



AI with IoT

PART IV. IoT with Voice & Vision

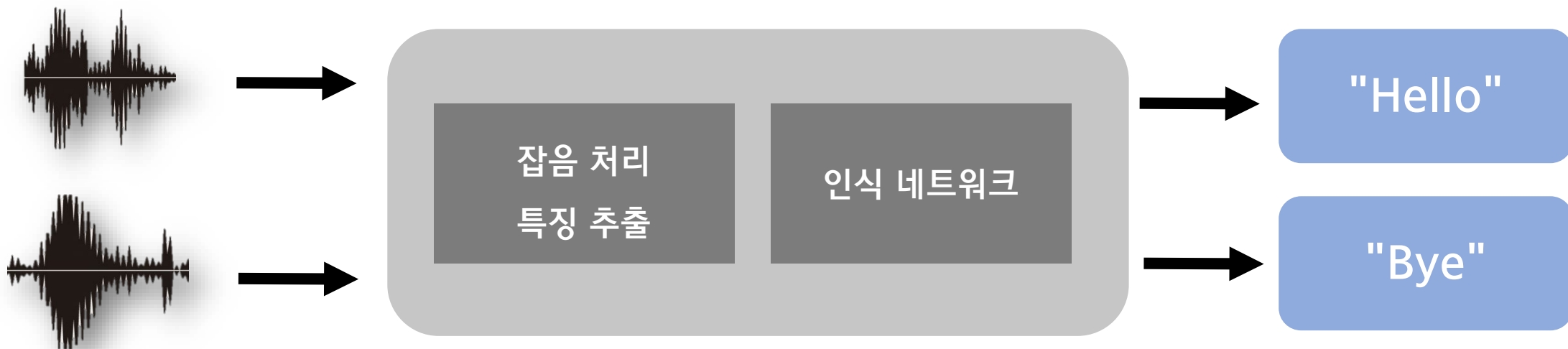
음성 인식 기술 – 자동 음성 인식



자동 음성 인식(ASR)

사용자가 말할 때 생기는 음성 정보를 해석해 문자 데이터로 전환하는 기술

- 발음이 유사한 단어를 구별하기 위하여 머신 러닝 모델을 활용
- 상황 별로 분리된 데이터를 바탕으로 학습시키고, 알맞은 경우에 적용하면 좋은 결과를 얻을 수 있음
(e.g. 전화 통화 음성과 뉴스 동영상을 분리시켜 학습하고 모델 형성)
- NLP 기술과 함께 사용해 문맥에 맞는 문장 단위 음성을 인식해낼 수 있음



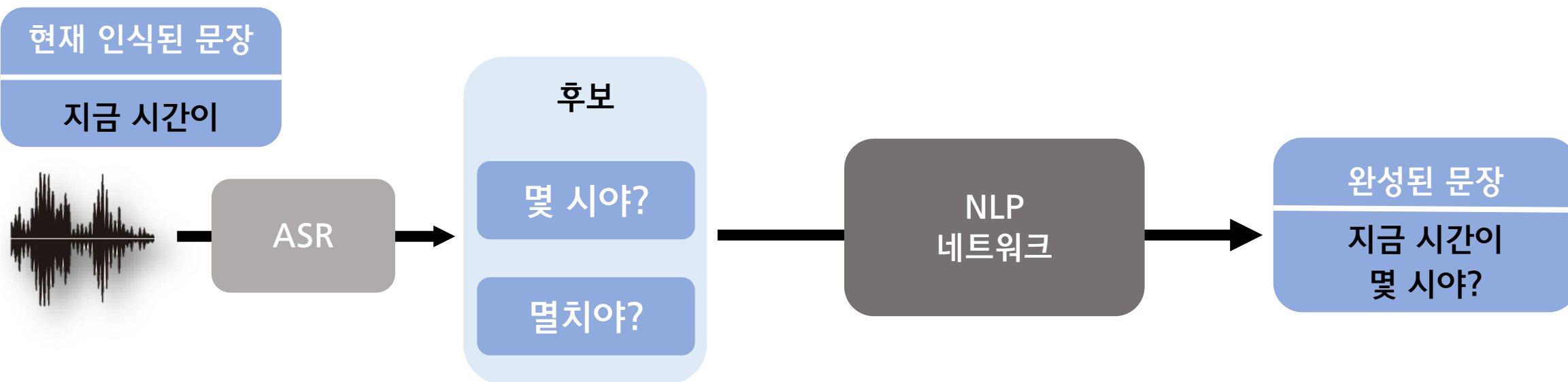
음성 인식 기술 – 자연어 처리



자연어 처리(NLP)

컴퓨터가 사람이 사용하는 언어인 자연어의 의미를 이해하고 분석하는 인공지능 기술

- 정돈되지 않은 문장에서 자연어 법칙을 얻어내어 컴퓨터가 이해할 수 있는 형태로 변환
- 문장에서 의미 있는 정보를 분석하고 추출해낼 수 있음
- 과거에는 언어학에 기반해 사람이 직접 추출한 Feature를 기반으로 구현됨
- 현재는 딥러닝을 통해 Feature를 생성해내어 빠르고 확장성 있는 처리 시스템 구축



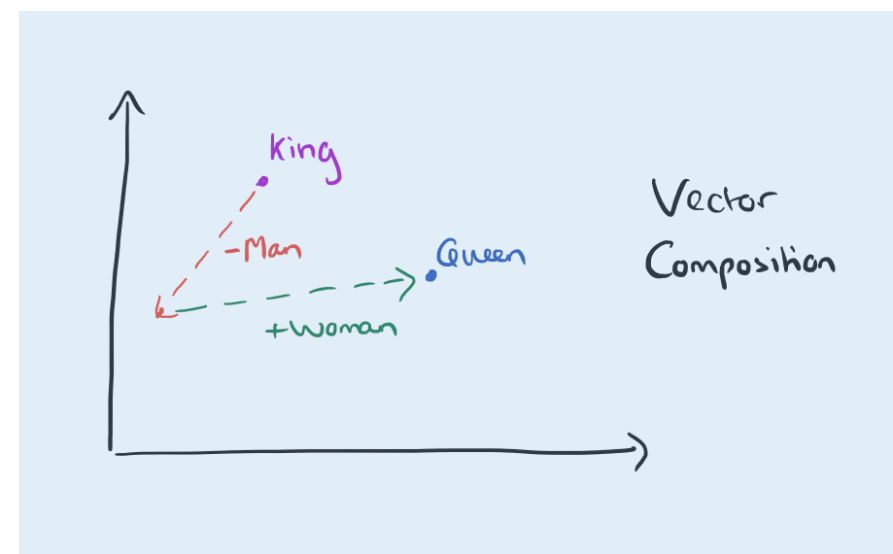
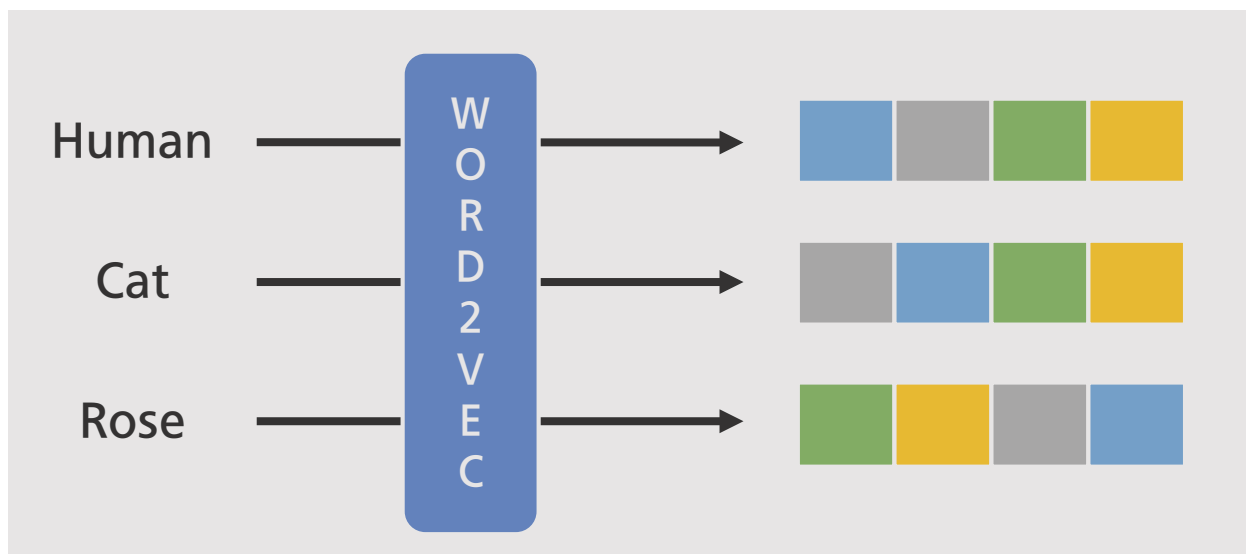
음성 인식 기술 – 기본 개념



