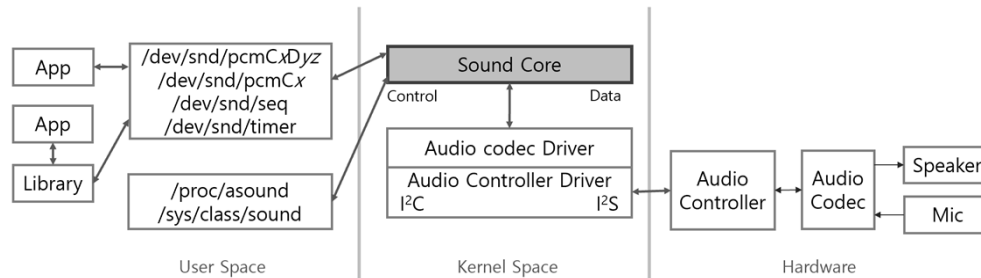


AIoT AutoCar Prime 으로 배우는 온디바이스 AI 프로그래밍

5 오디오 활용

오디오 활용 하위 시스템

- 오디오 하위 시스템은 다양한 사운드 카드를 추상화
 - ▣ 사용자가 일관된 방법으로 접근할 수 있도록 표준화된 사운드 인터페이스 제공



오디오 활용 하위 시스템

- 오디오 컨트롤러

- ▣ 마이크의 아날로그 입력을 디지털로 변환해 입력 스트림에 전달
- ▣ 디지털 출력 스트림을 아날로그로 변환해 스피커에 전달

- 코덱

- ▣ 입출력 스트림에 대해 압축/해제와 같은 인코딩/디코딩을 수행

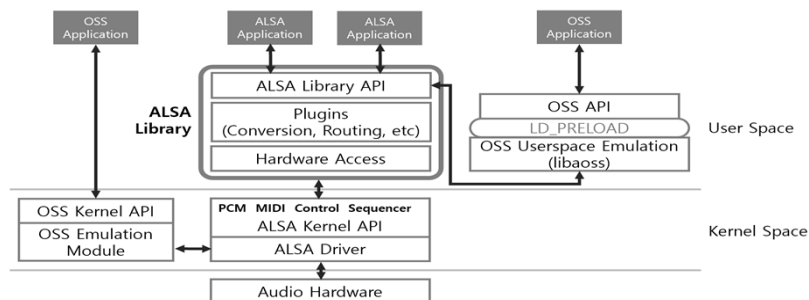
오디오 활용 하위 시스템

- 오디오 하위 시스템
- 오디오 코덱 드라이버, 오디오 컨트롤러 드라이버로 사운드 카드 제어
- 초기에는 OSS 사용
 - ▣ OSS : 유닉스 계열 운영체제에서 사운드 생성 및 캡처 인터페이스로 사용
- 현재는 새롭게 설계한 ALSA 사용

오디오 활용 하위 시스템

□ ALSA

- 완전히 모듈화 된 사운드 드라이버
- 하드웨어 기반 미디 합성 및 여러 개의 채널에 대한 하드웨어 믹싱 지원
- 멀티프로세서 및 스레드에 안전하게 설계
- OSS 바이너리 호환 인터페이스 제공



오디오 활용 하위 시스템

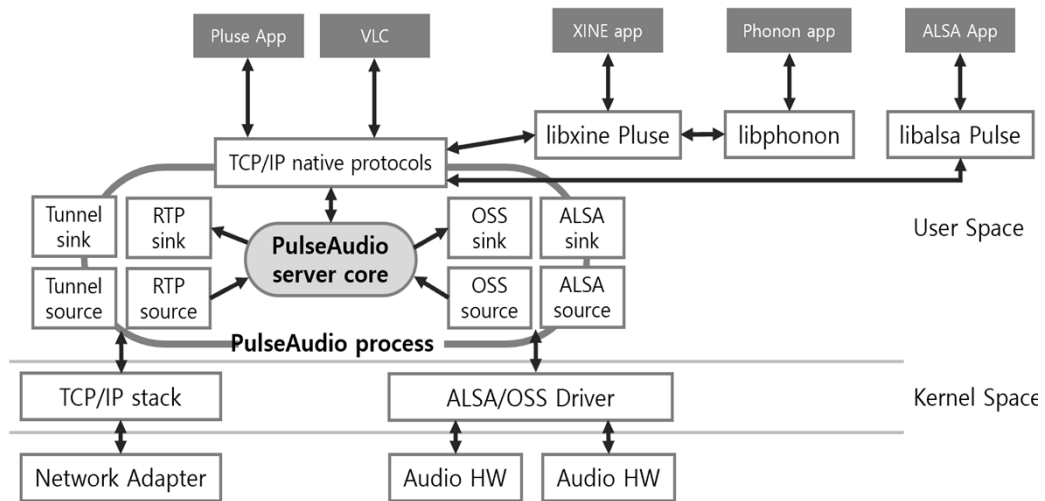
□ ALSA의 주요 기능

- Control: 사운드 카드 제어를 위한 레지스터 관리
- PCM: 디지털 오디오 캡처와 출력을 위한 인터페이스.
- Raw MIDI: 전자 음악 장비의 표준인 MIDI 지원
- Time: 사운드 이벤트의 동기화에 사용되는 사운드 카드의 하드웨어 타이밍 접근
- Sequencer: MIDI를 위한 고수준 인터페이스
 - 보다 많은 미디 프로토콜과 타이밍 관리
- Mixer: 최 상단에서 사운드 볼륨 제어. 입력 시그널의 선택과 사운드 카드 제어

오디오 라이브러리

- 오디오 라이브러리(ex. alsa-lib)
 - ▣ 커널의 ALSA 오디오 하위 시스템을 응용프로그램에서 접근 위해 추상화한 것
 - ▣ 한 응용프로그램의 오디오 출력을 다른 응용 프로그램에 전달
 - ▣ 실시간 오디오 및 음악 프로그램을 작성하고 기기 간 공유 지원
 - ▣ alsa-lib에서 제공하는 기능은 너무 많고 복잡
 - ▣ 쉬운 사용위해 JACK, PulseAudio, PortAudio 같은 고수준 라이브러리들 등장
 - 리눅스뿐만 아니라 윈도우나 맥과 같은 다른 플랫폼도 함께 지원

오디오 라이브러리



셸 환경에서 오디오 재생 및 녹음

□ AutoCAR는 셸 환경에서 오디오를 녹음하고 재생하는 명령 제공

□ speaker-test: 오디오 출력 테스트

■ speaker-test -c2 -D hw:1,0 -t wav

- 사운드 카드 1번, 장치 0번에서 2채널 테스트용 wav 재생

■ speaker-test -c2 -t wav:

- 기본 장치에서 2채널 테스트용 wav 재생

□ arecord: raw 또는 wav 포맷으로 오디오 녹음

■ arecord -c 2 -D hw:1,0 --format=S16_LE --duration=5 --rate=16000 --file-type=raw out.raw

- out.raw 파일에 5초간 16bit 16000 샘플링 레이트, stereo, raw 포맷 녹음

셸 환경에서 오디오 재생 및 녹음

▣ aplay: raw 또는 wav 포맷 재생

- `aplay -D hw:1,0 --format=S16_LE --rate=16000 out.raw`
 - 16bit 16000 샘플링 레이트, raw 포맷으로 녹음된 out.raw 재생

▣ sox: 다양한 포맷으로 오디오 녹음, 재생, 효과 지원

- `sox -t alsa plughw:1 -d`
 - 사운드 카드 1번의 마이크 입력을 기본 오디오 출력(스피커)으로 전달
- `rec`: 녹음용 심볼릭 링크
 - `rec hello.wav`: 오디오 입력을 wav 포맷으로 hello.wav에 녹음
- `play`: 재생용 심볼릭마이크
 - `play hello.wav`: wav 포맷의 hello.wav 재생
 - `play example.mp3`

셸 환경에서 오디오 재생 및 녹음

- WAV 파일에서 MP3 또는 OGG 파일을 만드는 명령
 - ▣ `lame hello.wav`: WAV에서 MP3 생성
 - ▣ `lame --decode my.mp3 my.wav`: MP3를 WAV 생성
 - ▣ `oggenc hello.wav`: WAV에서 OGG 생성
- `lame`는 반대로 MP3를 WAV로도 변환 가능

셸 환경에서 오디오 재생 및 녹음

□ SOX

- ▣ aplay, arecord 기능 외에도 다양한 형식의 오디오 파일의 형식 변환 가능
- ▣ 리눅스를 비롯해 윈도우, Mac OS X 등 다양한 플랫폼 지원
- ▣ lame 또는 oggenc 이 설치되어 있다면 MP3, OGG 녹음 가능

셸 환경에서 오디오 재생 및 녹음

□ sox 주요 기능

▣ 현재 마이크 입력을 .mp3로 변환하면서 녹음

- `rec -t wav - | lame - hello2.mp3`

▣ 현재 마이크 입력을 .ogg로 변환하면서 녹음

- `rec -t wav - | oggenc - -o hello2.ogg`

▣ 스테레오(2개 채널)로 일정 시간 녹음

- `rec -c 2 demo.wav trim 0 00:10`

셸 환경에서 오디오 재생 및 녹음

- ▣ 여러 효과를 적용한 후 16bit 깊이로 저장
 - `sox demo.wav -b 16 demo2.wav channels 1 rate 16k fade 3 norm`
 - 하나의 채널로 다운 믹스, 샘플 속도 변경, 페이드 인, 노말라이즈
- ▣ 속도 변환
 - `sox demo2.wav demo3.wav speed 1.527`
- ▣ 두 개의 오디오 파일 결합
 - `sox demo2.wav demo3.wav demo4.wav`

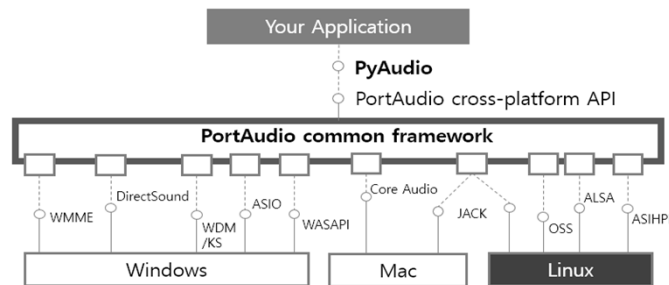
셸 환경에서 오디오 재생 및 녹음

- ▣ 두 개의 오디오 파일을 함께 믹싱
 - `sox -m demo2.wav demo3.wav demo5.flac`
- ▣ 저음 부스트 효과를 적용하면서 출력
 - `sox demo5.flac -d bass +20`
- ▣ 파이프 오르간 사운드로 합성된 '마이너 7th' 코드 연주
 - `play -n -c1 synth sin %-12 sin %-9 sin %-5 sin %-2 fade h 0.1 1 0.1`

PyAudio

□ PyAudio

- ▣ 고수준 오디오 라이브러리 중 하나인 PortAudio에 대한 파이썬 버전
- ▣ 리눅스를 비롯해 윈도우, Mac 등을 지원
- ▣ 파이썬으로 WAVE 파일을 재생, 녹음 프로그램 작성 가능



PyAudio

- PyAudio는 블로킹과 논블로킹 모드를 모두 지원
 - ▣ 블로킹 모드 : `read()`, `write()`로 녹음, 재생 수행
 - 커널이 작업을 완료하고 결과를 반환할 때까지 응용프로그램이 대기하는 것
 - ▣ 논블로킹 모드 : `start_stream()`으로 녹음, 재생 수행하는 내부 스레드 시작
 - 응용프로그램이 중단없이 자신의 작업을 이어감
 - ▣ `stop_stream()` 메소드 : 일시 정지
 - ▣ `close()` : 스트림 종료
 - ▣ `terminate()` : PyAudio 작업 종료

주요 메소드와 클래스

- Pyudio는 무손실 무압축 포맷인 WAV파일만 지원
 - ▣ MP3와 같은 무손실 압축 포맷을 지원 않음
 - ▣ 오디오 작업을 수행할 때는 pyaudio ,wave 모듈의 메소드를 함께 사용
- 오디오 작업 수행 시 사용하는 메소드
 - ▣ pyaudio 모듈에 포함된 메소드
 - ▣ wave 모듈의 open() 메소드와 Wave_read 및 Wave_write 객체의 메소드
 - ▣ 자세한 내용은 교재 참고

WAV 파일 재생

- wave 모듈의 open() 메소드로 WAV 파일을 읽기 전용으로 열 때
 - ▣ wave_read 객체 반환
- wave_read 객체의 readframes() 메소드
 - ▣ 프레임 수만큼 데이터를 읽으면서 파일 포인터를 이동
 - ▣ getnframes()로 전체 개수를 파악한 후 한 번에 전체를 읽을 수도 있음
- 예제들을 실행하려면 wav 파일 필요
 - ▣ '/usr/share/sounds/alsa/' 경로의 테스트용 WAV 파일 사용 가능

블로킹 모드 재생

□ 블로킹 모드 재생 예제

- WAV 파일 데이터를 Output Stream 객체의 write() 메소드에 전달 : 스피커 출력
- Output Stream 객체
 - PyAudio 객체의 open() 메소드 인자 중 output에 True를 전달해 만듦
- open() 메소드에는 샘플링 형식이나 채널 수, 샘플링 비율 등도 함께 전달

블로킹 모드 재생

```
01: import pyaudio
02: import wave
03:
04: w = wave.open("/usr/share/sounds/alsa/Side_Left.wav", "rb")
05: data = w.readframes(w.getnframes())
06: w.close()
07:
08: p = pyaudio.PyAudio()
09: stream = p.open(format=p.get_format_from_width(2),
10:                 channels=1,
11:                 rate=48000,
12:                 output=True)
13:
14: stream.write(data)
15:
16: stream.stop_stream()
17: stream.close()
18: p.terminate()
```

블로킹 모드 재생

- 프로그램을 실행하면 “Side Left” 가 스피커로 출력
 - ▣ 커널 버전에 따라 표준 출력에 “alsa... failed...” 오류가 표시
 - 소리 출력에 문제가 없다면 무시

```
Expression 'alsa_snd_pcm_hw_params_set_period_size_near( pcm, hwParams,
&alsaPeriodFrames, &dir )' failed in 'src/hostapi/alsa/pa_linux_alsa.c', line:
924
...
```

블로킹 모드 재생

- PyAudio 객체의 open() 메소드로 출력 스트림 객체를 만들 때
 - ▣ 포맷 폭과, 채널, 샘플링 비율은 WAV 파일을 만들 때 설정된 값과 같아야 함
 - ▣ 상수를 사용하는 것보다 WAVE_Read 객체의 해당 메소드 사용 권장

```
19:         stream = p.open(format=p.get_format_from_width(w.getsampwidth()),  
                           channels=w.getnchannels(),rate=w.getframerate(), output=True)
```

