物聯網與微處理機系統設計
Internet of Things and Microprocessor System Design

Lecture 10 - Smart Speaker

Lecturer: 陳彥安 Chen, Yan-Ann

YZU CSE

Modified from: NCTU IoT Course
https://github.com/coldwufish/RaspPI

Outline

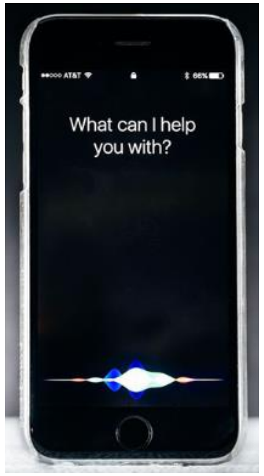
- Introduction
- Installation
- Speech to Text
- Text to Speech
- Hotword detector
- Discussion & Lab

Outline

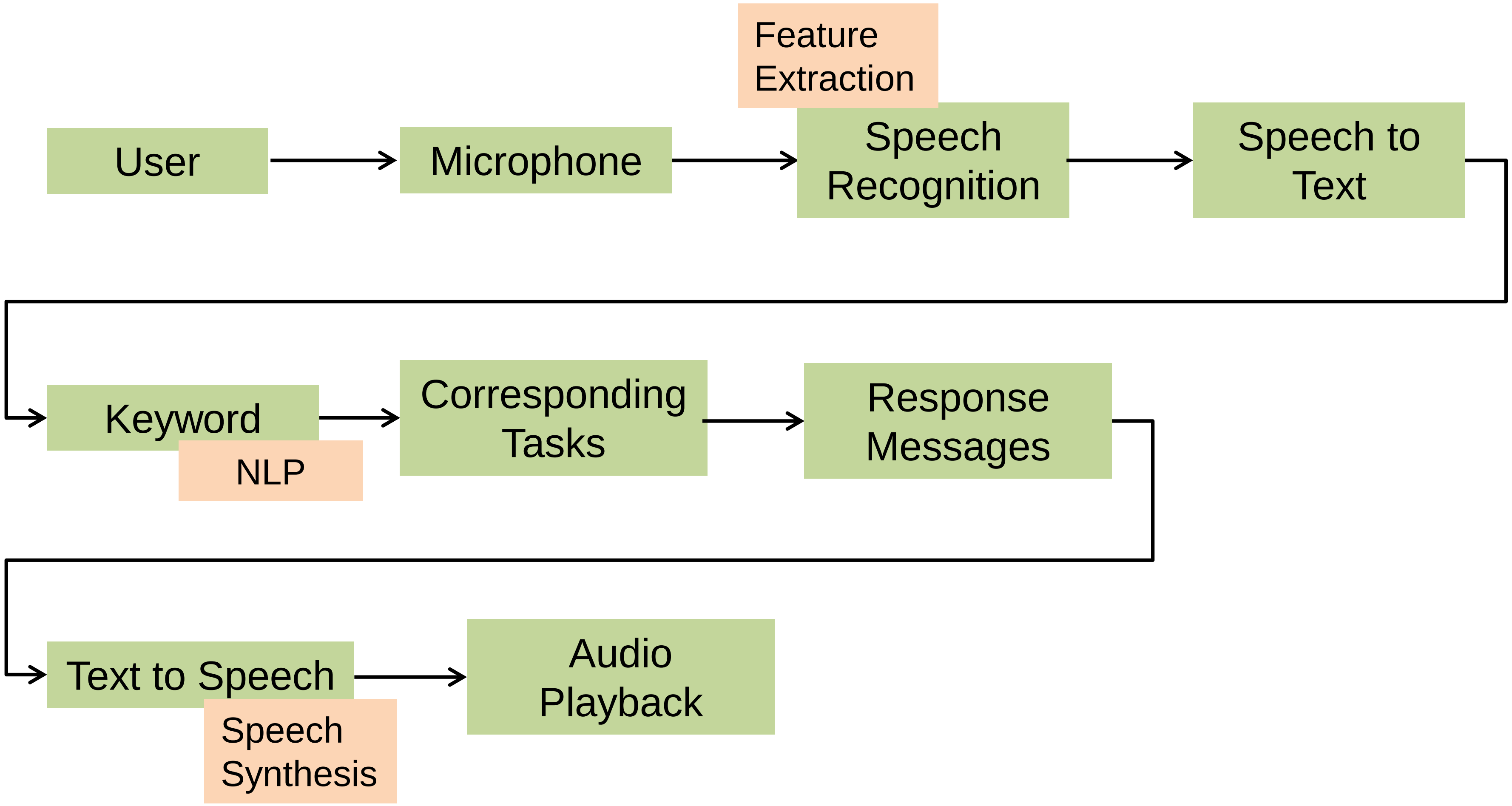
- Introduction
- Installation
- Speech to Text
- Text to Speech
- Hotword detector
- Discussion & Lab

Introduction

- Voice Assistant

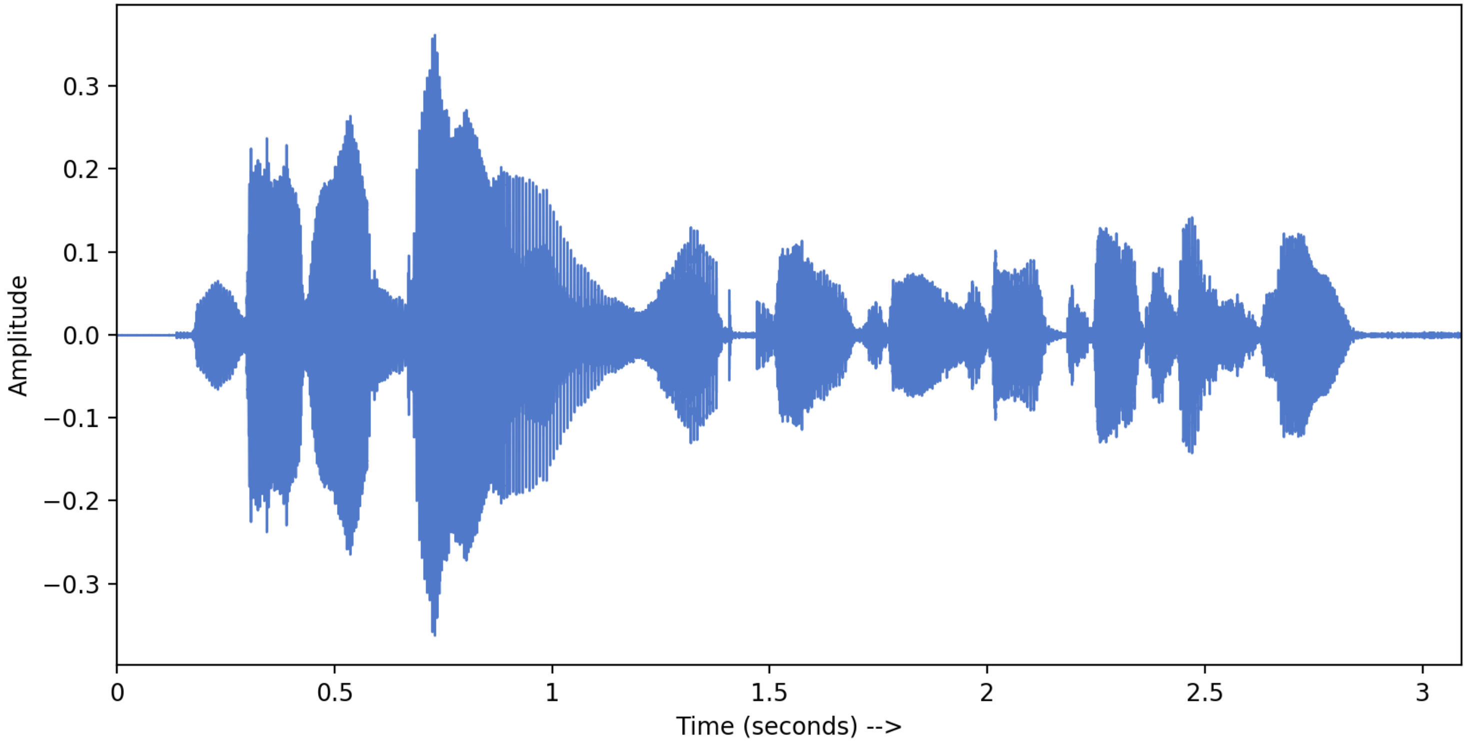


Flowchart of Voice Assistant



Audio Processing (1/3)

