<2020 캡스톤 디자인 최종 보고서>

위험 소리 감지 AI 스피커

2020.06.28

경상대학교 해양과학대학

정보통신공학과

2s팀: 김륜우,구주회, 박현주, 임석현

< 목 차 >

I.서론

1. 과제 목표 및 필요성

1) 목표

: 우리는 많은 경우 중 한 가지의 경우만 감지하는 시스템 개발

2) 필요성

- 사용자가 넘어져서 위급한 상황이 발생할 경우, 움직임 감지센서는 사용자가 일정 시간 이상 움직이지 않으면 작동함.

- BUT 골든 타임을 놓지는 상황이 발생할 가능성이 존재하며, 움직임만 보는 것이 아닌 소리를 듣고 감지센서와 함께 상황을 판단할 수 있으면 좀 더 정확하고 빠르게 위험 시 응급신호를 보낼 수 있음.

- 이는 노인 분들의 위험상황에 훨씬 정확하고 빠르게 대처가 가능하게 될 것이라고 생각됨.

II. 위험 소리 감지 AI 스피커

2.1. 시스템 요구사항

1) 기능적 요구사항

- 소리를 듣고 위험상황을 판단
: RNN(Recurrent Neural Network)를 이용해 소리 정보를 받아서 위험한 상황이 발생했는지 판단한다.
(작게는 큰 소리, 작은 소리 구별부터 추후엔 무슨 소리인지 알 수 있게 만들기)

- 전원을 켠다
: 따로 전원 버튼을 만들어서 버튼을 누르면 전원이 켜진다.

- 와이파이를 연결한다
: 전원 버튼과 마찬가지로 와이파이를 켤 수 있는 버튼을 만들어서 자동으로 연결하게 만들어 준다.

2) 비기능적 요구사항

정확성

: 최대한 정확한 프로그램을 개발해 변수들을 제거해야 한다. 위험한 상황인지 아닌지 확실하게 구분 할 수 있어야 한다.
(예: 많은 분들이 모여 놀면서 소리를 지르는 경우도 있음)

- 편의성

: 소리를 듣고 판단을 하는 기기를 개발하는 것이 목표임으로 크기가 작은게 좋다.

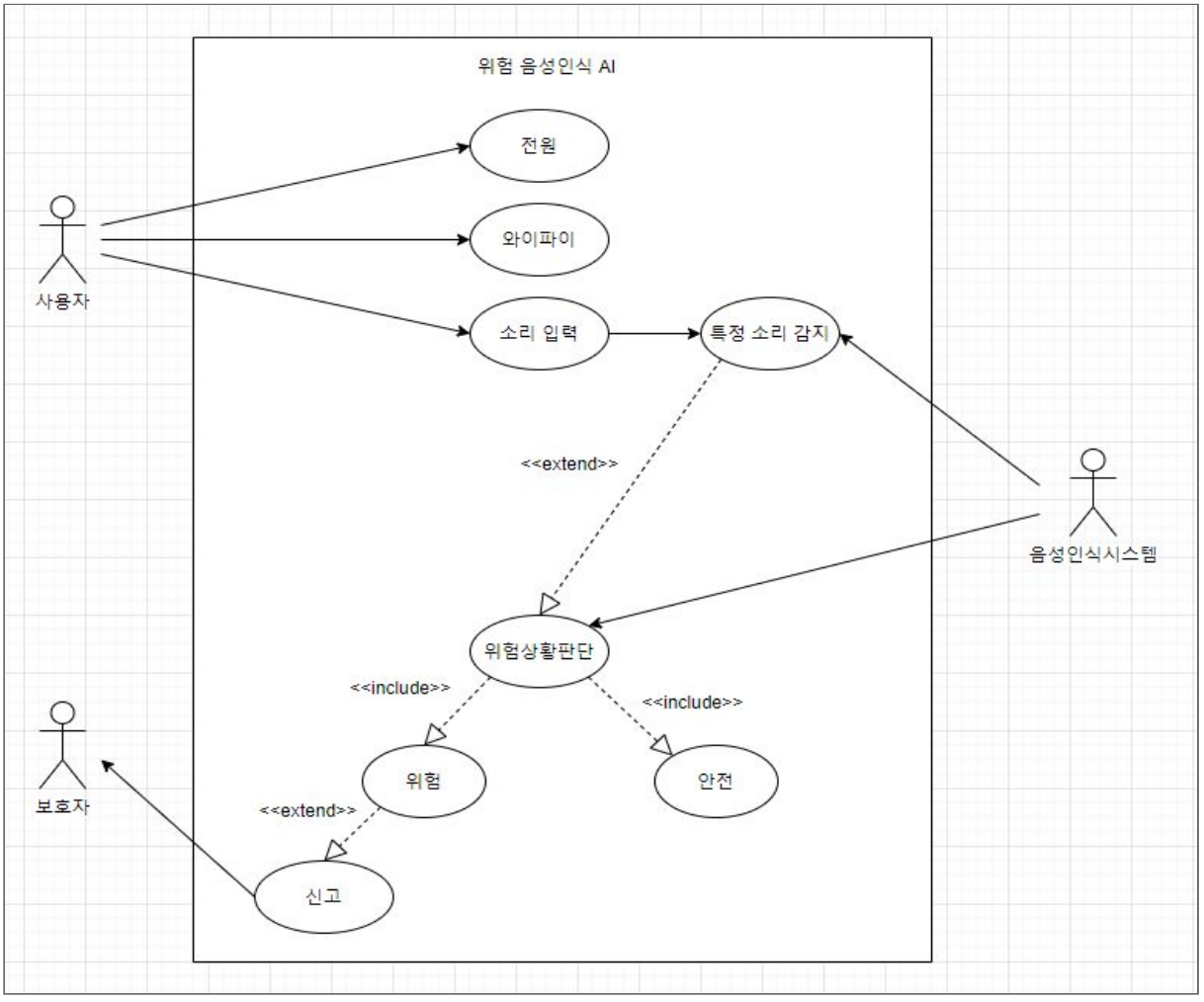
- 운영과 보수성

: 매번 업데이트 해주는 것이 좋지만, 기기를 만든다고 가정했을 때는 개발 시 최대한 많은 경우의 수를 판단할 수 있는 프로그램 개발이 필요하다.