Word2Vec

텍스트로 된 단어를 유의미한 벡터 형태로 바꿔주는 변환 알고리즘

- 단어 간의 상관성을 반영한 벡터를 형성하여 학습한 모델의 정확도를 향상시킴
- 참고 사이트: <https://word2vec.kr/>



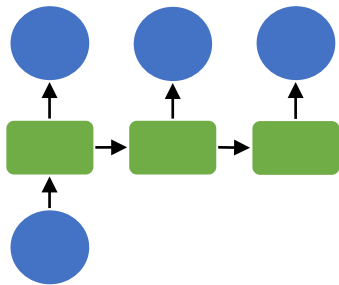
음성 인식 기술 – 기본 개념



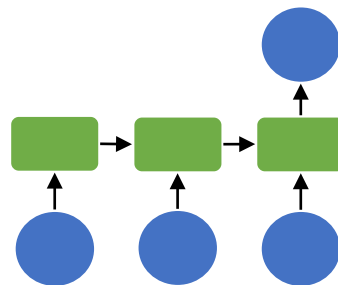
RNN

순환 신경망이라고 부르며, 은닉층 (Hidden state)의 결과값이 출력층으로도 보내고, 다시 은닉층의 계산 입력으로도 보내는 네트워크

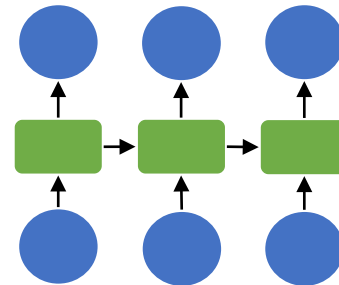
- 음성 데이터 혹은 텍스트 데이터처럼 길이가 다양하게 주어지는 시퀀스 데이터를 처리할 때 주로 사용
- 다양한 길이의 출력 구조를 가질 수 있음
- ASR이나 NLP 내의 네트워크로 주로 사용되는 네트워크



일 대 다



다 대 일



다 대 다

Google Assistant란?



Google Assistant

구글의 인공지능 비서 서비스

- 자연 음성 또는 장치를 통한 입력으로 상호작용할 수 있음
- 자동 음성 인식(ASR)과 자연어 처리(NLP) 기술을 사용해 사용자가 말한 문장을 생성

SDK (Software Development Kit)를 제공

- 핫워드 감지, 음성 제어, 자연어 이해 및 구글 스마트 기능을 장치에 추가할 수 있음
- 어시스턴트 요청과 응답의 오디오 바이트를 직접 조작하는 저수준 API까지 제공

Actions on Google 제공

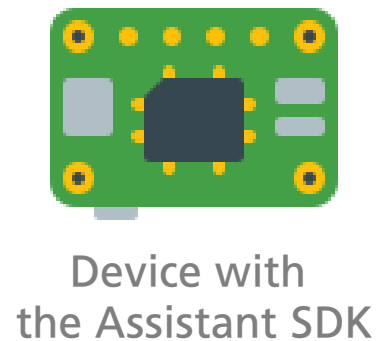
- 사용자가 Google Assistant로 장치를 제어할 수 있는 플랫폼



음성 인식을 통한 장치 제어



Actions on Google을 통한 장치 제어



Request: *"Turn on."*

Device Model: my-model
Device Instance: my-device-id



ASR
NLP
Device Matching



Request: *"Sure."*

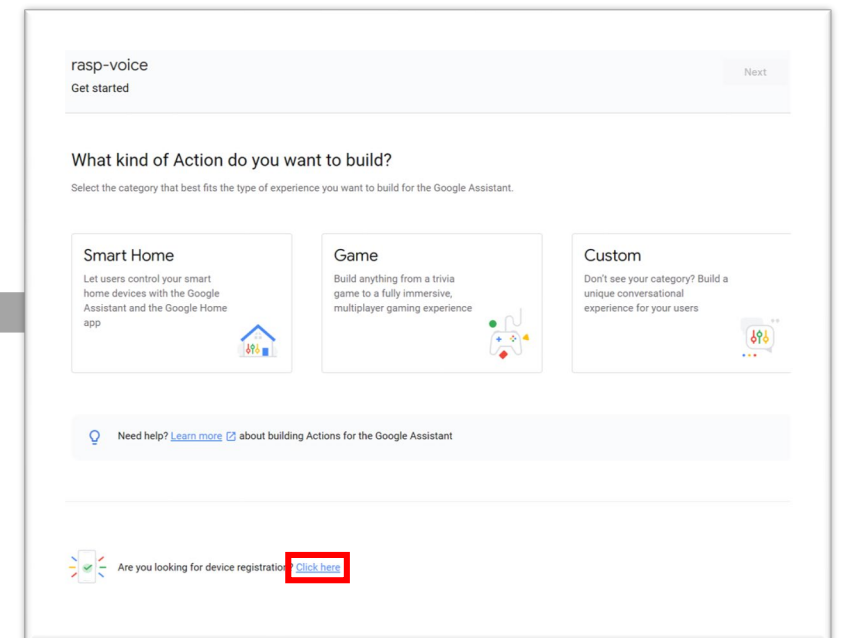
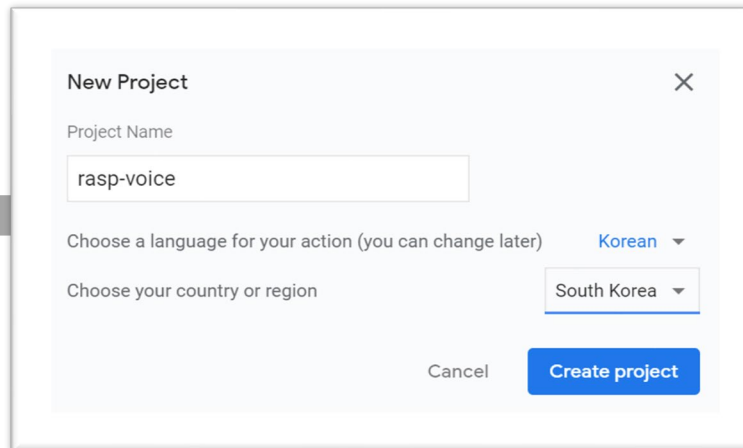
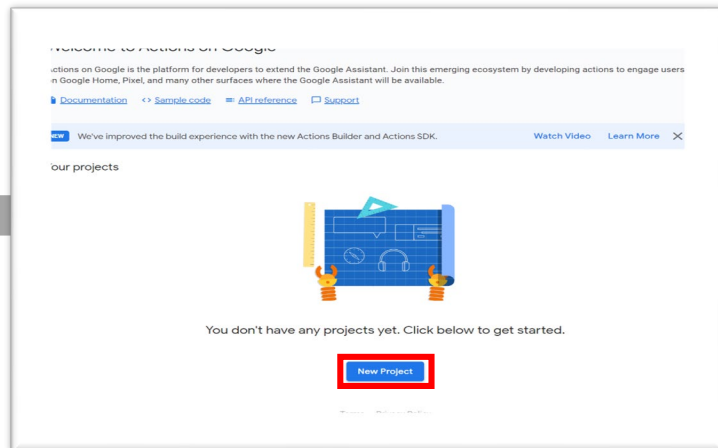
Trait: action.devices.traits.OnOff
Command: action.devices.commands.OnOff
Parameters: on (True)