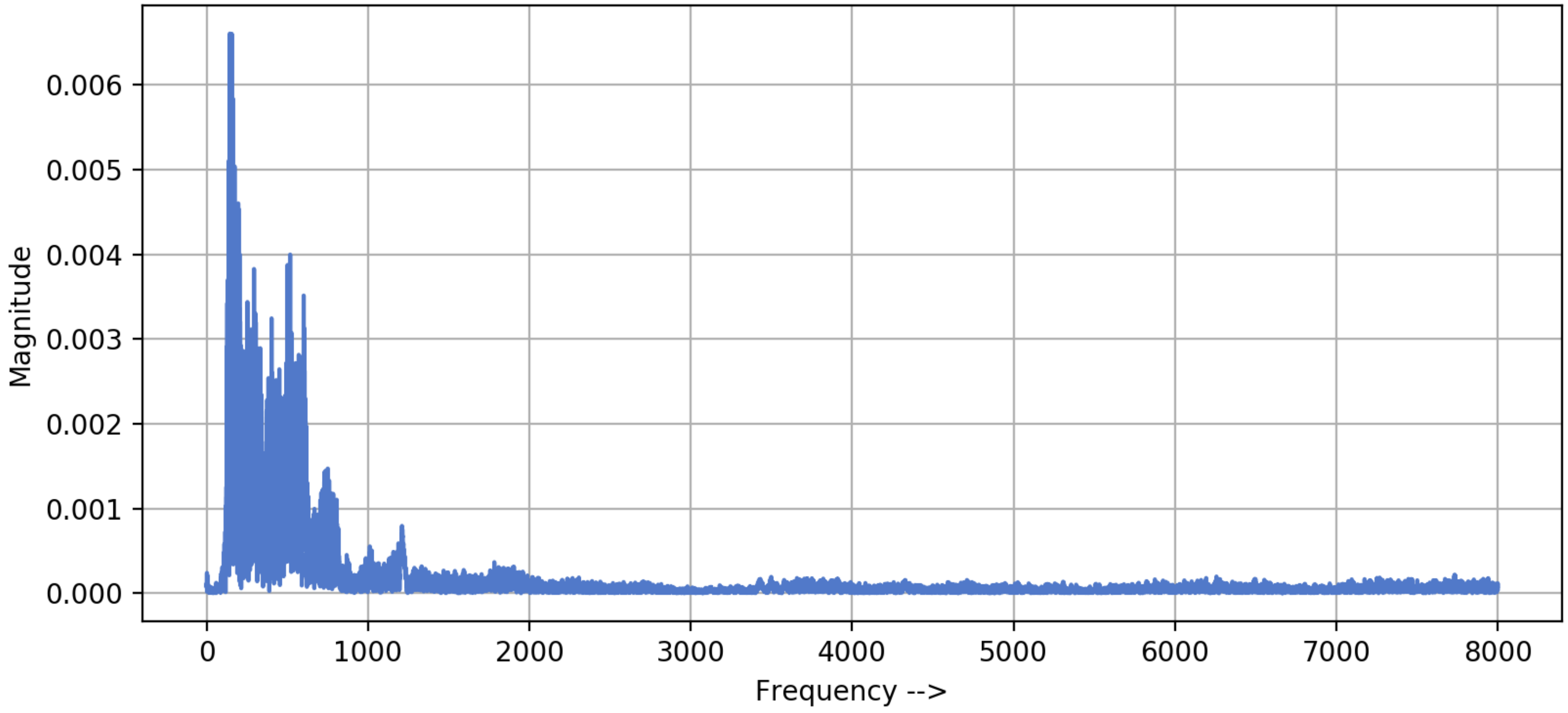
- Audio clip



Ref: https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520

Audio Processing (2/3)

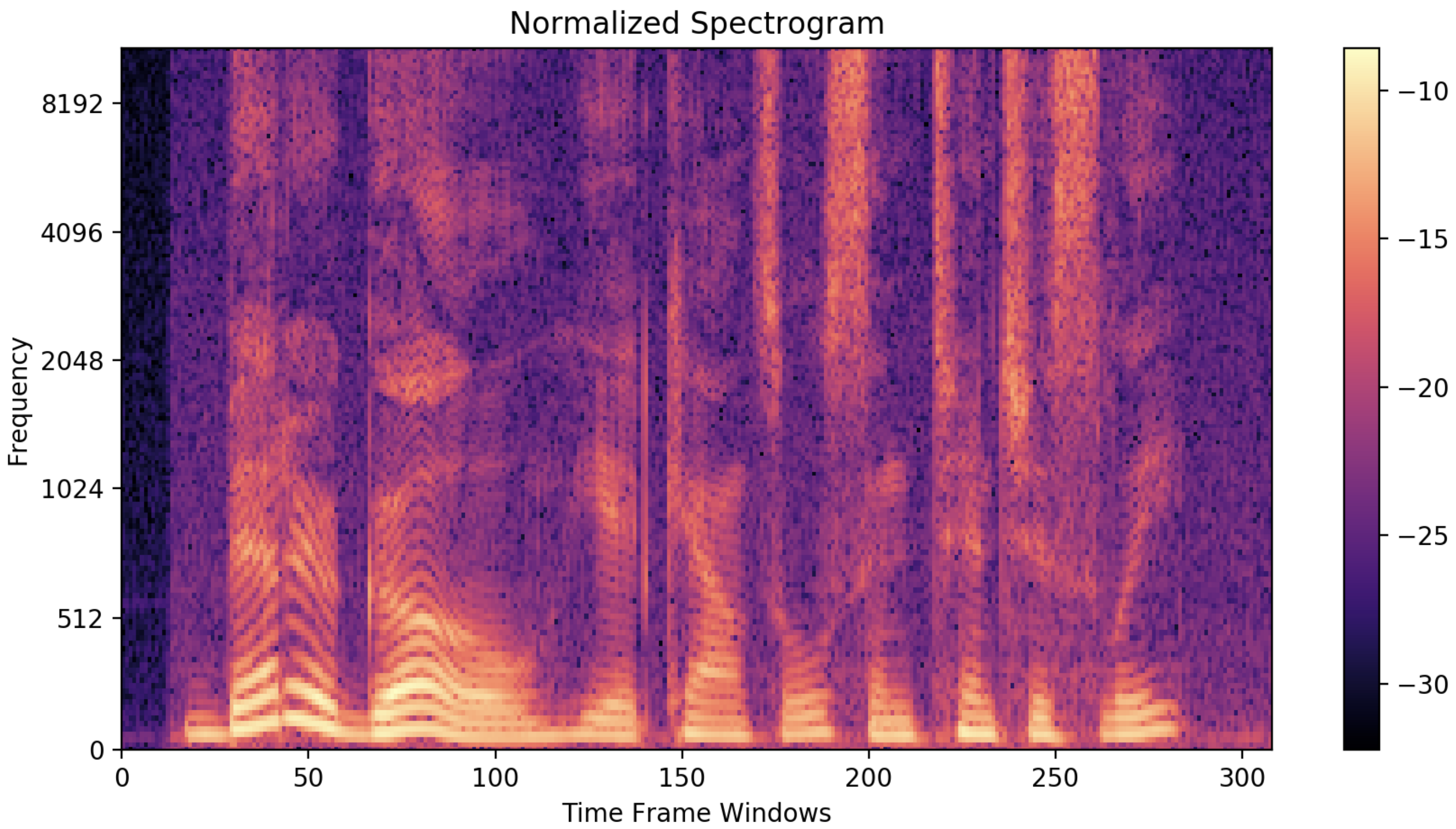
- Audio signals in frequencies



Ref: https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520

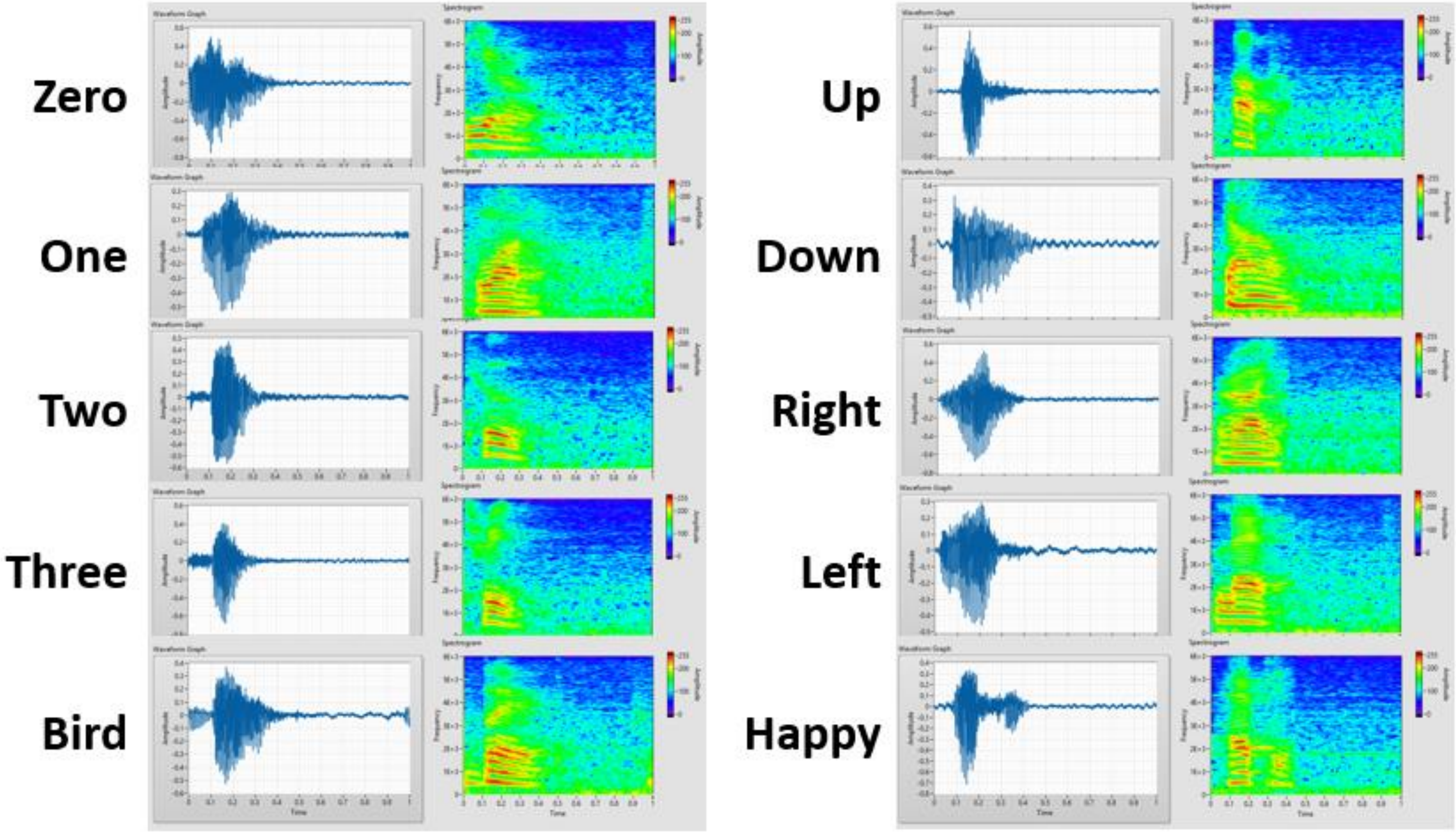
Audio Processing (3/3)

- Spectrogram
  - Colors represent magnitude (amplitude)