-효율성

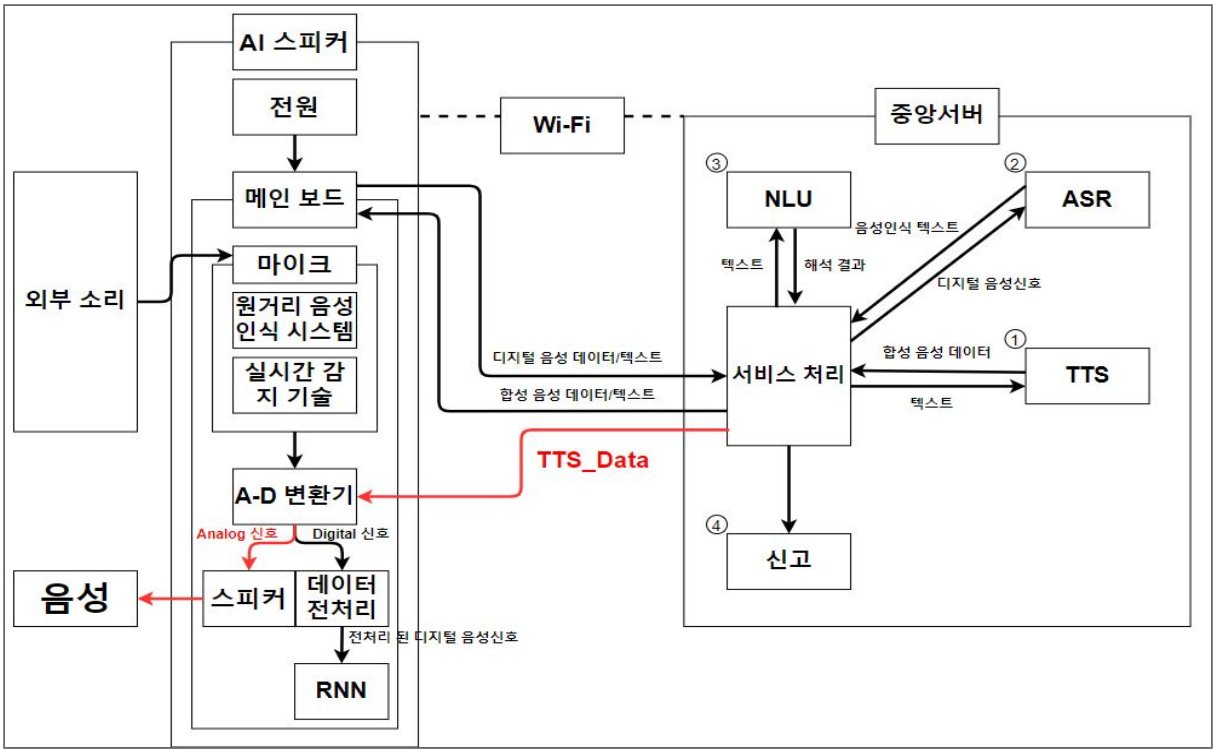
: 특정 소리가 나면 빠르게 상황을 판단하고 신호를 보내게 한다.

3) 요구사항 분석(use case)

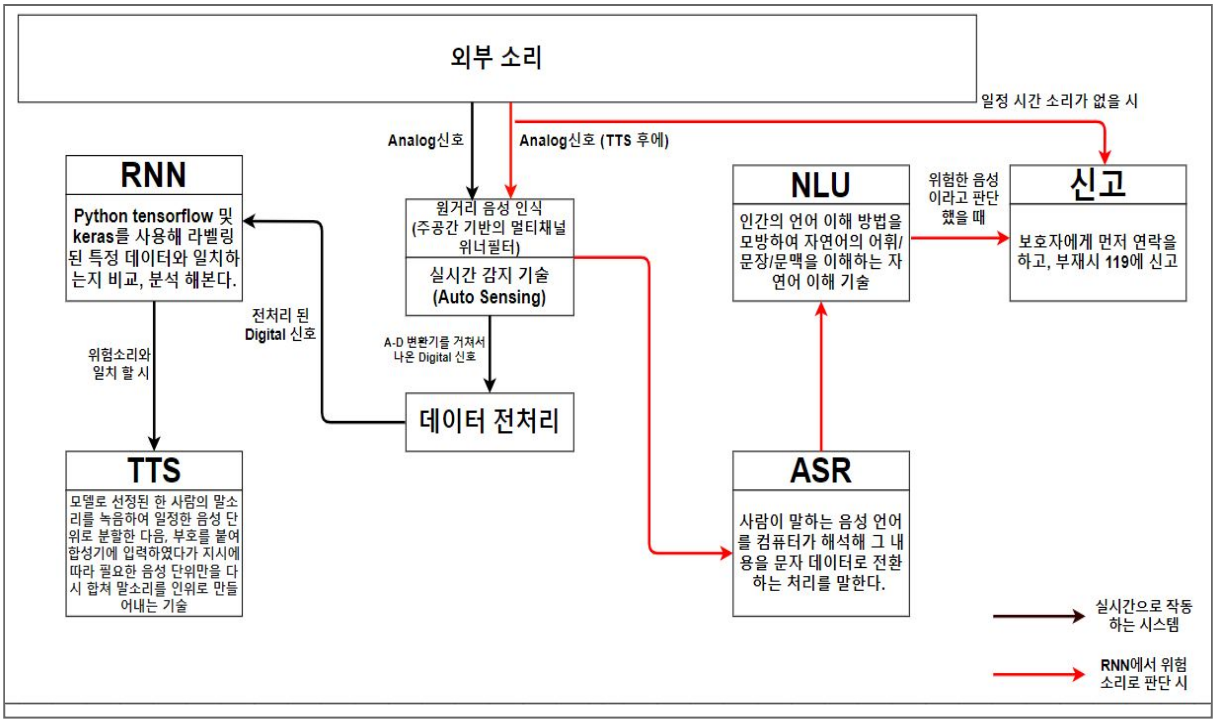


< 그림 1 > 위험 음성인식 AI Use case

2.2. 설계 결과



< 그림 2 > SW 전체 시스템 아키텍처 구상도



< 그림 3> SW 프로젝트 아키텍처 구상도

III. 시스템 구현 및 결과

3.1 데이터 수집

1) Python을 이용하여 Youtube 동영상 오디오를 추출하여 데이터를 수집함

① 사운드 데이터 수집에 사용한 파이썬 라이브러리

- Pytube
- youtube_dl

```python
import os
import subprocess
import youtube_dl

#다운로드 받고 싶은 음악의 유튜브 주소를 입력
url = "https://www.youtube.com/watch?v=tM-51bjU7kg"

# 필요하다면 옵션을 여러가지 사용할 수 있다.
ydl_opts = {}
with youtube_dl.YoutubeDL(ydl_opts) as ydl:
    ydl.download([url])

# 다운로드 받은 webm 파일을 리스트에 저장한다.
file_list = os.listdir('.')
video = [file for file in file_list if file.endswith("webm")]

# 파일을 mp3 로 변환한다.
for v in video:
    s = v.split('.')[0]+'.mp3'
    cmds = ['ffmpeg', '-i', v, '-vn', '-ab', '192K', '-y', s]
    subprocess.Popen(cmds)
    print('Converting',v,'to',s)