Actions on Google – 프로젝트 및 계정 설정



Actions on Google 프로젝트 구성

프로젝트 생성 (<https://console.actions.google.com>)

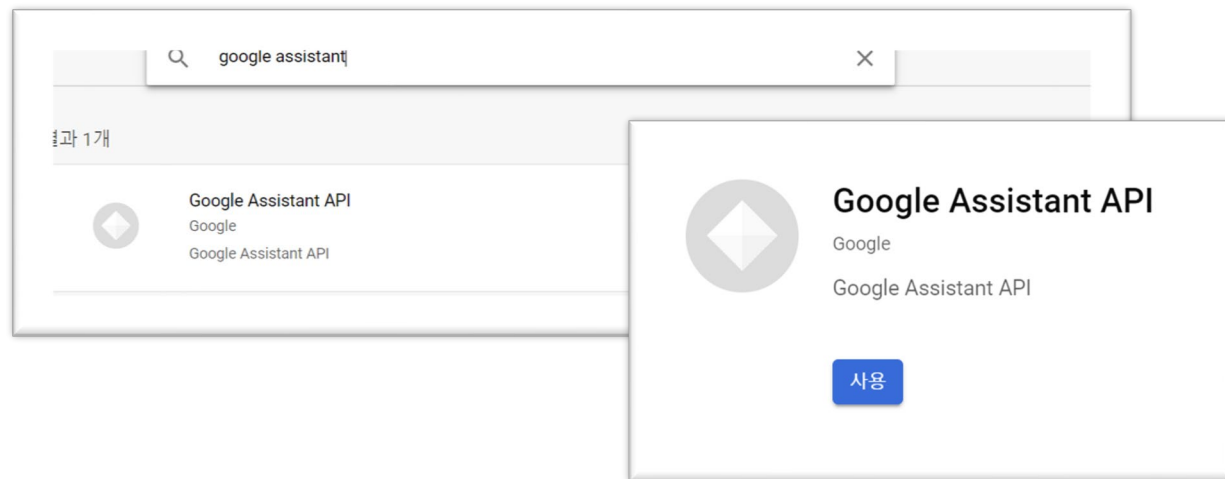
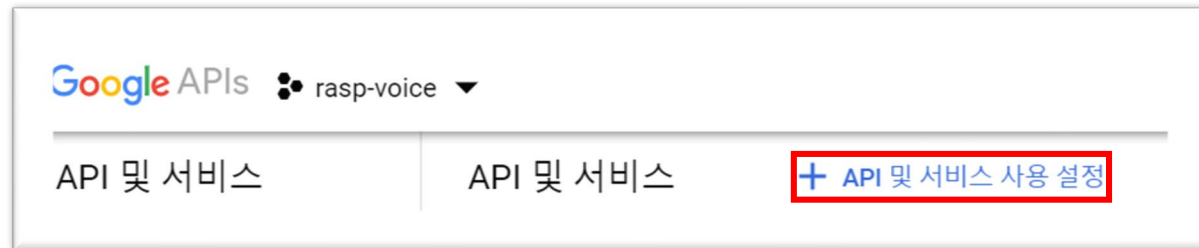
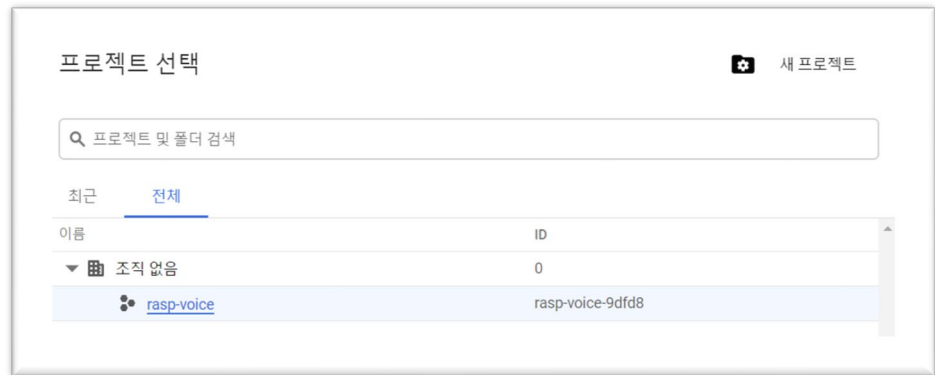
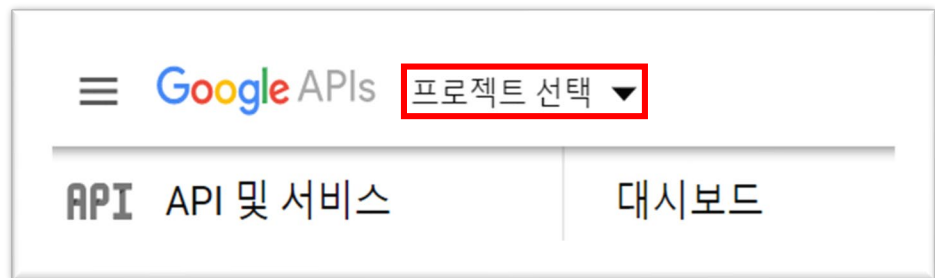


Actions on Google – 프로젝트 및 계정 설정



프로젝트 내 API 사용 설정

구글 어시스턴트 API 사용 설정 (<https://console.developers.google.com>)



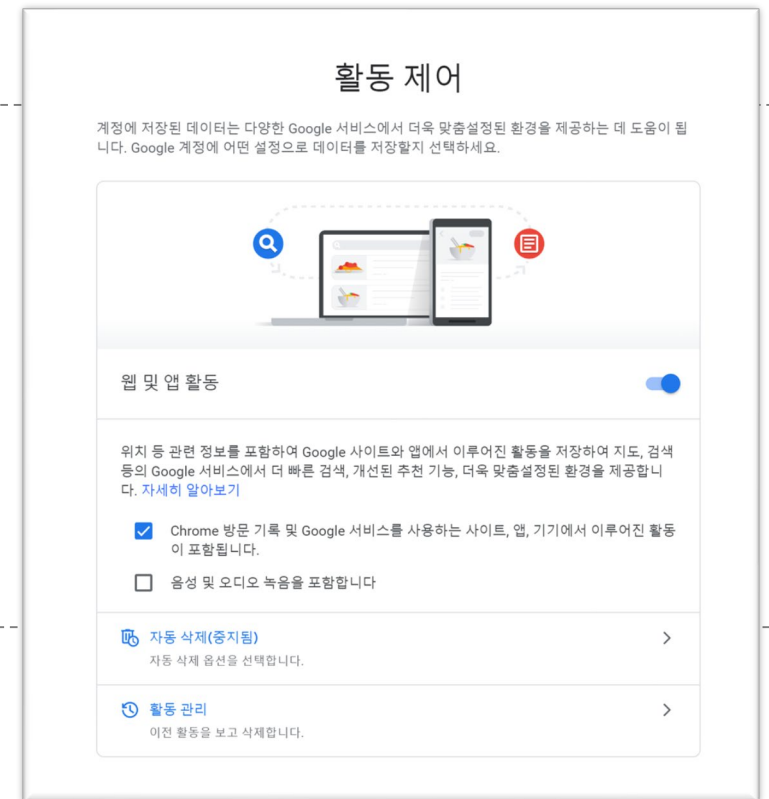
Actions on Google – 프로젝트 및 계정 설정



계정에 대한 활동 제어 설정

구글 어시스턴트 API 사용을 위해 특정 활동 데이터를 구글과 공유해야 함
(<https://myaccount.google.com/activitycontrols>)

