

## 목 차

1. 팀 소개
2. Gantt Chart
3. 개발 배경
4. 요구사항 분석
5. 스토리 보드
6. 설계 내용

Smart Speaker

GO



**1334609 최재훈**

팀장  
세부 기능 담당  
문서 작성



**1334085 손민성**

자료 수집 및 분석  
데이터 분류










**1334402 서진성**

API 분석 및 수정  
발표

# Gantt Chart

P2

	4월	5월			6월	
	4.29 ~ 5.3	5.4 ~ 5.10	5.11 ~ 5.17	5.18 ~ 5.24	5.25 ~ 6.1	6.2 ~ 6.8
사 전 조 사						
요 구 분 석						
제 품 설 계						
중 간 점 검						
세 부 기 능						
테 스 트						
발 표 준 비						

## 국내 스마트홈 시장 후끈... “누가 누가 잘하나”

통신사부터 ICT, 건설사까지

최진홍 기자 | rgdsz@econovill.com | 승인 2018.06.07 08:45:00

[이코노믹리뷰=최진홍 기자]국내 스마트홈 시장의 행보가 빨라지고 있다. 인공지능을 기반 플랫폼이 비즈니스 모델을 모색하며 그 대상을 ‘거주의 공간’으로 확장했기 때문이다. 모든 이상의 가치를 보여주는 시도는 ICT 기업부터 통신사, 심지어 전통의 건설사까지 관심



## [AI시대 성큼] "3년내 넷중 셋은 AI스피커 사용"

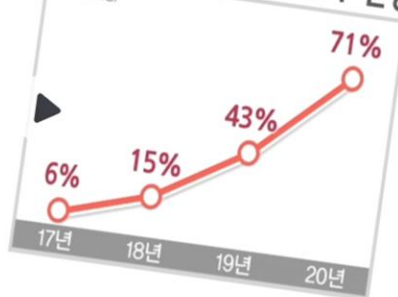
입력 2018-04-05 17:12 수정 2018-04-05 17:27

생활 속 파고든 AI 스피커



### 국내 AI스피커 보유 가구수 전망

(자료: 통계청)



많이 본 뉴스

- 1 [급성증]
- 2 [헬스케어]
- 3 포브스



12 이재명

투데

1 주식초

아마존 Echo

구글 홈

SK텔레콤 NUGU

KT 기가지니

카카오 카카오톡

네이버 클로바

⋮

## OS

- Raspberry Pi – Raspbian
- Windows 10

## 개발 도구

- Raspberry Pi – python3
- USB 마이크
- HDMI 케이블
- 스피커
- Micro SD 카드

## API

- Google – Google Cloud  
Google Cloud Speech
- Konlpys – KoNLPy's Twitter
- Naver – Clova Speech Synthesis  
Neural Machine Translation
- T – Weather Planet API

### 음성 인식률

- ✓ 한국어를 정확하게 인식할 수 있어야 한다.

### 동의어 문장 인식

- ✓ 다른 형태의 문장이지만 뜻이 같다면  
같은 정보를 제공해야 한다.

### 응답 속도

- ✓ 정보를 제공하기까지의 시간은 5초 이내로 한다.

### 인식 거리

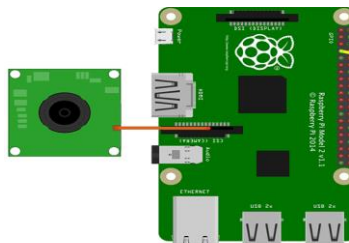
- ✓ 1M 정도의 거리에서도 인식을 가능하게 한다.

### 잡음의 영향

- ✓ 주위 잡음이 인식에 영향을 미치지 않는다.