```

< 그림 4 > 사용 라이브러리

2) 추가적으로 인터넷에 공개된 데이터를 검색 및 정제하여 데이터에 포함 시킴

3) 학습 데이터

① 사운드 종류 : 유리 깨지는 소리

② 파일 수 : 1008개

③ 총 시간 : 총 38분 30초

④ 파일 포맷 : .wav

3.2. 데이터 전처리 & 특징추출 과정

1) 데이터 전처리 과정

① Analog to Digital Converter ADC.

② DC Filtering.

③ Serial to 32-Bit Parallel Converter.

④ Integer to Floating-Point Converter.

⑤ Pre-Emphasis Filtering.

⑥ Window Advance Buffering.

⑦ Hamming Window.

2) MFCC란?

- 오디오 신호에서 추출할 수 있는 feature로, 소리의 고유한 특징을 나타내는 수치이다.

- 주로 음성 인식, 화자 인식, 음성 합성, 음악 장르 분류 등의 오디오 도메인 문제를 해결하는 데 사용된다.

3) MFCC의 사용 예시

예1) 화자 검증 , speaker verification

- **화자 검증이란 화자 인식의 세부 분류로서 말하는 사람이 그 사람이 맞는지를 확인하는 기술, 대표적인 예시로 시스템의 등록된 음성에만 반응하는 아이폰의 siri를 볼 수 있다.**

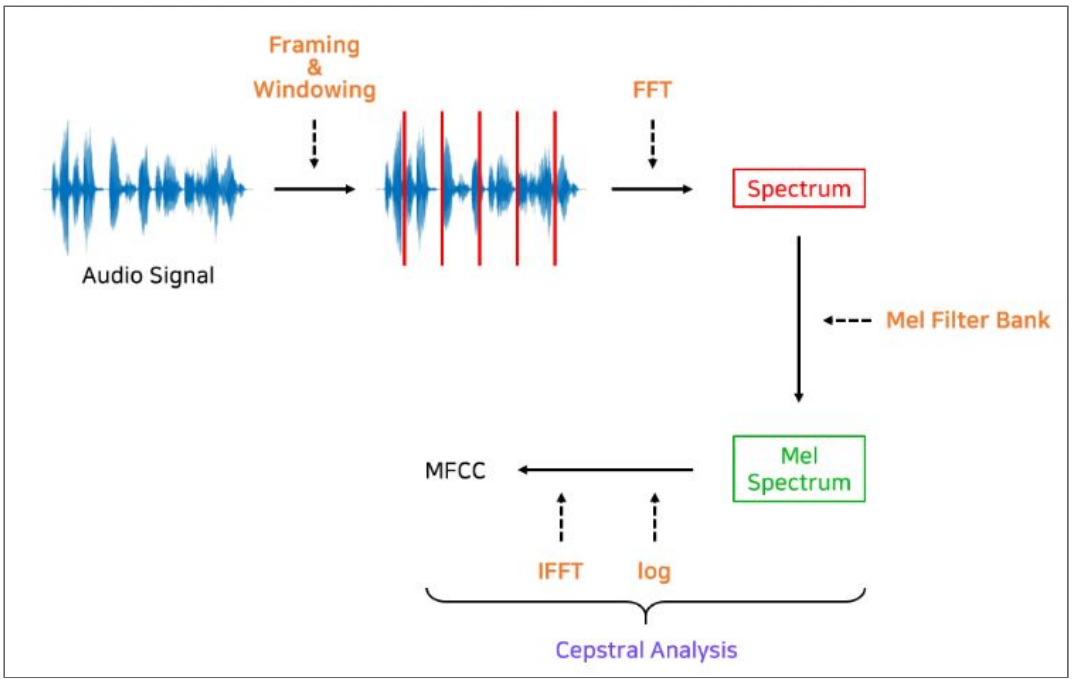
- **보통 유사도를 정해놓고 들어오는 인풋 음성이 그 유사도보다 큰지 아닌지를 화자가 맞는지를 판별하게 된다.**

- **즉, MFCC가 음성의 고유한 특징을 표현하는 값으로 사용된다는 사실을 알 수 있다.**

예2) 음악 장르 분류, music genre classfication

- MFCC는 음악 신호에서도 자주 사용된다.
- 화자 인식에서 화자의 특징을 표현할 수 있는 것처럼, 음악의 특징도 MFCC로 표현할 수 있다.

4) MFCC 추출 과정



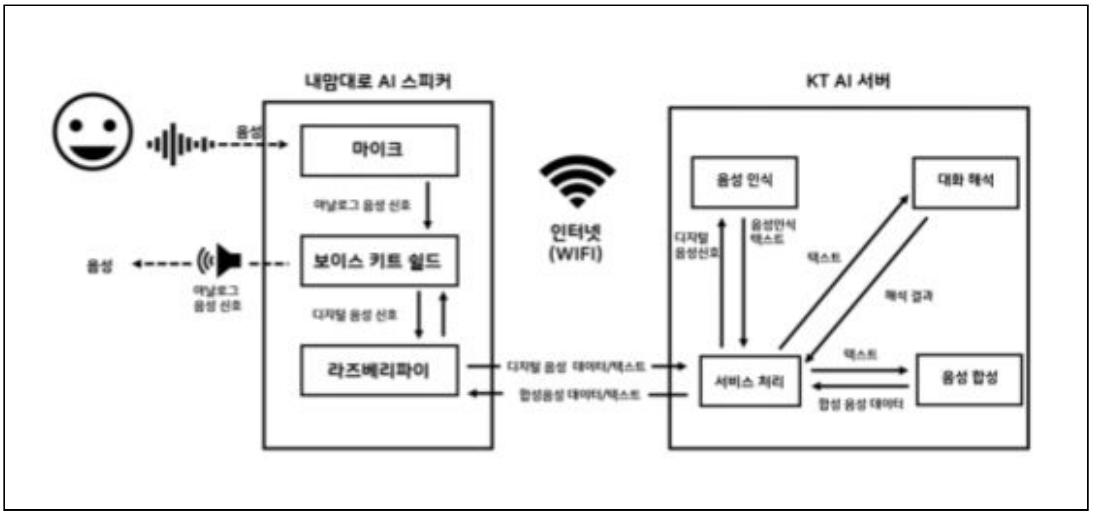
① 오디오 신호를 프레임별로 (보통 20ms- 40ms)로 나누어서 FFT를 적용해 Spectrum을 구한다. 오디오 신호는 시간,time(가로축)에 따른 음압(세로축)의 표현, 즉 시간 영역이 표현된다. 여기서 FFT를 수행하면 주파수(가로축)에 따른 음압(세로축)의 표현, 즉 주파수 영역(frequency domain)의 표현이 가능해지고, 그것이 spectrum이다.