블로킹 모드 재생

- WAV 파일 크기가 클 경우

- readframes() 메소드로 블록(또는 청크) 크기로 읽어 출력

- readframes() 메소드의 반환 값이 0일 때까지 반복

```
20:         data = w.readframes(1024)
21:         while len(data) > 0:
22:             stream.write(data)
23:             data = w.readframes(1024)
```

블로킹 모드 재생

□ 전체 코드

```
01: import pyaudio
02: import wave
03:
04: w = wave.open("/usr/share/sounds/alsa/Side_Left.wav", "rb")
05: p = pyaudio.PyAudio()
06:
07: stream = p.open(format=p.get_format_from_width(w.getsampwidth()),
                  channels=w.getnchannels(), rate=w.getframerate(), output=True)
08:
09: data = w.readframes(1024)
10: while len(data) > 0:
11:     stream.write(data)
12:     data = w.readframes(1024)
13:
14: w.close()
15: stream.stop_stream()
16: stream.close()
17: p.terminate()
```

논블로킹 모드 재생

□ 논블로킹 모드 재생 예제

- 소리 출력이 완료될 때까지 응용프로그램은 대기 상태
 - 다른 작업은 불가능
- 대기 상태에 빠지지 않아 배경 음악처럼 소리 출력 가능
- 커널이 재생 가능한 프레임 수를 사용자 메소드에 콜백으로 전달
 - 사용자 콜백 메소드는 호출될 때마다 WAV 파일에서 해당 개수만큼 프레임을 읽어 반환 값으로 커널에 전달

```
def callback(in_data, frame_count, time_info, status):  
    data = w.readframes(frame_count),  
    return (data, pyaudio.paContinue)
```

논블로킹 모드 재생

- ▣ 콜백 메소드를 open메소드의 stream_callback 인자로 전달

- start_stream() 메소드를 통해 비동기 재생을 시작

```
stream = p.open(format=p.get_format_from_width(w.getsampwidth()),
channels=w.getnchannels(), rate=w.getframerate(), output=True, stream_callback=callback)

stream.start_stream()
```

- ▣ 반복문과 is_active()로 재생이 끝났을 경우 프로그램을 종료시킵니다.

```
while stream.is_active(): # WAV 재생이 끝나면 False
    print("main work...")
    time.sleep(0.1)

stream.stop_stream()
stream.close()
p.terminate()
```

논블로킹 모드 재생

□ 전체 코드

```
01: import pyaudio
02: import wave
03: import time
04:
05: def callback(in_data, frame_count, time_info, status):
06:     data = w.readframes(frame_count)
07:     return (data, pyaudio.paContinue)
08:
09: w = wave.open("/usr/share/sounds/alsa/Side_Left.wav", "rb")
10: p = pyaudio.PyAudio()
11:
12: stream = p.open(format=p.get_format_from_width(w.getsampwidth()),
                  channels=w.getnchannels(), rate=w.getframerate(),
                  output=True, stream_callback=callback)
```

```
13:
14: stream.start_stream()
15:
16: while stream.is_active(): # WAV 재생이 끝나면 False
17:     print("main work...")
18:     time.sleep(0.1)
19:
20: stream.stop_stream()
21: stream.close()
22: p.terminate()
```

논블로킹 모드 재생

- 배경 음악처럼 재생이 끝날 때마다 다시 처음으로 되감아 재생할 때
 - ▣ 사용자 콜백 메소드가 커널이 요구한 프레임 수보다 더 적은 데이터를 반환
 - 커널은 해당 데이터만 출력하고 더는 사용자 콜백을 호출하지 않음
 - ▣ Wave_read 객체의 rewind() 메소드로 WAV 파일 포인터를 처음으로 옮김
 - ▣ 커널이 요구한 프레임 수에서 부족한 만큼 추가로 더 읽어 커널에 반환

```
def callback(in_data, frame_count, time_info, status):  
    data = w.readframes(frame_count)  
    mod = frame_count - len(data) // w.getsampwidth()  
    if mod != 0:  
        w.rewind()  
        data += w.readframes(mod)  
  
    return (data, pyaudio.paContinue)
```

웨이브폼 출력

- numpy로 출력할 샘플링 주파수와 옥타브 및 음계 주파수를 이용
 - ▣ WAV 파일이 없어도 소리 출력 가능
 - ▣ `numpy.sin(2 * numpy.pi * numpy.arange(fs * duration) * f / fs).astype(numpy.float32)`
 - `numpy.pi`: 파이 값
 - `fs`: 샘플링 주파수
 - `duration`: 지속 시간
 - `f`: 음계 주파수

특정 음 출력

□ 주파수가 440.0Hz인 4옥타브 ‘라’ 를 PyAudio로 출력하는 예제

□ numpy 웨이브 폼 식 사용

01: import pyaudio	09: data = (np.sin(2 * np.pi * np.arange(fs * duration) * f/fs)).astype(np. float32)
02: import numpy as np	10:
03:	11: p = pyaudio.PyAudio()
04: volume = 0.5	12: stream = p.open(format=pyaudio.paFloat32, channels=1, rate=fs, output=True)
05: fs = 48000	13: stream.write(volume * data)
06: duration = 1.0	14:
07: f = 440.0	15: stream.stop_stream()
08:	16: stream.close()
	17: p.terminate()

Tone 클래스 구현

□ Tone 클래스

▣ numpy 웨이브 폼 식 이용

▣ Tone 객체를 생성하는 __init__() 메소드

■ volume과 rate, channels를 인자로 받아 PyAudio 및 Output Stream 객체를 만듦

```
01:         import pyaudio
02:         import numpy as np
03:
04:         class Tone:
05:             def __init__(self, volume=.5, rate=48000, channels=1):
06:                 self.volume = volume
07:                 self.rate = rate
08:                 self.channels = channels
09:                 self.p = pyaudio.PyAudio()
10:                 self.stream = self.p.open(format=pyaudio.paFloat32, channels=self.channels, rate=self.rate, output=True)
```

Tone 클래스 구현

▣ play() 메소드

- octave와 note, duration을 인자로 받아 numpy 웨이브 폼 식으로 만든 데이터 생성
- duration은 4가 약 1초에 해당

```
11:         def play(self, octave, note, duration):
12:             f = 2**(octave) * 55 * 2**(((note) - 10) / 12)
13:             sample = (np.sin(2 * np.pi * np.arange(self.rate * duration) * f / self.rate)).astype(np.float32)
14:             self.stream.write(self.volume * sample)
```

Tone 클래스 구현

▣ stop() 메소드

- 출력을 멈추고 Stream과 PyAudio 객체를 닫음

```
15:         def stop(self):
16:             self.stream.stop_stream()
17:             self.stream.close()
18:             self.p.terminate()
```

Tone 클래스 구현

▣ `__enter__()` 메소드와 `__exit__()` 메소드

- `with` 구문에서도 사용할 수 있도록 구현
- `__enter__()` 는 Tone 개체를 반환
- `__exit__()` 는 `stop()` 메소드를 호출

```
19:         def __enter__(self):
20:             return self
21:
22:         def __exit__(self, type, value, traceback):
23:             self.stop()
```

Tone 클래스 구현

- ▣ 3옥타브 ‘도’ 에서 ‘시’ 까지 약 1초 간격으로 출력 예
 - Tone 객체를 with 구문으로 만들

```
24:         with Tone() as tone:
25:             for n in [1, 3, 5, 7, 8, 10, 12]:
26:                 tone.play(3, n, 4)
```

Tone 클래스 구현

□ 전체 코드

```
01: import pyaudio
02: import numpy as np
03:
04: class Tone:
05:     def __init__(self, volume=.5, rate=48000, channels=1):
06:         self.volume = volume
07:         self.rate = rate
08:         self.channels = channels
09:         self.p = pyaudio.PyAudio()
10:         self.stream = self.p.open(format=pyaudio.paFloat32,
11: channels=self.channels, rate=self.rate, output=True)
12:     def __enter__(self):
13:         return self
14:
15:     def __exit__(self, type, value, traceback):
```

```
16:         self.stop()
17:
18:     def stop(self):
19:         self.stream.stop_stream()
20:         self.stream.close()
21:         self.p.terminate()
22:
23:     def play(self, octave, note, duration):
24:         f = 2**(octave) * 55 * 2**(((note) - 10) / 12)
25:         sample = (np.sin(2 * np.pi * np.arange(self.rate *
26: duration) * f / self.rate)).astype(np.float32)
27:         self.stream.write(self.volume * sample)
28:
29:     with Tone() as tone:
30:         for n in [1, 3, 5, 7, 8, 10, 12]:
31:             tone.play(3, n, 4)
```

Pop 라이브러리의 Tone 클래스

- Pop 라이브러리의 Tone 클래스

- `Tone(tempo=100, volume=.5, rate=48000, channels=1)`: Tone 객체 생성

- tempo: 빠르기. 기본값은 100
 - volume: 볼륨. 기본값은 0.5
 - rate: 샘플링 비율. 기본값은 48000
 - channels: 채널 수. 기본값은 1

- `close()`: 오디오 자원 해제

Pop 라이브러리의 Tone 클래스

- ▣ setTempo(tempo): 빠르기 설정
 - tempo: 빠르기
- ▣ rest(duration): 쉼표
 - duration: 길이. 1, 1/2, 1/4, 1/8, 1/16, ...
- ▣ play(octave, pitch, duration): 음 출력
 - octave: 옥타브. 1 ~ 8
 - pitch: 음. “DO”, “DO#”, “RE”, “RE#”, “MI”, “FA”, “SOL”, “SOL#”, “RA”, “RA#”, “SI”
 - duration: 길이. 1, 1/2, 1/4, 1/8, 1/16, ...

Tone 클래스 구현

- Pop 라이브러리의 Tone 객체로 “학교종” 을 출력 예제
 - Tone 객체를 만든 후 setTempo()로 빠르기를 200으로 설정
 - play()에 옥타브와 음표, 길이 입력

```
01: from pop import *
02:
03: shoolBall1 = ((4, "SOL", 1/4), (4, "SOL", 1/4), (4, "RA", 1/4),
(4, "RA", 1/4), (4, "SOL", 1/4), (4, "SOL", 1/4), (4, "MI",
1/2), (4, "SOL", 1/4), (4, "SOL", 1/4), (4, "MI", 1/4), (4,
"MI", 1/4), (4, "RE", 1/2 + 1/4))
04:
05: shoolBall2 = (*shoolBall1[:7], (4, "SOL", 1/4), (4, "MI", 1/2),
(4, "SOL", 1/4), (4, "MI", 1/4), (4, "RE", 1/4), (4, "MI", 1/4),
(4, "DO", 1/2 + 1/4))
06:
```

```
07: with Tone() as tone:
08:     tone.setTempo(200)
09:
10:     for n in shoolBall1:
11:         tone.play(*n)
12:     tone.rest(1/4)
13:
14:     for n in shoolBall2:
15:         tone.play(*n)
16:     tone.rest(1/4)
```

WAV 파일 녹음

- ▣ wave 모듈의 open() 메소드로 WAV 파일을 쓰기 전용으로 열기
 - 해당 경로에 WAV 파일이 만들어지고 Wave_write 객체 반환
- ▣ wave_read 객체의 setnchannels(), setsampwidth(), setframerate() 설정
 - 샘플링 타입, 채널 수, 샘플링 비율을 설정
- ▣ PyAudio 객체의 open() 메소드로 만든 입력 Stream에서 읽은 데이터 쓰기
 - 모드에 따라 읽고 쓰는 방식이 다름

```
01:         w = wave.open("./out.wav", "wb")
02:         w.setsampwidth(p.get_sample_size(pyaudio.paInt16))
03:         w.setnchannels(1)
04:         w.setframerate(RATE)
```

블로킹 모드 녹음

- ▣ Input Stream 객체의 read() 메소드로 마이크 입력을 읽기
- ▣ Wave_read 객체의 writeframes()에 이를 전달해 스피커에 출력
- ▣ Ctrl + C를 누를 경우 녹음이 종료
 - Jupyter에서는 중지(Interrupt the Kernel) 버튼

블로킹 모드 녹음

□ 전체 코드

```
01: import pyaudio
02: import wave
03:
04: CHUNK = 1024
05: RATE = 48000
06:
07: p = pyaudio.PyAudio()
08: stream = p.open(format=pyaudio.paInt16, channels=1, rate=RATE,
09:                 input=True, frames_per_buffer=CHUNK)
10:
11: w = wave.open("./out.wav", "wb")
12: w.setsampwidth(p.get_sample_size(paudio.paInt16))
```

```
13: w.setnchannels(1)
14: w.setframerate(RATE)
15:
16: try:
17:     while True:
18:         w.writeframes(stream.read(CHUNK))
19: except KeyboardInterrupt:
20:     w.close()
21:
22: stream.stop_stream()
23: stream.close()
24: p.terminate()
```

블로킹 모드 녹음

- 마이크 입력, 스피커 출력 정상동작 확인
 - 옵션 plughw:1 : 캡처 장치(마이크)로 사운드 카드 1을 지정
 - 옵션 -d : 기본 사운드 카드로 설정된 스피커에 캡처한 오디오 데이터를 전달
 - Soda는 USB 사운드 카드가 사운드 카드 1이냐 기본 사운드 카드

```
sox -t alsa plughw:1 -d
```

블로킹 모드 녹음

- 마이크가 정상적으로 동작하면 프로그램을 실행해 녹음을 시작
 - 마이크 입력은 현재 경로의 out.wav에 저장
 - 결과를 확인하려면 Ctrl+C를 눌러 녹음을 멈춘 후 play 명령을 실행
 - Jupyter에서는 중지(Interrupt the Kernel) 버튼

```
play out.wav
```

블로킹 모드 녹음

- 프로그램에서 녹음 시간을 제한
 - ▣ for 루프 구문을 사용
 - ▣ 입력 데이터를 리스트에 추가한 후 루프를 탈출하면 리스트 내용을 모두 파일에 저장하는 방식
 - 녹음 시간을 지나치게 길게 설정하면 메모리가 부족할 수 있으므로 주의

```
TIME = 5 # 5초  
data = []
```

```
for _ in range(0, int(RATE / CHUNK * TIME)):  
    d = stream.read(CHUNK)  
    data.append(d)
```

```
w.writeframes(b''.join(data))
```