사람의 음성



Raspberry Pi



Google Cloud Speech



KoNLPy



명령에 따른 동작

기본적인 의사소통  
시간정보  
날짜정보  
날씨정보  
일정 Save & Load  
KR - EN 번역



Clova<sup>®</sup>

Clova Speech Synthesis

## Raspberry Pi 초기 설정

### ▶ 스피커 설정 - 초기설치

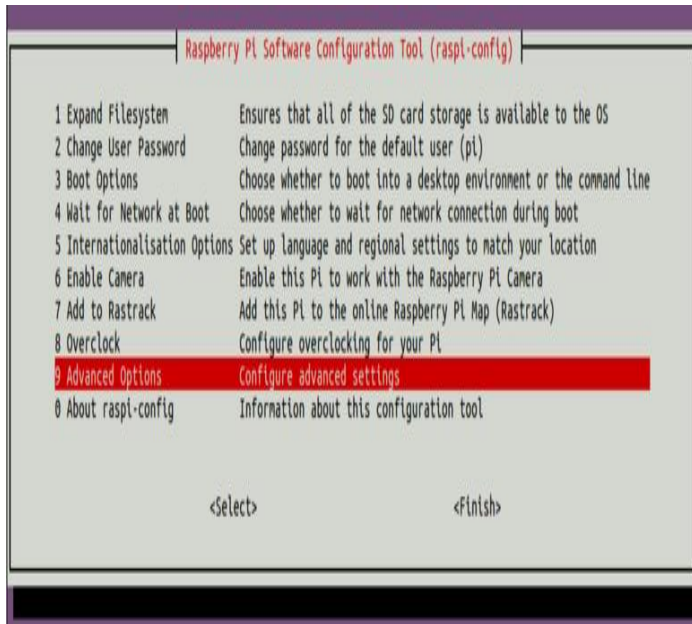
```
sudo apt install portaudio19-dev  
pip install pyaudio
```



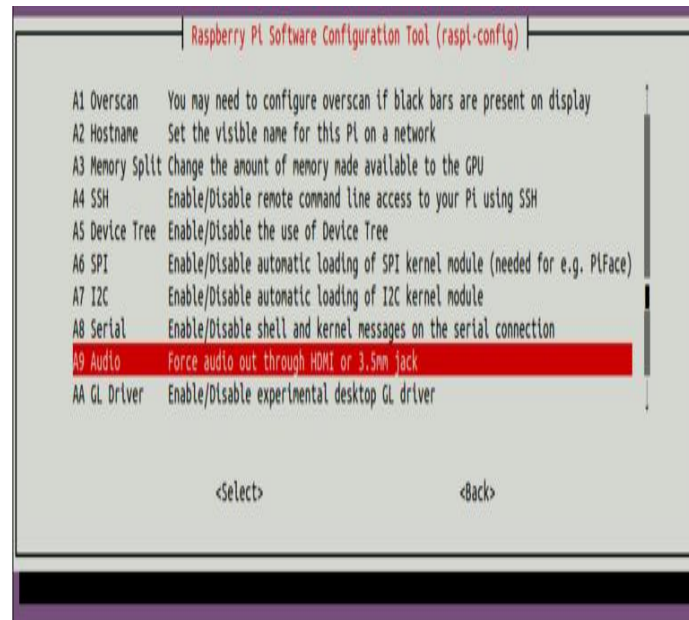
## Raspberry Pi 초기 설정

### ▶ 스피커 설정 - 스피커로 출력

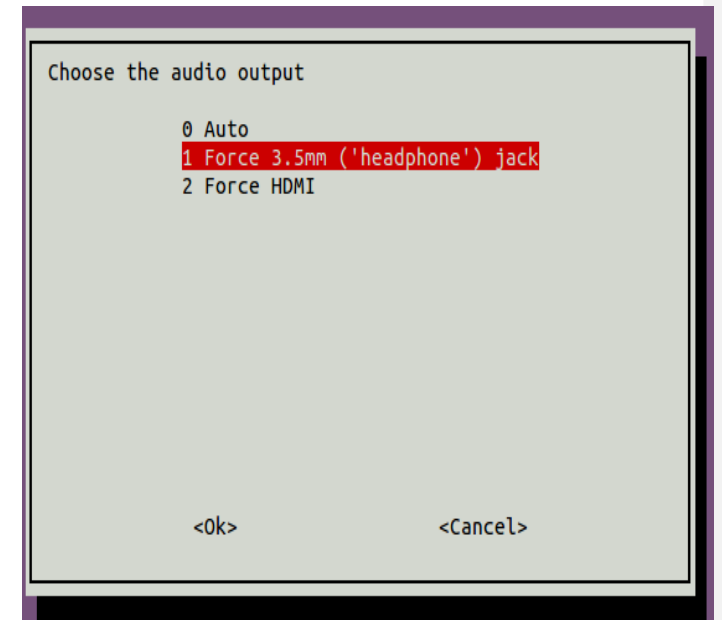
```
sudo apt-get install raspi-config  
sudo raspi-config
```