② Spectrum에서 Mel filter Bank를 적용해서 Mel spectrum을 구한다.

③ Mel spectrum에 Cepstral 분석을 적용해서 MFCC를 구한다.

3.3. AI 스피커 mechanism

- 본 과정에서는 KT AI 스피커 KIT의 코드분석을 통해 AI스피커에 필수 기술인 STT , TTS 기술에 대한 전반적인 매커니즘을 설명한 후 실행함.
- 라즈베리파이는 보이스 키트에 장착된 마이크와 버튼으로 사용자의 입력을 받아서 가공한 데이터를 API를 통해서 서버에 전송한 후, 처리된 데이터는 다시 라즈베리파이의 스피커를 통해서 출력되어 사용자에게 들려줌.



1) STT (Speech to Text) / wake-up word

:사람이 말하는 음성 언어를 컴퓨터가 해석해 그 내용을 문자 데이터로 전환하는 처리를 말한다.

```
def test(key_word = '2s'):
    rc = ktkws.init("../data/kwsmodel.pack")
    print ('init rc = %d' % (rc))
    rc = ktkws.start()
    print ('start rc = %d' % (rc))
    print ('\n호출어를 불러보세요~\n')
    ktkws.set_keyword(KWSID.index(key_word))
    rc = detect()
    print ('detect rc = %d' % (rc))
    print ('\n\n호출어가 정상적으로 인식되었습니다.\n\n')
    ktkws.stop()
    return rc

def main():
    test()
if __name__ == '__main__':
    main()
```

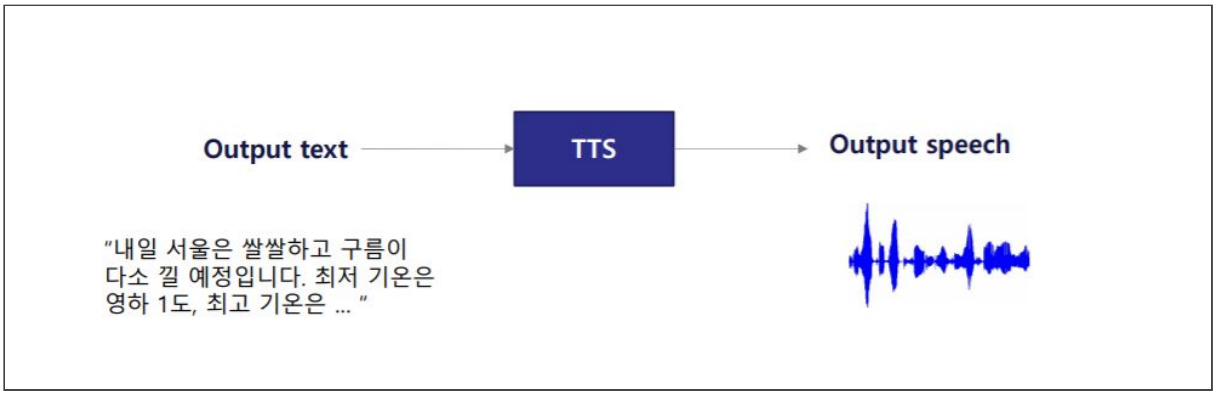
> def test(key_word = '2s')함수
마이크로 호출어를 인식하는 함수를 실행하고 진행 상황을 출력해주는 함수를 작성한 것임.

> def main 함수
main함수를 사용해 윗 함수를 음성 호출을 할 수 있음.

> wake-up word
본 코드는 호출 명령어를 간단히 작성한 코드임.

2) TTS (Text to Speech)

: 말소리의 음파를 기계가 자동으로 만들어 내는 기술로, 간단히 말하면 모델로 선정된 한 사람의 말소리를 녹음하여 일정한 음성 단위로 분할한 다음, 부호를 붙여 합성기에 입력하였다가 지시에 따라 필요한 음성 단위만을 다시 합쳐 말소리를 인위로 만들어내는 기술이다.




```
def generate_request():
    with MS.MicrophoneStream(RATE, CHUNK) as stream:
         audio_generator = stream.generator()

        for content in audio_generator:
          message = gigagenieRPC_pb2.reqVoice()
          message.audioContent = content
          yield message

          rms = audioop.rms(content,2)
          #print_rms(rms)
```

> 마이크를 통해 가져온 데이터를 기가지니 STT API에 입력할 수 있도록 변환해주는 함수

> def getText2VoiceUrl(inText):함수
메인 함수에서 출력할 텍스트를 받아서 TTS API에 입력하여 출력할 수 있는 음성을

```python
from __future__ import print_function
import grpc
import gigagenieRPC_pb2
import gigagenieRPC_pb2_grpc
import user_auth as UA
import os
HOST = 'gate.gigagenie.ai'
PORT = 4080
# TTS : getText2VoiceUrl
def getText2VoiceUrl(inText):
    channel = grpc.secure_channel('{}:{}'.format(HOST, PORT), UA.getCredentials())
    stub = gigagenieRPC_pb2_grpc.GigagenieStub(channel)
    message = gigagenieRPC_pb2.reqText()
    message.lang=0
    message.mode=0
    message.text=inText
    response = stub.getText2VoiceUrl(message)

    print ("\n\nresultCd: %d" % (response.resultCd))
    if response.resultCd == 200:
        print ("TTS 생성에 성공하였습니다.\n\n\n아래 URL을 웹브라우져에 넣어보세요.")
        print ("Stream Url: %s\n\n" % (response.url))

    else:
        print ("TTS 생성에 실패하였습니다.")
        print ("Fail: %d" % (response.resultCd))

def main():
    getText2VoiceUrl("안녕하세요. 반갑습니다.")
if __name__ == '__main__':
    main()
```

B

출력하는 URL을 출력해줍니다. (이 URL은 1분간 유효한 임시 URL입니다.).

