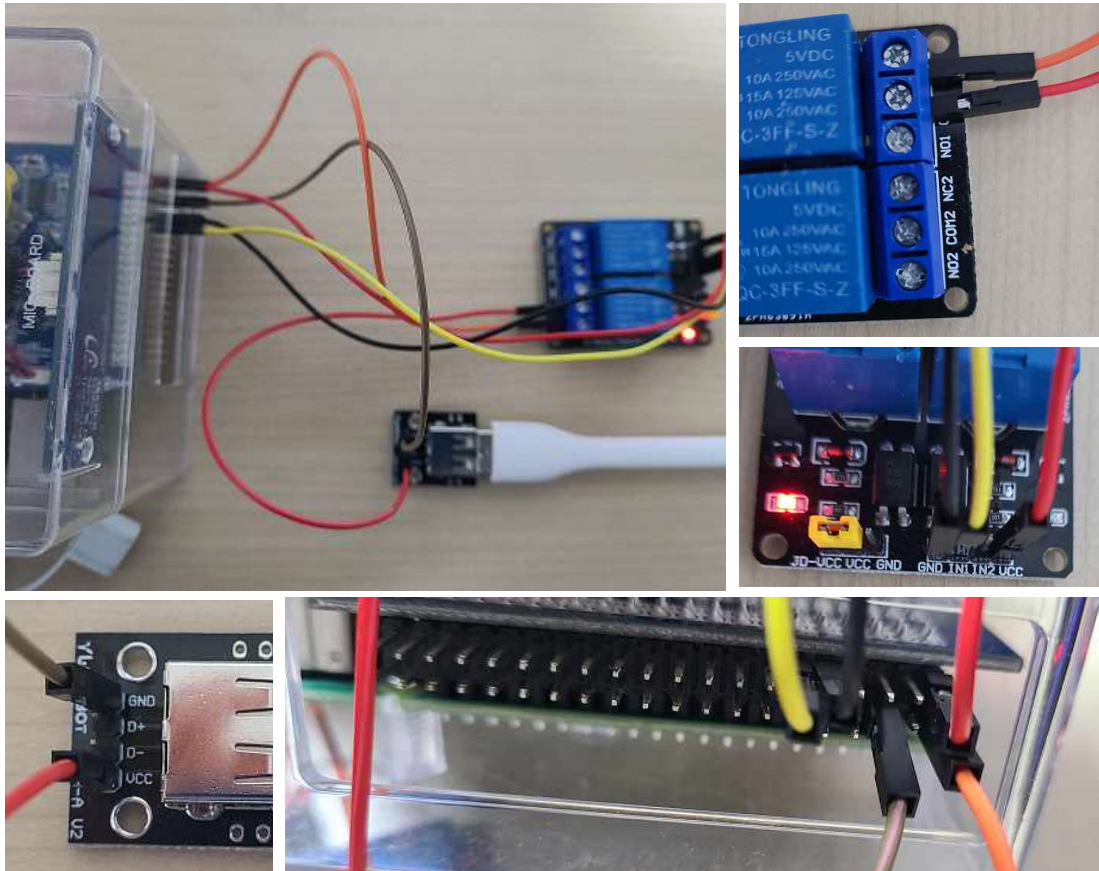
여기 있는 코드가 actions.joon 파일의 onoff()명령 을 처리합니다

```
    "deviceExecution": {
        "command": "com.example.commands.OnOff",
        "params": {
            "on": "$onoff"
        }
    }
```