블로킹 모드 녹음

□ 녹음 시간 설정 부분이 적용된 전체 코드

```
01: import pyaudio
02: import wave
03:
04: CHUNK = 1024
05: RATE = 48000
06:
07: p = pyaudio.PyAudio()
08: stream = p.open(format=pyaudio.paInt16, channels=1, rate=RATE,
09:                 input=True, frames_per_buffer=CHUNK)
10: w = wave.open("./out.wav", "wb")
11: w.setnchannels(1)
12: w.setsampwidth(p.get_sample_size(paudio.paInt16))
13: w.setframerate(RATE)
14:
15: TIME = 5 # 5초
16: data = [ ]
17:
18: for _ in range(0, int(RATE / CHUNK * TIME)):
19:     d = stream.read(CHUNK)
20:     data.append(d)
21:
22: w.writeframes(b''.join(data))
23:
24: w.close()
25: stream.stop_stream()
26: stream.close()
27: p.terminate()
```

논블로킹 모드 녹음

- 사용자 콜백 메소드를 사용
 - ▣ Wave_write 객체의 writeframes() 메소드로 오디오 데이터 저장
 - ▣ 저장한 오디오 데이터 크기만큼 빈 데이터를 만듦
 - paContinue 또는 paContinue와 함께 튜플로 반환
 - paContinue : 작업할 데이터가 더 남았다는 반환값
 - paComplete : 작업이 완료되었다는 반환값

```
01: def callback(in_data, frame_count, time_info, status):  
02:     w.writeframes(in_data)  
03:     data = chr(0) * len(in_data)  
04:     return (data, pyaudio.paContinue if not isStop else pyaudio.paComplete)
```

논블로킹 모드 녹음

- 녹음이 진행 중이라는 것을 알 수 있도록 “recording...” 을 출력

```
05: try:
06:     print("Recording..",end="")
07:     while True:
08:         print(".",end="")
09:         time.sleep(0.5)
10: except KeyboardInterrupt:
11:     isStop = True
12:     time.sleep(1) # 녹음을 완료할 때까지 대기
```

논블로킹 모드 녹음

□ 전체 코드

```
01: import pyaudio
02: import wave
03: import time
04:
05: CHUNK = 1024
06: RATE = 48000
07: isStop = False
08:
09: p = pyaudio.PyAudio()
10: w = wave.open("out.wav", 'wb')
11: w.setsampwidth(p.get_sample_size(pyaudio.paInt16))
12: w.setnchannels(1)
13: w.setframerate(RATE)
14:
15: def callback(in_data, frame_count, time_info, status):
16:     w.writeframes(in_data)
17:     data = chr(0) * len(in_data)
18:     return (data, pyaudio.paContinue if not isStop else pyaudio.paComplete)
```

```
20: stream = p.open(format=pyaudio.paInt16,
channels=1, rate=RATE, input=True,
frames_per_buffer=CHUNK, stream_callback=callback)
21: stream.start_stream()
22:
23: try:
24:     print("Recording..", end="")
25:     while True:
26:         print(".", end="")
27:         time.sleep(0.5)
28: except KeyboardInterrupt:
29:     isStop = True
30:     time.sleep(1) # 녹음을 완료할 때까지 대기
31:
32: w.close()
33: stream.stop_stream()
34: stream.close()
35: p.terminate()
```

마이크를 소음 측정 센서로 활용

- Input Stream에서 읽은 오디오 데이터의 RMS를 계산
- audioop 모듈
 - ▣ 원시 오디오 데이터 가공을 위한 여러 메소드를 제공
 - ▣ 그 중 RMS 계산용 rms() 메소드도 포함

```
01:         def callback(in_data, frame_count, time_info, status):
02:             rms = audioop.rms(in_data, 2)
03:             print(' ' * (rms//50), '*(', rms, ')')
04:             data = chr(0) * len(in_data)
05:             return (data, pyaudio.paContinue if not isStop else pyaudio.paAbort)
```

마이크를 소음 측정 센서로 활용

□ 전체 코드

```
01: import pyaudio
02: import audioop
03: import time
04:
05: CHUNK = 1024
06: RATE = 48000
07: isStop = False
08:
09: p = pyaudio.PyAudio()
10:
11: def callback(in_data, frame_count, time_info, status):
12:     rms = audioop.rms(in_data, 2)
13:     print('=' * (rms//50), '(' , rms, ')')
14:     data = chr(0) * len(in_data)
15:     return (data, pyaudio.paContinue if not isStop else pyaudio.paAbort)
16:
17: stream = p.open(format=pyaudio.paInt16,
```

```
18:         channels=1,
19:         rate=RATE,
20:         input=True,
21:         frames_per_buffer=CHUNK,
22:         stream_callback=callback)
23: stream.start_stream()
24:
25: try:
26:     while True:
27:         time.sleep(0.1)
28: except KeyboardInterrupt:
29:     isStop = True
30:
31: stream.stop_stream()
32: stream.close()
33: p.terminate()
```

마이크를 소음 측정 센서로 활용

□ 프로그램실행

▣ 마이크에서 감지한 소음과 그래프 출력

```
soda@soda:~ $ python3 code101.py
*( 284 )
*( 297 )
=====*( 302 )
=====*( 961 )
=====*( 2629 )
=====*( 3077
)
=====*(
3307 )
=====
=====*( 3791 )
=====
=====*( 3790 )
=====*( 32
24 )
=====*( 2741 )
=====*( 2287 )
=====*( 1503 )
=====*( 1030 )
...
```

Pop 라이브러리의 SoundMeter 클래스

- Pop 라이브러리의 SoundMeter 클래스
 - ▣ 마이크를 고성능 주변소음 측정기로 활용
 - ▣ `SoundMeter(sType=pyaudio.paInt16, chNum=1, chunk=1024, sRate=48000)`: SoundMeter 객체 생성
 - `sType`: 샘플링 타입. 기본값은 `paInt16`
 - `chNum`: 채널 수. 기본값은 1
 - `chunk`: 버퍼 크기. 기본값은 1024
 - `sRate`: 샘플링 비율. 기본값은 48000

Pop 라이브러리의 SoundMeter 클래스

- ▣ `setCallback(func, *args)`: 사용자 콜백 메소드 등록 및 측정 시작
 - `func`: 사용자 콜백 메소드
 - `args`: 사용자 콜백 메소드가 호출될 때 함께 전달할 인자. 생략 가능
- ▣ `callback(rms, inData, args, ...)`: 사용자 콜백 메소드
 - `rms`: 입력 데이터의 RMS
 - `inData`: 입력 데이터
 - `args, ...`: `setCallback()`에서 전달한 인자들. 생략 가능
- ▣ `stop ()`: 측정을 중단하고 오디오 자원 해제

Pop 라이브러리의 SoundMeter 클래스

□ 주변 소음 크기가 일정값 이상일 경우에 출력 예제

□ SoundMeter를 이용

```
01:         import time
02:         from pop import *
03:
04:         sm = SoundMeter()
05:
06:         def onSoundMeter(rms, inData):
07:             if(rms>600):
08:                 print(rms)
09:
10:         sm.setCallback(onSoundMeter)
11:
12:         input("input something")
13:
14:         sm.stop()
```

pop 라이브러리

- Pop.AudioPlay, Pop.AudioPlayList, Pop.AudioRecord 클래스
 - ▣ Pop 라이브러리에서 제공하는 PyAudio 기반 오디오 녹음 재생을 위한 클래스
 - ▣ AudioPlay 클래스
 - WAV 파일을 재생할 때 블로킹과 논블로킹 모드를 모두 지원
 - AudioPlay(file, blocking=True, cont=False): AudioPlay 객체 생성
 - file: 재생할 WAV 파일 이름
 - blocking: True이면 블로킹 모드, False는 논블로킹 모드. 기본값은 True
 - cont: 논블로킹 모드일 때 True이면 반복재생, False이면 단일 재생. 기본값은 False

pop 라이브러리

- `run()`: 재생 시작
- `isPlay()`: 논블로킹 모드일 때 재생 상태 반환
 - True이면 재생 중. False이면 종료
- `stop()`: 논블로킹 모드일 때 멈춤
- `close()`: 명시적으로 오디오 자원 해제
 - 프로그램이 종료 등으로 `AudioPlay` 객체가 제거될 때 자동 호출

pop 라이브러리

▣ AudioPlaylist 클래스

- 파일 리스트를 인자로 전달
- run() 메소드에 재생할 리스트의 인덱스를 지정 가능
- 그 외는 AudioPlay와 동일
- AudioPlaylist(files, blocking=True, cont=False): AudioPlaylist 객체 생성
 - files: 재생할 WAV 파일 리스트
 - blocking: True이면 블로킹 모드, False는 논블로킹 모드. 기본값은 True
 - cont: 논블로킹 모드일 때 True이면 연속 재생. False이면 단일 재생. 기본값은 False
- run(pos=0): 재생 시작
 - pos: 리스트 인덱스. 기본값은 0

pop 라이브러리

▣ AudioRecord 클래스

- 사용자가 원할 때 즉시 녹음을 중단할 수 있어야 하므로 논블로킹 모드만 지원
- `AudioRecord(file, sFormat=8, sChannel=1, sRate=48000, sFramePerBuffer=1024):`
AudioRecord 객체 생성
 - file: 저장할 WAV 파일 이름
 - sFormat: 샘플링 타입. 8 (paInt16), 2 (paInt32), 1 (paFloat32). 기본값은 8
 - sChannel: 채널 수. 기본값은 1
 - sRate: 샘플링 속도. 기본값은 48000
 - sFramePerBuffer: 버퍼 당 프레임 수 지정. 기본값은 1024
- `run():` 재생 시작

pop 라이브러리

- `isPlay()`: 논블로킹 모드일 때 재생 상태 반환
 - True이면 재생 중. False이면 종료
- `stop()`: 논블로킹 모드일 때 멈춤
- `close()`: 명시적으로 오디오 자원 해제
 - 프로그램이 종료 등으로 `AudioRecord` 객체가 제거될 때 자동 호출

pop 라이브러리

- ▣ `audio_play()` 메소드
 - 단순히 해당 파일을 재생합니다.
 - `audio_play(file):` 파일 재생
 - file: 재생할 WAV 파일 이름

pop 라이브러리

□ AudioRecord와 AudioPlay 클래스를 이용해 녹음 및 재생하는 예제

▣ 5초간 녹음한 내용을 “my_record.wav” 라는 이름으로 저장

```
01:         with AudioRecord("my_record.wav") as record:
02:             record.run()
03:             print("Start Recording...")
04:
05:             for _ in range(5):
06:                 time.sleep(1)
07:
08:             record.stop()
09:             print("Stop Recording...")
```

pop 라이브러리

- ▣ 논블로킹으로 12초간 “my_record.wav” 를 반복재생

```
10:         with AudioPlay("my_record.wav", False, True) as play:
11:             play.run()
12:             print("Start Play...")
13:             for _ in range(12):
14:                 time.sleep(1)
15:
16:             play.stop()
17:             print("Stop play...")
```

pop 라이브러리

□ 전체 코드

```
01:         from pop import *
02:
03:         with AudioRecord("my_record.wav") as record:
04:             record.run()
05:             print("Start Recording...")
06:
07:             for _ in range(5):
08:                 time.sleep(1)
09:
10:             record.stop()
11:             print("Stop Recording...")
```

```
12:
13:         # 논블로킹 모드, 반복재생
14:         with AudioPlay("my_record.wav", False, True) as
15:             play:
16:                 play.run()
17:                 print("Start Play...")
18:                 for _ in range(12):
19:                     time.sleep(1)
20:
21:                 play.stop()
22:                 print("Stop play...")
```

텍스트 음성 변환

□ TTS

▣ 텍스트를 음성으로 변환할 때 사용하는 모델

- 선정한 한 사람의 말소리를 녹음
- 일정한 음성 단위로 나눈 후 부호를 붙여 압생기에 저장
- 필요에 따라 해당 음성 단위만을 다시 합쳐 인위적인 말소리를 만드는 기술

▣ 분절음의 경계를 기준으로 음성의 앞과 뒤를 함께 기록해 음성합성

- 일반적으로 음성의 분절음을 합성하는 것이 어려움

텍스트 음성 변환

□ 텍스트 음성 변환 시스템

□ 프론트엔드와 백엔드로 나눔

□ 프론트엔드

- 사용자가 입력한 텍스트를 정규화해 숫자나 생략된 표연을 어떻게 처리할지 결정
- 각 단어를 발음 기호로 변환하면서 단어나 숙어, 문장 단위로 분할한 결과를 백엔드로 전달

□ 백엔드

- 운율 등을 조정한 다음 실제 음성 데이터로 합성에 출력

구글 텍스트 음성 변환기

- 구글 텍스트 음성 변환기

- ▣ 과거 : 스마트폰 화면의 텍스트를 소리 내어 읽도록 개발
- ▣ 현재 : DeepMind의 WaveNet에 이식된 클라우드 버전

- DeepMind의 AI 음성 합성 기술

- ▣ 기계 학습으로 음성 생성
- ▣ 사람의 음성 데이터베이스를 통해 파형 생성
- ▣ 최종 결과는 악센트와 같은 미묘한 처리까지 포함
- ▣ 애플의 Siri를 포함한 대부분의 음성 합성기는 연결합성 방식을 사용

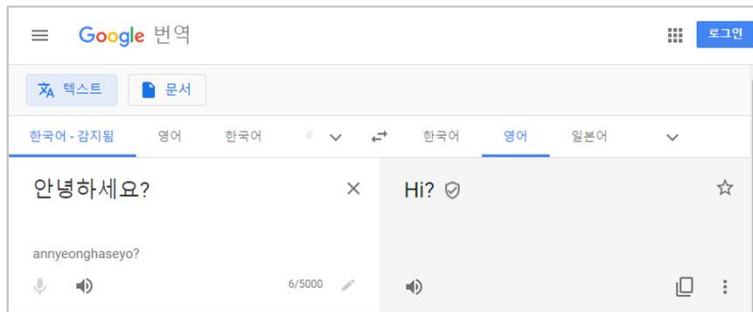
gTTS

- gTTS

- ▣ 클라우드 기반 텍스트 음성 변환 API에 대한 파이썬 라이브러리
 - 구글 번역기에 포함
 - 명령행 기반 툴 제공
- ▣ 구글 클라우드 서비스로 텍스트를 전달하면 결과를 mp3 파일로 전달

gTTS

- ▣ 맞춤식 음성 분석기 및 언어 자동 감지 기능을 제공
 - 발음을 교정할 수 있는 유연한 전처리와 적절한 억양, 약어, 숫자 등을 유지
 - 길이 제한 없이 텍스트를 읽을 수 있음