> B part

언어를 설정하고(한국어의 경우 lang이 0입니다.) 텍스트를 입력해 변환합니다.

3.4. 실시간 감지 및 AI모델

1) AI 모델 구축 및 실시간 감지

- AI 모델 구축 과정 및 추후 개선점
- 실시간 감지 기술에 대한 과정 및 추후 개선점

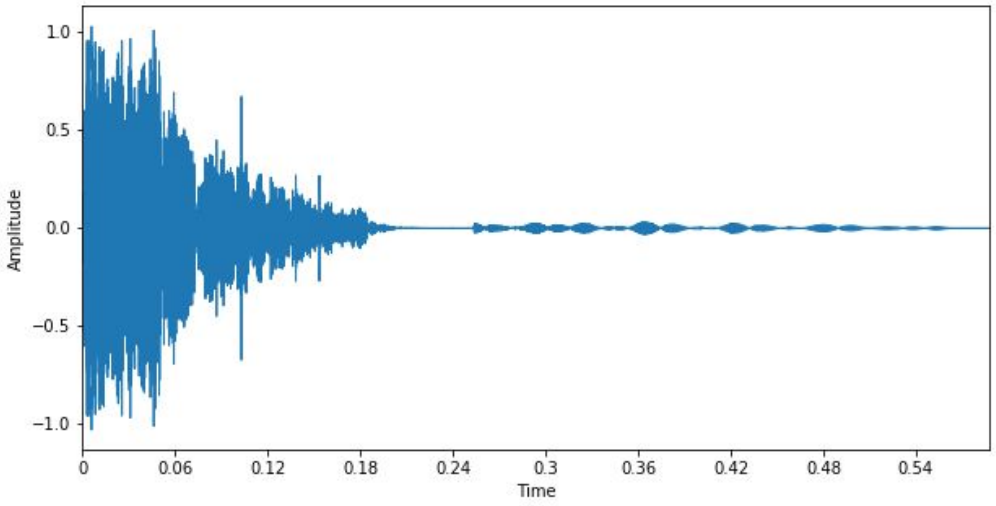
① AI 모델 구축 과정

< 초기 아이디어>

: RNN모델을 사용하려 했으나, 대부분의 모델 구조가 대부분 단어를 인식하는 구조를 가지고 있어, 우리 팀이 하려는 소리 구분에는 맞지않다고 판단, 음성데이터를 시각화 하는 방법을 찾아 사진으로 저장 후 CNN에 적용시켜보기로함.

< Code >

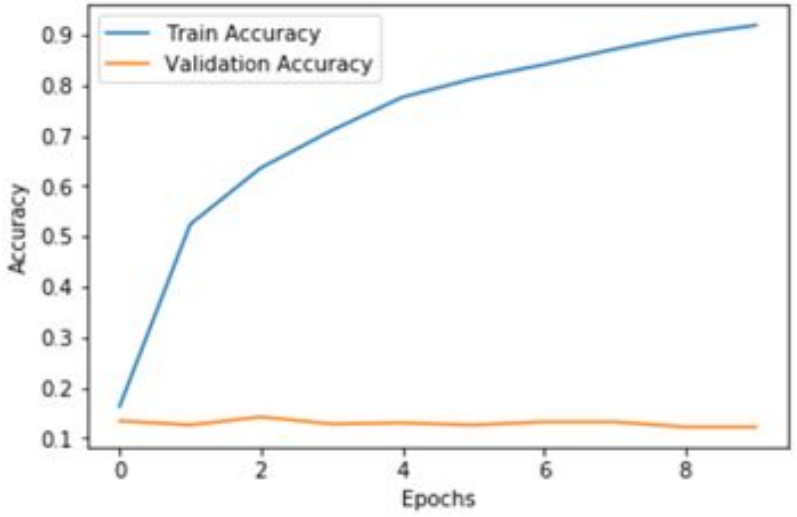
```
import os
import matplotlib.pyplot as plt
import numpy as np
import natsort
import librosa, librosa.display

path_dir = 'C:/Users/say23/Downloads/유리 파일/유리 파일/'
file_list = os.listdir(path_dir) #os.listdir를 사용해 디렉토리 내의 모든 파일 이름을
                                 #list 형식으로 만듦
file_list = natsort.natsorted(file_list)

for i in range(0, 1) :
    wav = path_dir + file_list[i]
    y, sr = librosa.load(wav, sr = 16000)
    Value = np.max(y)/10
    Index = np.where(np.abs(y) > Value)
    time = np.linspace(0, len(y)/sr, len(y))
    (file_dir, file_id) = os.path.split(wav)
    plt.figure(figsize = (10,5))
    plt.xlabel("Times")
    plt.ylabel("Amplitude")
    y = y[Index[0][0] : Index[0][0] + 16000]
    librosa.display.waveplot(y, sr, alpha = 1)
    plt.savefig('C:/Users₩say23/Picture/' + str(file_id) + '.png', dpi = 50)
    librosa.output.write_wav('2.wav',y,sr)
```



< 그림 5 > 결과 사진

< 문제점 >

: 위 그래프만으로 학습을 하기에는 저런 파형을 가진 소리가 비단 유리깨지는 소리에만 존재한다고 할 수 없기 때문에 학습이 잘 되지 않을 것이다.



➤ 위의 그래프는 직접 학습해본것은 아니고(데이터 수가 적어서), 다른 사람이 파형 사진만을 가지고 학습시켰을 때 나온 정확도를 나타내는 그래프이다. 그렇다면 다른 방식의 시각화는 불가능한가에 대해 생각을 해보았다.