- 웹 및 앱 활동
- 기타 데이터 설정은 이후 프로젝트 진행 중에 설정 예정
(기기 정보, 음성 및 오디오 활동)



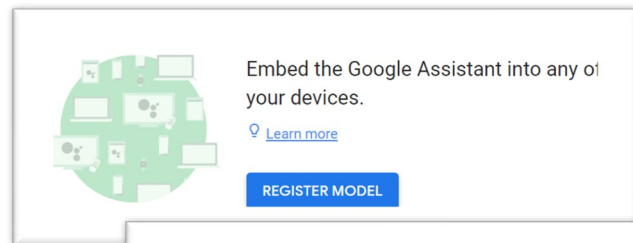
Actions on Google – 프로젝트 및 계정 설정



장치 모델 등록

구글 어시스턴트가 사용자의 장치에 접근하고 명령에 응답하려면 사용자 특정 장치에 관한 정보가 필요

- 장치 유형과 제조업체와 같은 정보 제공



Register model

1 Create model 2 Download credentials 3 Specify traits

Product name ⓘ
mCube-AI

Manufacturer name ⓘ
cndi

Device type ⓘ
Speaker

Model id ⓘ
rasp-voice-9dfd8-mcube-ai-ty63lg

Cancel REGISTER MODEL

Register model

1 Create model 2 Download credentials 3 Specify traits

★ Keep this file secure, do not upload it to public repositories like GitHub. It's possession allows client applications to make call to the Google Assistant Service and will consume your project quota (see the [OAuth 2.0](#) documentation for more information).

1. **Download OAuth 2.0 credentials** /home/pi 에 저장

2. Place the client_secret_*.json file in the folder where you're running the Assistant SDK.
To copy over SSH to a remote device, run the following command from your current computer:

```
scp ~/Downloads/client_secret_*.json <username>@<device-ip-address>:
</path/to/assistant-sdk/project>
```

password: password-for-device

Next

Register model

1 Create model 2 Download credentials 3 Specify traits

Search All traits ▾

☐ All 7 traits

☐ Brightness
This trait covers how to control the brightness of a device. Absolute brightness setting is in a normalized range from 0 to 100 (individual lights may not support every point in the range based on their LED configuration).
[View details](#)

☐ ColorSpectrum
This trait belongs to any device that is able to set a color spectrum. This applies to 'full' color bulbs that take RGB color ranges. Lights may have any combination of...

SKIP SAVE TRAITS

Actions on Google – 프로젝트 및 계정 설정



SDK 및 샘플 코드 설치

구글 어시스턴트 SDK 패키지를 다운로드하여, 샘플 코드를 포함해 장치에서 구글 어시스턴트를 실행하는 데 필요한 모든 코드를 설치함

구글 어시스턴트 실행과 자격 증명을 위해 필요한 코드를 다운로드

```
>>> sudo apt-get update
>>> sudo apt-get install portaudio19-dev libffi-dev libssl-dev
>>> python3 -m pip install --upgrade google-assistant-sdk[samples]
>>> python3 -m pip install --upgrade google-auth-oauthlib[tool]
```

Actions on Google – 프로젝트 및 계정 설정



자격 증명 생성

프로젝트에서 사용할 장치임을 증명하기 위하여 자격 증명을 생성

장치의 자격 증명을 생성

```
>>> google-oauthlib-tool --scope  
https://www.googleapis.com/auth/assistant-sdk-prototype  
--save --headless --client-secrets  
/home/pi/client_secret_client-id.json
```

(client-id는 json 파일의 이름을 참고하여 입력한다.)

Please visit this URL to authorize this application:

https://... (URL에 접속하여 구글 계정 로그인)


Enter the authorization code: 위에서 얻은 코드 입력

Actions on Google – 프로젝트 및 계정 설정



자격 증명 생성 Troubleshooting

오류가 발생할 경우, 지원 이메일을 추가하기 위한 링크로 이동


승인 오류

403 오류: restricted_client

This app is not yet configured to make OAuth requests. To do that, set up the app's OAuth consent screen in the Google Cloud Console.

[자세히 알아보기](#)

요청 세부정보

- access_type=offline
- response_type=code
- redirect_uri=urn:ietf:wg:oauth:2.0:oob
- state=cRTXVgUdo7BCStlpjrAftDoFU6nVZc
- prompt=consent
- client_id=247674462625-kgg2s0ef4kennj8la7tcv9jnbhe3pt4e.apps.googleusercontent.com
- scope=https://www.googleapis.com/auth/assistant-sdk-prototype



OAuth 동의 화면

대상 사용자를 비롯해 앱을 구성하고 등록하려는 방식을 선택하세요. 프로젝트에는 하나의 앱만 연결할 수 있습니다.

User Type

☐ 내부 ?

조직 내 사용자만 사용할 수 있습니다. 확인을 위해 앱을 제출할 필요는 없습니다.

☒ 외부 ?

Google 계정이 있는 모든 사용자가 사용할 수 있습니다.

[만들기](#)

OAuth 동의 화면

사용자는 인증 전에 이 동의 화면을 통해 비공개 데이터에 대한 액세스 권한을 부여할지 선택할 수 있으며 서비스 약관 및 개인정보처리방침 링크도 제공됩니다. 이 페이지에서 프로젝트에 속한 모든 애플리케이션의 동의 화면을 구성합니다.

확인 상태

게시되지 않음

애플리케이션 이름 ?

동의를 요청하는 앱의 이름

애플리케이션 로고 ?

사용자가 앱을 알아보는 데 도움이 되는 동의 화면의 이미지입니다.

[탐색](#)



지원 이메일 ?

