

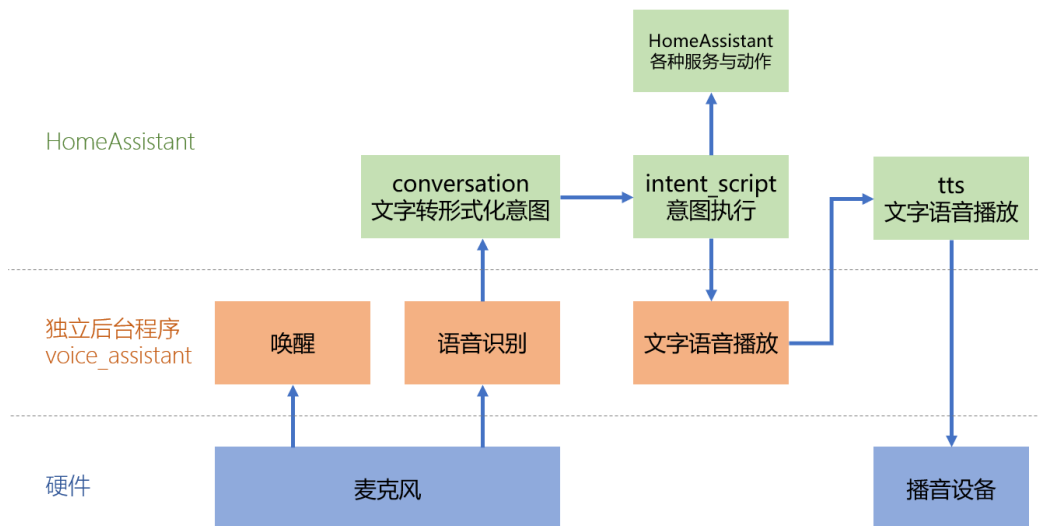
DIY 智能音箱（1）——整体架构、硬件安装

【操作步骤】

1. 项目目标
2. 整体架构讲解
3. 拾音与播音设备安装
4. 设置缺省音频输入与输出，并进行测试

【参考】

- 项目目标
 - 具有一般智能音箱的功能
 - 尽量使用成熟的开源的开放的项目
 - 架构模块化，保持组装与配置的自由度
- 架构图



- seeed 双麦克风树莓派扩展板
http://wiki.seeedstudio.com/ReSpeaker_2_Mics_Pi_HAT/



驱动安装：

```
git clone https://github.com/respeaker/seeed-voicecard
cd seeed-voicecard
sudo ./install.sh
sudo reboot
```

- 播放与录音命令

```
arecord -l #列出所有录音设备
arecord -f cd -d 6 -Dhw:1,0 test.wav #以 cd 音质录制 6 秒钟音频，保存到 test.wav 文件，hw:1,0 为录音设备
aplay -l #列出所有播放设备
aplay test.wav #播放 test.wav
```

- 注：如果大家不使用本地麦克风，直接使用远程麦克风，可以参见后续的视频《音乐灯带》与《远程麦克风》

DIY 智能音箱（2）——snowboy、speech_recognition

【操作步骤】

1. 安装需要的基础库
2. 下载与测试唤醒词服务 snowboy
3. 安装与测试 SpeechRecognition

【参考】

- 安装必须的基础库

```
sudo apt-get install python-pyaudio python3-pyaudio flac libpcrc3 libpcrc3-dev libatlas-base-dev swig
```

- 下载 snowboy 唤醒服务

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

```
cd snowboy/swig/Python3
```

```
make
```

- snowboy 测试

```
cd ../../examples/Python3
```

修改 snowboydecoder.py 中

```
from . import snowboydetect 变为 import snowboydetect
```

```
python3 demo.py ../../resources/models/snowboy.umd1
```