명령행 툴

▣ 명령행 툴

■ gtts-cli 명령

- 웹에서 사용자가 입력한 텍스트를 구글 클라우드로 보낸 후 mp3 결과 파일을 받아 현재 경로에 저장
- 재생은 play 명령 사용

■ gtts-cli의 주요 옵션

- -f, --file <file>: 파일로부터 텍스트 읽음
- -s, --slow: 천천히 발음
- -l, --lang=<lang_word>: IETF 언어 태그로 언어 설정. 기본값은 en(영어) (ko:한국어)
- --all: 적용 가능한 IETF 언어 태그 출력
- -o, --output <.mp3>: 결과를 파일로 저장. 기본값은 표준 출력으로 결과(mp3 내용) 전달

명령행 툴

- 기본 명령어를 사용한 “안녕하세요” 출력
 - ▣ `gtts-cli “안녕하세요” | play -t mp3 -`
 - 영어 발음이 적용되어 어색하게 들림
- ‘`--lang=ko`’ 옵션 추가 후 “안녕하세요” 출력
 - ▣ `gtts-cli “안녕하세요” --lang=ko | play -t mp3 -`
 - 자연스러운 출력됨

명령행 툴

- ‘--output’ 옵션
 - ▣ 결과를 파일로 저장
 - ▣ play로 재생
 - ▣ `gtts-cli “안녕하세요” --lang=ko --output hello.mp3`
 - ▣ `play hello.mp3`

gTTS 클래스

- ▣ `gTTS(text, lang='en', slow=False, lang_check=True, ...)`: gTTS 객체 생성
 - `text`: 읽을 텍스트
 - `lang`: 텍스트를 읽을 언어. IETF 언어 태그
 - 기본값은 영어. 한글은 'ko'
 - `slow`: True이면 더 느리게 읽음
 - `lang_check`: True이면 언어 오류 감지
 - 언어 선택이 잘못되면 `ValueError` 예외 발생
- ▣ `save(savefile)`: TTS API 요청 수행 및 결과 저장
 - `savefile`: 저장할 파일 이름
- ▣ `write_to_fp(fp)`: `save()` 메소드와 같으나 결과를 파일 객체에 저장
 - `fp`: 쓰기 모드의 바이너리 파일 객체

gTTS 클래스

- gTTS를 이용한 텍스트 변환 음성 출력 예제
 - 객체를 만든 후 save() 메소드를 통해 영문 텍스트를 압성한 후 MP3로 저장
 - 결과는 play 명령으로 확인

```
01:         from gtts import gTTS
02:
03:         text = "And the saddest thing. under the sun above"
04:
05:         tts = gTTS(text)
06:         tts.save("en_tts.mp3")
```

gTTS 클래스

- 한글 텍스트 합성

- gTTS() 생성자의 lang 인자에 'ko' 를 전달

- subprocess 모듈의 Popen() 메소드에 쉘 명령 play와 파일 이름 인자로 전달

- 코드에서 MP3 파일도 함께 재생

```
07:         with subprocess.Popen(['play', FILE_NAME]) as p:  
08:             p.wait()
```

gTTS 클래스

□ gTTS() 메소드를 통해 ‘너의 하늘을 보아’ 낭송 예제

```
01: from gtts import gTTS
02: import subprocess
03:
04: FILE_NAME = '너의 하늘을 보아.mp3'
05:
06: TEXT = """네가 자꾸 쓰러지는 게으
07: 네가 꼭 이룰 게으 있기 때문이야
08: 네가 지금 기으 잃어버린 게으
09: 네가 가야만 할 기으 있기 때문이야
10: 네가 다시 올 게으 가는 게으"""
```

```
11: 네가 꼭피워 내 게으 있기 때문이야
12: 힘들고 앞이 안 보일 때느
13: 너의 하늘을 보아"""
14:
15: tts = gTTS(TEXT, lang='ko')
16: tts.save(FILE_NAME)
17:
18: with subprocess.Popen(['play', FILE_NAME]) as p:
19:     p.wait()
```

gTTS 클래스

- 파일 이름, 텍스트, 언어를 입력하면 음성 합성 결과를 생성,재생 코드
 - Kernel이 shutdown 될 때까지 지속

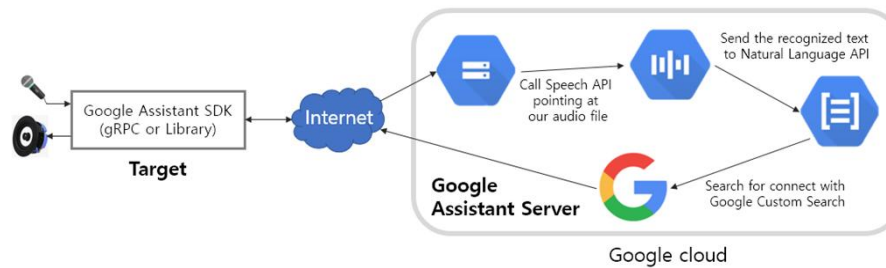
```
01: from gtts import gTTS
02: import subprocess
03:
04: try:
05:     while True:
06:         f = input("Enter of file name: ")
07:         t = input("Enter of Text: ")
08:         l = input("Select language (ko | en): ")
```

```
09:
10:         tts = gTTS(t, lang=l)
11:         tts.save(f + ".mp3")
12:
13:         with subprocess.Popen(["play", f + ".mp3"]) as p:
14:             p.wait()
15:     except KeyboardInterrupt:
16:         pass
```

구글 어시스턴트

□ 구글 어시스턴트

- 구글 클라우드에서 제공하는 AI 기반 음성 인식 및 행동 실행 서비스
- 구글 클라우드 서버와 연동하는 어시스턴트 SDK
 - 안드로이드 폰과 옛지 디바이스로 나뉨
 - 라즈베리 파이는 옛지 디바이스용 SDK를 사용해야 함
 - 학습이 아닌 상용 배포는 구글과의 계약 필요



구글 어시스턴트 사용 인증

- 옛지 디바이스에 구글 어시스턴트를 사용
 - ▣ Google Assistant SDK 설치를 비롯해 구글 어시스턴트 사용 인증 필요
 - ▣ 구글 어시스턴트 사용 인증은 구글 계정을 기반으로 진행
 - ▣ Actions on Google에 새 프로젝트를 만들고 모델을 등록해 자격 증명서를 다운
 - ▣ Google Cloud Platform에서 Google Assistant API 사용을 허용
 - ▣ 다운받은 자격 증명서로 옛지 디바이스를 인증

구글 어시스턴트 사용 인증

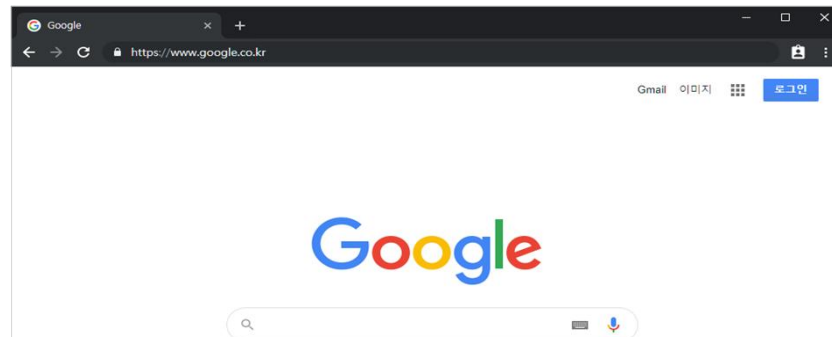
- Soda OS에서 구글 어시스턴트 사용
 - ▣ 등록 절차 적용 되어 있어 바로 서비스를 사용 가능
 - ▣ 계정마다 옛찌 디바이스의 무료 할당량이 1일 500회, 1분 60회로 제한
 - ▣ 자신의 계정으로 구글 어시스턴트 사용 인증을 새로 진행 권장
 - ▣ 옛찌 디바이스에서 구글 어시스턴트 사용에 필요한 세부 절차 안내
 - <https://developers.google.com/assistant/sdk/guides/service/python/>

활동 제어 설정

□ 첫 번째 단계

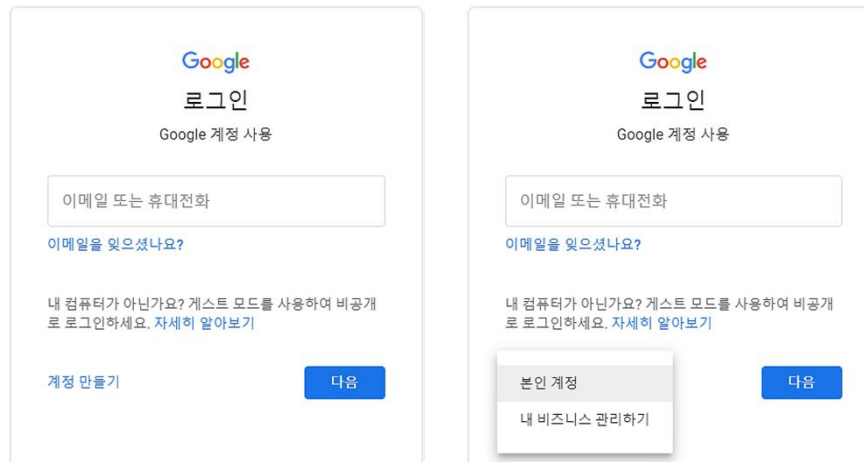
▣ 자신의 구글 계정에 대한 활동 제어를 설정

- PC에서 크롬 브라우저를 실행
- 구글 계정에 로그인 (프로젝트용 계정을 새로 만드는 것을 추천)
- 프로젝트 계정을 만들기 위해 게스트 세션으로 크롬 브라우저 실행



활동 제어 설정

- 오른쪽 상단의 ‘로그인’ 누르고 “계정 만들기” 에서 “본인 계정” 을 선택



Google
로그인
Google 계정 사용

이메일 또는 휴대전화

[이메일을 잊으셨나요?](#)


내 컴퓨터가 아닌가요? 게스트 모드를 사용하여 비공개로 로그인하세요. [자세히 알아보기](#)

계정 만들기 다음

본인 계정
내 비즈니스 관리하기

활동 제어 설정

- 계정 정보를 입력한 후 '다음' 을 선택



Google 계정 만들기

성

이름

사용자 이름

@gmail.com

문자, 숫자, 마침표를 사용할 수 있습니다
[대신 현재 이메일 주소 사용](#)


비밀번호

확인

문자, 숫자, 기호를 조합하여 8자 이상을 사용하세요

[대신 로그인하기](#)

다음



하나의 계정으로 모든 Google 서비스를 이용할 수 있습니다.


활동 제어 설정

- 자신의 휴대폰 번호를 입력한 후 ‘다음’ 을 선택


Google

전화번호 인증

Google에서는 보안을 위해 본인임을 확인하고자 합니다. Google에서 6자리 인증 코드가 포함된 문자 메시지를 전송합니다. 표준 요금이 적용됩니다

 ▼


[뒤로](#)[다음](#)



개인정보를 비공개로 안전하게 유지합니다.


활동 제어 설정

- 전화번호를 인증하기 위한 인증 코드가 문자로 오면 이를 입력



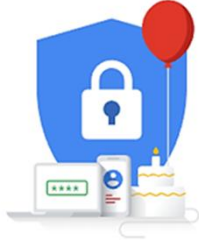
전화번호 인증

Google에서는 보안을 위해 본인임을 확인하고자 합니다. Google에서 6자리 인증 코드가 포함된 문자 메시지를 전송합니다. *표준 요금이 적용됩니다*

 **전화번호**

인증 코드 입력
G-

[뒤로](#) [전화로 대체](#) [확인](#)



개인정보를 비공개로 안전하게 유지합니다.

활동 제어 설정

- 개인 정보를 입력한 후 ‘다음’ 을 선택
 - 생일과 성별을 제외한 옵션 항목은 생략 가능

Google

Google에 오신 것을 환영합니다

👤

🇰🇷 ▼ 전화번호(선택사항)

전화번호는 계정 보안용으로 사용됩니다. 다른 사용자에게는 전화번호가 표시되지 않습니다.

복구 이메일 주소(선택사항)

계정을 안전하게 보호하기 위해 사용합니다


연도 월 일

생일

성별

이 정보가 필요한 이유


뒤로 다음



개인정보를 비공개로 안전하게 유지합니다.

활동 제어 설정

- 스크롤바를 아래로 내려 개인정보 보호 및 약관에 동의한 후 ‘계정 만들기’ 를 선택



개인정보 보호 및 약관


은 그 이후에 열어 씁니다.

자신의 데이터를 직접 관리
계정 설정에 따라 데이터의 일부가 Google 계정과 연결될 수 있으며, Google에서는 이 데이터를 개인정보로 취급합니다. 아래 '옵션 더보기'를 클릭하여 Google에서 이 데이터를 수집하고 사용하는 방식을 관리할 수 있습니다. 나중에 언제든지 내 계정(myaccount.google.com)으로 이동하여 설정을 변경하거나 동의를 철회할 수 있습니다.

[옵션 더보기](#) ▼

- ☒ Google의 서비스 약관에 동의합니다
- ☒ 위와 같은 주요 사항을 비롯하여 Google 개인정보 처리방침에 설명되어 있는 개인정보 수집·이용 목적·보유기간에 동의하며, 위치서비스 이용약관에 동의합니다.

[취소](#) [계정 만들기](#)



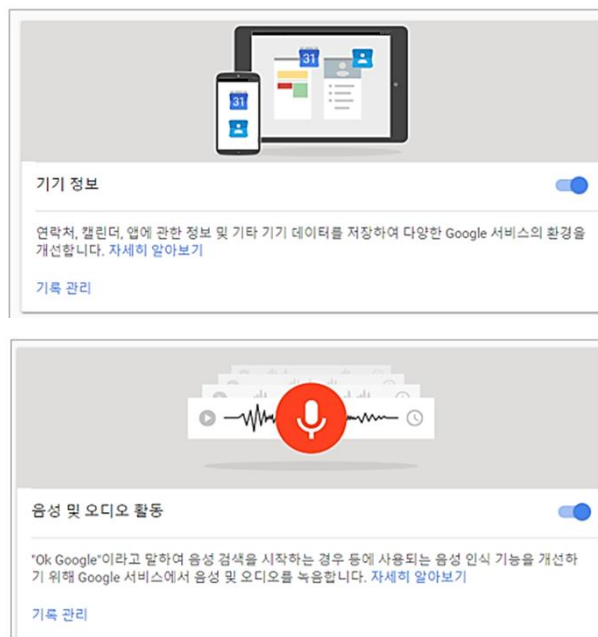
Google에서 수집하는 데이터와 사용 방법을 관리할 수 있습니다.