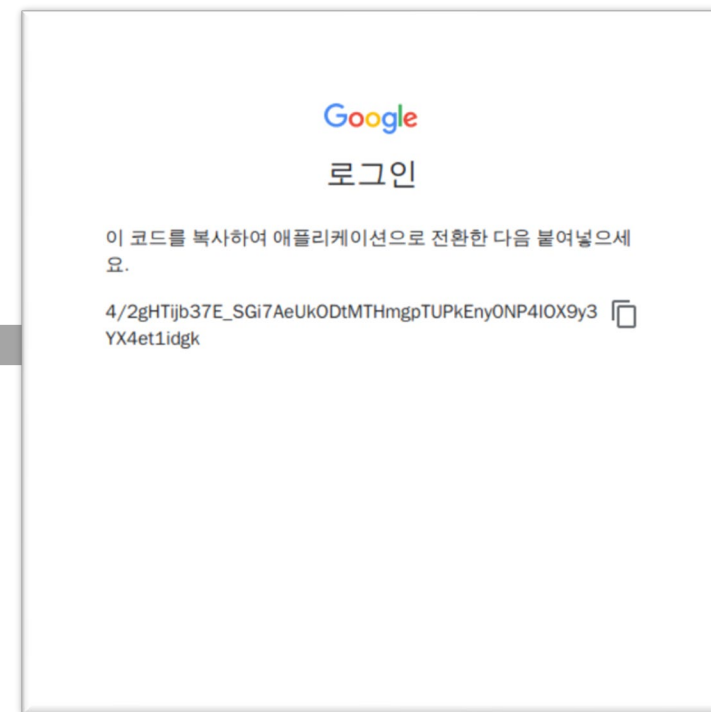
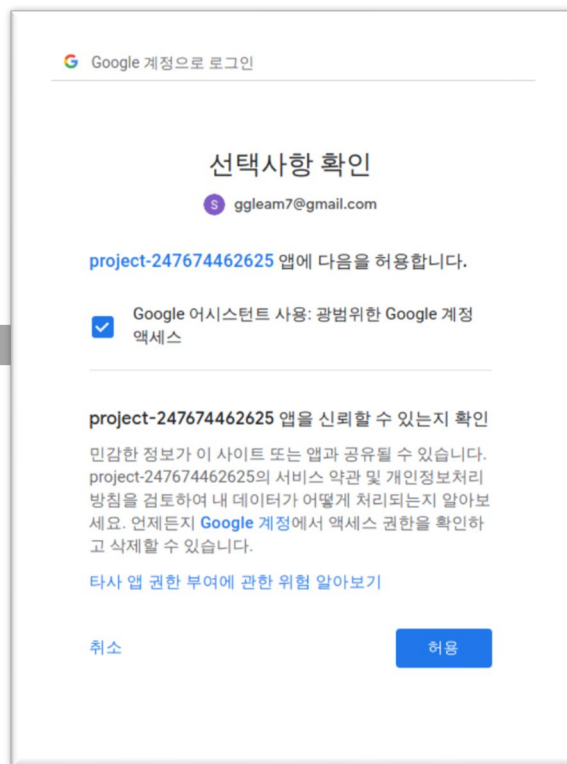
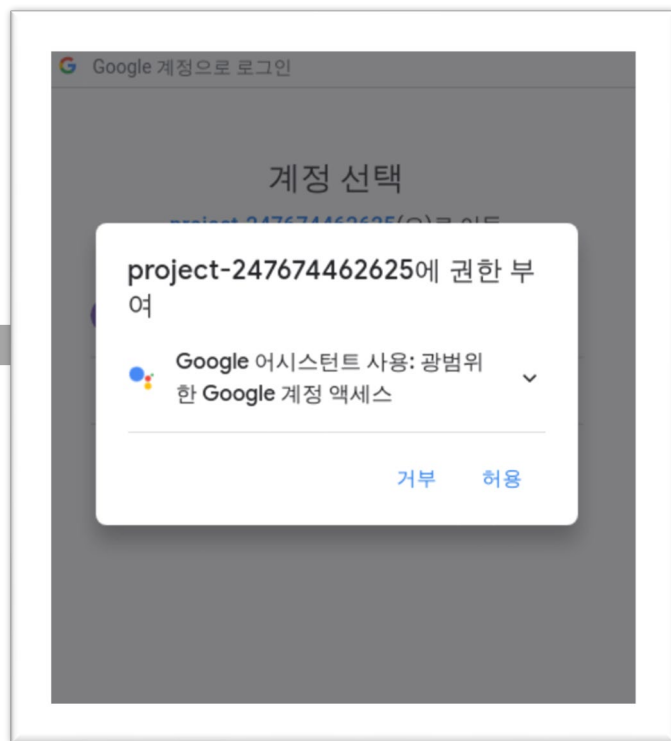
사용자 지원을 위해 동의 화면에 표시됩니다.

Actions on Google – 프로젝트 및 계정 설정



자격 증명 생성 코드

입력해야 할 코드를 생성 받기 위해 순서대로 진행



Actions on Google – 프로젝트 및 계정 설정



마이크 세팅

실습을 진행하기 전에 라즈베리 파이에서 오디오 시스템을 구성해야 함

- Aplay -l 명령어로 스피커 장치 확인
- Arecord -l 명령어로 마이크 장치 확인
- .asoundrc 파일을 생성하고 알맞게 작성

Card 1
Device 0

```
pi@raspberrypi:~$ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: b1 [bcm2835 HDMI 1], device 0: bcm2835 HDMI 1 [bcm2835 HDMI 1]
  Subdevices: 4/4
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
card 1: Headphones [bcm2835 Headphones], device 0: bcm2835 Headphones [bcm2835 Headphones]
  Subdevices: 4/4
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
pi@raspberrypi:~$ arecord -l
**** List of CAPTURE Hardware Devices ****
card 2: DeviceEEPR0M [USB PnP Audio Device(EEPROM)], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Card 2
Device 0

```
pcm.!default{
    type asym
    capture.pcm "mic"
    playback.pcm "speaker"
}
pcm.mic{
    type plug
    slave{
        pcm "hw:2,0"
    }
}
pcm.speaker{
    type plug
    slave{
        pcm "hw:1,0"
    }
}
```

Card, Device
순서

Actions on Google – 프로젝트 및 계정 설정



샘플 코드 실행

다음 쿼리 중 일부를 말하여 시도하고 어시스턴트가 정상적으로 응답하는 지 확인

- Who am I?
- What time is it now?
- What is the weather in San Fransisco?

구글 어시스턴트 샘플 코드 실행

```
>>> googlesamples-assistant-pushtotalk --project-id project-id  
--device-model-id model-id  
(프로젝트 설정과 모델 설정을 확인하여 project-id와 model-id 기입)
```

Actions on Google – 라즈베리 파이 Flask 서버 구성



Flask

Jinja2 템플릿 엔진 기반 웹 프레임워크

- 최소한의 기능만 포함하고 있어 작고 가벼움
- 구글 어시스턴트로부터 명령을 전달받기 위한 서버로 Flask를 활용



Flask 프로그램 설치

```
>>> pip3 install Flask
```

- `@app.route('/example/<ex>/')`
 - 괄호 안에 해당하는 규칙의 URL로 요청이 들어왔을 때, 등록된 함수를 실행시킬 것을 지시
 - 위의 예시에서 꺾쇠 안에 적힌 ex는 문자열 변수이고, URL 내 해당하는 자리의 문자열을 변수로 전달받아 함수 내에서 사용 가능함
- `app.run(host = '주소' ,port = '번호')`
 - Port 번호를 명시하고, 지정한 주소의 host만 접근이 가능하게 함
 - 주소를 '0.0.0.0'으로 설정할 경우, 어떤 호스트든 간에 접근 가능

Actions on Google – 라즈베리 파이 Flask 서버 구성



LED 장치 제어 Flask 서버 구성

```
import RPi.GPIO as GPIO

from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "hello world"

@app.route('/led/<onoff>')
def ledonoff(onoff):
    if onoff == "on":
        print("LED Turn on")
        GPIO.output(4,1)
        return "LED on"
    elif onoff == "off":
        print("LED Turn off")
        GPIO.output(4,0)
        return "LED off"

if __name__ == "__main__":
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(4,GPIO.OUT,initial=GPIO.LOW)
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Actions on Google – 라즈베리 파이 Flask 서버 구성

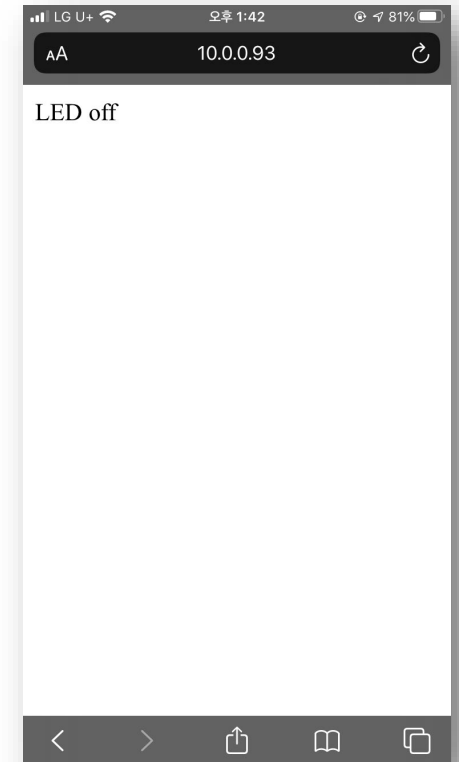


동일한 WiFi를 사용하는 디바이스를 사용해 접근

ifconfig을 통해 라즈베리 파이의 ip 주소를 얻고, Flask 서버의 포트에 맞춰 URL을 입력

- 내부 ip이므로, 동일한 WiFi를 사용하는 디바이스를 통해서만 접근이 가능
- 외부에서도 접근이 가능하게 하기 위해서는 공인 ip를 부여해야 함