GPIO를 이용한 LED 제어

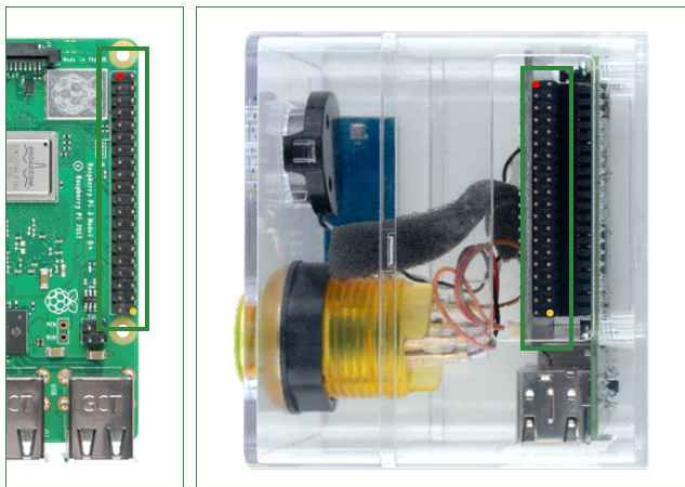
1. 샤오미 USB 램프를 릴레이 스위치에 연결하여 GPIO를 이용하여 제어해 보겠습니다.

[준비물] 라즈베리파이 키트, 릴레이, USB 소켓, USB LED램프, 전선

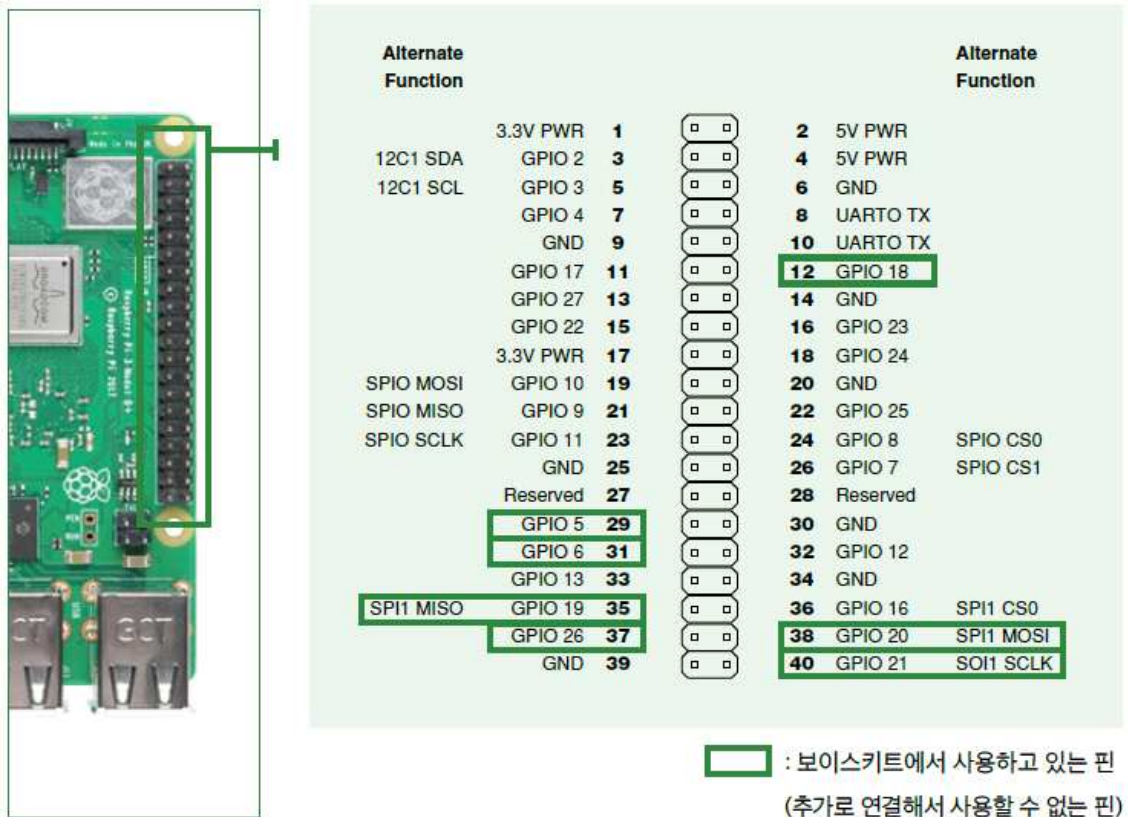


2. 보이스키트 핀 배치도

보이스 키트의 핀배치는 라즈베리파이의 기존 배치와 동일합니다.