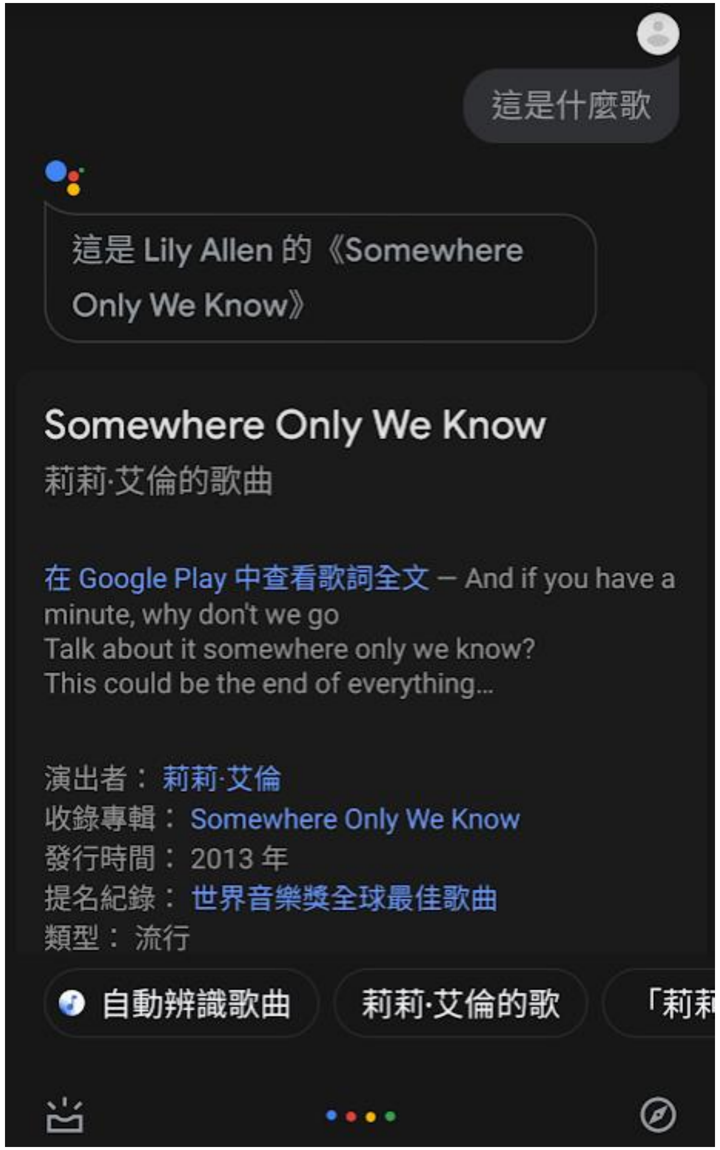
Ref: https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520

Speech Recognition



Ref: https://www.ngene.co/post/speech-recognition-with-deep-learning

Google Assistant



Google Assistant

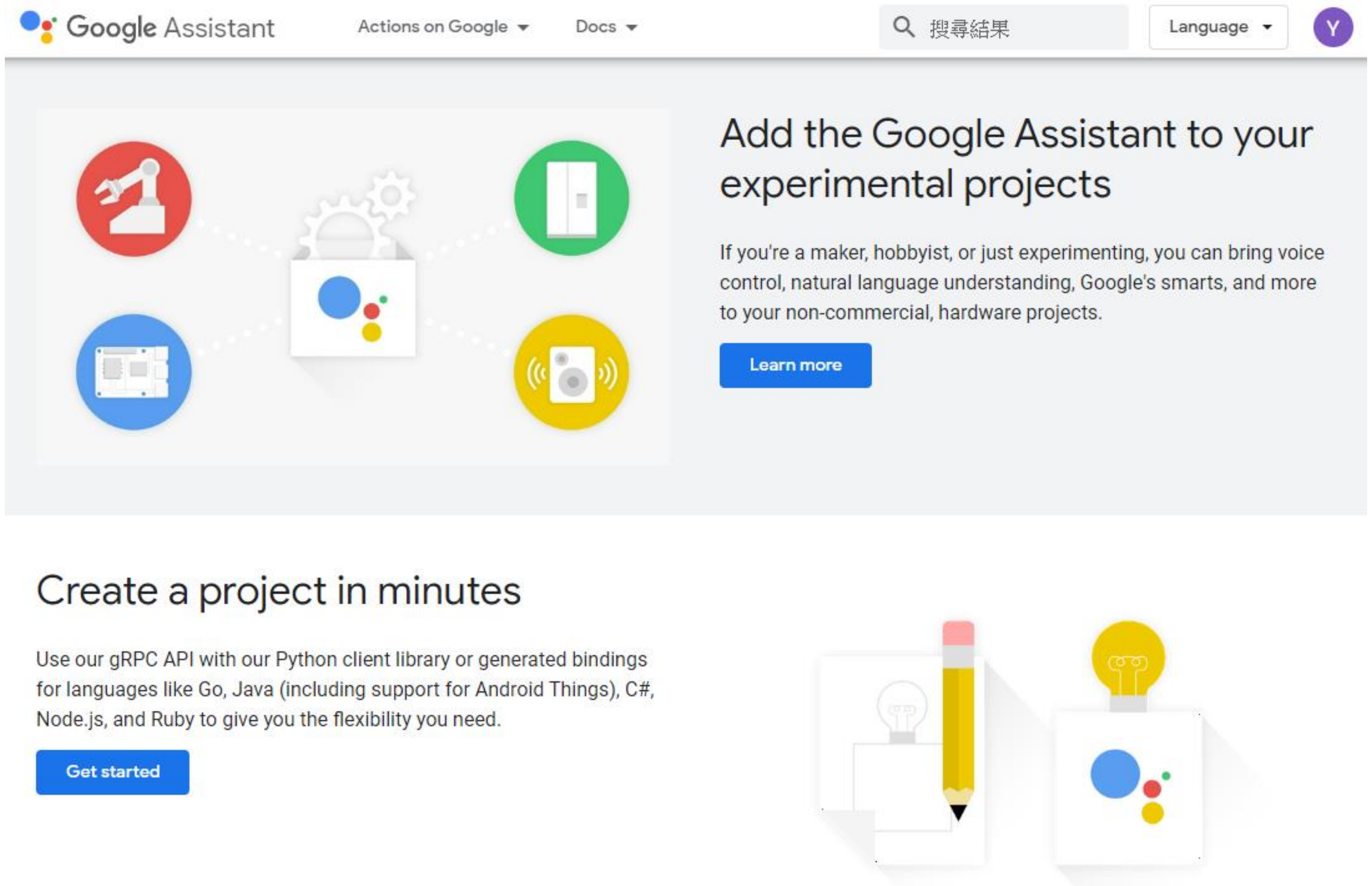
- Google Assistant calling a restaurant for a reservation
- https://www.youtube.com/watch?v=7gh6_U7Nfjs

Google Assistant

- Google IO 2019 Next Gen Google Assistant (2019/5/7)
- https://www.youtube.com/watch?v=3kODsHcrs2c

Google Assistant SDK

- https://developers.google.com/assistant/sdk/



Voice Kit

- https://aiyprojects.withgoogle.com/voice/

Microsoft Azure

- https://azure.microsoft.com/zh-tw/services/cognitive-services/speech-services/

Speech to Text – Converts spoken audio to text for intuitive interaction

Easily add real-time speech-to-text capabilities to your applications for scenarios like voice commands, conversation transcription, and call center log analysis.

Tailor your speech recognition models to adapt to users' speaking styles, expressions, and unique vocabularies, and to accommodate background noises, accents, and voice patterns.

Learn more >

Text to Speech – Give natural voice to your apps

Build smart apps and services that speak to users naturally with the Text to Speech service. Convert text to audio in near real time, tailor to change the speed of speech, pitch, volume, and more.

Give your application a one-of-a-kind, recognizable brand voice using custom voice models. Simply record and upload training data, and the service will create a unique voice font tuned to your recording.

Learn more >

Speech Translation

Give your app real-time speech translation capabilities in any of the supported languages and receive either a text or speech translation back. Speech Translation models are based on leading-edge speech recognition and neural machine translation (NMT) technologies. They're optimized to understand the way people speak in real life and generate translations of exceptional quality.