Advanced Options



A9 Audio

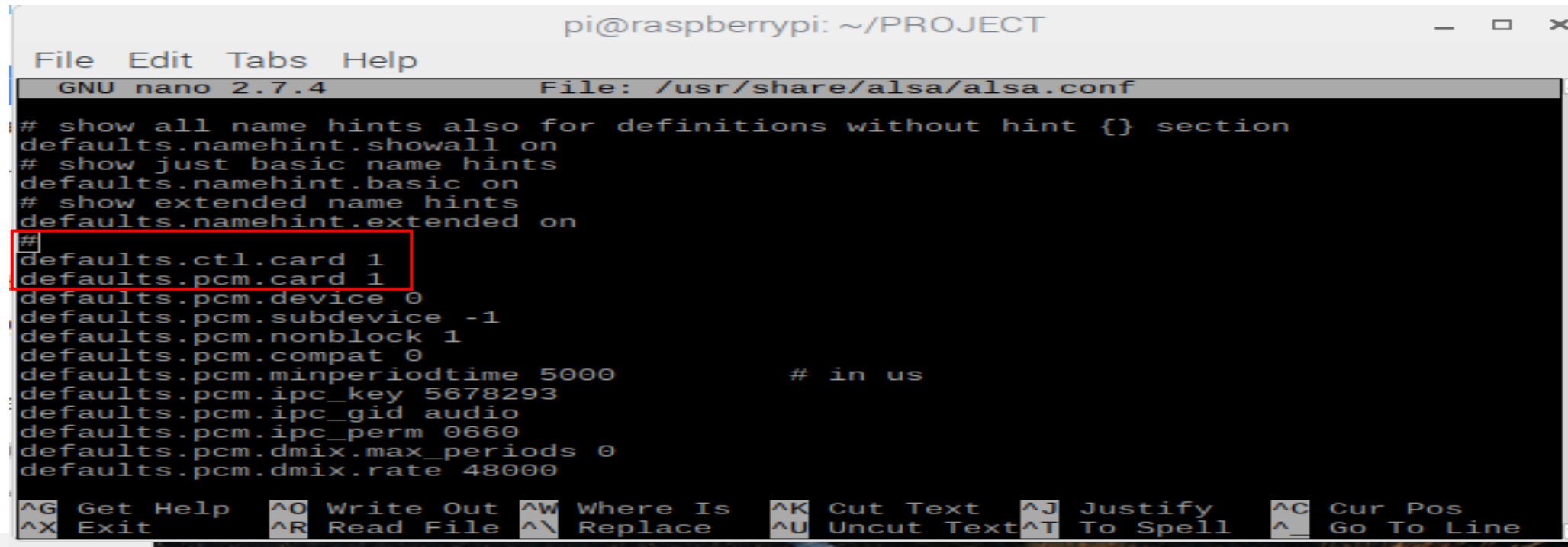


Force 3.5 mm jack

## Raspberry Pi 초기 설정

### ▶ 스피커 설정 - USB 마이크

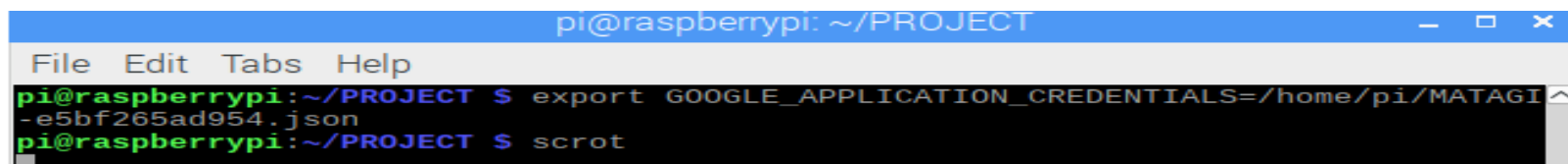
```
sudo nano /usr/share/alsa/alsa.conf
```



```
pi@raspberrypi: ~/PROJECT
File Edit Tabs Help
GNU nano 2.7.4 File: /usr/share/alsa/alsa.conf
# show all name hints also for definitions without hint {} section
defaults.namehint.showall on
# show just basic name hints
defaults.namehint.basic on
# show extended name hints
defaults.namehint.extended on
#
defaults.ctl.card 1
defaults.pcm.card 1
defaults.pcm.device 0
defaults.pcm.subdevice -1
defaults.pcm.nonblock 1
defaults.pcm.compat 0
defaults.pcm.minperiodtime 5000 # in us
defaults.pcm.ipc_key 5678293
defaults.pcm.ipc_gid audio
defaults.pcm.ipc_perm 0660
defaults.pcm.dmix.max_periods 0
defaults.pcm.dmix.rate 48000
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

## 음성 인식

### ▶ API 인증키 입력