3. 라즈베리파이 GPIO 핀 배치도



4. RPi.GPIO 라이브러리

setmode(numbering)

확장 커넥터 핀의 번호 할당 방법을 지정

numbering: GPIO.BOARD 또는 GPIO.BCM

GPIO.BOARD는 커넥터 핀번호, GPIO.BCM은 CPU 핀번호

setup(channel, input/output, [pull_up_down 또는 initial])

사용할 GPIO 핀을 설정, GPIO 핀을 사용

channel: 핀 번호를 setmode()에 설정한 할당 방법으로 지정

input/output: 입력이면 GPIO.IN, 출력이면 GPIO.OUT

pull_up_down: 입력 핀이면 풀업으로 할지 풀다운으로 할지 지정.

풀업은 GPIO.PUD_UP, 풀다운은 GPIO.PUD_DOWN

initial: 출력할 초기 상태를 지정, GPIO.HIGH는 하이레벨, GPIO.LOW 는 로우 레벨

output(channel, state)

channel로 지정한 핀에 state로 지정한 값을 출력

channel: 핀 번호는 setmode()에 설정한 할당 방법으로 지정

state: 출력할 값을 지정. 0/GPIO.LOW/False는 로우 레벨

1/GPIO.HIGH/True는 하이 레벨

cleanup(channel)

channel에서 사용한 GPIO를 패러미터(인수)가 없으면 모든 GPIO를 개방

프로그램을 종료할 때 반드시 실행

channel: 핀 번호는 setmode()에 설정한 할당 방법으로 지정

time.sleep(seconds)

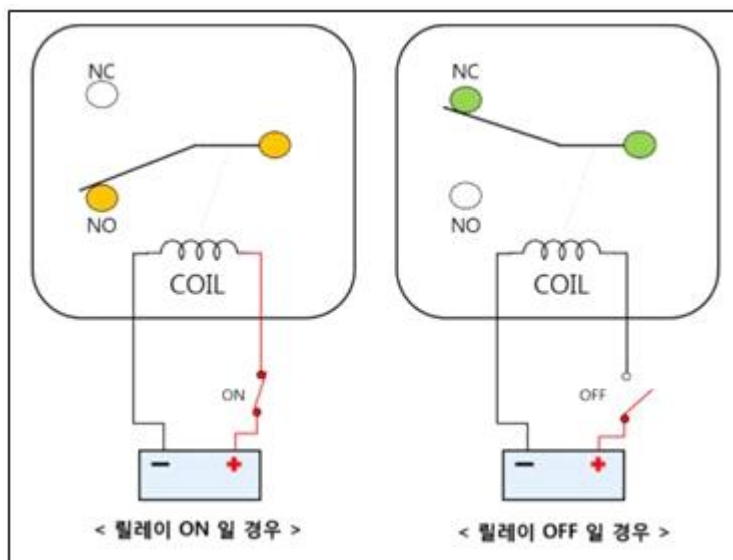
일정 시간동안 프로세스를 일시정지할 수 있다

seconds: 1을 입력하면 1초 대기하고 0.5를 입력하면 0.5초 대기

5. 릴레이 스위치

릴레이 (Relay)는 전자석의 원리를 이용한 스위치 기능을 가지고 있습니다. 따라서 릴레이에 센서나 모듈 또는 가정에서 사용하는 멀티탭이나 형광등 스위치 등을 연결하여 스위치로써 ON / OFF 를 제어할 수 있습니다. 단, 220V같이 높은 전압을 공급하는 장치를 제어할 때는 위험할수 있기 때문에 제품의 회로 등을 정확하게 파악하고 사용하여야 합니다.