활동 제어 설정

- 계정 만들기가 완료되면 크롬 브라우저는 새로 만든 계정으로 로그인한 상태
- 이 계정에 대한 활동 제어를 설정하기 위해 다음 주소로 이동
 - <https://myaccount.google.com/activitycontrols>

활동 제어 설정

- “웹 및 앱 활동” 과 “기기 정보” , “음성 및 오디오 활동” 활성화



Actions on Google 프로젝트 생성

□ 두 번째 단계

▣ Actions on Google 프로젝트 생성

■ 옛지 디바이스 자격 증명서 다운로드

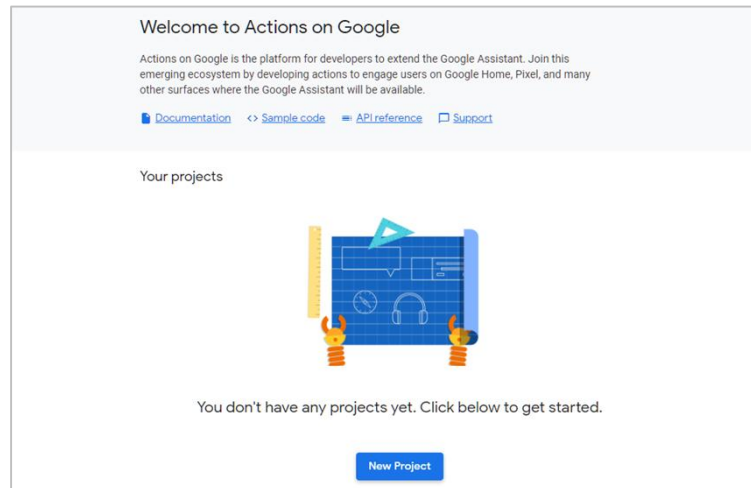
- Actions on Google 콘솔에서 새 프로젝트를 만든 후 장치 모델 등록 필요

■ Actions on Google은 구글 어시스턴트의 핵심 서비스 중 하나

- 장치마다 탑재된 서비스가 요청한 음성 명령을 분석해 지정된 액션 실행
- 대부분은 구글이 AI로 학습시킨 음성 답변이나 웹 브라우저 실행
- 옛지 디바이스에서 새로운 액션을 정의하는 것도 가능

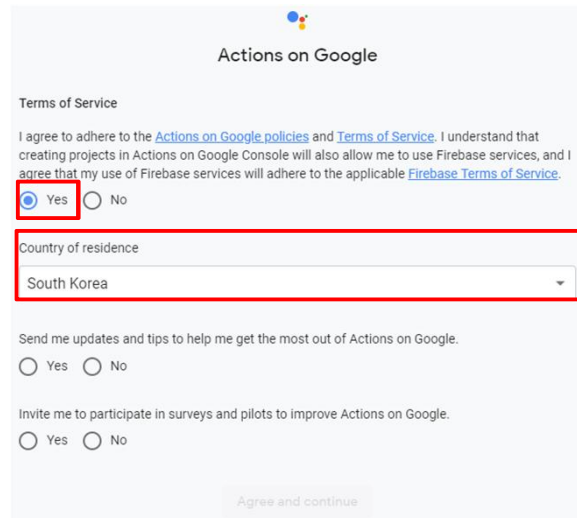
Actions on Google 프로젝트 생성

- Actions on Google 콘솔에서 새 프로젝트를 만들기 위해 다음 주소로 이동
 - <https://console.actions.google.com/?pli=1>
- “New Project” 를 선택해 새 프로젝트 만들기 시작



Actions on Google 프로젝트 생성

- 서비스 약관에 동의한 후 “Agree and continue” 를 선택
 - 업데이트 및 팁 받거나 설문조사, 파일럿 프로그램 참여는 옵션
- 프로젝트 이름을 입력한 후 언어는 “Korean” , 나라는 “South Korea” 를 선택



The screenshot shows the 'Actions on Google' setup interface. At the top, the Google logo is displayed above the text 'Actions on Google'. Below this, the 'Terms of Service' section is visible, containing a paragraph of text and two radio buttons: 'Yes' (selected) and 'No'. A red box highlights the 'Yes' radio button. Below the terms, there is a 'Country of residence' dropdown menu with 'South Korea' selected. Another red box highlights this dropdown. At the bottom, there are two more radio button options: 'Send me updates and tips to help me get the most out of Actions on Google.' (Yes/No) and 'Invite me to participate in surveys and pilots to improve Actions on Google.' (Yes/No). A greyed-out 'Agree and continue' button is located at the very bottom.

Actions on Google 프로젝트 생성

- “Create project” 를 선택하면 프로젝트가 만들어지는데 시간이 약간 걸릴 수 있음

New Project

Project Name

soda

Choose a language for your Actions project (you can add more Korean languages later)

Choose your country or region South Korea

Cancel Create project

Actions on Google 장치 모델 등록

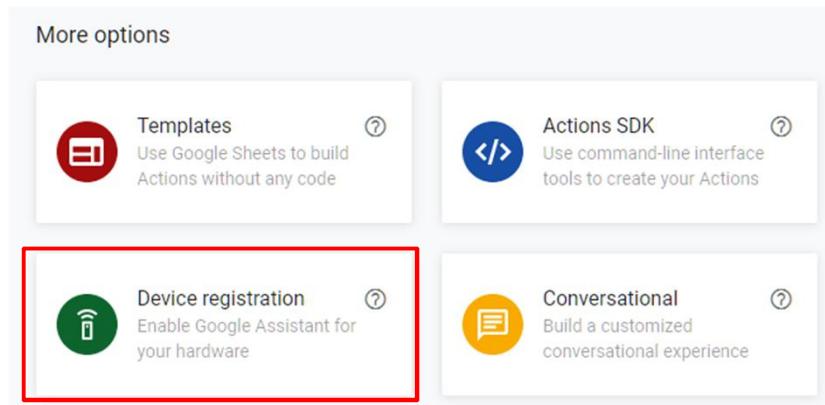
□ 세 번째 단계

▣ Actions on Google 장치 모델 등록

- 옛지 디바이스를 위한 간단한 정보를 입력하고 이를 토대로 자격 증명서를 발급
- 선택적으로 간단한 액션을 지정하는 절차
- 발급받은 자격 증명서로 옛지 디바이스를 인증해야 Google Assistant API를 사용 가능

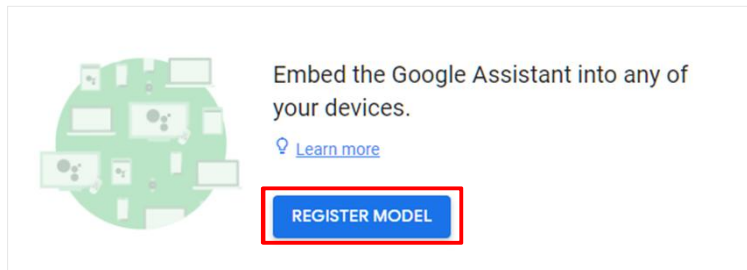
Actions on Google 장치 모델 등록

- 스크롤바를 가장 아래로 내린 후 More options에서 “Device registration” 선택



Actions on Google 장치 모델 등록

- 옛지 디바이스 모델 등록 화면이 표시되면 “REGISTER MODEL” 을 선택해 등록 시작



Actions on Google 장치 모델 등록

- 제품명과 제조사를 입력한 후 목록에서 장치 종류를 선택
 - 가장 중요한 모델 ID는 제품명과 제조사를 조합해 자동으로 만들어짐
 - 구글 어시스턴트 응용프로그램을 처음 실행할 때 프로젝트 ID와 함께 인자로 전달
 - 프로젝트 ID 는 제품명 앞의 2개 필드. **반드시 별도 메모 필요**
- “REGISTER MODEL” 선택

Register model

1 Create model 2 Download credentials 3 Specify traits

Product name ⓘ
pop

Manufacturer name ⓘ
soda

Device type ⓘ
Camera

Model id ⓘ
soda-fde1c0pop-mo7mwj

Cancel REGISTER MODEL

Actions on Google 장치 모델 등록

- Download OAuth 2.0 credentials” 을 선택해 자격 증명서를 다운
 - 이 증명서는 나중에 구글 어시스턴트를 사용할 엡지 디바이스를 인증하는데 필요
- Next 클릭

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](#)

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

Actions on Google 장치 모델 등록

- 옛지 디바이스를 위한 몇 가지 서비스 특성을 선택 가능
- 선택한 서비스 특성에 따라 Actions on Google에는 특정 음성 명령에 대한 액션이 추가
- 지금은 선택하지 않고 “SKIP” 를 선택하여 모델 등록 완료

Register model

Create model Download credentials 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

SKIP SAVE TRAITS

Actions on Google 장치 모델 등록

- 장치 모델 등록이 완료되면 목록이 표시
- 모델 제거, 해당 모델의 자격 증명서를 다시 다운로드 가능
- 특성과 같은 설정을 변경 가능


Device registration				
				REGISTER MODEL
Product name	Manufacturer name	Model ID	Device Type	Last updated time
pop	soda	soda-fdeltc-pop-m...	action.devices.typ...	Aug 08, 2019, 10:22...

Google Assistant API 사용 설정

- Google Assistant API를 사용 설정
 - ▣ 다음 주소의 Google Cloud Platform 콘솔로 이동
 - <https://console.developers.google.com/apis/library/embeddedassistant.googleapis.com>
 - ▣ 이때 크롬 브라우저는 프로젝트 계정 로그인 유지되어야 함

Google Assistant API 사용 설정

- ▣ 처음 접속하면 Google Cloud Platform에 대한 서비스 약과 동의 화면 표시
- ▣ “동의 및 계속하기” 선택

 Google Cloud Platform

seamo님, 환영합니다

Google Cloud Platform 인스턴스, 디스크, 네트워크, 기타 리소스를 한 곳에서 만들고 관리할 수 있습니다.

서비스 약관

☒ Google Cloud Platform 서비스 약관과 [관련 서비스 및 API](#)의 서비스 약관에 동의합니다.

거주 국가

대한민국 ▼

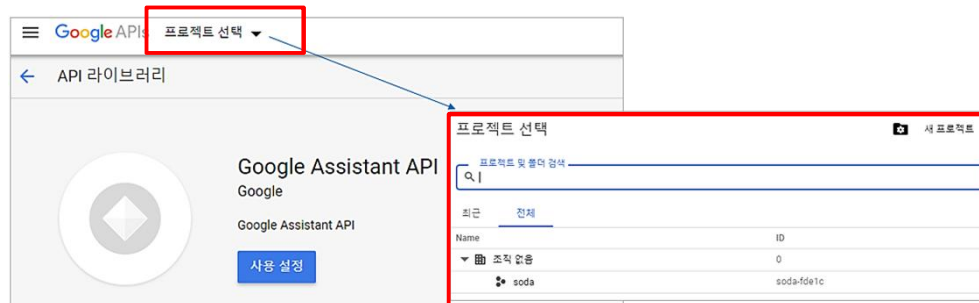
이메일 업데이트

☐ Google Cloud 및 Google Cloud 파트너가 보내는 뉴스, 제품 업데이트, 특별 이벤트에 대한 정기적인 이메일을 수신하겠습니다.

동의 및 계속하기

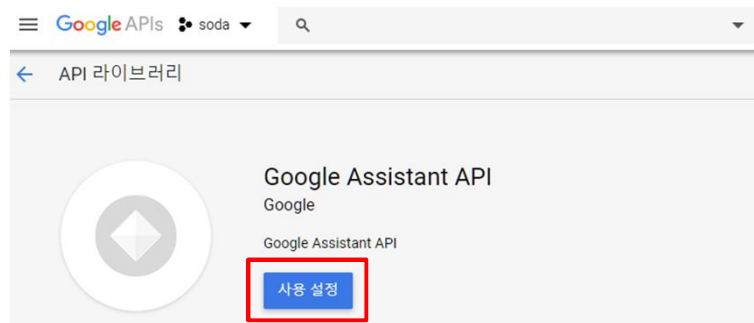
Google Assistant API 사용 설정

- ▣ 상단 메뉴바에서 “프로젝트 선택” 선택
- ▣ Actions on Google에서 만든 프로젝트를 선택한 후 “열기” 를 선택
- ▣ 만약 목록에 자신이 만든 프로젝트가 표시되지 않으면 기다렸다가 다시 시도



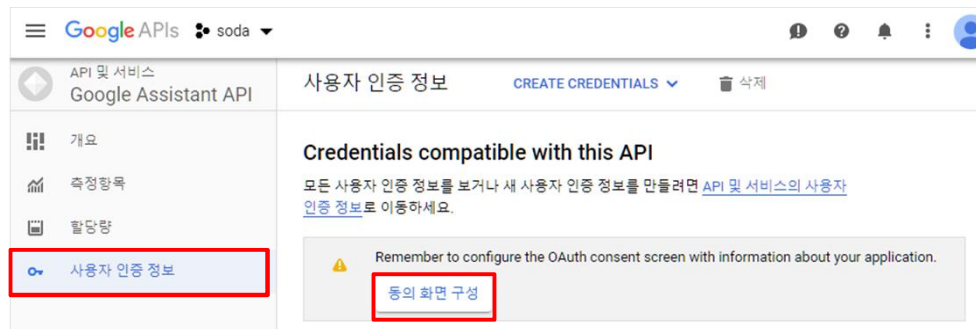
Google Assistant API 사용 설정

- ▣ Google Assistant API “사용 설정” 선택
- ▣ Google Assistant API를 사용할 수 있도록 허용



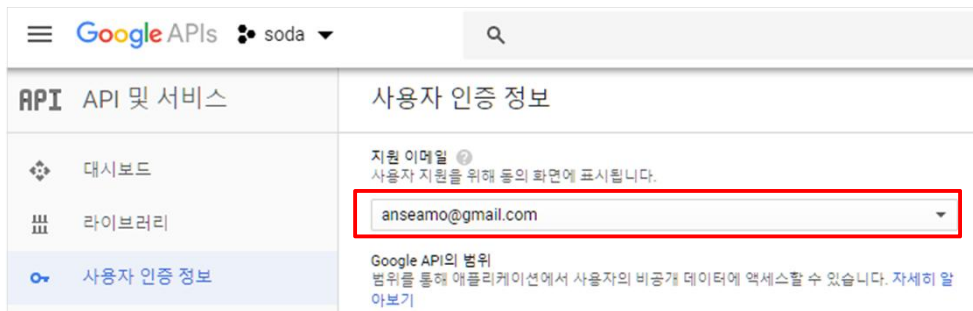
Google Assistant API 사용 설정

- 왼쪽 메뉴바에서 “사용자 인증 정보” 선택
 - OAuth 동의 화면을 구성
 - 사용자 정보를 제3자에게 제공하지 않고 인증 또는 인가를 허용하는 개방형 표준 프로토콜
- 처음에는 저장한 동의 화면이 없으므로 “동의 화면 구성” 을 선택