Learn more >

SpeechRecognition

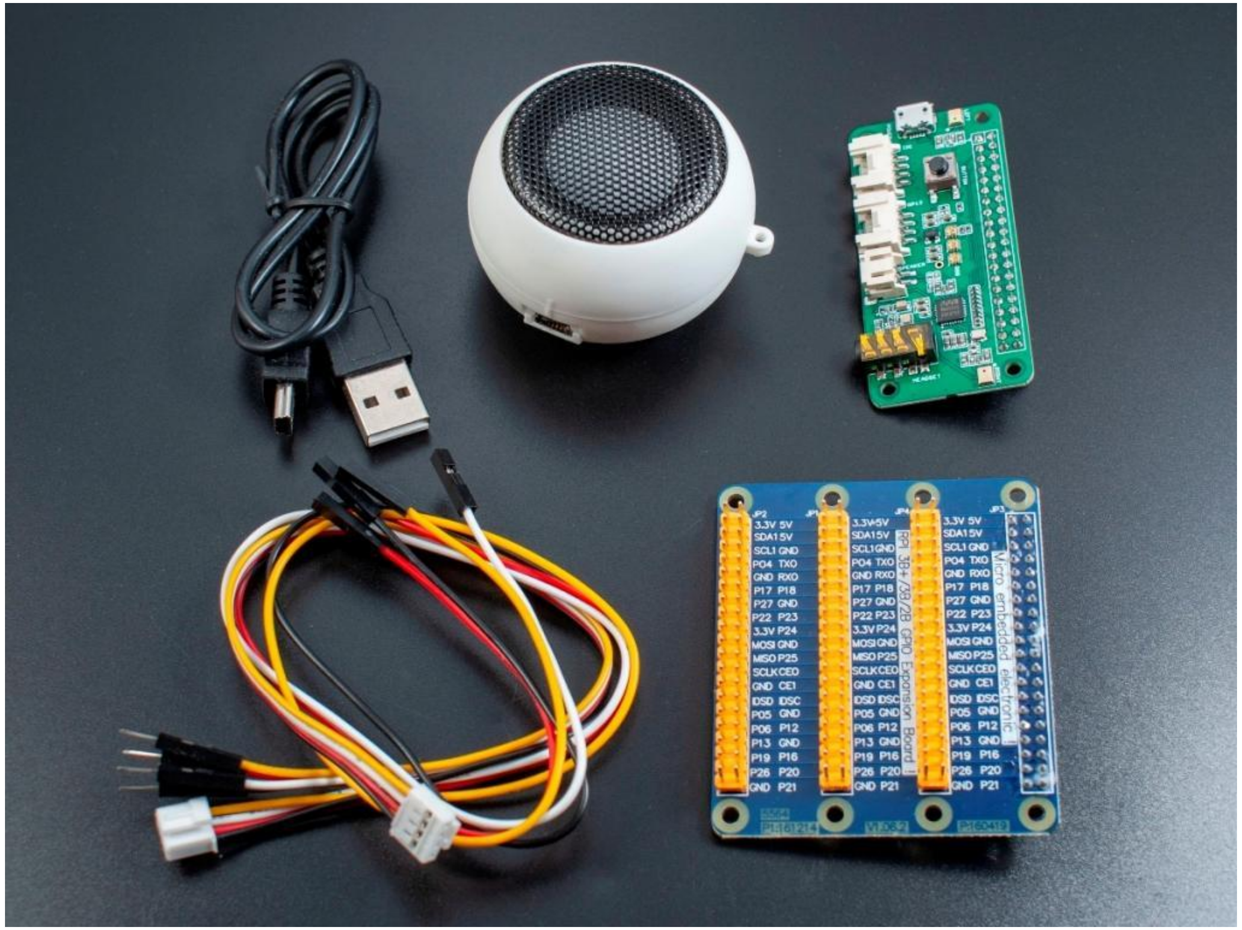
- Library for performing speech recognition, with support for several engines and APIs, online and offline.
- https://pypi.org/project/SpeechRecognition/
- Speech recognition engine/API support:
  - CMU Sphinx (works offline)
  - Google Speech Recognition
  - Google Cloud Speech API
  - Wit.ai (Facebook, Messenger ChatBot)
  - Microsoft Bing Voice Recognition
  - Houndify API (SoundHound,音樂識別平台)
  - IBM Speech to Text
  - Snowboy Hotword Detection (works offline)

Outline

- Introduction
- Installation
- Speech to Text
- Text to Speech
- Hotword detector
- Discussion & Lab

Smart Speaker Kit

- https://www.raspberrypi.com.tw/30038/pi-smart-speaker-kit-v2



《特色》

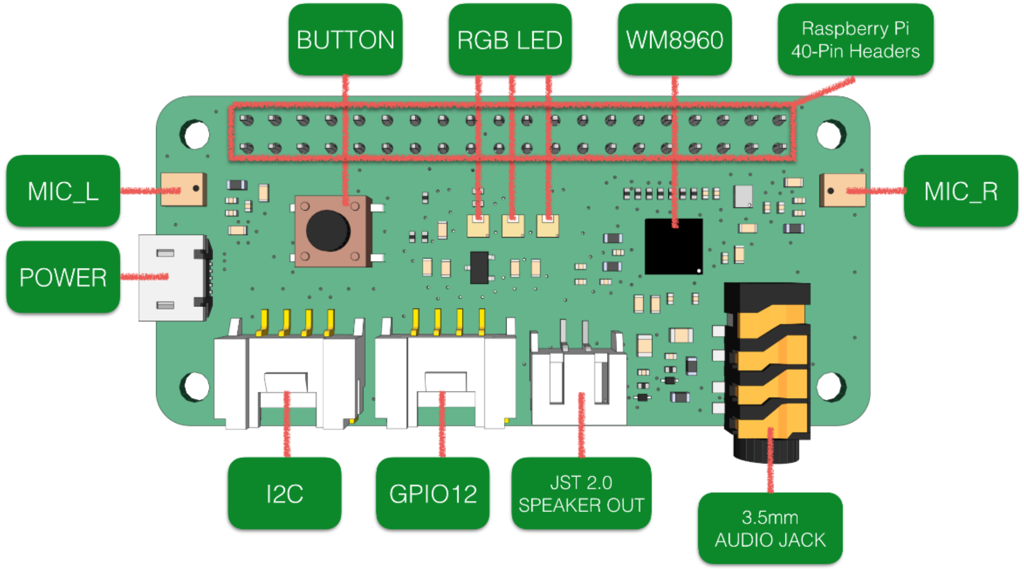
1. 手把手教學，附完整範例程式，可結合智慧插座套件做語音控制開關。
2. 可使用 NLTK 和結巴(Jieba)做中英文斷詞和自然語言處理。
3. 可嵌入 Google Assistant 或是雲端語音辨識功能，例如 Alexa、Olami 等。
4. 可串接 snowboy 自訂喚醒詞，串接 Dialogflow 自然語言理解平台。
5. 全系列 Pi 都可以使用(Pi4 也可以使用)。

《規格》

1. ReSpeaker雙麥克風擴充板(2-Mics Pi HAT)
2. Grove 母排線 x1
3. Grove 公排線 x1
4. 高品質攜帶式喇叭(USB 充電)
5. GPIO擴充板(一對三)

ReSpeaker 2-Mics

- https://wiki.seeedstudio.com/ReSpeaker_2_Mics_Pi_HAT_Raspberry/
- ReSpeaker 2-Mics Pi HAT is a dual-microphone expansion board for Raspberry Pi designed for AI and voice applications.
  - Raspberry Pi compatible(Support Raspberry Pi Zero and Zero W, Raspberry Pi B+, Raspberry Pi 2 B, Raspberry Pi 3 B, Raspberry Pi 3 B+, Raspberry Pi 3 A+ and Raspberry Pi 4)
  - 2 Microphones
  - 2 Grove Interfaces
  - 1 User Button
  - 3.5mm Audio Jack
  - JST2.0 Speaker Out
  - Max Sample Rate: 48Khz



Hardware Setup

- Connect the 2-Mics Pi HAT before powering on.

Driver Installation

- https://wiki.seeedstudio.com/ReSpeaker_2_Mics_Pi_HAT/

$ git clone https://github.com/respeaker/seeed-voicecard

$ cd seeed-voicecard

$ sudo ./install.sh

⋮

```
### Start service seeed-voicecard
    see /var/log/seeed-voicecard.log for more service information
Created symlink /etc/systemd/system/sysinit.target.wants/seeed-voicecard.service
 → /lib/systemd/system/seeed-voicecard.service.
------------------------------------------------------
Please reboot your device to apply all settings
Enjoy!
------------------------------------------------------
pi@rpi4-A00:~/seeed-voicecard $
```

$ sudo reboot

Device Checking

$ aplay -l

```
pi@rpi4-A00:~ $ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: b1 [bcm2835 HDMI 1], device 0: bcm2835 HDMI 1 [bcm2835 HDMI 1]
  Subdevices: 4/4
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
card 1: Headphones [bcm2835 Headphones], device 0: bcm2835 Headphones [bcm2835 H
eadphones]
  Subdevices: 4/4
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
card 2: seeed2micvoicec [seeed-2mic-voicecard], device 0: bcm2835-i2s-wm8960-hif
i wm8960-hifi-0 [bcm2835-i2s-wm8960-hifi wm8960-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

$ arecord -l

```
pi@rpi4-A00:~ $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 2: seeed2micvoicec [seeed-2mic-voicecard], device 0: bcm2835-i2s-wm8960-hif
i wm8960-hifi-0 [bcm2835-i2s-wm8960-hifi wm8960-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Output Setting

$ sudo raspi-config

- Press ESC to exit the configuration tool

```
Raspberry Pi Software Configuration Tool (raspi-config)

1 System Options         Configure system settings
2 Display Options        Configure display settings
3 Interface Options      Configure connections to peripherals
4 Performance Options    Configure performance settings
5 Localisation Options   Configure language and regional settings
6 Advanced Options       Configure advanced settings
8 Update                 Update this tool to the latest version
9 About raspi-config     Information about this configuration tool

            <Select>                  <Finish>
```

```
Raspberry Pi Software Configuration Tool (raspi-config)

S1 Wireless LAN       Enter SSID and passphrase
S2 Audio              Select audio out through HDMI or 3.5mm jack
S3 Password           Change password for the 'pi' user
S4 Hostname           Set name for this computer on a network
S5 Boot / Auto Login  Select boot into desktop or to command line
S6 Network at Boot    Select wait for network connection on boot
S7 Splash Screen      Choose graphical splash screen or text boot
S8 Power LED          Set behaviour of power LED

            <Select>                  <Back>
```

```
Choose the audio output

        0 HDMI 1
        1 Headphones
        2 bcm2835-i2s-wm8960-hifi wm8960-hifi-0

            <Ok>                  <Cancel>
```

Voice Recording & Playback

$ sudo mv /usr/share/piwiz/srprompt.wav /usr/share/piwiz/srprompt.wav.bak

- Record you voice for 5 seconds

$ arecord -d 5 -D plughw:2,0 -f cd voice.mp3

  - "-d": 5 seconds
  - "-D": card number and device number
  - "-f": CD quality

- Play the recorded audio
  - Turn on the speaker

$ omxplayer -o local -p voice.mp3

$ aplay -Dhw:1,0 voice.mp3

Package Installations

```
$ sudo pip3 install SpeechRecognition
$ sudo pip3 install gTTS
$ sudo apt-get install libasound2-dev
$ sudo apt-get install python3-pyaudio
$ sudo apt-get install flac
$ sudo apt-get install jackd
```

Sample Codes

$ wget https://github.com/yachentw/yzucseiot/raw/main/lec10/audio.tar.gz

$ tar -zxvf audio.tar.gz

```
pi@rpi4-A00:~ $ tar -zxvf audio.tar.gz
audio/
audio/stt_file.py
audio/tts_hello_tw.py
audio/stt_realtime.py
audio/tts_hello.py
audio/google.wav
```

$ cd audio

Outline

- Introduction
- Installation
- Speech to Text
- Text to Speech
- Hotword detector
- Discussion & Lab

Speech to Text (Realtime)