[출처] 아두이노 1채널 5V 릴레이 모듈 / Arduino Relay Module|작성자 에듀이노 오픈랩



6. 소스 코드

LED가 켜지도록 소스코드를 참고하여 수정하세요.

출력 핀을 초기에 LOW 상태로 설정합니다. on 명령이 수신되면 핀을 HIGH 상태로 설정합니다. off 명령이 수신되면 핀을 LOW 상태로 설정합니다.

[참고] 이 소스코드에서는 GPIO 번호로 핀을 참조합니다. GPIO 17번은 Raspberry Pi의 물리적 핀 11번을 나타냅니다.

샤오미 LED 램프는 전원을 끌 때 두 번을 꺼줘야 합니다. 즉 GPIO 핀 상태를 LOW-HIGH-LOW 순으로 해야합니다.

```
...
...
...
from aiy.assistant import auth_helpers, device_helpers, action_helpers
from aiy.assistant.grpc import AssistantServiceClientWithLed
from aiy.board import Board

import RPi.GPIO as GPIO
import time

...
..
..

device_handler = action_helpers.DeviceRequestHandler(device_id)

GPIO.setmode(GPIO.BOARD)
LED = 11
GPIO.setup(LED, GPIO.OUT, initial=GPIO.LOW)

@device_handler.command('com.example.commands.OnOff')
def onoff(on):
    if on == "on":
        logging.info('Power on')
        GPIO.output(LED, GPIO.HIGH)
    elif on == "off":
        logging.info('Power off')
        GPIO.output(LED, GPIO.LOW)
        time.sleep(0.5)
        GPIO.output(LED, GPIO.HIGH)
        time.sleep(0.5)
        GPIO.output(LED, GPIO.LOW)
```

```
with Board() as board:
    assistant = AssistantServiceClientWithLed(board=board,
volume_percentage=args.volume,

language_code=args.language,

device_handler=device_handler)
    while True:
        logging.info('Press button to start conversation...')
        board.button.wait_for_press()
        logging.info('Conversation started!')
        assistant.conversation()

GPIO.cleanup(LED)
...
..
..
```

[참고] ActionPackage

<https://developers.google.com/assistant/conversational/df-asdk/reference/action-package/rest/Shared.Types/ActionPackage>

ActionPackage holds the content for the draft of an App as well as each deployed version. This includes directory listing details, conversation configuration and account linking.

JSON representation

```
{
  "manifest": {
    object (Manifest)
  },
  "accountLinking": {
    object (AccountLinking)
  },
  "actions": [
    {
      object (Action)
    }
  ],
  "types": [
    {
      object (Type)
    }
  ],
  "conversations": {
    string: {
      object (ConversationFulfillment)
    },
    ...
  },
  "locale": string
}
```

Examples

```
{
  "manifest": {
```

```

        "displayName": "Blinky light",
        "invocationName": "Blinky light",
        "category": "PRODUCTIVITY"
    },
    "actions": [
        {
            "name": "com.example.actions.BlinkLight",
            "availability": {
                "deviceClasses": [
                    {
                        "assistantSdkDevice": {}
                    }
                ]
            },
            "intent": {
                "name": "com.example.intents.BlinkLight",
                "parameters": [
                    {
                        "name": "number",
                        "type": "SchemaOrg_Number"
                    },
                    {
                        "name": "speed",
                        "type": "Speed"
                    }
                ],
                "trigger": {
                    "queryPatterns": [
                        "blink ($Speed:speed)? $SchemaOrg_Number:number time s",
                        "blink $SchemaOrg_Number:number times ($Speed:speed)?"
                    ]
                }
            },
            "fulfillment": {
                "staticFulfillment": {
                    "templatedResponse": {
                        "items": [
                            {