Google Assistant API 사용 설정

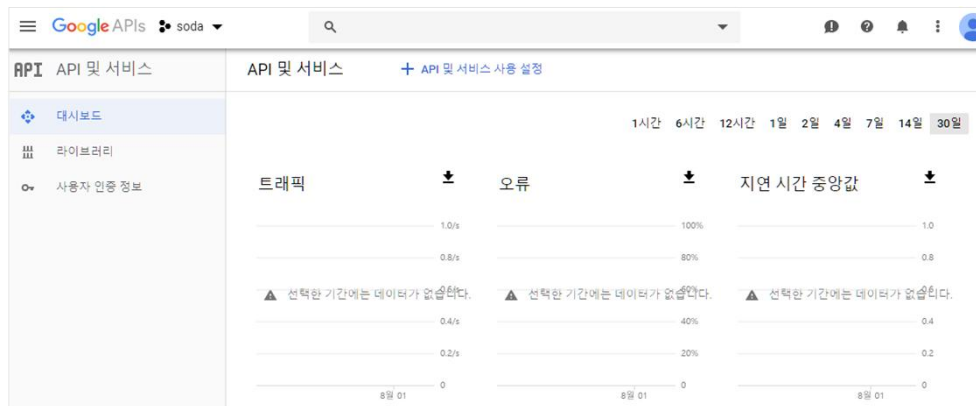
- 가운데 “OAuth 동의 화면” 선택
- 지원 이메일에서 자신의 이메일을 선택
- 다음 아래로 이동해 “저장” 선택



The screenshot shows the Google APIs console interface. At the top, there's a header with the Google APIs logo, a user profile icon labeled 'soda', and a search bar. Below the header, there's a sidebar on the left with navigation links: 'API 및 서비스' (APIs and Services), '대시보드' (Dashboard), '라이브러리' (Library), and '사용자 인증 정보' (User Consent Information). The '사용자 인증 정보' link is highlighted. The main content area is titled '사용자 인증 정보' (User Consent Information). It contains a section for '지원 이메일' (Support Email) with a subtext '사용자 지원을 위해 동의 화면에 표시됩니다.' (Displayed on the consent screen for user support). Below this, there's a dropdown menu showing 'anseamo@gmail.com', which is highlighted with a red rectangular box. At the bottom, there's a section for 'Google API의 범위' (Google API Scope) with a subtext '범위를 통해 애플리케이션에서 사용자의 비공개 데이터에 액세스할 수 있습니다. 자세히 알아보기' (You can use the scope to access user's private data in your application. Learn more).

Google Assistant API 사용 설정

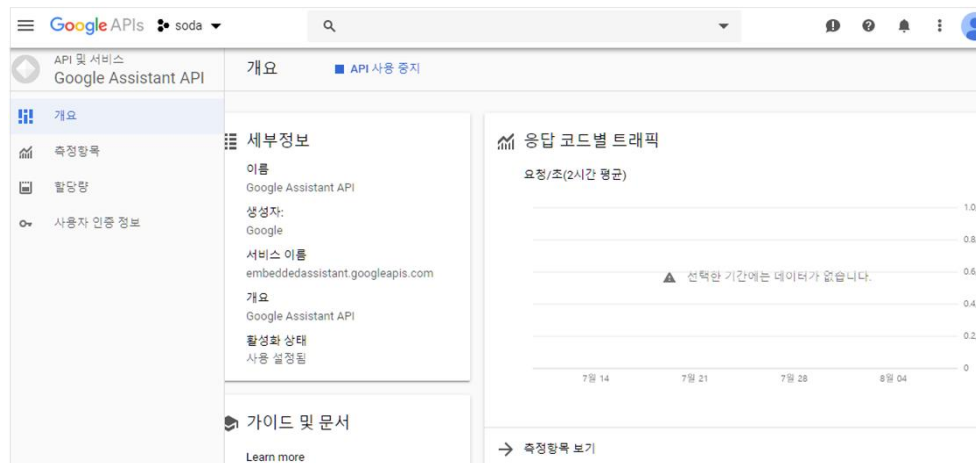
- 대시보드에 표시되는 트래픽 관련 정보는 Google Cloud Platform 전체 정보



Google Assistant API 사용 설정

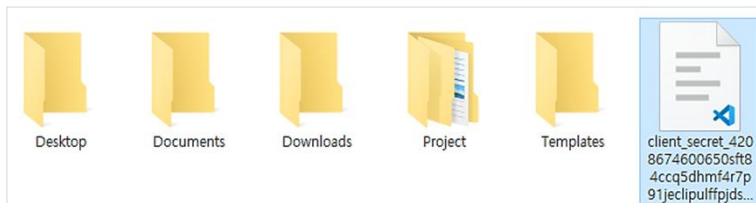
▣ Google Assistant API

■ 아래 필터 목록에서 Google Assistant API를 선택



엣지 디바이스 자격 증명

- ▣ AloT AutoCar를 네트워크 드라이버로 연결
- ▣ 자격 증명서를 AloT AutoCar에 복사
 - Actions on Google 콘솔에서 장치 모델 등록을 통해 다운받은 자격증명서



- ▣ 셸에서 다음 명령으로 Soda OS에 포함된 기존 자격 증명을 제거

```
rm -rf ~/.config/google*
```

엣지 디바이스 자격 증명

- ▣ 복사한 자격 증명서 경로에서 새 자격 증명을 등록위해 아래 명령어 실행
 - <json_file> 은 “client_secret_” 으로 시작하는 자격 증명서 파일 이름

```
google-oauthlib-tool --scope https://www.googleapis.com/auth/assistant-sdk-prototype \  
--save --headless --client-secrets <json_file>
```

옛지 디바이스 자격 증명

- ▣ 명령을 실행하면 URL이 출력, 인증 코드 입력 메시지 출력
 - OAuth 기반으로 응용프로그램을 인증하는데 필요한 코드를 받을 수 있는 URL
 - URL은 명령을 실행할 때마다 바뀜
 - 반드시 해당 URL을 통해 생성한 인증 코드 전체를 입력해야 InvalidGrantError가 발생 없음

```
google-oauthlib-tool --scope https://www.googleapis.com/auth/assistant-sdk-prototype  
--save --headless --client-secrets client_secret_4208674600650sft84ccq5dhmf4r7p91j  
pulff.json  
Please visit this URL to authorize this applications: https://accounts.google.com/o/oauth  
uth84ccq5dhmf4r4p92jeclpulffpjds.apps.googleusercontent.com\&redirect_uri=urn%3Aietf%3Aw  
oauth$.com%2Fault%2Fassistant-sdk-prototype\&state=L3uRbuAjMpFiMsZgmZdupPVNiuF30U\&prom  
consent\&acceLcedPPiP5EcPHdJikk0dfd_GHo\&code-hallenge_method=S256  
Enter the authorization code:
```


엣지 디바이스 자격 증명

- ▣ 웹 작업을 잠시 멈추고, 해당 URL을 크롬 브라우저로 복사해 이동
- ▣ OAuth 인증을 위한 계정 선택 화면이 표시
- ▣ Actions on Google 콘솔에서 새 프로젝트를 만들 때 사용한 계정을 선택




옛지 디바이스 자격 증명

▣ 모두 “허용” 을 선택

■ 구글 어시스턴트 사용에 관한 권한 부여 선택 창


project-420867460065에 권한 부여

 Google 어시스턴트 사용: 광범위한 Google 계정 액세스 ^

사용자를 대신하여 Google 어시스턴트와 소통합니다. 이 서비스는 Google 계정에 있는 대부분의 데이터에 액세스할 수 있습니다.

1 / 2 거부 **허용**

project-420867460065에 권한 부여

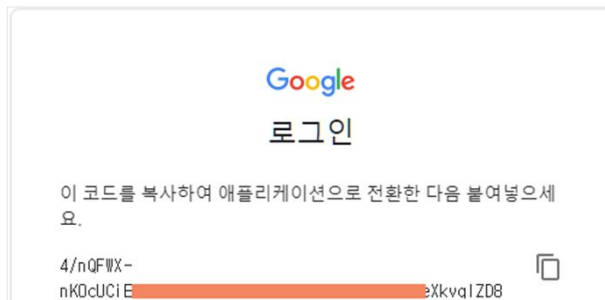
 내 Android 기기로 정보를 보냅니다. ^

내 Android 기기에 설치된 앱으로 정보를 보냅니다.

2 / 2 거부 **허용**

옛지 디바이스 자격 증명

- “어용” 을 선택
 - 최종적으로 부여한 권한 확인 화면이 표시되면
- 이후 “4/” 로 시작하는 인증 코드 복사



옛지 디바이스 자격 증명

- ❑ 멈췄던 웹 작업으로 돌아가기
- ❑ 복사한 인증코드를 마지막 줄에 붙여넣고 <Enter>를 눌러 인증 완료
- ❑ 인증 성공시 Google Assistant API를 호출하는데 필요한 액세스 토큰 저장
 - 저장 위치 : ~/.config/google-oauthlib-tool/ 경로

```
google-oauthlib-tool --scope https://www.googleapis.com/auth/assistant-sdk-prototype \
--save --headless --client-secrets client_secret_4208674600650sft84ccq5dhmf4r7p91jeclj
pulff.json
Please visit this URL to authorize this applications: https://accounts.google.com/o/oauth2/
uth84ccq5dhmf4r4p92jeclpulffpjds.apps.googleusercontent.com\&redirect_uri=urn%3Aietf%3Awg%3
oauth$.com%2Fault%2Fassistant-sdk-prototype\&state=L3uRbuAjMpFiMsZgmZdupPVNiuF30U\&prompt\=
consent\&acceLcedPPiP5EcPHdJikkOdfd_GHo\&code-hallenge_method\=S256
Enter the authorization code: 4/nQG-i4mLzkHKVAhFsvZkFPi4NGK8I11PAIh2JjwoBvzZlnFotW90tEI
credentials saved: /home/soda/.config/google-oauthlib-tool/credentials.json
```

인스턴스 ID 생성

- 디바이스 모델과 연관된 인스턴스를 통해 프로그램 실행
 - 인스턴스는 모델 ID, 프로젝트 ID로 응용프로그램과 디바이스 모델을 연관 시킴
 - 재사용을 위해 인스턴스 ID를 로컬에 저장
 - 같은 디바이스 모델을 사용할 때는 인자 생략 가능
 - 인스턴스 ID가 저장되지 않은 상태에서 인자 없이 응용프로그램을 실행 시
 - 오류 메시지를 출력하고 프로그램 종료

```
googlesamples-assistant-hotword
Traceback (most recent call last):
  File "/usr/local/bin/googlesamples-assistant-hotword", line 10, in <module>
    sys.exit(main())
  File "/usr/local/lib/python3.7/site-packages/googlesamples/assistant/library/hotword.py", line 119, in main
    raise Exception('Missing --device-model-id option')
Exception: Missing --device-model-id option_
```

인스턴스 ID 생성

- Soda OS는 인스턴스 ID를 제공. 인자없이 응용프로그램 실행 가능
- 사용자가 자신의 계정으로 새로 구글 어시스턴트 사용 인증 시
 - ▣ 최초 한 번은 프로젝트 ID와 모델 ID를 인자로 전달 필요

인스턴스 ID 생성

- gRPC 서비스 기반의 googlesamples-assistant-pushtotal
- 다음과 실행해 같이 인스턴스 ID를 생성
- <Enter> 키를 눌러 음성 명령을 시작하며 기본 언어는 영어

```
googlesamples-assistant-pushtotal --project-id <project_id> --device-model-id <model_id>
```

구글 어시스턴트 SDK 설치

- 구글 어시스턴트 SDK
 - ▣ 라이브러리, SDK 예제, gRPC, OAuth 인증 툴 구성
 - ▣ Soda OS에는 미리 설치되어 있음
- 구글 어시스턴트 라이브러리에서 사용하는 오픈소스 소프트웨어
 - ▣ portAudio : 오디오 입출력에 사용
 - ▣ libffi : 파이썬과 라이브러리 사이 호출 인터페이스 맞춰 줌
 - ▣ libssl : Google Cloud Platform과 옛지 디바이스 사이 암호화된 통신 채널 만듦
 - ▣ libmpg123 : MP3 포맷의 데이터 압축과 해제 수행

구글 어시스턴트 SDK 설치

- ▣ 어시스턴트 라이브러리에서 사용하는 오픈소스 소프트웨어 설치 명령

```
sudo apt install portaudio19-dev libffi-dev libssl-dev libmpg123-dev
```

- ▣ 구글 어시스턴트 라이브러리 설치 명령

```
sudo pip3 install --upgrade google-assistant-library
```

구글 어시스턴트 SDK 설치

- 구글 어시스턴트 예제와 OAuth 인증 툴을 설치할 때
 - 일반 사용자 계정에서 sudo 명령을 함께 사용하면 실패
 - “sudo su” 명령으로 루트 계층용 셸을 만들어 진행
- 구글 어시스턴트 예제와 gRPC, OAuth 인증 툴을 설치하는 명령

```
sudo su
pip3 install --upgrade google-assistant-sdk[samples]
pip3 install --upgrade google-assistant-grpc
pip3 install --upgrade google-auth-oauthlib[tool]
exit
```

구글 어시스턴트 SDK 설치

- 구글 어시스턴트 예제
 - ▣ 파이썬 확장 라이브러리 설치 경로 아래 두개의 폴더에 위치
 - ▣ 자신의 응용프로그램에 Google Assistant API를 적용할 때는 이들을 분석

구글 어시스턴트 SDK 설치

- ▣ grpc 폴더: gRPC 서비스 기반 Google Assistant 클라이언트
 - pushtotalk.py: 키 입력이 있을 때마다 Google Assistant 실행
 - 한국어를 포함한 다국어 지원
 - /usr/local/bin/googlesamples-assistant-pushtotalk에서 호출

구글 어시스턴트 SDK 예제

- 구글 어시스턴트 SDK 예제
 - ▣ Google Assistant API의 사용법을 보여주는 응용프로그램들
 - ▣ 라이브러리와 gRPC 서비스 버전이 있음