注：一些 usb 麦克风不支持 16000 的采样率，可能无法通过测试——这种情况并不影响后续的使用；可以参见后面的注

- 安装 SpeechRecognition 并修正

```
sudo pip3 install SpeechRecognition
```

```
cd /usr/local/lib/python3.5/dist-packages/speech_recognition/ #若 Speech_Recognition 库在其它位置，修改此处目录
```

```
sudo mv __init__.py __init__.py.bak
```

```
sudo wget https://github.com/zhujiasheng/Home-Assistant-DIY/raw/master/__init__.py
```

- 测试 SpeechRecognition

创建文件~/voice_assistant/voice_assistant.py，权限 755，内容为：

```
#!/usr/bin/env python3
```

```
import speech_recognition as sr
```

```
# 从麦克风获得音频
```

```
r = sr.Recognizer()
```

```
with sr.Microphone(sample_rate=16000) as source:
```

```
    print("开始监听.....")
```

```
    audio = r.listen(source, phrase_time_limit=6)
```

```
# 使用 Google Speech Recognition CN 进行语音文字识别
```

```
print("开始识别.....")
```

```
result = r.recognize_google_cn(audio, language='zh-CN')
```

```
print("识别结果: " + result)
```

注：一些 usb 麦克风不支持 16000 的采样率，可以将程序中 sample_rate=16000 去除

- SpeechRecognition 项目

https://github.com/Uberi/speech_recognition

- SnowBoy 项目

<https://snowboy.kitt.ai/>

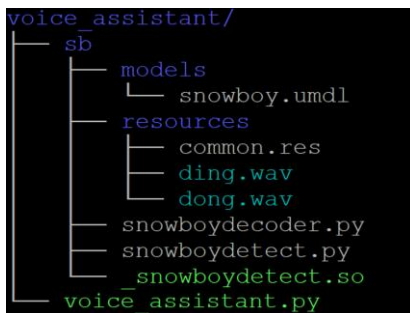
DIY 智能音箱（3）——完成主程序架构

【操作步骤】

1. 调整文件结构
2. 唤醒后再进行语音识别
3. 增加唤醒后提示音
4. 唤醒-识别，无限循环

【参考】

- 构建合理的文件结构



```
voice_assistant/
├── sb
│   ├── models
│   │   └── snowboy.umdl
│   ├── resources
│   │   ├── common.res
│   │   ├── ding.wav
│   │   └── dong.wav
│   ├── snowboydecoder.py
│   ├── snowboydetect.py
│   ├── snowboydetect.so
│   └── voice_assistant.py
```

```
mkdir -p ~/voice_assistant/sb/resources
cp ~/voice_assistant/snowboy/resources/common.res ~/voice_assistant/sb/resources/
cp ~/voice_assistant/snowboy/resources/d*.wav ~/voice_assistant/sb/resources/
mkdir -p ~/voice_assistant/sb/models
cp ~/voice_assistant/snowboy/resources/models/snowboy.umdl ~/voice_assistant/sb/models/
cp ~/voice_assistant/snowboy/swig/Python3/_snowboydetect.so ~/voice_assistant/sb/
cp ~/voice_assistant/snowboy/swig/Python3/snowboydetect.py ~/voice_assistant/sb/
cp ~/voice_assistant/snowboy/examples/Python3/snowboydecoder.py ~/voice_assistant/sb/
```

- 样例程序

```
#!/usr/bin/env python3

import speech_recognition as sr

snowboy_location = '/home/pi/voice_assistant/sb/'
snowboy_models = ['/home/pi/voice_assistant/sb/models/snowboy.umdl']
snowboy_config = (snowboy_location, snowboy_models)

import sys
sys.path.append(snowboy_location)
import snowboydecoder
sys.path.pop()

r = sr.Recognizer()

with sr.Microphone(sample_rate=16000) as source:
    while True:
        try:
            print("开始监听.....")
            audio = r.listen(source,
                              phrase_time_limit=6,
                              snowboy_configuration=snowboy_config,
                              hot_word_callback=snowboydecoder.play_audio_file
                              )

            print("开始识别.....")
            snowboydecoder.play_audio_file(fname=snowboy_location+'resources/dong.wav')
            result = r.recognize_google_cn(audio, language='zh-CN')
        except sr.UnknownValueError:
            result = ''
        except Exception as e:
            print("识别错误: {0}".format(e))
            continue
        print("识别结果: " + result )
```