```

```

        "simpleResponse": {
            "textToSpeech": "Blinking $number times"
        }
    },
    {
        "deviceExecution": {
            "command": "com.example.commands.BlinkLig
ht",
            "params": {
                "speed": "$speed",
                "number": "$number"
            }
        }
    }
]
}
}
},
],
"types": [
    {
        "name": "$Speed",
        "entities": [
            {
                "key": "SLOWLY",
                "synonyms": [
                    "slowly",
                    "slow"
                ]
            },
            {
                "key": "NORMALLY",
                "synonyms": [
                    "normally",
                    "regular"
                ]
            },
            {
                "key": "QUICKLY",

```

```
    "synonyms": [  
        "quickly",  
        "fast",  
        "quick"  
    ]  
}  
]  
}
```

Google Cloud Speech-to-Text

Google AI 기술로 지원되는 API를 사용하여 음성을 텍스트로 정확하게 변환할 수 있습니다.

1. Cloud Speech API 키 발급 받기

키 발급을 위해서 체크 카드 혹은 신용 카드가 필요합니다. 오늘은 선생님이 등록해 둔 키를 이용하여 간단히 살펴보겠습니다.

키 발급을 직접 해보고 싶은 학생은 요기를 참고하세요.

<https://webnautes.tistory.com/1247>

설명에 있는 “14. JSON을 선택하고 만들기를 클릭합니다.”에서 json 파일을 다운 받고 파일명을 “cloud_speech.json” 으로 바꿔주세요. 그리고 /home/pi 폴더로 복사합니다.

2. demo 파일 실행

```
$ cd ~/aiyprojects-raspbian/src/examples/voice
$ python3 cloudspeech_demo.py
```

다음 순으로 말해 봅니다.

```
turn on the light
turn off the light
blink the light
goodbye
```

2. 소스 코드

[참고] cloudspeech_demp.py

<https://github.com/google/aiyprojects-raspbian/tree/aiyprojects/src/examples/voice>

<https://aiyprojects.withgoogle.com/voice/#makers-guide--custom-voice-user-interface>

```
#!/usr/bin/env python3
# Copyright 2017 Google Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
```



```

#
#   http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# implied.
# See the License for the specific language governing permissions and
# limitations under the License.

"""A demo of the Google CloudSpeech recognizer."""
import argparse
import locale
import logging

from aiyo.board import Board, Led
from aiyo.cloudspeech import CloudSpeechClient

def get_hints(language_code):
    if language_code.startswith('en_'):
        return ('turn on the light',
                'turn off the light',
                'blink the light',
                'goodbye')
    return None

def locale_language():
    language, _ = locale.getdefaultlocale()
    return language

def main():
    logging.basicConfig(level=logging.DEBUG)

    parser = argparse.ArgumentParser(description='Assistant service example.')
    parser.add_argument('--language', default=locale_language())
    args = parser.parse_args()

    logging.info('Initializing for language %s...', args.language)
    hints = get_hints(args.language)

```

```

client = CloudSpeechClient()
with Board() as board:
    while True:
        if hints:
            logging.info('Say something, e.g. %s.' % ', '.join(hints))
        else:
            logging.info('Say something.')
        text = client.recognize(language_code=args.language,
                                hint_phrases=hints)

        if text is None:
            logging.info('You said nothing.')
            continue

        logging.info('You said: "%s"' % text)
        text = text.lower()
        if 'turn on the light' in text:
            board.led.state = Led.ON
        elif 'turn off the light' in text:
            board.led.state = Led.OFF
        elif 'blink the light' in text:
            board.led.state = Led.BLINK
        elif 'goodbye' in text:
            break

if __name__ == '__main__':
    main()

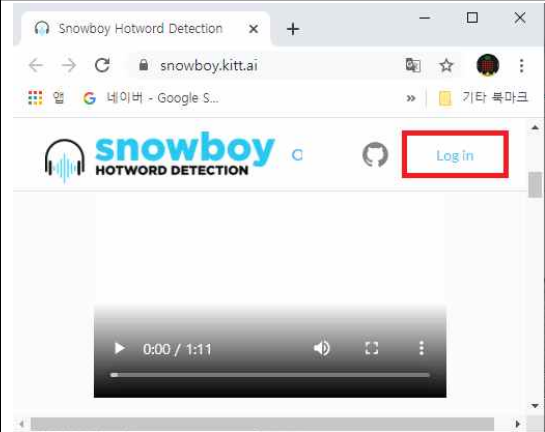
```