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.93 netmask 255.255.255.0 broadcast 10.0.0.255
    inet6 fe80::3a31:8eda:b75f:26fb prefixlen 64 scopeid 0x20<link>
    ether dc:a6:32:b0:f5:45 txqueuelen 1000 (Ethernet)
    RX packets 77966 bytes 110850733 (105.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 31146 bytes 2813860 (2.6 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```



Actions on Google – ngrok를 사용한 외부 망 접속 구성



ngrok

로컬 서버를 안전하게 공개 인터넷으로 연결 시켜주는 프로그램

- Public address를 제공하여, 외부 망에서도 서버에 연결이 가능하게 해줌
- 라즈베리 파이를 외부 서버처럼 사용 가능

The ngrok logo, consisting of the word "ngrok" in a bold, dark blue, sans-serif font.

ngrok 설치 및 실행

```
>>> wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-arm.zip
```

(또는 <https://ngrok.com/download>에 접속하여 download 후 zip 파일 이동)

```
>>> unzip ./ngrok-stable-linux-arm.zip
```

Actions on Google – IFTTT를 이용한 음성 인식 서비스 활용



IFTTT

IF This Then That의 약자로, 다양한 장치와 서비스를 연결하는 자동화 서비스

- 특정 이벤트에 대한 응답으로 다른 서비스를 제공하는 형태
- 구글 어시스턴트의 음성인식을 Trigger로 사용
- 명령에 알맞게 Flask 서버에 접근하여 라즈베리 파이에 동작을 요청하는 시스템 구축 가능

IFTTT

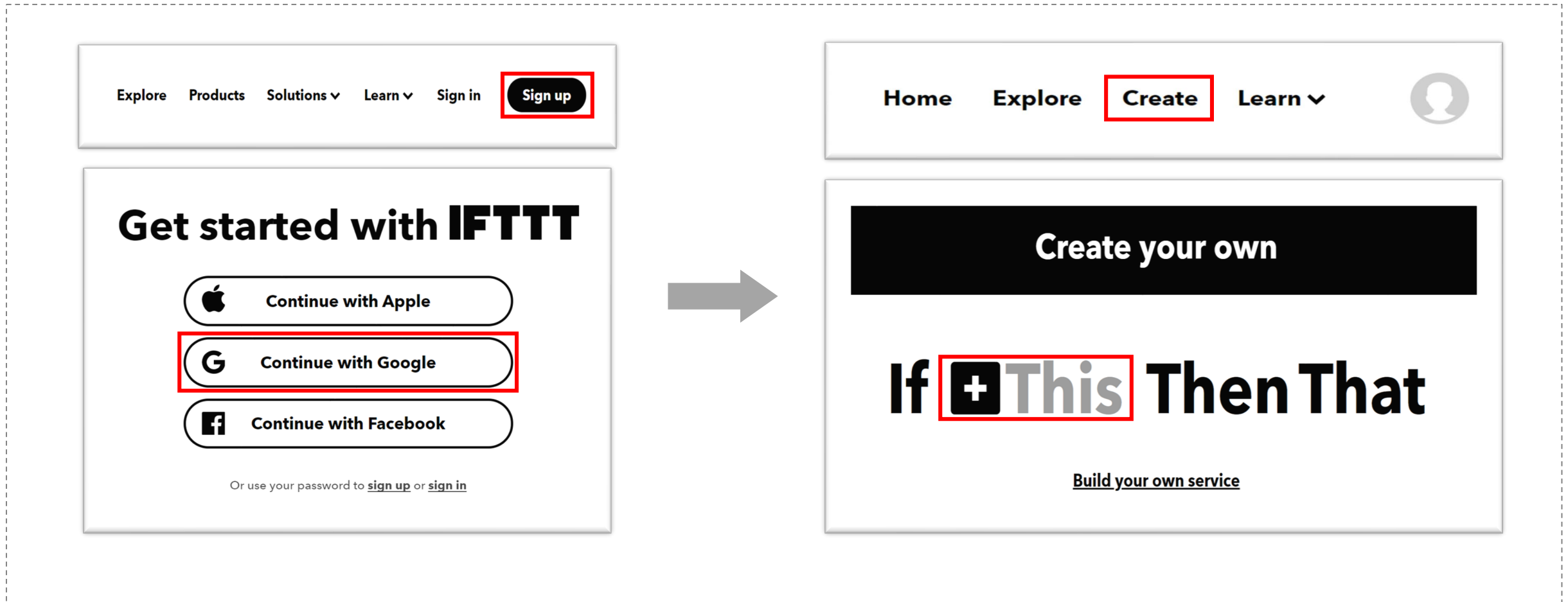
If  **This** **Then**  **That**

구글 어시스턴트 Webhooks

Actions on Google – IFTTT를 이용한 음성 인식 서비스 활용

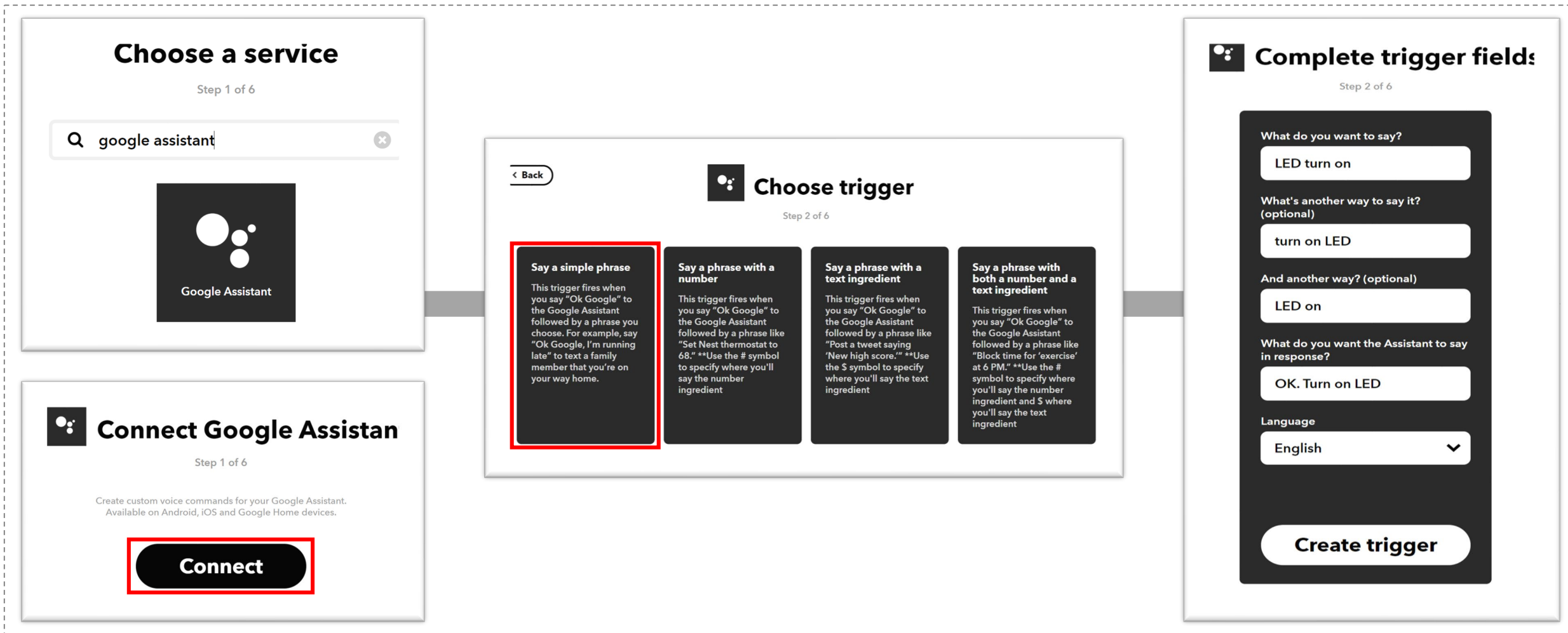


IFTTT 서비스 사용을 위한 계정 생성



Actions on Google – IFTTT를 이용한 음성 인식 서비스 활용

IFTTT 서비스 생성 (Google Assistant)