서비스 기반 응용프로그램

- **인사없이 google-assistant-pushtotalk 실행**
 - ▣ 로컬에 저장된 모델 ID와 프로젝트 ID를 찾아 인스턴스를 만듦
 - ▣ 디폴트 마이크와 스피커를 초기화하고 키 입력이 있을 때까지 대기
 - ▣ 키가 입력되면 나머지 절차는 google-assistant-pushtotalk 와 동일
 - ▣ 스피커로 결과가 출력한 뒤에는 다시 키 입력 대기 상태

```
INFO:root:Connecting to embeddedassistent.googleapis.com
INFO:root:Using device model soda-fde1c-pop-m07mwj and device id 69fddfaa2-b9ab-11e9-9ae9-
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript to user request: "안녕하세요".
INFO:root:Transcript to user request: "안녕하세요 나".
INFO:root:Transcript to user request: "안녕하세요".
INFO:root:Transcript to user request: "안녕하세요 알러".
INFO:root:Transcript to user request: "안녕하세요 알러 주세요".
INFO:root:Transcript to user request: "안녕하세요 알러 줘".
INFO:root:Transcript to user request: "안녕하세요 알러 주세요".
INFO:root:Transcript to user request: "안녕하세요 알러 주세요".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript to user request: "안녕하세요 알러 주세요".
INFO:root:Transcript to user request: "안녕하세요 알러 주세요".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to sned a new request...
```

Google Assistant API 활용

- Pop 라이브러리의 popAssist 모듈
 - ▣ 옛지 디바이스에서 사용자가 좀 더 쉽게 Google Assistant API 사용 지원
 - ▣ create_conversation_stream() 메소드
 - 타기의 마이크와 스피커를 제어하는 ConversationStream 객체를 생성
 - 내부에서는 PortAudio 라이브러리의 파이썬 바인더인 sounddevice를 사용
 - ConversationStream 객체는 오디오 소스로 마이크, 오디오 싱크로 스피커 사용
 - 응답으로 받은 오디오 데이터는 volume_percentage 프로퍼티에 의해 스케일이 조정된 상태로 출력
 - create_conversation_stream() 메소드
 - ConversationStream 객체 반환
 - 주로 인자 없이(기본값 사용) 호출

Google Assistant API 활용

- `create_conversation_stream(audio_sample_rate=16000, audio_sample_width=2, audio_block_size=6400, audio_flush_size=25600, audio_iter_size=3200):`
ConversationStream 객체를 만들어 반환
- `audio_sample_rate`: 샘플 비율. 기본값은 16000
- `audio_sample_width`: 바이트 단위의 각 샘플 크기. 기본값인 2 이상 허용
- `audio_block_size`: 바이트 단위의 각 읽기 및 쓰기 작업의 크기. 기본값은 6400
- `audio_flush_size`: 플러시 동작 때 만들어진 바이트 단위 묶음 데이터 크기. 기본값은 25600
- `audio_iter_size`: 각 반복마다 읽을 데이터 크기 (바이트 단위). 기본값은 3200

Google Assistant API 활용

▣ create_device_handler() 메소드

- DeviceRequestHandler 객체를 반환
- 이를 이용해 Actions on Google에서 정의한 사용자 액션 처리
- 주로 생성자와 Assistant를 실행하는 assist() 메소드 사용
- 이때 이 둘에 공통적으로 적용되는 conversation_stream 인자는 서로 배타적
 - 안 쪽에만 ConversationStream 객체 전달

Google Assistant API 활용

- `GAssistant(conversation_stream=None, local_device_handler=None, google_device_handler=None, lang="ko-KR", display=False):`

GAssistant 객체 생성

- `conversation_stream`: 음성을 인식하고 결과를 출력할 `ConversationStream` 객체.
 - `None`(기본값)일 때는 `assist()` 메소드의 인자에 전달해야 함
- `local_device_handler`: 로컬 사용자 액션 핸들러. 기본값은 `None`
- `google_device_handler`: Actions on Google 서비스 연계 액션 핸들러 객체. 기본값은 `None`
- `lang`: 사용 언어. "ko-KR" (한국어), "en-US" (미국식 영어) 등. 기본값은 "ko-KR"
 - <https://developers.google.com/actions/localization/languages-locales> 참조
- `display`: 결과를 화면에 출력할 때 사용하는 웹 브라우저 객체. 기본값은 `False`

Google Assistant API 활용

- `conversation_stream_close()`: `ConversationStream` 객체의 오디오 접근을 닫음
 - 주로 `ConversationStream` 객체의 `Close()` 메소드 직접 호출
- `get_device_id()`: `DeviceRequestHandler` 객체를 만들 때 필요한 `device_id` 반환
- `get_device_handler()`: 현재 `GAssistant` 객체에 설정된 `DeviceRequestHandler` 객체 반환
- `assist(rec_request_handler=None, resp_handler=None, conversation_stream=None)`:

Assistant 실행

- `rec_request_handler`: 음성 인식을 시작할 때 호출되는 사용자 메소드. 기본값은 `None`
- `resp_handler`: 응답을 출력한 후 호출되는 사용자 메소드. 기본값은 `None`
- `conversation_stream`: `ConversationStream` 객체
 - `None`(기본값)일 때는 생성자의 인자로 전달해야 함

Google Assistant API 활용

▣ ConversationStream 객체

- GAssistant 객체 내부에서 오디오 접근을 위해 사용
- 몇 가지 기능은 사용자가 호출 가능
- volume_percentage: 볼륨 조절 프로퍼티
 - volume_percentage = <n>: 현재 볼륨 설정. 1 ~ 100
 - volume = volume_percentage: 현재 볼륨 반환
- close(): 오디오 자원 해제

구글 어시스턴트 실행

- 구글 어시스턴트를 일괄성으로 실행

- PopAssist의 GAssistant 클래스를 이용

- ConversationStream 객체는 create_conversation_stream() 메소드로 만듦

- 옛지 디바이스는 반드시 인터넷에 연결되어 있어야함

```
01:         from popAssist import *
02:
03:         stream = create_conversation_stream()
04:         ga = GAssistant(stream)
05:
06:         print("Taking about...")
07:         ga.assist()
08:
09:         print("Bye...")
10:         stream.close()
```

구글 어시스턴트 실행

- ▣ 프로그램을 실행한 후 “Taking about...” 메시지가 출력
- ▣ “오늘 날씨가 어때?” 와 같은 음성 명령을 입력
- ▣ Google Cloud Platform에서 가져온 결과를 출력한 후 종료
- ▣ 10초 동안 소리 입력이 없어도 종료

```
soda@soda:~ $ python3 code111.py ↵
```

```
Taking about... ↵
```

```
Bye... ↵
```

음성 명령 처리 ↵

구글 어시스턴트 실행

- ▣ ConversationStream 객체의 volume_percentage 프로퍼티 이용
 - 값의 범위는 1 ~ 100
 - 응답에 데이터의 스케일을 늘리거나 줄여 스피커 볼륨 조절 효과를 볼 수 있음

```
01:         from popAssist import *
02:
03:         stream = create_conversation_stream()
04:         ga = GAssistant(stream)
05:
06:         print("Current volume: %d"%(stream.volume_percentage))
07:         stream.volume_percentage = 100 #1 ~ 100 사이 값 사용
08:
09:         print("Taking about...")
10:         ga.assist()
11:
12:         print("Bye...")
13:         stream.close()
```

구글 어시스턴트 실행

- ▣ 응답 스케일은 ConversationStream 객체가 만들어질 때 50으로 설정

```
soda@soda:~ $ python3 code112.py
Current volume: 50
Taking about...
Bye...
```

구글 어시스턴트 실행

▣ assist() 메소드의 rec_request_handler와 resp_handler 인자

- 사용자 메소드를 전달하면 음성 인식이 시작되거나 응답이 완료될 때 해당 메소드를 호출

```
01:         from popAssist import *
02:
03:         stream = create_conversation_stream()
04:         ga = GAssistant(stream)
05:
06:         def onStart(): # 음성 인식을 시작할 때 호출
07:             print(">>> start recording...")
08:
09:         def onStop(): # 응답의 출력이 완료되면 호출
10:             print(">>> stop response...")
11:
12:         ga.assist(onStart, onStop)
13:         stream.close()
```

구글 어시스턴트 실행

- ▣ 프로그램 실행 후 음성 인식을 시작할 수 있을 때 onStart() 메소드 호출
- ▣ 응답이 완료되면 onStop() 메소드 호출

```
soda@soda:~ $ python3 code113.py  
>>> start recording...  
>>> stop response...
```

구글 어시스턴트 실행

- ▣ 응답이 완료될 때 호출되는 사용자 메소드에서 다시 assist() 메소드를 호출 시
 - 계속해서 구글 어시스턴트를 실행 가능
 - 응답은 내부 스레드에서 처리
 - assist() 메소드가 실행될 때 프로그램이 종료하지 않도록 주의

```
01: from popAssist import *
02:
03: stream = create_conversation_stream()
04: ga = GAssistant(stream)
05:
06: try:
07:     def onStart():
08:         print(">>> start recording...")
09:
```

```
10:     def onStop():
11:         ga.assist(onStart, onStop) #응답의 처리가 완료되면 다시 assist() 호출
12:
13:     ga.assist(onStart, onStop)
14:     while True:
15:         print("main work...")
16:         time.sleep(2)
17: except KeyboardInterrupt:
18:     stream.close()
```

구글 어시스턴트 실행

- ▣ 프로그램을 실행한 후 시작 메시지가 출력되면 음성 인식이 가능한 상태
 - 내부 스레드를 통해 응답을 처리
 - 묵음이 10초 이상 지속되어 음성 인식을 다시 시작하거나 응답을 출력 중에도 계속 실행

```
soda@soda:~ $ python3 code114.py
>>> start recording...
main work...
>>> start recording...
main work...
main work...
main work...
```

사용자 장치 액션

□ 사용자 장치 액션

- ▣ 음성 인식 결과를 옯지 디바이스의 특정 명령에 반영하는 행위
- ▣ 사용자 계정과 연계된 Actions on Google 서비스를 사용
- ▣ 반드시 “구글 어시스턴트 사용 인증” 을 수행한 후 진행

□ Actions on Google에서 관리하는 액션

- ▣ 특성 : 밝기 제어, 장치를 켜고 끄는 것과 같이 구글에서 미리 지정한 액션
- ▣ Actions SDK와 연계되는 사용자 장치 액션 : 사용자가 추가한 액션

사용자 장치 액션

- 옛지 디바이스가 액션 패키지를 정의한 후 Actions on Google에 등록
 - ▣ Google Assistant 서버는 음성 인식 결과를 액션 패키지와 비교
 - ▣ 일치하는 명령을 옛지 디바이스에 반환
 - ▣ 옛지 디바이스는 그에 대응하는 사용자 코드 실행

사용자 장치 액션

- JSON 포맷으로 정의한 액션 패키지에 포함되는 내용
 - ▣ 사용자 질의와 일치시키려는 패턴
 - ▣ 일치하는 질의와 연결할 사용자 장치 액션
 - ▣ 장치가 액션을 지원하면 사용자에게 전달할 응답
 - ▣ 매개 변수와 함께 장치로 전송되는 명령 이름

사용자 장치 액션

▣ 확장자가 .json인 JSON 포맷의 액션 패키지 구조

```
01:      {  
02:        "manifest": {  
03:          object(Manifest)  
04:        },  
05:        "accountLinking": {  
06:          object(AccountLinking)  
07:        },  
08:        "actions": [  
09:          {  
10:            object(Action)  
11:          }  
12:        ],
```

```
13:        "types": [  
14:          {  
15:            object(Type)  
16:          }  
17:        ],  
18:        "conversations": {  
19:          string: {  
20:            object(ConversationFulfillment)  
21:          },  
22:          ...  
23:        },24:      }
```


사용자 장치 액션

▣ 액션 패키지의 항목 의미

manifest	<code>object(Manifest)</code> 앱의 세부 정보
accountLinking	<code>object(AccountLinking)</code> 앱의 계정 연결에 대한 세부 정보
actions[]	<code>object(Action)</code> 앱이 처리할 수 있는 작업 목록.
types[]	<code>object(Type)</code> 개발자가 정의한 타입 목록
conversations	<code>map (key: string, value: object(ConversationFulfillment))</code> 액션 간에 공유할 수 있는 대화 맵 "key": value 쌍 목록으로 구성 예: { "name": "wrench", "mass": "1.3kg", "count": "3" }.

사용자 장치 액션

□ 음성 명령으로 사진을 찍는 예제

□ 사진을 찍는 액션 패키지를 picture.json 파일로 정의

```
01: {  
02:   "manifest": {  
03:     "displayName": "Take Picture",  
04:     "invocationName": "Take Picture",  
05:     "category": "PRODUCTIVITY"  
06:   },  
07:   "locale": "ko",  
08:   "actions": [  
09:     {  
10:       "name": "TaskPicture",  
11:       "availability": {  
12:         "deviceClasses": [ { "assistantSdkDevice": {} } ]  
13:       },  
14:       "intent": {  
15:         "name": "com.example.intents.TakePicture",  
16:         "trigger": { "queryPatterns": [ "사진 찍어" ] }  
17:       },
```

```
18:     "fulfillment": {  
19:       "staticFulfillment": {  
20:         "templatedResponse": {  
21:           "items": [  
22:             {  
23:               "simpleResponse": { "textToSpeech": "사진을 찍습니다" }  
24:             },  
25:             {  
26:               "deviceExecution": { "command": "TakePicture" }  
27:             }  
28:           ]  
29:         }  
30:       }  
31:     }  
32:   }  
33: ]  
34: }
```

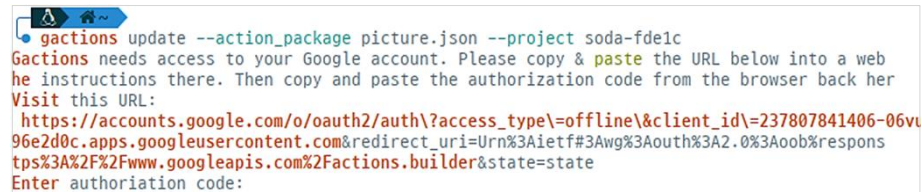
사용자 장치 액션

- 한국어 음성 인식 사용
 - “locale” 값을 “ko” 로 지정
 - “intent” 에 포함된 “trigger” 의 “queryPatterns” 값
 - 사진을 찍는 구문 패턴으로 여러 개일 수 있음
 - “deviceExecution” 의 “command” 값
 - 옛지 디바이스로 반환되는 명령 내용

사용자 장치 액션

- ▣ 정의한 액션 패키지는 gactions 명령으로 Actions on Google에 등록
- ▣ gactions 명령으로 액션 패키지 등록을 요청
- ▣ 이를 허용하도록 프로젝트 ID를 함께 전달

```
gactions update --action_package <action.json> --project <project-id>
```



```
gactions update --action_package picture.json --project soda-fde1c
Gactions needs access to your Google account. Please copy & paste the URL below into a web
browser. Then copy and paste the authorization code from the browser back here.
Visit this URL:
https://accounts.google.com/o/oauth2/auth?access_type=offline&client_id=237807841406-06vu
96e2d0c.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Aauth%3A2.0%3Aaob%3Arespon
se%3A%2F%2Fwww.googleapis.com%2Factions.builder&state=state
Enter authorization code:
```

사용자 장치 액션

- ▣ 처음 액션 패키지를 등록하면 인증 코드를 얻을 수 있는 URL 표시
- ▣ 작업을 잠시 멈추고 크롬 브라우저에 붙여넣어 이동
- ▣ 프로젝트 계정을 선택하면 Assistant CLI 사용 여부에 “허용” 선택

Assistant CLI이(가) 내 Google 계
정에 액세스하려고 합니다



이렇게 하면 Assistant CLI에서 다음 작업을 할 수 있
습니다.