< 해결책 >

- 시각화 할 수 있는 다른 방법이 존재하는지 찾아보았고, Spectrogram, MFCC 기법을 이용해 시각화 하는 방법을 찾아냄

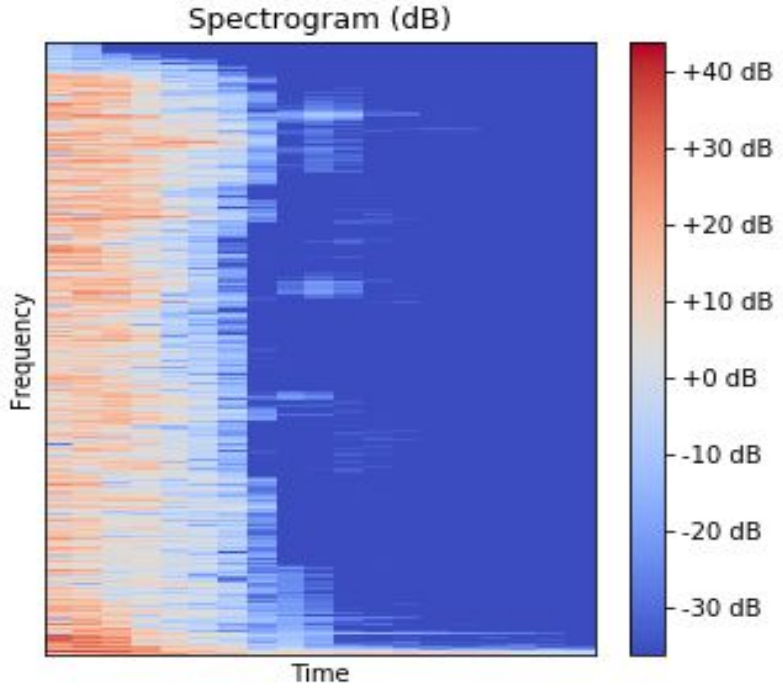
Spectrogram

- 스펙트로그램(Spectrogram)은 소리나 파동을 시각화 하여 파악하기 위한 도구로 파형(waveform)과 스펙트럼(spectrum)의 특징이 조합되어 있다.

- dB 단위로 구별이 가능해 파형만 그려져있는 사진보다는 학습이 더 잘 될 것으로 예상

< Code >

```
# STFT -> spectrogram
hop_length = 512  # 전체 frame 수
n_fft = 2048  # frame 하나당 sample 수

# calculate duration hop length and window in seconds
hop_length_duration = float(hop_length)/sr
n_fft_duration = float(n_fft)/sr

# STFT
stft = librosa.stft(y, n_fft=n_fft, hop_length=hop_length)

# 복소공간 값 절댓값 취하기
magnitude = np.abs(stft)

# magnitude > Decibels
log_spectrogram = librosa.amplitude_to_db(magnitude)

# display spectrogram
plt.figure(figsize=(5,5))
librosa.display.specshow(log_spectrogram, sr=sr, hop_length=hop_length)
plt.xlabel("Time")
plt.ylabel("Frequency")
plt.colorbar(format="%+2.0f dB")
plt.title("Spectrogram (dB)")
```



< 그림 6 > 결과 사진

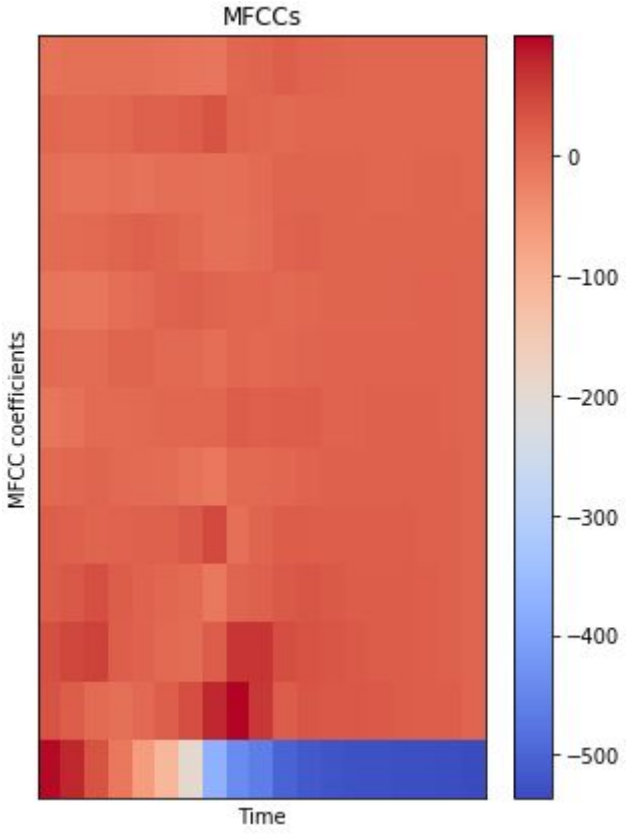
< MFCC >

- MFCC(Mel-frequency cepstral coefficients)란

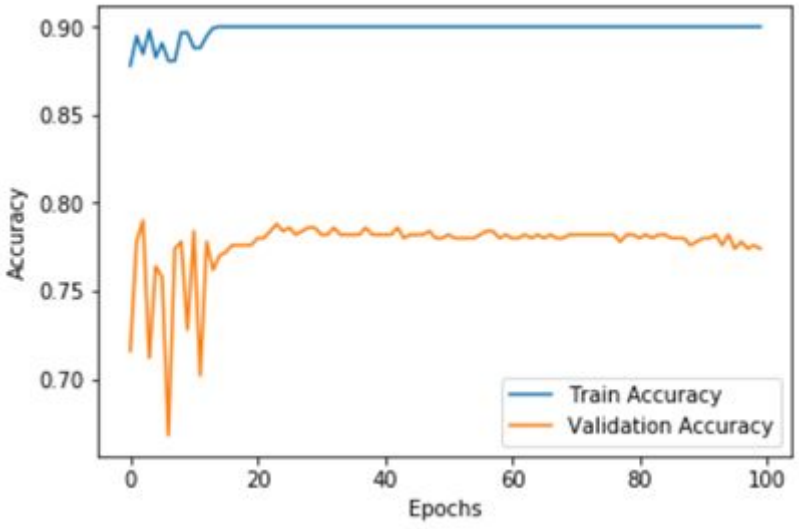
  : 음성의 특징을 추출하는 방법 중 최근에 많이 사용되어지고 있는 방법이고 입력된 신호에서 노이즈 및 배경 소음이 있을 텐데 그중 실제 유효한 소리의 특징을 추출하는데 쓰인다.