- stt_realtime.py

```python
import speech_recognition as sr

#obtain audio from the microphone
r=sr.Recognizer()

with sr.Microphone() as source:
    print("Please wait. Calibrating microphone...")
    #listen for 1 seconds and create the ambient noise energy level
    r.adjust_for_ambient_noise(source, duration=1)
    print("Say something!")
    audio=r.listen(source)

# recognize speech using Google Speech Recognition
try:
    print("Google Speech Recognition thinks you said:")
    print(r.recognize_google(audio, language='zh-TW'))
except sr.UnknownValueError:
    print("Google Speech Recognition could not understand audio")
except sr.RequestError as e:
    print("No response from Google Speech Recognition service: {0}".format(e))
```

$ python3 stt_realtime.py 2>/dev/null

```
Please wait. Calibrating microphone...
Say something!
Google Speech Recognition thinks you said:
你好
```

Speech to Text (Audio file)

- stt_file.py

```
import speech_recognition as sr

#obtain audio from the microphone
r=sr.Recognizer()

myvoice = sr.AudioFile('google.wav')
with myvoice as source:
    print("Use audio file as input!")
    audio = r.record(source)

# recognize speech using Google Speech Recognition
try:
    print("Google Speech Recognition thinks you said:")
    print(r.recognize_google(audio, language='zh-TW'))
except sr.UnknownValueError:
    print("Google Speech Recognition could not understand audio")
except sr.RequestError as e:
    print("No response from Google Speech Recognition service: {0}".format(e))
```

Speech to Text (Audio file)

- Input format: PCM WAV, AIFF/AIFF-C, or Native FLAC
- Recognize the voice in “google.wav” to text.
- Run the program

$ python3 stt_file.py

```
pi@rpi:~/esd/lec11 $ python3 stt_file.py
Use audio file as input!
Google Speech Recognition thinks you said:
我是谷歌小姐
```

SpeechRecognition

- https://pypi.org/project/SpeechRecognition/
  - r.recognize_sphinx(audio)
  - r.recognize_google(audio)
  - r.recognize_google_cloud(audio, credentials_json=GOOGLE_CLOUD_SPEECH_CREDENTIALS)
  - r.recognize_wit(audio, key=WIT_AI_KEY)
  - r.recognize_azure(audio, key=AZURE_SPEECH_KEY)
  - r.recognize_bing(audio, key=BING_KEY)
  - r.recognize_houndify(audio, client_id=HOUNDIFY_CLIENT_ID, client_key=HOUNDIFY_CLIENT_KEY)
  - r.recognize_ibm(audio, username=IBM_USERNAME, password=IBM_PASSWORD)

Examples

See the `examples/` directory in the repository root for usage examples:

- Recognize speech input from the microphone
- Transcribe an audio file
- Save audio data to an audio file
- Show extended recognition results
- Calibrate the recognizer energy threshold for ambient noise levels (see `recognizer_instance.energy_threshold` for details)
- Listening to a microphone in the background
- Various other useful recognizer features

Outline

- Introduction
- Installation
- Speech to Text
- Text to Speech
- Hotword detector
- Discussion & Lab

Text to Speech (En)

- tss_hello.py

```
from gtts import gTTS
import os

tts = gTTS(text='hello', lang='en')
tts.save('hello.mp3')
os.system('omxplayer -o local -p hello.mp3 > /dev/null 2>&1')
```

$ python3 tts_hello.py

Text to Speech (TW)

- tts_hello_tw.py

```python
from gtts import gTTS
import os

tts = gTTS(text='你好我是google小姐', lang='zh-TW')
tts.save('hello_tw.mp3')
os.system('omxplayer -o local -p hello_tw.mp3 > /dev/null 2>&1')
```

$ python3 tts_hello_tw.py

gTTS (Google Text-to-Speech)

- An interface to Google Translate's Text-to-Speech API.
  - https://gtts.readthedocs.io/en/latest/

gTTS

`gTTS` (*Google Text-to-Speech*), a Python library and CLI tool to interface with Google Translate's text-to-speech API. Writes spoken `mp3` data to a file, a file-like object (bytestring) for further audio manipulation, or `stdout`. It features flexible pre-processing and tokenizing, as well as automatic retrieval of supported languages.

gTTS (`gtts.gTTS`)

```
class gtts.tts.gTTS(text, lang='en', slow=False, lang_check=True, pre_processor_funcs=[<function tone_marks>, <function end_of_line>, <function abbreviations>, <function word_sub>], tokenizer_func=<bound method Tokenizer.run of re.compile('(?<=\?).|(?<=\!).|(?<=\？).|(?<=\！).|(?<!\.[a-z])\. |(?<!\.[a-z])\, |(?<!\d)\:|\：|\、|\—|\，|\¿|\…|\n|\]|\،|\。|\(|\¡|\;|\..|\[|\)', re.IGNORECASE) from: [<function tone_marks>, <function period_comma>, <function colon>, <function other_punctuation>]>)
```
[source]

gTTS – Google Text-to-Speech.

An interface to Google Translate's Text-to-Speech API.

Parameters

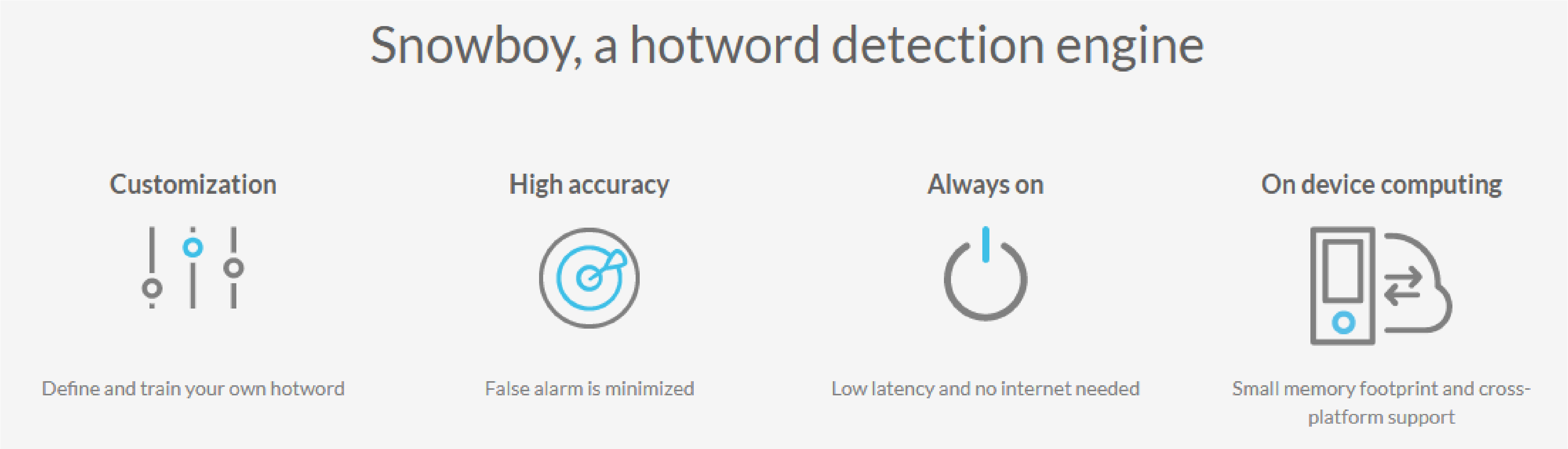
- **text** (*string*) – The text to be read.
- **lang** (*string, optional*) – The language (IETF language tag) to read the text in. Defaults to 'en'.
- **slow** (*bool, optional*) – Reads text more slowly. Defaults to `False`.
- **lang_check** (*bool, optional*) – Strictly enforce an existing `lang`, to catch a language error early. If set to `True`, a `ValueError` is raised if `lang` doesn't exist. Default is `True`.

Outline

- Introduction
- Installation
- Speech to Text
- Text to Speech
- Hotword detector
- Discussion & Lab

Trigger Word Detection

- https://github.com/Kitt-AI/snowboy
- Snowboy, a Customizable Hotword Detection Engine



Installation

$ sudo apt-get install sox swig

$ git clone https://github.com/Kitt-AI/snowboy.git

$ cd snowboy/swig/Python3/

$ make

$ cd ../../examples/Python3

$ nano snowboydecoder.py

```
import collections
import pyaudio
from . import snowboydetect
import time
import wave
import os
import logging
from ctypes import *
from contextlib import contextmanager
```

```
import collections
import pyaudio
import snowboydetect
import time
import wave
import os
import logging
from ctypes import *
from contextlib import contextmanager
```

Demo

$ python3 demo.py resources/models/computer.umdl

- Say “computer”

```
pi@rpi4-A00:~/snowboy/examples/Python3 $ python3 demo.py resources/models/comput
er.umdl
Listening... Press Ctrl+C to exit
INFO:snowboy:Keyword 1 detected at time: 2020-12-07 00:33:50
```

- You can train your personal model on their website.
  - arecord --format=S16_LE --duration=5 --rate=16000 --file-type=wav 1.wav

- https://github.com/Kitt-AI/snowboy/blob/master/examples/Python3/demo.py

```python
import snowboydecoder
import sys
import signal

interrupted = False


def signal_handler(signal, frame):
    global interrupted
    interrupted = True


def interrupt_callback():
    global interrupted
    return interrupted

if len(sys.argv) == 1:
    print("Error: need to specify model name")
    print("Usage: python demo.py your.model")
    sys.exit(-1)

model = sys.argv[1]

# capture SIGINT signal, e.g., Ctrl+C
signal.signal(signal.SIGINT, signal_handler)

detector = snowboydecoder.HotwordDetector(model, sensitivity=0.5)
print('Listening... Press Ctrl+C to exit')

# main loop
detector.start(detected_callback=snowboydecoder.play_audio_file,
               interrupt_check=interrupt_callback,
               sleep_time=0.03)

detector.terminate()
```

Demo4

- Integrate with google speech recognition.