```
pi@raspberrypi: ~/PROJECT
File Edit Tabs Help
pi@raspberrypi:~/PROJECT $ export GOOGLE_APPLICATION_CREDENTIALS=/home/pi/MATAGI
-e5bf265ad954.json
pi@raspberrypi:~/PROJECT $ scrot
```

### ▶ 음성 인식



```
streaming.py ✕

def main():
    # See http://g.co/cloud/speech/docs/languages
    # for a list of supported languages.
    #language_code = 'en-US' # a BCP-47 language tag
    language_code = 'ko-KR' # 한국어로 변경

    client = speech.SpeechClient()
    config = types.RecognitionConfig(
        encoding=enums.RecognitionConfig.AudioEncoding.LINEAR16,
        sample_rate_hertz=RATE,
        language_code=language_code)
    streaming_config = types.StreamingRecognitionConfig(
        config=config,
        interim_results=True)

    with MicrophoneStream(RATE, CHUNK) as stream:
        audio_generator = stream.generator()
        requests = (types.StreamingRecognizeRequest(audio_content=content)
                    for content in audio_generator)

        responses = client.streaming_recognize(streaming_config, requests)

    # Now, put the transcription responses to use.
    listen_print_loop(responses)
```

## 음성 인식

## ▶ 문장에서 형태소 분리

streaming.py

```
def CommandProc(stt):  
    t = time.localtime()  
  
    # 문자 양쪽 공백 제거  
    cmd = stt.strip()  
  
    # 입력 받은 문자 화면에 표시  
    print('나 : ' + str(cmd))  
    posList = twitter.pos(str(cmd), norm=True, stem=True)  
    pprint(posList)  
  
    #명사 골라내기  
    print("명사: ")  
    NList = []  
    for i in posList:  
        if i[1] == 'Noun':  
            NList.append(i[0])  
            print(i[0])
```



```
[('이', 'Determiner'), ('것', 'Noun'), ('은', 'Josa'), ('테스트', 'Noun'), ('하다', 'Verb'), ('문장', 'Noun'), ('이다', 'Adjective'), ('.', 'Punctuation')]
```

## 의사 소통

### ▶ 기본적인 의사소통 Rule

streaming.py

```
# Rule
for word in NList:
    if word == '안녕':
        print('스피치: 반갑습니다')
        tts.play('반갑습니다.')

        return 1
    elif word == '너':
        for next_word in NList:
            if next_word == '이름':
                print('저는 마타기 입니다.')
                tts.play('저는 마타기 입니다.')
                return 1
    elif word == '오늘':
        for next_word in NList:
            if next_word == '날씨':
                weather_txt = TWeather.requestSummaryWeather(subData = 0)
                print(weather_txt)
                tts.play(weather_txt)
                return 1
    elif word == '지금':
        for next_word in NList:
            if next_word == '몇':
                for next_word in NList:
                    if next_word == '시야':
                        time_txt = "현재 " + str(t.tm_hour) + "시 " + str(t.tm_min) + "분 입니다"
                        print(time_txt)
```

## 날짜 및 시간 정보

### ▶ 시스템의 날짜와 시간 정보 Text화

```
streaming.py ✕  naverTTS.py ✕  TWeather.py ✕  Translate.py ✕  
    print( '저는 마타기 입니다. ' )  
    tts.play('저는 마타기 입니다. ')  
    return 1  
elif word == '오늘':  
    for next_word in NList:  
        if next_word == '날씨':  
            weather_txt = TWeather.requestSummaryWeather(subData = 0)  
            print(weather_txt)  
            tts.play(weather_txt)  
            return 1  
elif word == '지금':  
    for next_word in NList:  
        if next_word == '몇':  
            for next_word in NList:  
                if next_word == '시야':  
                    time_txt = "현재 " + str(t.tm_hour) + "시 " + str(t.tm_min) + "분 입니다"  
                    print(time_txt)  
                    tts.play(time_txt)  
                    return 1  
            elif next_word == '시간':  
                time_txt = "현재 " + str(t.tm_hour) + "시 " + str(t.tm_min) + "분 입니다"  
                print(time_txt)  
                tts.play(time_txt)  
                return 1
```

## T – Weather Planet API

### ▶ 받은 정보를 가공