DIY 智能音箱（4）——与 HomeAssistant 交互

【操作步骤】

1. conversation 与 intent_script 组件配置
2. 修改主程序
 - a) 引入 ha_cli 库
 - b) 获得 HA 的访问 token
 - c) 增加命令文本处理、播放反馈文本
3. 调整未定义命令的处理

【参考】

- 配置样例

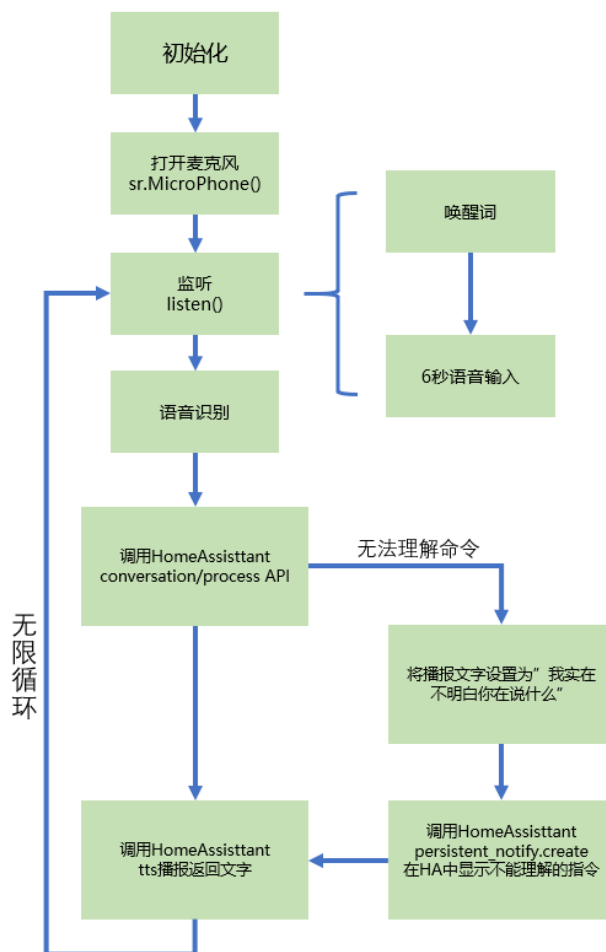
```
conversation:
  intents:
    AboutEat:
      - ".*(?:吃|饿|饱).*"

intent_script:
  AboutEat:
    speech:
      text: 饿了就吃点, 吃饱了就歇歇
```

- 获得 ha_cli.py

wget https://github.com/zhujisheng/Home-Assistant-DIY/raw/master/ha_cli.py

- 程序架构



● 样例程序

```
#!/usr/bin/env python3

import speech_recognition as sr
from ha_cli import ha_cli

#此处替换为你的 HomeAssistant 的 token
ha_token='eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJhbnNwI2OTZlZmM0ZjU3YTlhMmExOTYzNjBjNmEwNSIsImV4cCI6MTg2NTY2NTYxMiwiYWFWIjoxNTUwMzA1NjE5fQ.H_4vjf-Ta0HbNVw7bNe8pwh9n9BwQb1lfFR67XeX_pi'
snowboy_location = '/home/pi/voice_assistant/sb/'
snowboy_models = ['/home/pi/voice_assistant/sb/models/snowboy.umd1']
snowboy_config = (snowboy_location, snowboy_models)

import sys
sys.path.append(snowboy_location)
import snowboydecoder
sys.path.pop()

r = sr.Recognizer()
ha = ha_cli(token=ha_token)

with sr.Microphone(sample_rate=16000) as source:
    while True:
        try:
            print("开始监听.....")
            audio = r.listen(source,
                              phrase_time_limit=6,
                              snowboy_configuration=snowboy_config,
                              hot_word_callback=snowboydecoder.play_audio_file
                              )

            print("开始识别.....")
            result = r.recognize_google_cn(audio, language='zh-CN')
        except sr.UnknownValueError:
            result = ''
        except Exception as e:
            print("识别错误: {0}".format(e))
            continue
        print("识别结果: " + result)

    try:
        speech = ha.process(result)
        if speech == "Sorry, I didn't understand that":
            speech = result + "? 我实在不明白你在说什么"
            ha.note(message=result)
        # 如果 HA 中使用其它的 tts 组件, 此处改为对应的服务名 (缺省为 google_say)
        ha.speak(speech, tts='google_say')
    except Exception as e:
        print("与 HomeAssistant 通讯失败: {0}".format(e))
        continue
```