< Code >

```
MFCCs = librosa.feature.mfcc(y, sr, n_fft=n_fft, hop_length=hop_length, n_mfcc=13)

# display MFCCs
plt.figure(figsize=(5,10))
librosa.display.specshow(MFCCs, sr=sr, hop_length=hop_length)
plt.xlabel("Time")
plt.ylabel("MFCC coefficients")
plt.colorbar()
plt.title("MFCCs")

# show plots
plt.show()
```



< 그림 7 > 구현한 MFCC 사진



- 실제로 구축을 해서 학습을 시켜보진 못했지만 위에 파형 그래프로 학습시킨 사람이 MFCC를 사용해서도 학습을 시켜보았는데 위와 같은 결과가 나왔다.

- (간단한 모델로 했음에도 불구하고 높은 정확도를 도출해냄)

2) AI 모델 구축 문제점 및 개선점

1. 그래프 파형만으로는 정확한 모델 구축이 불가능하다고 판단.
2. Spectrogram 사진을 이용한 학습
3. MFCC 사진을 이용한 학습
4. 2, 3 번의 학습으로도 정확도가 낮게 나온다면 두 모델을 함께 사용하는 방안 모색

② 실시간 감지 기술 과정

< Code >

```
import pyaudio
import wave
import numpy as np
import time

CHUNK = 2**18
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 44100
WAVE_OUTPUT_FILENAME = "output.wav"
start_time = time.time()
p = pyaudio.PyAudio()

stream = p.open(format = FORMAT,
                channels = CHANNELS,
                rate = RATE,
                input = True,
                frames_per_buffer = CHUNK)

print("Start to record the audio")

frames = []
while(True) :
    data = np.fromstring(stream.read(CHUNK), dtype = np.int16)
    frames.append(data)
    if time.time() - start_time > 5 :
        break
print("Recording is finished.")

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

wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(CHANNELS)
wf.setsampwidth(p.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b''.join(frames))
wf.close()
```

< 상세 코드 설명 >

```
import pyaudio
import wave
import numpy as np
import time
```

pyaudio 모듈을 통해 음성을 받을 수 있다.
wave 모듈을 통해 받은 음성을 저장할 수 있다.

```
CHUNK = 2**18
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 44100
WAVE_OUTPUT_FILENAME = "output.wav"
start_time = time.time()
p = pyaudio.PyAudio()

stream = p.open(format = FORMAT,
                channels = CHANNELS,
                rate = RATE,
                input = True,
                frames_per_buffer = CHUNK)
```

- CHUNK 는 음성데이터를 불러올 때 한번에 몇개의 정수를 불러올 지 정하는 것 FORMAT 은 비트의 깊이 현재는 16비트

- CHANNELS 는 1일 때 모노, 2일 때 스테레오로 지정해주는 것

- RATE는 초당 몇 번의 샘플링을 할 것인지 정하는 것

- WAVE_OUTPUT_FILENAME 은 말 그대로 저장되어질 파일명

- p 는 PyAudio 모듈의 객체 생성

- p.open 함수를 통해 음성 데이터 스트림을 열어준다.
  (input 은 입력 스트림인지 아닌지 설정하게 하는 매개 변수)

```
print("Start to record the audio")

frames = []
while(True) :
    data = np.fromstring(stream.read(CHUNK), dtype = np.int16)
    frames.append(data)
    if time.time() - start_time > 5 :
        break
print("Recording is finished.")

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

wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(CHANNELS)
wf.setsampwidth(p.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b''.join(frames))
wf.close()
```

- 녹음한 음성 데이터들을 frames 이라는 list형 변수를 통해 저장시킬 수 있다.

- stream.read 함수는 음성 데이터를 문자열로 반환하는데, 이 문자열들을 np.fromstring()함수를 이용해 정수 numpy 배열로 바꿔준다. (다양하게 사용할 수 있을 것으로 예상)

- while문을 이용해 5초간 음성데이터를 받게 한다.

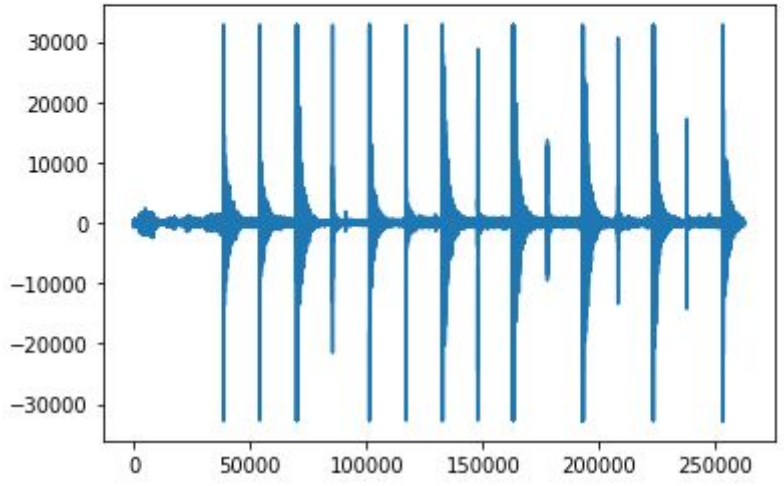
- stream.stop_stream()은 재생/녹음을 일시 정지
- stream.close() 는 스트림 종료
- p.terminate() 는 포트 오디오 세션을 종료

- wave.open()함수를 이용해 파일명을 지정, 용도를 설정해줌 (‘wb’는 쓰기전용, ‘rb’는 읽기전용)
- setnchannels() 로 채널 수 설정
- setframerate() 로 프레임 속도 설정
- writeframes() 로 오디오 프레임을 기록한다.
- close() 는 끝내기

< 실시간 감지 응용 >

```
1 import matplotlib.pyplot as plt
2 x = np.arange(0, 2**18)
3 y = data
4 plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x137cc4c5448>]
```



- 위 슬라이드에 있던 np.fromstring 함수를 이용해 얻은 data 값으로 음성데이터 그래프를 그려보았다.
  (왼쪽 그래프는 박수치는 소리를 5초간 받은 소리이다.)