```
streaming.py ✕  naverTTS.py ✕  TWeather.py ✕  
  
def changeSkyName(txt):  
    dic = {  
        #과거  
        'Y':  
        {'01': '맑았',  
         '02': '구름조금 있었',  
         '03': '구름이 많았',  
         '04': '구름이 많았고 비 내렸',  
         '05': '구름이 많았고 눈 내렸',  
         '06': '구름이 많았고 비 또는 눈 내렸',  
         '07': '흐렸',  
         '08': '흐리고 비가 내렸',  
         '09': '흐리고 눈이 내렸',  
         '10': '흐리고 비 또는 눈 내렸',  
         '11': '흐리고 낙뢰 있었',  
         '12': '뇌우와 비이 내렸',  
         '13': '뇌우와 눈이 내렸',  
         '14': '뇌우와 비 또는 눈이 내렸'},  
        #현재  
        'A':  
        {'01': '맑',  
         '02': '구름이 조금 있',  
         '03': '구름이 많',  
         '04': '구름이 많고 비가 내리',  
         '05': '구름이 많고 눈 내리',  
         '06': '구름이 많고 비 또는 눈 내리',  
         '07': '흐리',
```

## T – Weather Planet API

### ▶ 텍스트 형태로 만들어서 Return

```
tempMin = temp + str(fTemper)

txt = ''
txt = sky_name + ' 최고 기온은 '
txt = txt + tempMax + ' 도 최저 기온은 '
txt = txt + tempMin + ' 도 ' + endofText
return txt
```



```
>>> %Run TWeather.py
```

```
오늘 하늘은 구름이 많고 비가 내립니다. 최고 기온은 29.0 도 최저 기온은 19.0 도 입니다.
```

```
>>>
```



## Naver – Neural Machine Translation

### ▶ 한국어 문장 -> 영어 문장

```
request = urllib.request.Request(url)
request.add_header("X-Naver-Client-Id",client_id)
request.add_header("X-Naver-Client-Secret",client_secret)
response = urllib.request.urlopen(request, data=data.encode("utf-8"))
rescode = response.getcode()
if(rescode==200):
    response_body = response.read()
    print(response_body.decode('utf-8'))

    #번역결과
    content = json.loads(response_body.decode('utf-8'))
    tr_txt = content.get('message')['result']['translatedText']
    print('데이터 추출 : ' + tr_txt)
    return tr_txt
else:
    print("Error Code:" + rescode)

a = '한 학기 동안 고생하셨습니다.'
translate_kr_to_en(a)
```



```
>>> %Run Translate.py
```

```
{"message":{"@type":"response","@service":"naverservice.nmt.proxy","@version":"1.0.0","result":{"srcLangType":"ko","tarLangType":"en","translatedText":"Thank you for your hard work during the semester."}}}
```

데이터 추출 : Thank you for your hard work during the semester.

```
>>>
```

## Naver – Clova Speech Synthesis

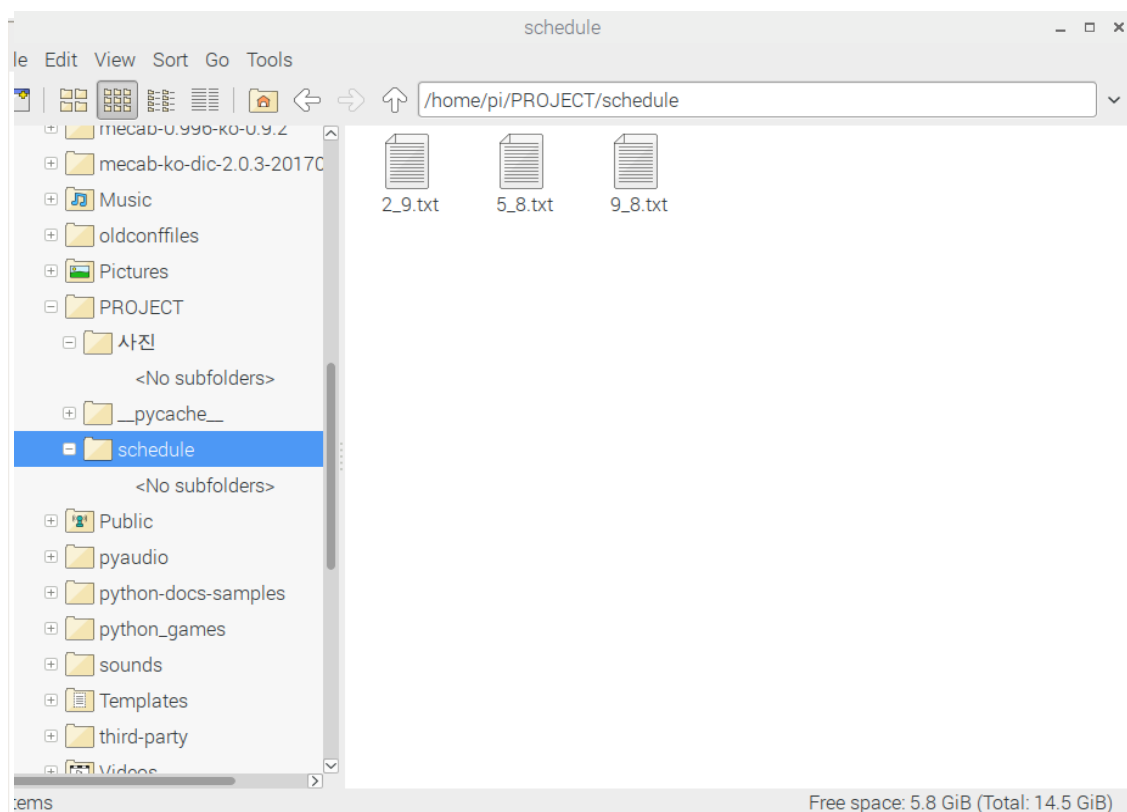
### ▶ 텍스트를 한국어 음성으로 출력