完善（1）——更好的音色、更多的指令

【操作步骤】

1. 使用 tts.baidu 组件
2. 赋予智能音箱更多的命令执行能力
 - a) 定义查询温度意图的命令文本与执行内容
 - b) 定义开关灯意图的命令文本与执行内容

【参考】

- tts.baidu 组件

<https://www.home-assistant.io/components/tts.baidu/>

配置：

```
tts:
  - platform: baidu
    app_id: 9931748
    api_key: yaEF9KGd6WvoXpyGMZxtX3Qj
    secret_key: 70e71c2425ddccb67439dafdcf9b999f
    person: 4
```

- 正则表达式参考

<http://tool.oschina.net/uploads/apidocs/jquery/regexp.html>

- conversation 组件

<https://www.home-assistant.io/components/conversation/>

- 视频中的意图配置

```
conversation:
  intents:
    RoomTemperature:
      - "现在多热"
      - "现在[室内]几度"
      - "[需][要]开空调吗"
      - ".*(?:温度|冷).*"
    OpenLight:
      - "打开(?:小米|小米网关|过道)?灯"
      - "把(?:小米|小米网关|过道)?灯打开"
    CloseLight:
      - "关闭(?:小米|小米网关|过道)?灯"
      - "把(?:小米|小米网关|过道)?灯关闭"

intent_script:
  RoomTemperature:
    speech:
      text: 当前室内{{states.sensor.entity_id.state}}度
  OpenLight:
    speech:
      text: 正在打开小米灯
    action:
      service: light.turn_on
      data:
        entity_id: light.entity_id
  CloseLight:
    async_action: true
    speech:
      text: 正在关闭小米灯
    action:
      service: light.turn_off
      data:
        entity_id: light.entity_id
```