[참고] <https://pimylifeup.com/raspberry-pi-snowboy/>

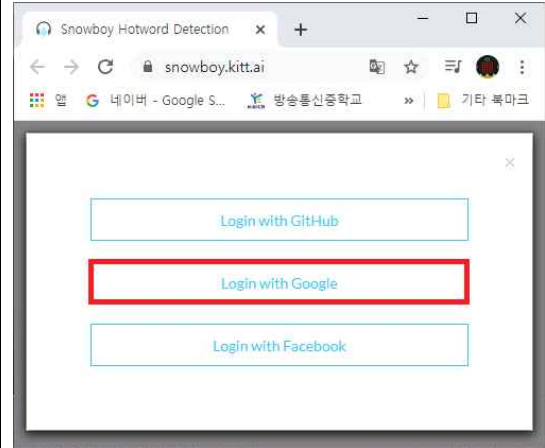
Snowboy, 사용자 호출어, Custom Hotword

Raspberry Pi에서 Snowboy를 설정하고, 호출어(hotword)를 훈련하고 감지하는 방법을 알아보겠습니다.

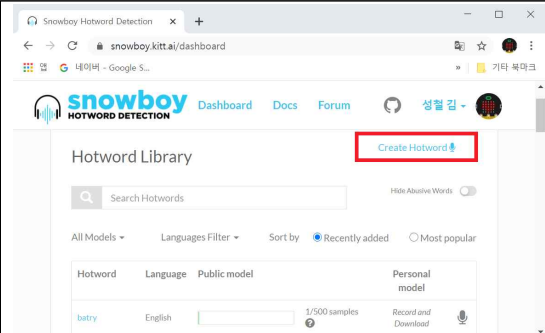
1. Snowboy API 키 얻기



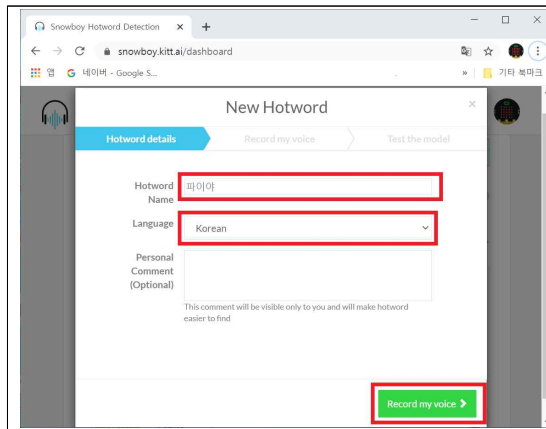
<https://snowboy.kitt.ai/>
Snowboy 웹사이트에 방문하여 “Log in” 버튼을 클릭합니다.



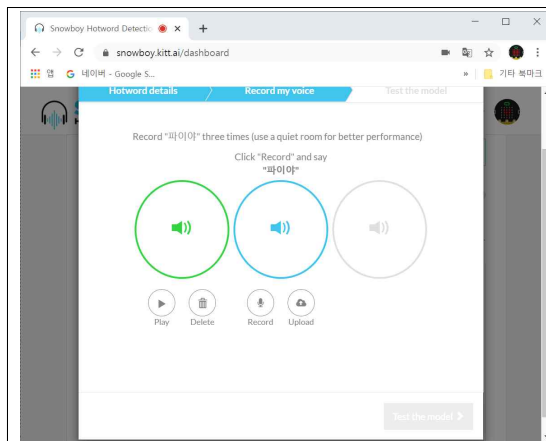
Snowboy 서비스에 로그인하는 세 가지 옵션 중에서 선택하라는 메시지가 표시됩니다.
원하는 방식으로 로그인하세요.