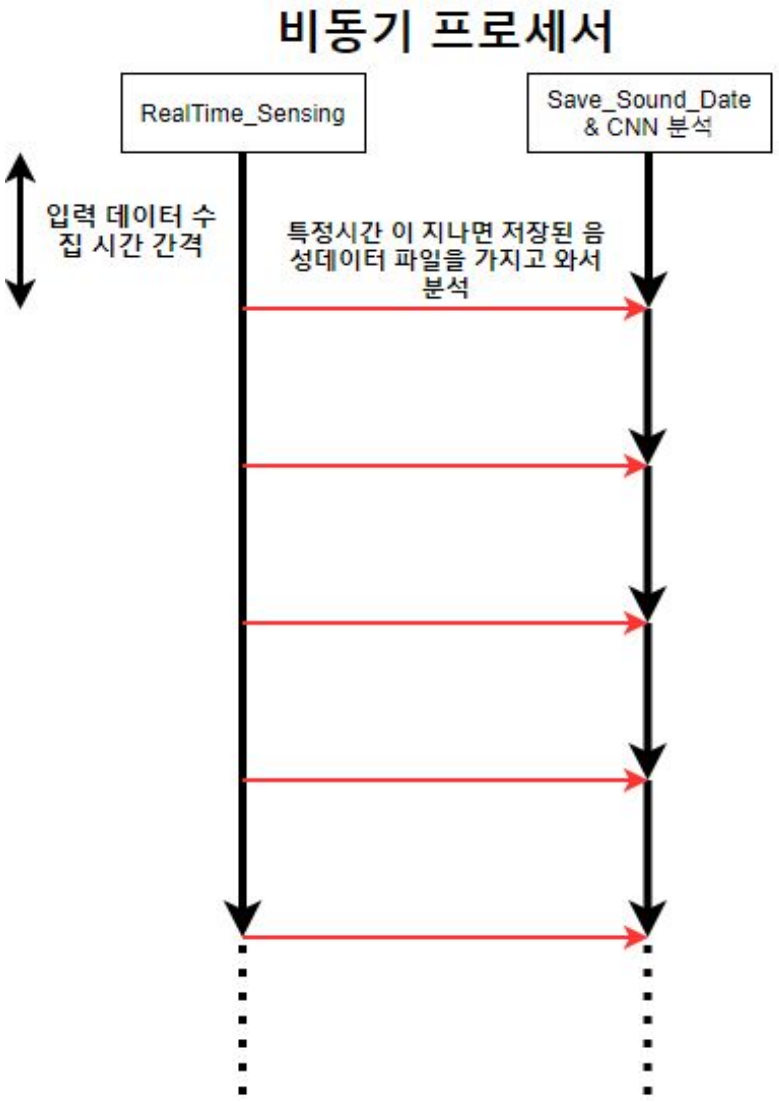
<실시간 감지 개선>

```
while True :
    frames = []
    while True :
        data = np.fromstring(stream.read(CHUNK), dtype = np.int16)
        frames.append(data)
        if time.time() - start_time > 0.5 :
            break
    stream.stop_stream()
    stream.close()
    p.terminate()

    wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
    wf.setnchannels(CHANNELS)
    wf.setsampwidth(p.get_sample_size(FORMAT))
    wf.setframerate(RATE)
    wf.writeframes(b''.join(frames))
    wf.close()
```

- 위와 같이 while문을 한번 더 사용해 일정시간 동안 계속

음성데이터를 받고, 특정 시간만큼 잘라서 저장한 음성데이터를 MFCC 기법을 통해 시각화 시켜 MFCC를 이용해 학습시킨 CNN 모델에 넣어 소리를 인식 시켜보는 방법이 좋을 것 같음

< 비동기 프로그래밍 활용 >



IV. 평가

4.1. 목표 대비 달성

① 데이터 수집 및 방법
② 데이터 전처리 및 특징추출 과정
③ AI 스피커 mechanism
④ 실시간 감지 방법 및 AI모델 구축 개선안

4.2. 과제를 통해 배운점과 향후 이 과제를 이어나갈 팀을 위한 조언

- 2S팀은 본 프로젝트를 통하여 AI 스피커에 대한 전반적인 기술 체계와 소프트웨어 설계과정 등을 배울 수 있는 기회를 가졌다. 무엇보다 기존의 다른 과목과 다르게 PBL 활동을 수행하여 팀원들간의 협동 및 개별 공부를 통해서 소규모 그룹을 통한 학습활동 사회적 환경 안에서 의사소통하며, 다른사 람들의 다양한 시각을 이해하면서 학습함

4.3. 공유(Git Repository)에 관한 설명

https://github.com/Lyun-Woo-Kim/MC.git 에 파일들을 다 정리 해놨습니다.

V. 참고문헌 및 사이트

1. **'파이썬으로 유튜브(Youtube) 영상을 저장과 mp3 파일 변환', woongheelee.com/, 2020년 6월 1일 접속 https://woongheelee.com/entry/파이썬으로-유튜브-영상을-저장과-mp3-파일-변환**
2. **'wake-up-word feature extraction on fpga','scientific research' 2020년 6월 14일 접속**

   https://file.scirp.org/Html/1-1560031_42600.htm
3. **'MFCC(Mel-Frequency Cepstral Coefficient)란?','네이버블로그',2020년 6월16일 접속**

   https://blog.naver.com/nm1lee/221915475247
4. **'Glass break search', FreeSound, 2020년 6월 14일 접속**

   https://freesound.org/
5. **'Glass break search', ZqpSplat, 2020년 6월 16일 접속**

https://www.zapsplat.com/

6. ‘Glass break search’, soundeffects+, 2020년 6월 16일 접속
https://www.soundeffectsplus.com/

7. ‘음악’, 공유마당, 2020년 6월 14일 접속
https://gongu.copyright.or.kr/gongu/main/main.do

8. ‘[Sound AI #11]오디오 데이터 전처리’
https://hyunlee103.tistory.com/36

9. ‘Pyaudio로 음성 데이터 입력받기’
https://kcal2845.tistory.com/35

10. ‘라즈베리파이에서 pyAudio 모듈 사용하기’
https://snowdeer.github.io/python/2017/11/15/python-raspberry-pyaudio-example/

11. ‘WAV파일 읽고 쓰기’
https://docs.python.org/ko/3/library/wave.html

12. ‘인공 지능 메이커 키트’, 깃허브,2020년 5월 28일 접속
https://github.com/gigagenie/ai-makers-kit/wiki/%EC%9D%8C%EC%84%B1%EC%9D%B8%EC%8B%9D

13. ‘AI 스피커 만들기’, tistory.com, 2020 5월~6월 접속
https://jybaek.tistory.com/766

14. ‘stt’, 깃허브, 2020 6월 접속
https://github.com/jybaek/hanvi/blob/master/controllers/stt.js

15. ‘kt ai 스피커 ’, blog.naver.com
https://m.blog.naver.com/PostView.nhn?blogId=roboholic84&logNo=221840576143&targetKeyword=&targetRecommendationCode=1