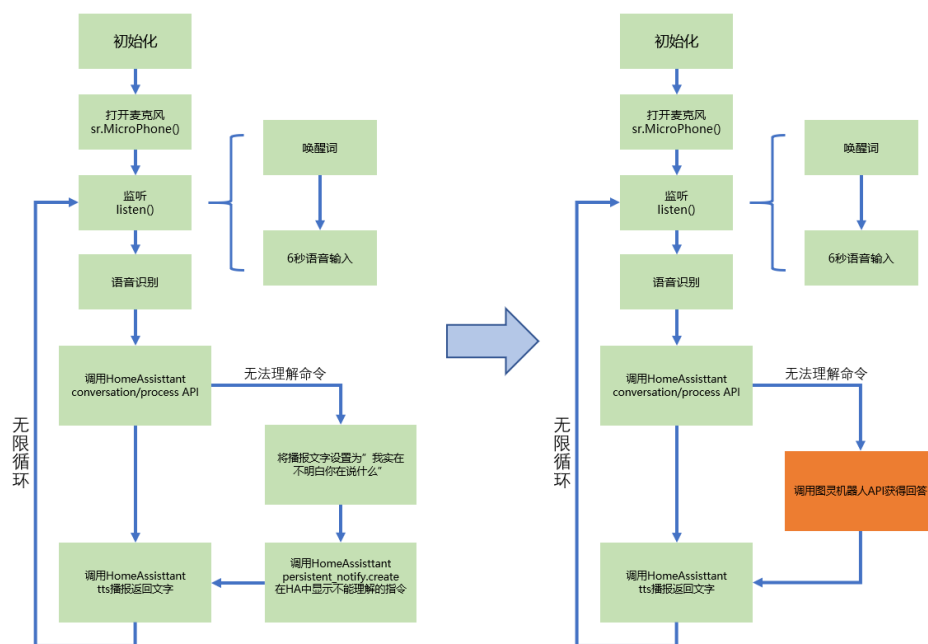
完善（2）——准确回答任意问题

【操作步骤】

1. 创建属于你的图灵机器人
2. 获得 ais_cli.py 文件
3. 修改 voice_assistant.py 访问图灵机器人

【参考】

- 修改程序



- 图灵机器人

<http://www.turingapi.com/>

- 获得图灵 API 访问 python 代码

`wget https://github.com/zhujisheng/Home-Assistant-DIY/raw/master/ais_cli.py`

- 程序代码

```
#!/usr/bin/env python3
```

```
import speech_recognition as sr
from ha_cli import ha_cli
from ais_cli import tuling123
```

```
tuling_user_id = '403981'
```

```
tuling_api_key = 'ddb64bbf5f47466eae4f3ccb5fab9410'
```

```
ha_token='eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsInR5cCI6IkpzZW50b3R0IiwiaWF0IjoiMTYxMjY2NTYxMiwiaWAwF0IjozNTUwMzA1NjE5fQ.H_4vjF-Ta0HbNVw7bNe8pwh9n9BwQb11fFR67XeX_pI'
```

```
snowboy_location = '/home/pi/voice_assistant/sb/'
snowboy_models = ['/home/pi/voice_assistant/sb/models/snowboy.umd1']
snowboy_config = (snowboy_location, snowboy_models)
```

```
import sys
sys.path.append(snowboy_location)
import snowboydecoder
sys.path.pop()
```

```
r = sr.Recognizer()
ha = ha_cli(token=ha_token)
tuling = tuling123(user_id=tuling_user_id, api_key=tuling_api_key)
```

```

with sr.Microphone(sample_rate=16000) as source:
    while True:
        try:
            print("开始监听.....")
            audio = r.listen(source,
                              phrase_time_limit=6,
                              snowboy_configuration=snowboy_config,
                              hot_word_callback=snowboydecoder.play_audio_file
                              )

            print("开始识别.....")
            snowboydecoder.play_audio_file(fname=snowboy_location+'resources/dong.wav')
            result = r.recognize_google_cn(audio, language='zh-CN')
        except sr.UnknownValueError:
            result = ''
        except Exception as e:
            print("识别错误: {0}".format(e))
            continue
        print("识别结果: " + result)

    try:
        speech = ha.process(result)
        if speech == "Sorry, I didn't understand that":
            speech = tuling.command(result)
            ha.note(message=result)
            ha.speak(speech, tts='baidu_say')
    except Exception as e:
        print("与 HomeAssistant 通讯失败: {0}".format(e))
        continue

```


完善 (3) ——自定义唤醒词与敏感度

【操作步骤】

1. 在程序中设置多个唤醒词
2. 制作自己的唤醒词模型文件
 - a) 安装 sox
 - b) 录制三个 wav 音频文件
 - c) 下载并修改制作程序，制作模型文件
 - d) 将模型文件放置在对应位置，并修改主程序
3. 修订 speech_recognition 中唤醒词敏感度
4. 实验你的唤醒词