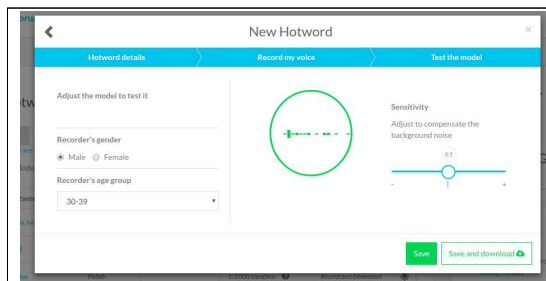
Create Hotword 버튼을 누릅니다.



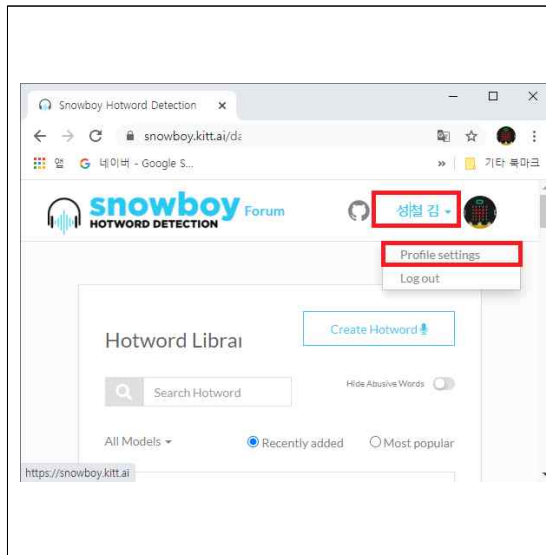
Hotword Name을 정하고, Language를 Korean으로 선택합니다.
Recording voice 버튼을 눌러 다음 단계로 진행합니다.



Record 버튼을 누르고 “파이야” 같은 자신의 호출어를 말하고 기다리다 다시 호출어를 말하고 3번 말합니다.
완료후 오른쪽 밑에 Test the model 버튼을 누릅니다.

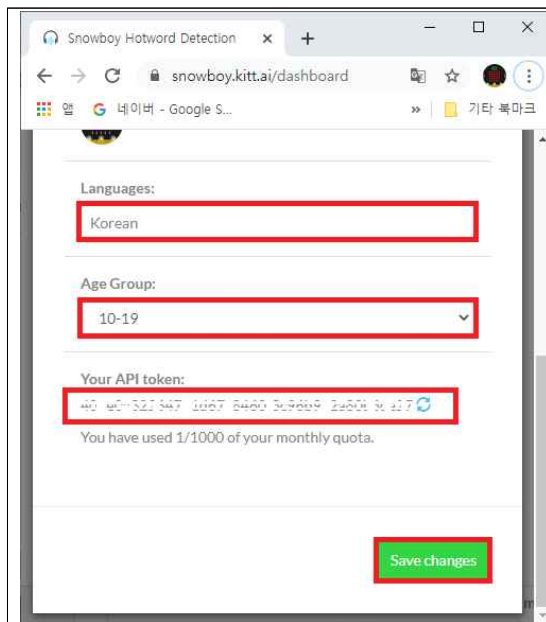


테스트를 완료해서 생성된 파일을 라즈베리 파이의 `/home/pi/aiyprojects-raspbian/src/examples/voice` 로 복사합니다.
라즈베리파이에서 실행하여 확인해봅니다.
`python3 assistant_grpc_demo_snowboy.py --model <my_model>.pmdl`



이 과정은 terminal에서 호출어를 만드는 방법입니다. 실제 해보니 마이크 문제로 잘 안 됩니다. 여러분 환경에 따라 가능할 수 있으니 조용한 공간에서 확인 해봅니다.