Actions on Google – IFTTT를 이용한 음성 인식 서비스 활용

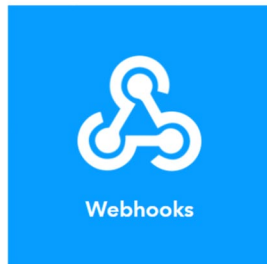
IFTTT 서비스 생성 (Webhooks)

If  Then  That

Choose action service

Step 3 of 6

webhooks



Connect Webhooks

Step 3 of 6

Integrate other services on IFTTT with your DIY projects. You can create Applets that work with any device or app that can make or receive a web request. If you'd like to build your own service and Applets, check out the IFTTT platform.

Connect



Choose action

Step 4 of 6

Make a web request

This action will make a web request to a publicly accessible URL. NOTE: Requests may be rate limited.



Complete action field:

Step 5 of 6

URL

<http://7153bd848111.ngrok.io/led/on>

Surround any text with "<<>>" to escape the content

Add ingredient

Method

GET

The method of the request e.g. GET, POST, DELETE

Content Type

Please select

Optional

Body

Surround any text with "<<>>" to escape the content

Add ingredient

Create action

Actions on Google – 프로젝트 완성 및 점검



동작 확인

프로그램 실행

```
>>> python3 iot.py  
>>> ./ngrok http 5000  
>>> googlesamples-assistant-pushtotalk
```




Exercise I – LED

LED 발광 모드 변경

- 1) LED가 모두 꺼진 초기(initial) 상태
- 2) "Turn on LED"/ "Turn off LED" 명령에 맞춰 전체 LED ON/OFF
- 3) LED가 모두 켜진 상황에서, "Party mode"를 명령하면 0.5초 간격으로 전체 LED 점멸

원하는 번호의 LED 제어

- 1) 왼쪽 위 LED부터 시계 방향으로 번호를 매김
- 2) “Turn on(off) LED number X” 명령에 맞춰 해당 번호 LED 제어
- 3) X는 1부터 4까지의 번호
- 4) Hint: IFTTT에 등록 시, X에 대한 숫자와 영어 모두 대응되도록 입력 (e.g., Turn on LED number 1 // Turn on LED number one)

Example – FAN



FAN 장치 제어

앞서 작성한 `iot.py`에 이어서 코드를 작성

```
@app.route('/fan/<onoff>')
def fanonoff(onoff):
    if onoff == "on":
        print("FAN Turn on")
        GPIO.output(18,1)
        GPIO.output(27,0)
        return "FAN on"
    elif onoff == "off":
        print("FAN Turn off")
        GPIO.output(18,0)
        GPIO.output(27,0)
        return "FAN off"

if __name__ == "__main__":
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(4,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(18,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(27,GPIO.OUT,initial=GPIO.LOW)
    app.run(host='0.0.0.0', port=5000, debug=True)
```



Exercise II – FAN

FAN 타이머 작동

- 1) "fan on for X second(s)" 명령에 대해 X초 동안 작동하도록 구현
- 2) X는 {1, 2, 3} 중 하나

OpenCV를 이용한 시각 인식



OpenCV

컴퓨터 비전과 기계 학습 라이브러리를 제공하는 오픈 소스

- 상용 제품에서 기계 인식 사용을 가속화 하기 위해 개발
- BSD 라이선스 제품으로 기업에서 코드를 쉽게 사용하고 수정할 수 있음
- C++, 파이썬, 자바 및 MATLAB 인터페이스를 제공
- Windows, 리눅스, 안드로이드, Mac OS 등 다양한 OS 지원



OpenCV를 이용한 시각 인식



OpenCV 설치

```
>>> sudo apt install -y libhdf5-dev libatlas-base-dev libjasper-dev libqtgui4  
libqt4-test  
>>> git clone https://github.com/opencv/opencv.git  
>>> pip3 install opencv-python  
>>> pip3 install opencv-contrib-python==4.1.0.25
```

OpenCV를 이용한 시각 인식 – 얼굴 감지



Haar Cascade를 활용한 얼굴 감지

Haar이라는 특징을 기반으로 cascade 분류기를 사용하여 얼굴 및 눈을 탐지함

- "Rapid Object Detection using a Boosted Cascade of Simple Features" 논문에서 제안된 객체 탐지 방법
- Opencv 내의 `cv::CascadeClassifier` 클래스를 통해 사용 가능
- `Cv::CascadeClassifier::load`
 - Xml 확장자로 저장해놓은 분류기를 불러오는 함수
- `cv::CascadeClassifier::DetectMultiScale`
 - 객체 탐지를 수행하는 함수

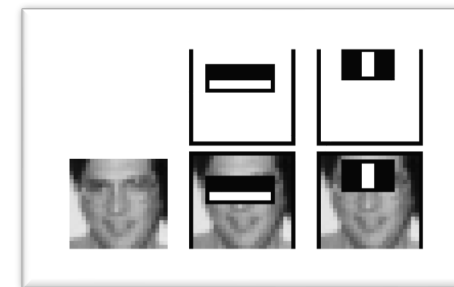
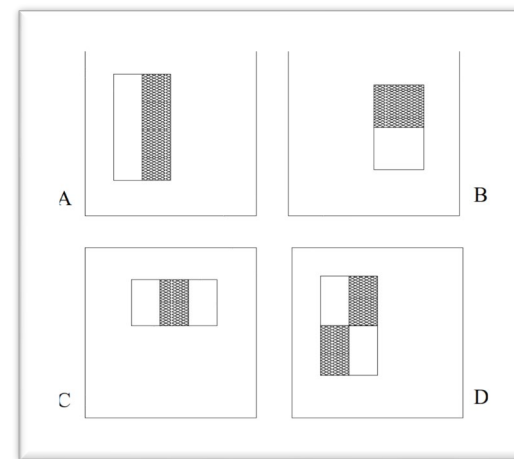
OpenCV를 이용한 시각 인식 – 얼굴 감지



Haar 특징

두 개의 직사각형으로 구성된 영역 내의 픽셀의 합과 차를 이용해 이미지 특정 부분에서 뽑아내는 특징

- 이미지 전체를 스캔하면서 각 구간으로부터 특징을 뽑아냄
- 빛금친 영역의 픽셀 합계에서 흰색 영역의 픽셀 합계를 뺀 단일 값 계산
- 각기 다르게 구성된 영역들을 통해 다양한 특징 추출
- 부분 이미지에서 가장 뚜렷하게 나타나는 특징을 찾아야 함
- 사람의 얼굴에서는 대개 눈 근처에서 의미있는 값을 보임



Cascade 분류기

각 영역에 대해 이미지 전체를 스캔하는 것은 매우 비효율적이므로, 약한 수준의 분류기로 후보를 줄이고 점차 분류 수준을 높게 적용하여 단계적으로 분류를 해내는 기술