```
speakers = [  
    'mijin',      #한국어 여성  
    'jinho',      #한국어 남성  
    'clara',      #영어 여성  
    'matt',       #영어 남성  
    'yuri',       #일본어 여성  
    'shinji',     #일본어 남성  
    'meimei',     #중국어 여성  
    'liangliang', #중국어 남성  
    'jose',       #스페인어 남성  
    'carmen'      #스페인어 여성  
]
```

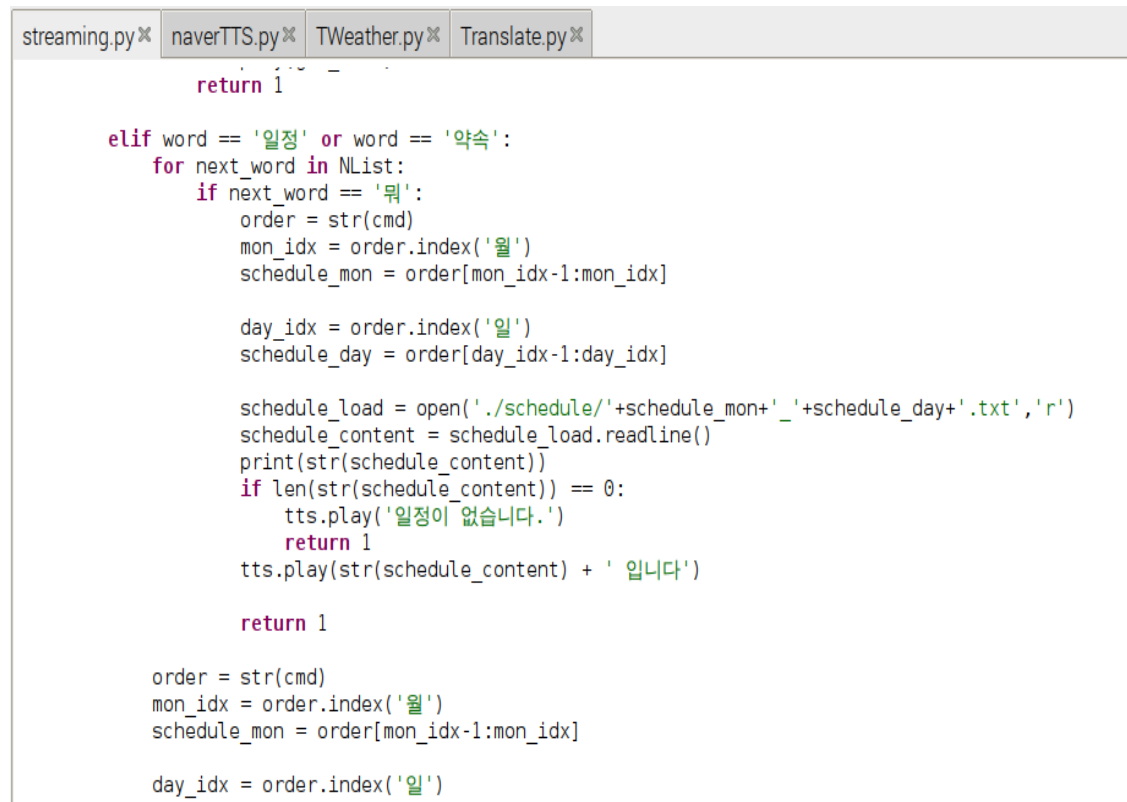
```
streaming.py ✕  naverTTS.py ✕  TWeather.py ✕  Translate.py ✕  
tmpPlayPath = './tmp.wav'  
  