로그인하면 화면에 현재 계산중인 모든 핫 워드가 나열됩니다. 우리가 원하는 것은 오른쪽 상단 모서리에 있습니다. 이름을 클릭 한 다음 “Profile Settings” 클릭하면 API키가 표시되는 페이지로 이동합니다.



Snowboy 서버와 통신하기 위해 필요한 API 토큰을 복사합니다.

언어와 그룹설정은 옵션으로 필수 사항이 아닙니다.

Languages:에 Korean을 입력하고, Age Group에서 10-19를 선택합니다. Save changes 버튼을 클릭합니다.

2. API 토큰 복사

```
$ cd ~/aiyprojects-raspbian/src/examples/voice
$ nano training_service.py
```

```
9 endpoint = "https://snowboy.kitt.ai/api/v1/train/"
10
11 token = "복사한 토큰을 여기 붙여넣어주세요"
12 hotword_name = "ENTER_HOTWORD"
13 language = "en"
14 age_group = "20_29"
15 gender = "M"
16 microphone = "usb microphone"
```

3. 사용자 호출어 파일 만들기

arecord 명령을 이용하여 자신의 목소리로 호출어(Howword)를 만듭니다.

```
$ arecord --format=S16_LE --duration=5 --rate=16000 --file-type=wav 1.wav
$ arecord --format=S16_LE --duration=5 --rate=16000 --file-type=wav 2.wav
$ arecord --format=S16_LE --duration=5 --rate=16000 --file-type=wav 3.wav
$ python3 training_service.py 1.wav 2.wav 3.wav saved_model.pmdl
Saved model to 'saved_model.pmdl'.
```

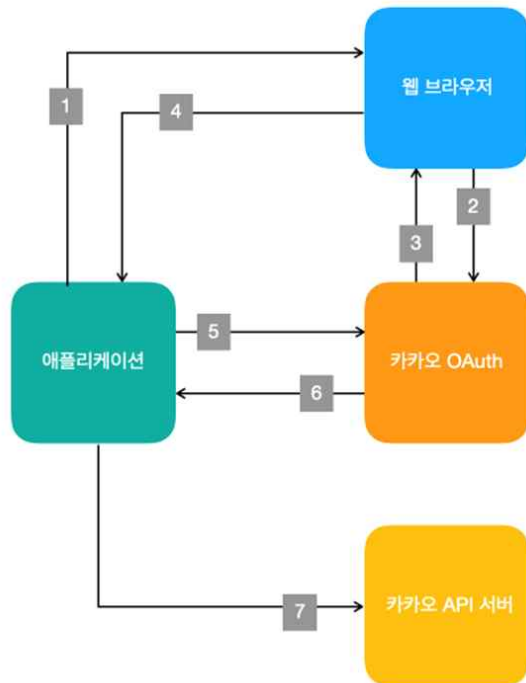
4. 실행

```
$ python3 assistant_grpc_demo_snowboy.py --model saved_model.pmdl
...
...
...
INFO:root:Listening... Press Ctrl+C to exit
DEBUG:root:detecting...

<호출어 인식후>

INFO:root:Keyword 1 detected at time: 2020-11-06 20:51:57
```

OAuth 2.0



0, 카카오 서비스를 이용하려면 사전에 앱을 등록 후 앱의 API 키를 발급받아야 합니다.

1, 카카오 서비스에 접근하기 위해 카카오 로그인 페이지를 웹브라우저에 표시합니다.

2, 사용자는 로그인 페이지에 ID, 비밀번호를 입력해 인증을 요청합니다.

3, 카카오 인증서버는 사용자를 인증하고 사용자 인증완료 페이지로 리다이렉션합니다. 이때 인증 코드를 함께 전달합니다.

4, 애플리케이션은 리다이렉션 URL을 분석해 인증코드를 획득합니다.

5, 애플리케이션은 카카오 인증서버에 제대로 인증된 사용자인지 확인 요청을 합니다.

6, 카카오 인증서버는 인증된 사용자임을 확인하고, 액세스 토큰(Access Token)을 발급합니다.