$ python3 demo4.py resources/models/computer.umdl

```
^[[A^Cpi@rpi4-A00:~/snowboy/examples/Python3 $ python3 demo4.py resources/models
/computer.umdl
Listening... Press Ctrl+C to exit
INFO:snowboy:Keyword 1 detected at time: 2020-12-07 00:43:17
recording audio...converting audio to text
Internet of Things
```

- https://github.com/Kitt-AI/snowboy/blob/master/examples/Python3/demo4.py

Personal Models

- https://github.com/seasalt-ai/snowboy
- DNN based hotword and wake word detection toolkit
- Build your own personal models (Ubuntu 16.04 and macOS)

Alternatives

- https://github.com/MycroftAI/mycroft-precise
  - A lightweight, simple-to-use, RNN wake word listener.
- https://picovoice.ai/
  - Picovoice is the end-to-end platform for adding voice to anything.

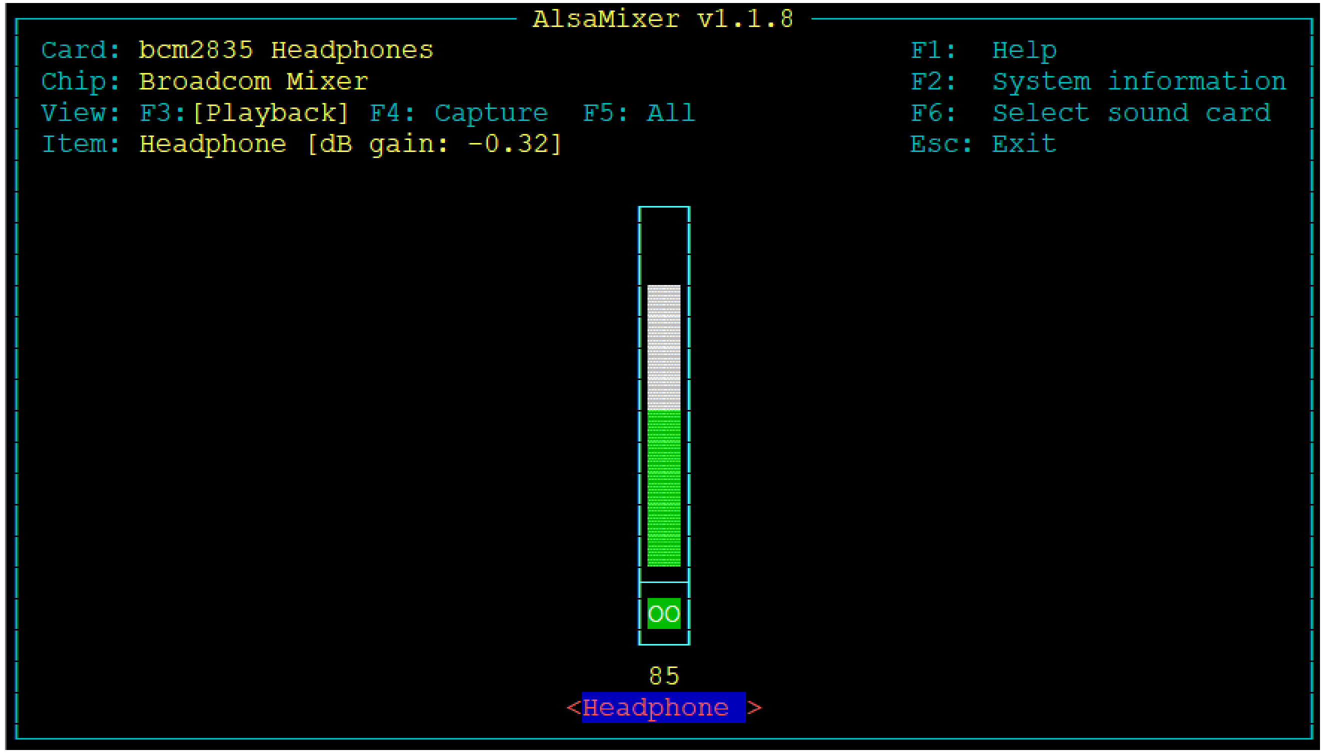
Outline

- Introduction
- Installation
- Speech to Text
- Text to Speech
- Discussion & Lab