OpenCV를 이용한 시각 인식 – 얼굴 감지



예시 코드 구성

#BGR 컬러를 GRAY 컬러로 변경하고, 밝기 값을 평균화하여 영상을 선명하게 만듦

```
frame_gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
```

```
frame_gray = cv.equalizeHist(frame_gray)
```

#HAAR 특징에 기반해 얼굴을 검출하고, 얼굴의 형태를 타원으로 표현

```
faces = face_cascade.detectMultiScale(frame_gray)
```

```
for (x,y,w,h) in faces:
```

```
    center = (x + w//2, y + h//2)
```

```
    frame = cv.ellipse(frame, center, (w//2, h//2), 0, 0, 360, (255, 0, 255), 4)
```

OpenCV를 이용한 시각 인식 – 얼굴 감지



예시 코드 구성 및 실행

```
#HAAR 특징에 기반해 눈을 검출하고, 눈의 형태를 원으로 표현
eyes = eyes_cascade.detectMultiScale(faceROI)
for (x2,y2,w2,h2) in eyes:
    eye_center = (x + x2 + w2//2, y + y2 + h2//2)
    radius = int(round((w2 + h2)*0.25))
    frame = cv.circle(frame, eye_center, radius, (255, 0, 0 ), 4)
```

```
parser = argparse.ArgumentParser(description='Code for Cascade Classifier tutorial.')
parser.add_argument('--face_cascade', help='Path to face cascade.', default='/home/pi/opencv/data/haarcascades/haarcascade_frontalface_alt.xml')
parser.add_argument('--eyes_cascade', help='Path to eyes cascade.', default='/home/pi/opencv/data/haarcascades/haarcascade_eye_tree_eyeglasses.xml')
parser.add_argument('--camera', help='Camera divide number.', type=int, default=0)
args = parser.parse_args()
```



경로를 맞춰 변경

프로그램 실행

```
>>> cd opencv/samples/python/tutorial_code/objectDetection/cascade_classifier
>>> python3 objectDetection.py
```

OpenCV를 이용한 시각 인식 – 얼굴 인지



Openface를 활용한 얼굴 인지

감지된 얼굴에서 드러나는 차이점을 학습하여 어떤 사람의 얼굴인지 인지. 두 단계로 진행됨

- Face Recognition: 영상으로부터 사람의 얼굴 부분을 검출
- Face Identification: 검출된 얼굴을 기존 학습된 데이터를 통해 누구인지 파악

기본 코드 Cloning

```
>>> git clone https://github.com/dozen12/OpenCV-Python-Series
```

OpenCV를 이용한 시각 인식 – 얼굴 인지 예제 코드



얼굴 감지 분류기 생성

```
face_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_alt2.xml')
eye_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_eye.xml')
smile_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_smile.xml')
```

얼굴 인지 모델 생성

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("./recognizers/face-trainer.yml")
```

캡처 및 검출

```
# 프레임마다 얼굴을 검출해냄
ret, frame = cap.read()
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5)
```

OpenCV를 이용한 시각 인식 – 얼굴 인지 예제 코드



얼굴 인지 및 정보 저장

```
# 딥러닝을 통해 구현된 recognizer를 통해 누구의 얼굴인지 추론
id_, conf = recognizer.predict(roi_gray)
if conf >= 4 and conf <= 85:
    #print(5: #id_)
    #print(labels[id_])
    font = cv2.FONT_HERSHEY_SIMPLEX
    name = labels[id_]
    color = (255, 255, 255)
    stroke = 2
    cv2.putText(frame, name, (x,y), font, 1, color, stroke, cv2.LINE_AA)
```

예제 코드 실행

모델 학습 및 프로그램 실행

```
>>> cd opencv/OpenCV-Python-Series/src/ (OpenCV-Python-Series 설치 경로에 맞춰 설정)
>>> python3 faces-train.py
>>> python3 faces.py
```

OpenCV를 이용한 얼굴 인지와 장치 제어



얼굴 인식에 따른 LED 제어

이전에 실행시켰던 faces.py에 코드를 추가하여 LED 제어 기능을 부여

```
import numpy as np
import cv2
import pickle

import RPi.GPIO as GPIO
from time import sleep
import threading

detect_state = False
led_status = 0
```

추가 코드

OpenCV를 이용한 얼굴 인지와 장치 제어



```
def controlDevice(detect_state):  
    while detect_state:  
        if(led_status == 1):  
            GPIO.output(4,1)  
        else:  
            GPIO.output(4,0)
```

```
GPIO.setmode(GPIO.BCM)  
GPIO.setwarnings(False)  
GPIO.setup(4,GPIO.OUT,initial=GPIO.LOW)
```

```
global t  
detect_state = True  
t = threading.Thread(target=controlDevice, args=(detect_state,))  
t.daemon = True  
t.start()
```

추가 코드

OpenCV를 이용한 얼굴 인지와 장치 제어



```
face_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_alt2.xml')
eye_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_eye.xml')
smile_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_smile.xml')

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("./recognizers/face-trainer.yml")

labels = {"person_name": 1}
with open("pickles/face-labels.pickle", 'rb') as f:
    og_labels = pickle.load(f)
    labels = {v:k for k,v in og_labels.items()}

cap = cv2.VideoCapture(0)
```

OpenCV를 이용한 얼굴 인지와 장치 제어



```
while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5)
    for (x, y, w, h) in faces:
        #print(x,y,w,h)
        roi_gray = gray[y:y+h, x:x+w] #(ycord_start, ycord_end)
        roi_color = frame[y:y+h, x:x+w]
        # recognize? deep learned model predict keras tensorflow pytorch scikit learn
        id_, conf = recognizer.predict(roi_gray)
        if conf >= 4 and conf <= 85:
            #print(5: #id_)
            #print(labels[id_])
            font = cv2.FONT_HERSHEY_SIMPLEX
            name = labels[id_]
            color = (255, 255, 255)
            stroke = 2
            cv2.putText(frame, name, (x,y), font, 1, color, stroke, cv2.LINE_AA)

            if name == "obama":
                led_status = 1
            else:
                led_status = 0
```

추가 코드

OpenCV를 이용한 얼굴 인지와 장치 제어



```
img_item = "7.png"
cv2.imwrite(img_item, roi_color)

color = (255, 0, 0) #BGR 0-255
stroke = 2
end_cord_x = x + w
end_cord_y = y + h
cv2.rectangle(frame, (x, y), (end_cord_x, end_cord_y), color, stroke)
```

```
GPIO.setup(4,GPIO.OUT,initial=GPIO.LOW)
```

추가 코드

```
# Display the resulting frame
cv2.imshow('frame',frame)
if cv2.waitKey(20) & 0xFF == ord('q'):
    break
```

```
# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```



Exercise III – 시각 IoT

얼굴 인식을 통한 도어락 이중 잠금

- 1) Text LCD를 통해 입력 받은 비밀번호 출력 예제에 얼굴 인식 기능 추가
- 2) 힐러리의 얼굴이 인지된 경우, 비밀번호 일치 여부와 관계 없이 ACCESS DENIED 출력
- 3) 오바마의 얼굴이 인지된 경우, 이전에 구현한 예제와 같이 작동 (일치할 경우, CORRECT / 불일치할 경우, FAILED)



Exercise IV – 시각+음성 IoT

얼굴 인식을 통한 선택적 음성 인식 및 장치 제어

- 1) 오바마의 얼굴이 인지된 경우에만 음성 인식이 가능하도록 구성
- 2) 다른 얼굴이 인지된 경우, 음성이 인식되더라도 명령 수행 X
- 3) Hint: flask 서버 실행 인자로 debug=False를 보낸다.
- 4) Hint: 얼굴 인지 부분을 함수로 바꾼 후 thread로 실행시키고, flask 서버는 main에서 바로 실행한다.