class NaverTTS():  
    def __init__(self, speaker=0, speed=0):  
        self.speaker = speakers[speaker]  
        self.speed=str(speed)  
    def play(self, txt):  
        encText = urllib.parse.quote(txt)  
        data = "speaker=" + self.speaker + "&speed=" + self.speed + "&text=" + encText;  
  
        request = urllib.request.Request(url)  
        request.add_header("X-NCP-APIGW-API-KEY-ID",client_id)  
        request.add_header("X-NCP-APIGW-API-KEY",client_secret)  
        response = urllib.request.urlopen(request, data=data.encode('utf-8'))  
        rescode = response.getcode()  
        if(rescode==200):  
            response_body = response.read()  
            with open(tmpPlayPath, 'wb') as f:  
                f.write(response_body)  
  
            #외부 프로그램 사용 vlc  
            #os.system('cvlc ' + tmpPlayPath + ' --play-and-exit')  
            #라즈베리파이  
            os.system('omxplayer ' + tmpPlayPath)  
        else:  
            print("Error Code:" + rescode)
```

## 일정 저장

### ▶ Text Slice를 이용해 날짜별로 일정을 파일로 저장

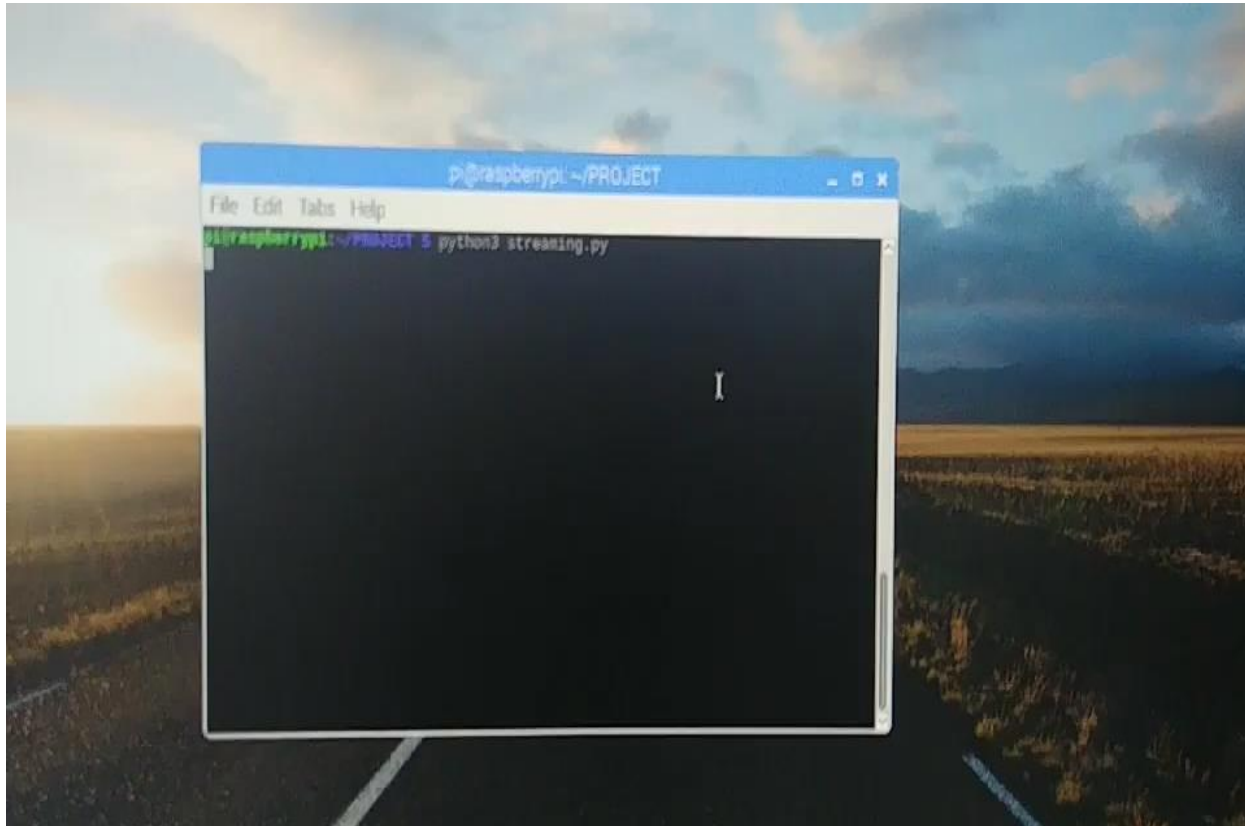


일정을 txt 파일로 저장

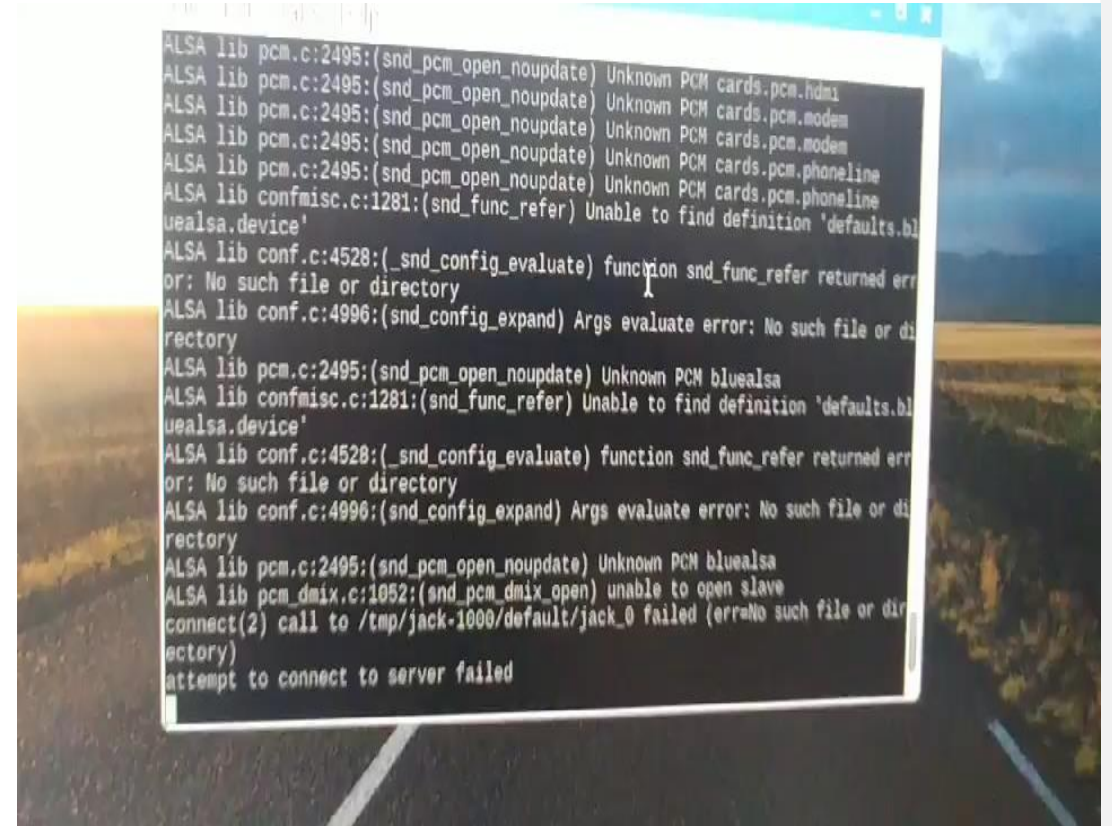


일정 txt 파일 로드

## 실행 동영상



기본 의사 소통



생활 정보

## 프로그램 실행 유지 시간

- ✓ Google Cloud Speech는 1분의 제한시간이 존재한다.
- ✓ 유료버전으로 전환하면 제한이 없어진다.

## 조사, 동사 인식

- ✓ 현재는 사람들의 명령을 Rule로 만들어서 명사만 인식.
- ✓ 조사와 동사를 이용한다면 좀 더 좋은 인식을 할 것으로 예상됨.

## 단어 학습

- ✓ 단어들을 학습시키면 뜻이 같은 문장을 판별하고 인식할 수 있을 것으로 예상

## 음악 실행

- ✓ Raspberry Pi에 저장된 파일이 아닌 뮤직 플레이어를 통한 재생.

T H A N K  
Y O U

---