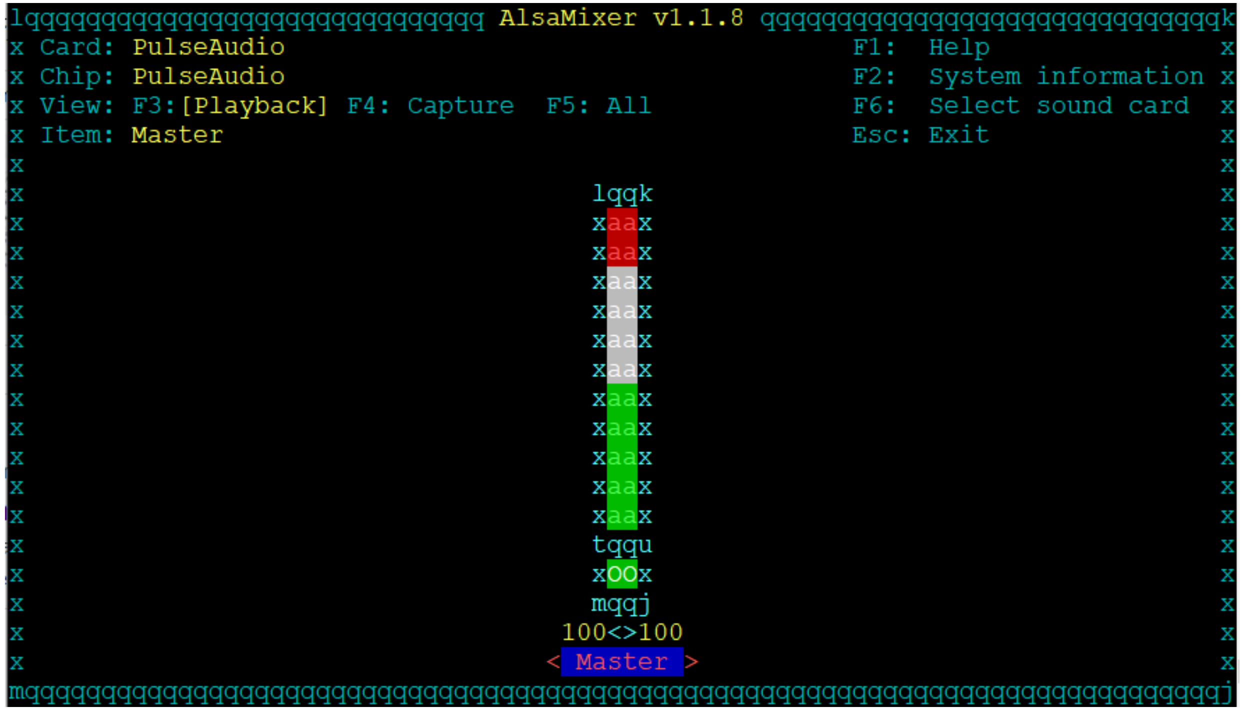
Gain Adjustment

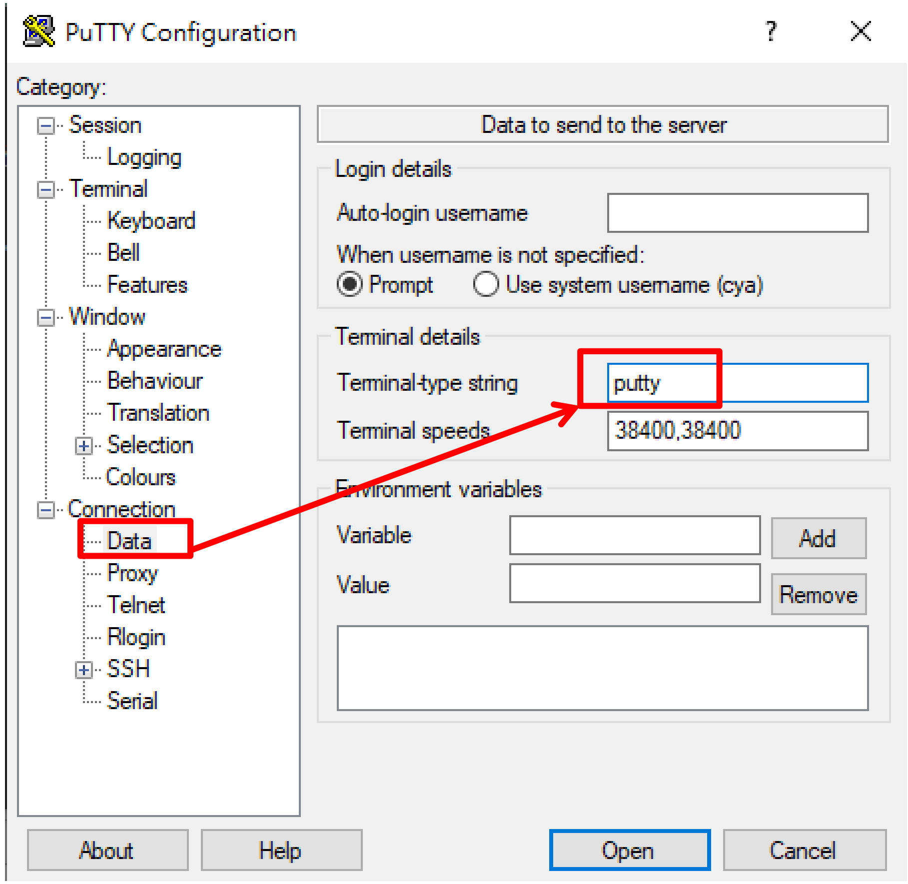
$ alsamixer

- Tune the volume of the output from 3.5 audio jack

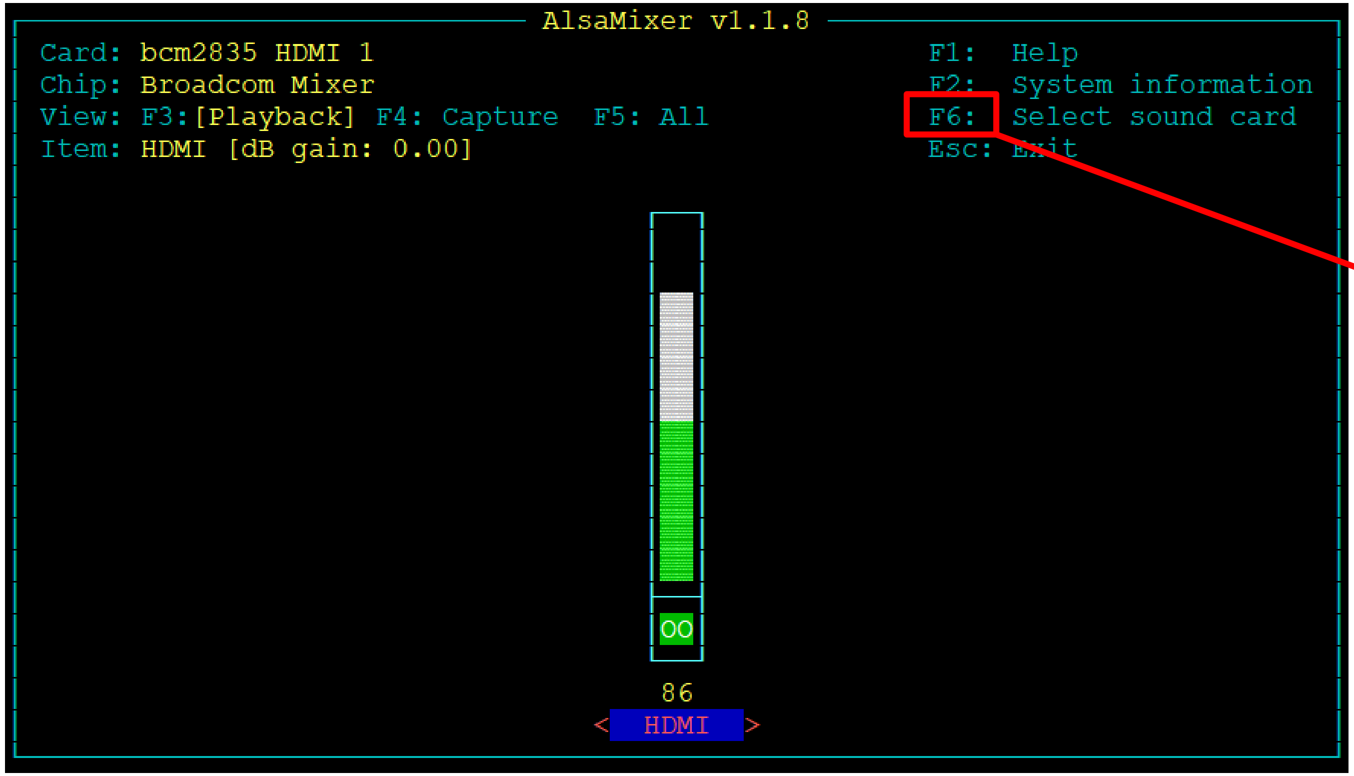


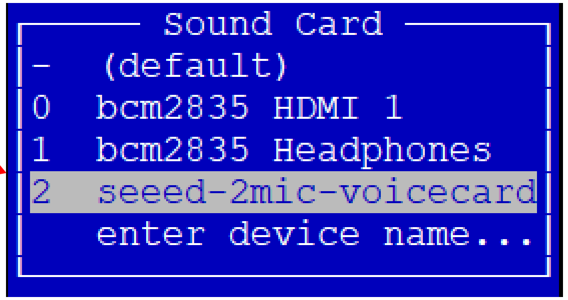
- If you don’t want the garbled text, change the string type





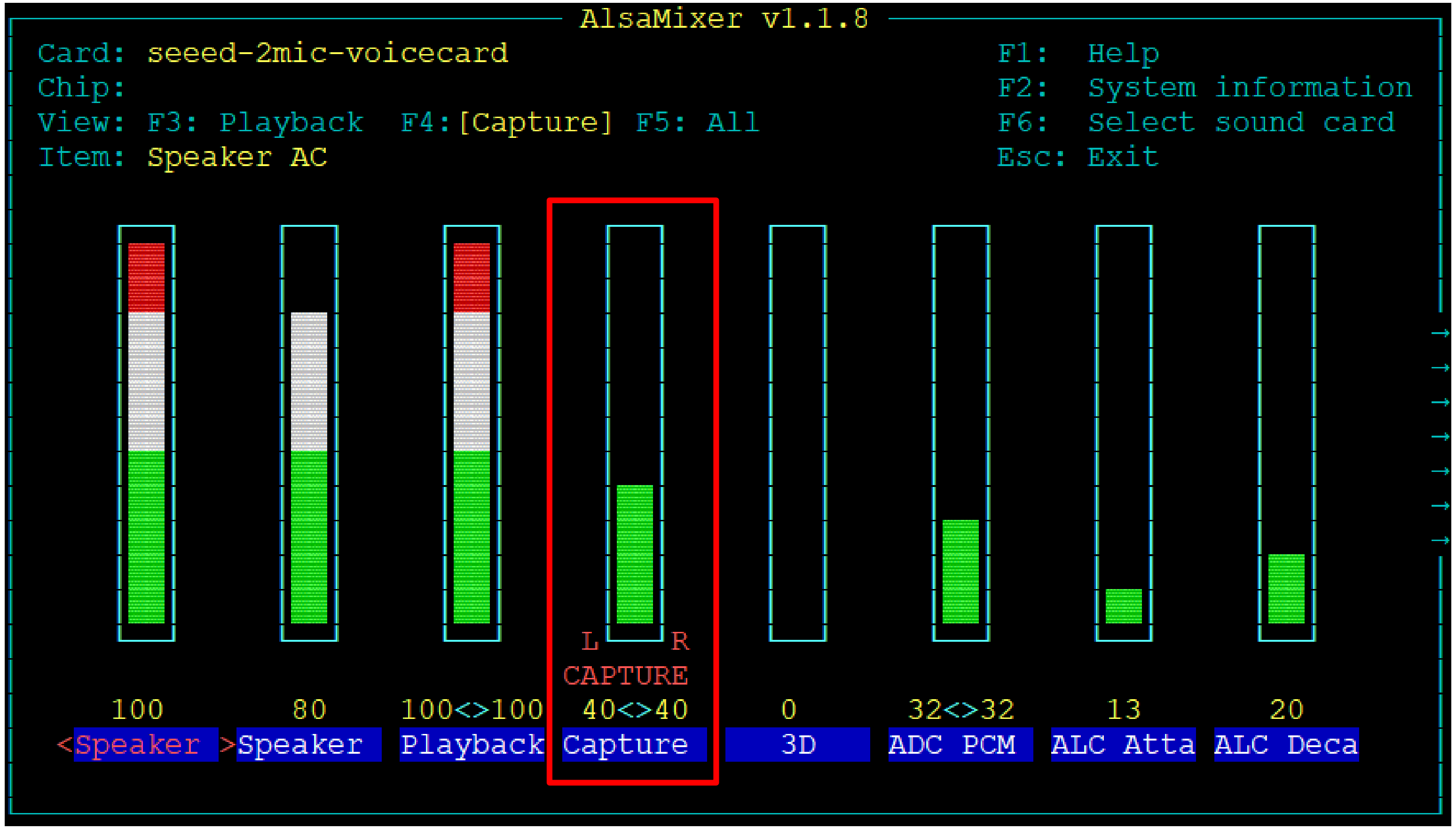
- Press “F6” to select the voice card.





Gain Adjustment

- Tune the capturing gain of seeed-2mic



Discussion

- How to make r.recognize_google(audio) understand other languages?
- https://github.com/Uberi/speech_recognition/blob/master/speech_recognition/__init__.py

```
def recognize_google(self, audio_data, key=None, language="en-US", pfilter=0, show_all=False):
```

- The recognition language is determined by ``language``, an RFC5646 language tag like ``"en-US"`` (US English) or ``"fr-FR"`` (International French), defaulting to US English.
- A list of supported language tags can be found in this StackOverflow answer <http://stackoverflow.com/a/14302134>.

Discussion

- How to make gTTS speak other language?

gTTS (`gtts.gTTS`)

class `gtts.tts.gTTS`(*text, lang='en', slow=False, lang_check=True, pre_processor_funcs=[<function tone_marks>, <function end_of_line>, <function abbreviations>, <function word_sub>], tokenizer_func=<bound method Tokenizer.run of re.compile('(?<=\?).|(?<=\!).|(?<=\？).|(?<=\！).|(?<!\.[a-z])\. |(?<!\.[a-z])\, |(?<!\d)\:|\：|\、|\—|\，|\¿|\…|\n|\]|\،|\。|\(|\¡|\;|\‥|\[|\)', re.IGNORECASE) from: [<function tone_marks>, <function period_comma>, <function colon>, <function other_punctuation>]>)* [source]

- **lang** (*string, optional*) – The language (IETF language tag) to read the text in. Defaults to 'en'.

Lab - Voice Controlled LED

- Integrate the following functions
  - Realtime Speech-to-Text
  - Text-to-Speech
  - Control on-board LEDs
- Periodically listen to the speech from the user.
  - You can specify a timeout to control the recording time.
  - Or use hotword detector.

```
recognizer_instance.listen(source: AudioSource, timeout: Union[float, None] = None, phrase_time_limit: Union[float, None] = None, snowboy_configuration: Union[Tuple[str, Iterable[str]], None] = None) -> AudioData
```

The `timeout` parameter is the maximum number of seconds that this will wait for a phrase to start before giving up and throwing an `speech_recognition.WaitTimeoutError` exception. If `timeout` is `None`, there will be no wait timeout.

- Turn the LED on when you say “開燈”.
  - Recognize the speech by STT “開燈” in the string.
  - Then output an audio “燈已開啟” by TTS.
- Turn the LED off when you say “關燈”.
  - Recognize the speech by STT “關燈” in the string.
  - Then output an audio “燈已關閉” by TTS.