Google에서 작업 보기 및 관리 ⓘ

Assistant CLI 앱을 신뢰할 수 있는지 확인

민감한 정보가 이 사이트 또는 앱과 공유될 수 있습니다.
Assistant CLI의 서비스 약관 및 개인정보처리방침을 검
토하여 내 데이터가 어떻게 처리되는지 알아보세요. 언
제든지 Google 계정에서 액세스 권한을 확인하고 삭제
할 수 있습니다.

[타사 앱 권한 부여에 관한 위험 알아보기](#)

취소

허용

사용자 장치 액션

- 인증 코드가 표시되면 이를 복사
- 멈췄던 작업으로 돌아가 “Enter authorization code:” 줄 아래 붙여넣기
- <Enter>를 눌러 액션 패키지 등록 완료
- 인증이 완료되면 현재 경로에 creds.data 파일이 만들어짐
- 다음에 수정된 내용을 반영할 때는 인증 절차 생략

```
gactions update --action_package picture.json --project soda-fdelc
Gactions needs access to your Google account. Please copy & paste the URL below into a web browser. Then copy and paste the authorization code from the browser back here.
Visit this URL:
https://accounts.google.com/o/oauth2/auth?access_type=offline&client_id=237807841406-96e2d0c.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Aauth%3A2.0%3Aaob%3Aresponse%3A%2F%2Fwww.googleapis.com%2Factions.builder&state=state
Enter authorization code:
4/nQGlaz7UE_TIc2eu4jpPYWJJ8XRN84_nov08ydHTEZc2fLnLhLCxWRrE
Your app for the Assistant for project soda-fdelc was successfully updated with your changes. You can now use the app on Google Home. To finish registering your app and submit it for review at https://console.cloud.google.com/project/soda-fdelc/overview
```

사용자 장치 액션

- ▣ Actions on Google에 등록된 액션 목록은 다음 주소에서 확인 가능
 - <https://console.actions.google.com/project/<project-id>/overview>
 - “Build your Action” > “Add Actions” 선택

Korean		Modify languages
Actions Below is a list of Actions you have built		
Test		
Name	Fulfillment type	Fulfillment tool
com.example.intents.TakePicture	Conversational	Actions SDK >

사용자 장치 액션

- ▣ 목록에서 해당 액션을 선택
 - 세부 내용을 확인 가능
 - 액션 호출 구문이나 언어 변경과 같은 수정 가능

The screenshot shows the Google Assistant Actions console interface for the action 'com.example.intents.TakePicture'. The interface is in Korean. At the top, there's a header with 'Actions' and a back arrow. Below the header, the action name 'com.example.intents.TakePicture' is displayed, along with a 'Save' button. The main content area is divided into three sections: 'Action invocation phrases', 'Parameters', and 'Fulfillment'. The 'Action invocation phrases' section is expanded, showing a list of phrases with '사진 찍어' (Take a picture) as the first entry. The 'Parameters' section is collapsed, showing 'No parameters added for this Action.' The 'Fulfillment' section is also collapsed, showing 'Conversational' as the fulfillment type. A 'Modify languages' link is visible in the top right corner of the main content area.

Actions

← com.example.intents.TakePicture [Save](#)

[Korean](#) [Modify languages](#)

Action invocation phrases ^

Invocation phrases help users find and implicitly or explicitly invoke your Action.

사진 찍어

Parameters ^

No parameters added for this Action.

Fulfillment Conversational v

사용자 장치 액션

- ▣ Actions on Google에 사진을 찍는 액션 패키지를 등록완료 후
 - 사진을 찍는 사용자 코드를 구현
 - 아래 코드를 실행하면, 카메라 데이터를 picture.png라는 이름의 파일에 저장

```
from pop import Camera  
cam = Camera(width=300, height=300)  
  
cv2.imwrite("picture.png", cam.value)
```

사용자 장치 액션

□ 사용자 메소드 등록

- “deviceExecution” 의 “command” 가 일치할 때 실행
 - Actions on Google에서 수신한 액션 항목
- create_device_handler() 메소드로 DeviceRequestHandler 객체를 만듦
- 만들어진 객체를 데커레이터로 사용자 메소드를 정의
- 자동으로 DeviceRequestHandler 객체에 추가
- 동일한 방법으로 다른 액션에 대한 사용자 메소드를 정의, 추가 가능

사용자 장치 액션

```
01:         device_handler = create_device_handler()
02:         cam = Camera(width=300, height=300)
03:
04:         @device_handler.command("TakePicture")
05:         def takePicture():
06:             print("Take picture...")
07:             cv2.imwrite("picture.png", cam.value)
```

사용자 장치 액션

▣ GAssistant 객체

- google_device_handler 인자에 사용자 메소드가 추가된 DeviceRequestHandler 객체 전달
- assist() 메소드가 호출될 때마다 내부에서 음성 인식에 대한 액션 응답을 분석
- 실행할 사용자 메소드가 있으면 호출

사용자 장치 액션

```
01: from popAssist import *
02: import subprocess
03: import time
04:
05: device_handler = create_device_handler()
06: stream = create_conversation_stream()
07: cam = Camera(width=300, height=300)
08:
09: @device_handler.command("TakePicture")
10: def takePicture():
11:     print("Take picture...")
12:     cv2.imwrite("picture.png", cam.value)
13:
14: ga = GAssistant(stream, google_device_handler=device_handler)
```

```
16: try:
17:     def onStart():
18:         print(">>> Start recording....")
19:
20:     def onStop():
21:         ga.assist(onStart, onStop)
22:
23:     ga.assist(onStart, onStop)
24:
25:     while True:
26:         time.sleep(1)
27: except KeyboardInterrupt:
28:     stream.close()
```

사용자 장치 액션

- ▣ 프로그램을 실행
- ▣ Start recording...” 메시지가 출력되면 음성 인식 시작
- ▣ “사진 찍어”란 음성이 인식되면 “사진을 찍습니다”란 응답 출력
 - AIoT Home에 장착된 카메라로 사진을 찍어 연재 경로에 저장

```
soda@soda:~ $ python3 code117.py
>>> Start recording...
Take picture...
>>> Start recording...
```

GAssistant 기반 사용자 장치 액션

- GAssistant 기반 사용자 장치 액션
 - ▣ GAssistant 클래스에 자체적으로 사용자 장치 액션을 처리하는 기능
 - ▣ 음성을 인식할 때마다 수신한 텍스트를 인자로 사용자 메소드를 호출
 - GAssistant 객체를 만들 때 `local_device_handler` 인자에 사용자 메소드를 전달
 - ▣ 사용자 메소드는 이 문자열을 분석해 원하는 작업을 수행
 - ▣ 사용자 메소드가 `True`를 반환하면 수신한 응답 파일 재생 안함

GAssistant 기반 사용자 장치 액션

```
01:         from popAssist import *
02:         import subprocess
03:
04:         def userAction(text): # 로컬 사용자 액션 핸들러
05:             print(text)
06:
07:             return True
08:
09:         stream = create_conversation_stream()
10:         ga = GAssistant(stream, local_device_handler=userAction)
11:
12:         try:
13:             def onStart():
14:                 print(">>> Start recording....")
15:
16:             while True:
17:                 ga.assist(onStart)
18:         except KeyboardInterrupt:
19:             stream.close()
```

GAssistant 기반 사용자 장치 액션

- ▣ 프로그램을 실행
- ▣ ‘>>> Start recording...’ 메시지가 출력되면 음성 인식 시작
- ▣ 인식된 내용은 텍스트로 출력

```
soda@soda:~ $ python3 code118.py
>>> Start recording...
오늘 며칠이야
>>> Start recording...
오늘 무슨 요일이지
>>> Start recording...
```

GAssistant 기반 사용자 장치 액션

□ GAssistant 기반 사용자 장치 액션으로 음성 명령으로 사진 찍기

□ ‘사진’ 과 ‘찍어’ 가 순서대로만 나오면 나머지 단어와 관계없이 사진 찍기

```
01: from popAssist import *
02: from pop import Camera
03: import cv2
04: import subprocess
05:
06: cam = Camera(width=300, height=300)
07:
08: def userAction(text):
09:     action = False
10:
11:     r = text.find("사진")
12:     if r != -1 and text.find("찍어", r) != -1:
13:         print("Take picture...")
14:         cv2.imwrite("picture.png", cam.value)
15:
```

```
16:     action = True
17:
18:     return action
19:
20: stream = create_conversation_stream()
21: ga = GAssistant(stream, local_device_handler=userAction)
22:
23: try:
24:     def onStart():
25:         print(">>> Start recording....")
26:
27:     while True:
28:         ga.assist(onStart)
29: except KeyboardInterrupt:
30:     stream.close()
```

음성으로 AIoT AutoCar 제어

- 애플워치가 인식되면 AIoT AutoCar의 모터를 제어하는 예제
 - ▣ 음성으로 ‘전진’ , ‘후진’ , ‘정지’ 명령어에 따라 모터 제어
 - ▣ GAssistant 객체를 만들 때 로컬 사용자 장치 액션을 실행하도록 설정
 - ▣ GAssistant 객체의 assist() 메소드 호출하도록 설정
 - ▣ userAction() 메소드가 호출될 때 인자로 받은 문자열과 비교해 모터 구현

음성으로 AIoT AutoCar 제어

```
01: from pop import Pilot
02: from popAssist import *
03:
04: Car = Pilot.AutoCar()
05:
06: Car.stop()
07:
08: def userAction(text):
09:     action = False
10:
11:     print(text)
12:
13:     if text.find("전진") != -1:
14:         Car.forward()
15:         action = True
16:     elif text.find("후진") != -1:
17:         Car.backward()
18:         action = True
19:     elif text.find("정지") != -1:
```

```
20:         Car.stop()
21:         action = True
22:
23:     return action
24:
25: stream = create_conversation_stream()
26: ga = GAssistant(stream, local_device_handler=userAction)
27:
28: try:
29:
30:     def onStart():
31:         print(">>> Start recording....")
32:
33:     while True:
34:         ga.assist(onStart)
35:
36: except KeyboardInterrupt:
37:     stream.close()stream.close()
```

음성으로 AIoT AutoCar 제어

- ▣ 프로그램 실행
- ▣ ‘Start recording’ 메시지가 출력되면 음성 인식 시작
- ▣ ‘전진’ , ‘후진’ , ‘정지’ 음성 명령으로 모터 제어

내용 정리

- 리눅스 커널은 오디오 하위 시스템을 제공
- 오디오 하위 시스템
 - ▣ 다양한 사운드 카드를 추상화해 표준화된 사운드 인터페이스를 제공.
 - ▣ ALSA 사용

내용 정리

- PyAudio
 - ▣ PortAudio에 대한 파이썬 버전
 - ▣ 파이썬으로 WAVE 파일을 재생하거나 녹음하는 프로그램을 작성 가능
 - ▣ 블로킹과 논블로킹 모드 지원
 - ▣ WAV 파일만 지원
- TTS : 텍스트를 음성으로 변환할 때 사용

내용 정리

- PyAudio
 - ▣ PortAudio에 대한 파이썬 버전
 - ▣ 파이썬으로 WAVE 파일을 재생하거나 녹음하는 프로그램을 작성 가능
 - ▣ 블로킹과 논블로킹 모드 지원
 - ▣ WAV 파일만 지원
- TTS : 텍스트를 음성으로 변환할 때 사용

내용 정리

□ gTTS

- ▣ 클라우드 기반 구글 번역기에 포함된 텍스트 음성 변환 API에 대한 파이썬 라이브러리
- ▣ 명령행 기반 툴도 함께 제공
- ▣ 구글 클라우드 서비스로 텍스트를 전달하면 음성합성 결과를 mp3 파일로 변환

내용 정리

- 구글 어시스턴트
 - ▣ 구글 클라우드에서 제공하는 AI 기반 음성 인식 및 행동 실행 서비스
- 사용자 장치 액션
 - ▣ 음성 인식 결과를 옯지 디바이스의 특정 명령에 반영

연습문제

- 문제 4. 다음 코드는 wave 라이브러리를 이용하여 오디오 파일을 읽어 변수에 저장하는 코드입니다. 질문을 읽고 답해보세요.

```
01: import wave
02:
03: w = wave.open( )
04: data = w.readframes(w.getnframes())
05: w.close()
```

연습문제

- ▣ A. 'sound.wav' 라는 오디오 파일을 읽으려 할 때 빈 칸에 들어갈 코드를 작성해보세요.
- ▣ B. getnframes() 메소드의 역할이 무엇인지 답해보세요.
- ▣ C. PyAudio 라이브러리를 이용해 읽어온 오디오 데이터를 재생하는 코드를 작성해보세요.

연습문제

- 문제 5. gTTS 라이브러리를 사용하여 ‘Audio output’ 이라는 문장을 음성 파일로 저장하는 코드를 작성해보세요.

연습문제

- 문제 6. 다음 코드는 popAssist 라이브러리를 이용해 음성으로 AutoCar를 제어하는 코드입니다. gTTS를 이용하여 ‘전진’ 이 인식되면 ‘전진합니다’ 와 같이 음성으로 응답을 하는 코드를 작성해보세요.

연습문제

```
01: from pop import Pilot
02: from popAssist import *
03:
04: Car = Pilot.AutoCar()
05:
06: Car.stop()
07:
08: def userAction(text):
09:     action = False
10:
11:     print(text)
12:
13:     if text.find("정지") != -1:
14:         Car.forward()
15:         action = True
16:     elif text.find("후진") != -1:
17:         Car.backward()
18:         action = True
```

```
19:     elif text.find("정지") != -1:
20:         Car.stop()
21:         action = True
22:
23:     return action
24:
25: stream = create_conversation_stream()
26: ga = GAssistant(stream,
                  local_device_handler=userAction)
27:
28: try:
29:
30:     def onStart():
31:         print(">>> Start recording....")
32:
33:     while True:
34:         ga.assist(onStart)
35:
36: except KeyboardInterrupt:
37:     stream.close()
```