On-Board LEDs

$ wget https://raw.githubusercontent.com/respeaker/mic_hat/master/interfaces/apa102.py

$ nano led.py

```
import apa102
import time

LED_NUM = 3

leds = apa102.APA102(num_led=3)
colors = [[255,0,0],[0,255,0],[0,0,255]] # LED0: R, LED1: G, LED2: B

try:
    while True:
        for i in range(LED_NUM):
            leds.set_pixel(i, colors[i][0], colors[i][1], colors[i][2], 10)
        leds.show()
        time.sleep(1)
        leds.clear_strip()
        time.sleep(1)
except:
    pass
finally:
    leds.clear_strip()
    leds.cleanup()
```

$ python3 led.py

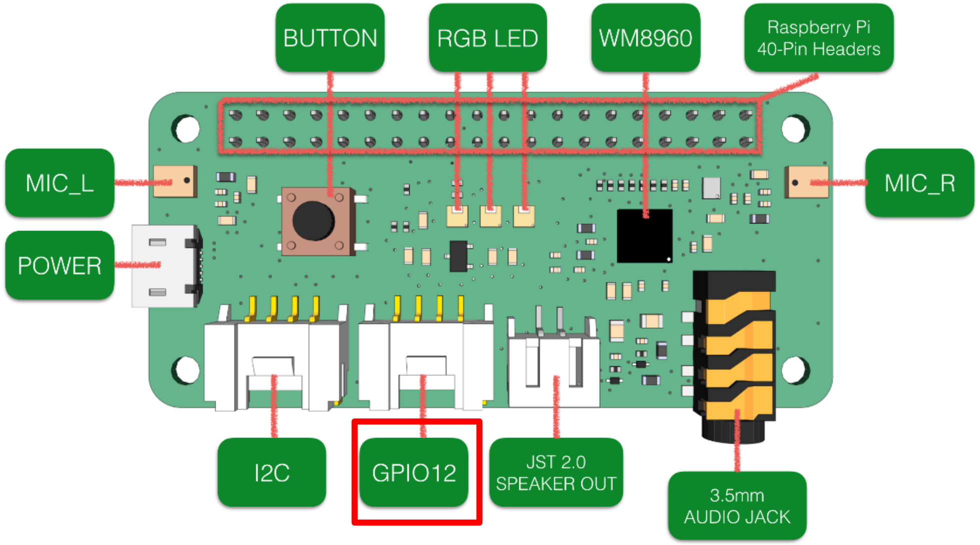
Hint

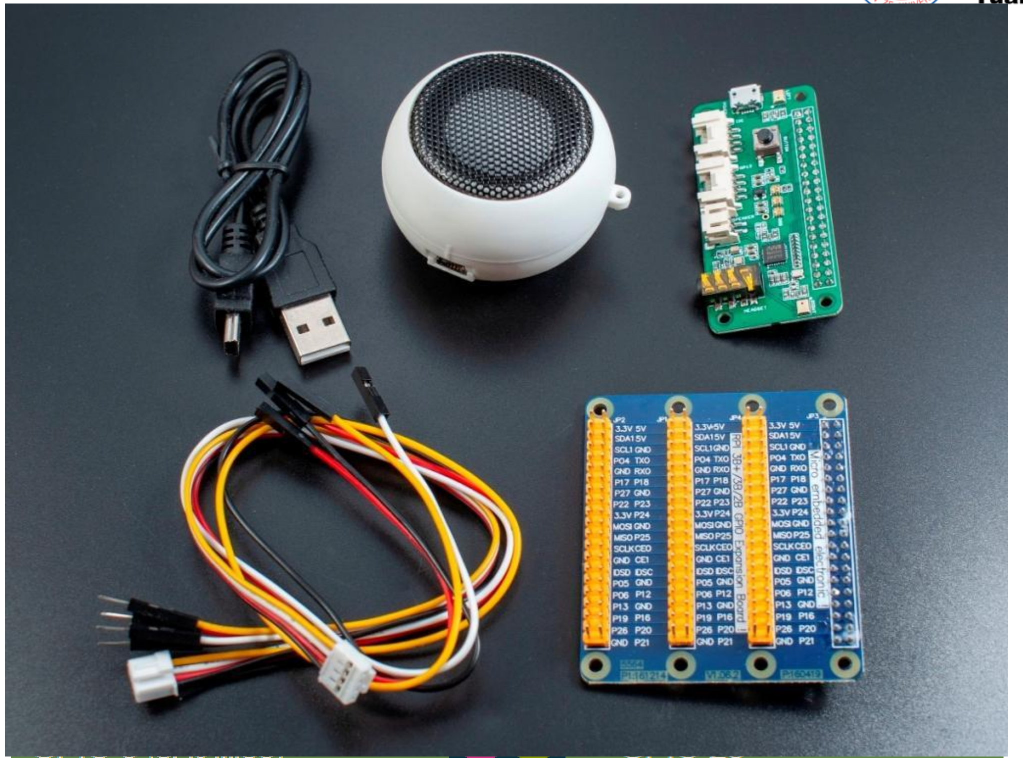
- Substring comparison

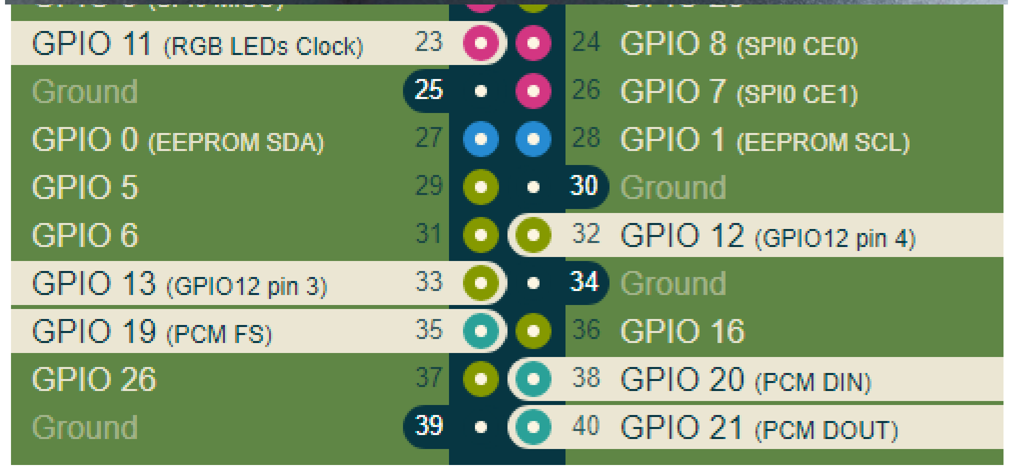
```
Type "help", "copyright", "credits" or "license" for more information.
>>> cmd = "我要開燈"
>>> "開燈" in cmd
True
>>>
```

Reference

- Use PIN32 by groove cable.
- Or use expansion board.



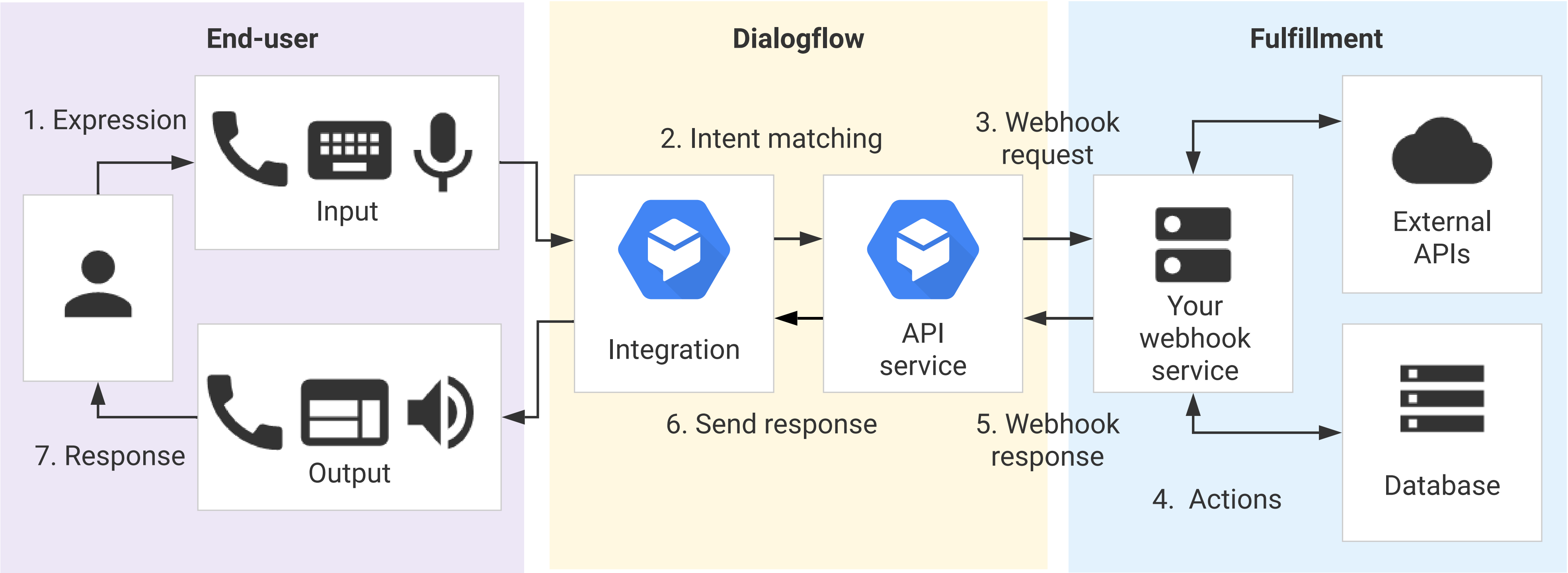




GPIO 11 (RGB LEDs Clock)	23	24	GPIO 8 (SPI0 CE0)
Ground	25	26	GPIO 7 (SPI0 CE1)
GPIO 0 (EEPROM SDA)	27	28	GPIO 1 (EEPROM SCL)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (GPIO12 pin 4)
GPIO 13 (GPIO12 pin 3)	33	34	Ground
GPIO 19 (PCM FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM DIN)
Ground	39	40	GPIO 21 (PCM DOUT)

NLP

- Dialogflow is a natural language understanding platform



Ref: https://cloud.google.com/dialogflow/es/docs/basics