【参考】

- 复制 snowboy 项目中的唤醒词模型文件
`cp ~/voice_assistant/snowboy/resources/models/*.umdl ~/voice_assistant/sb/models/`
- snowboy 自定义唤醒词 API
<http://docs.kitt.ai/snowboy/#restful-api-calls>
- 安装 sox
`sudo apt-get install sox`
- 录音命令
`rec -r 16000 -c 1 -b 16 -e signed-integer 1.wav`
- 制作唤醒词命令
`python2 training_service.py 1.wav 2.wav 3.wav saved_model.pmdl`
- 修订 speech_recognition 中唤醒词敏感度
在 `/usr/local/lib/python3.5/dist-packages/speech_recognition/__init__.py` 文件中设置敏感度
`detector.SetSensitivity("".join(["0.45"] * len(snowboy_hot_word_files)).encode())`
- 关于唤醒词的敏感度
 - ✓ 如果仅是非常有限样本生成的唤醒词模型，建议将敏感度设置在 0.5 以下，防止误识别
 - ✓ 如果使用 snowboy 项目中带的唤醒词模型，可以将敏感度设置在 0.8 以上，防止漏识别
 - ✓ 所以，不是很建议两者混用
 - ✓ 如果要降低自己生成模型的误识别率和漏识别率，可以在登录 <https://snowboy.kitt.ai/dashboard> 后使用更多样本进行训练。
 - ✓ 敏感度参数目前是作为常数写在程序中的，你如果有兴趣，可以调整 `/usr/local/lib/python3.5/dist-packages/speech_recognition/__init__.py`，让它作为一个传入 `listen` 函数的参数。
 - ✓ 你还可以调整程序，让设置的多个唤醒词模型具有不同的敏感度（也是调整 `/usr/local/lib/python3.5/dist-packages/speech_recognition/__init__.py`）

完善（4）——使用微软语音识别服务

【操作步骤】

1. 申请微软 azure 云/认知服务/语音 API 的密钥
2. 在主程序中，使用微软语音识别服务
3. 使用 logging 库

【参考】

- SpeechRecognition 项目
https://github.com/Uber/speech_recognition

- 微软 azure 云
<https://azure.microsoft.com/>

- 微软语音识别服务的调用

```
result = r.recognize_azure(audio, language='zh-CN', key='17940a7f7d99461b858bf5cd39fe0ced', location='westus')
import re
result = re.sub(r'^\w\s', '', result)
```

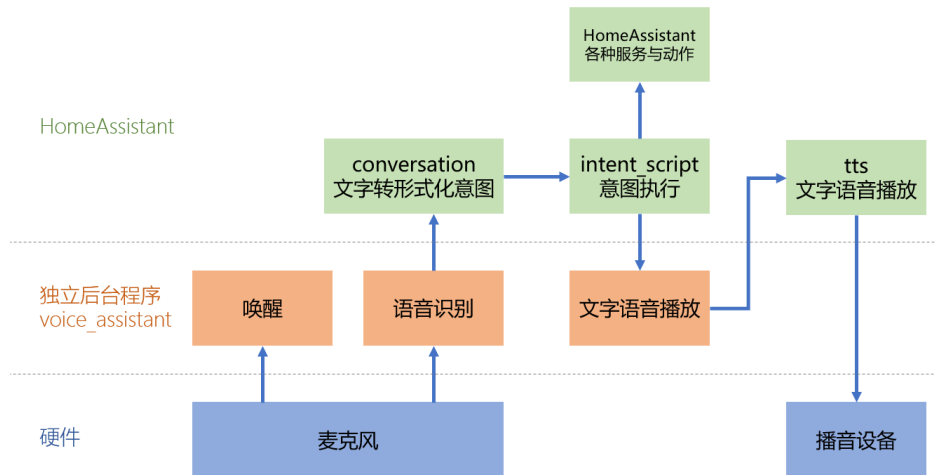
- 使用 logging 替代 print

```
import logging
logging.basicConfig(format='[%(levelname)s] %(asctime)s %(message)s', level=logging.INFO)
# 使用 logging.info(或者 warning、error 等，详见 logging 库的说明)替代 print
```

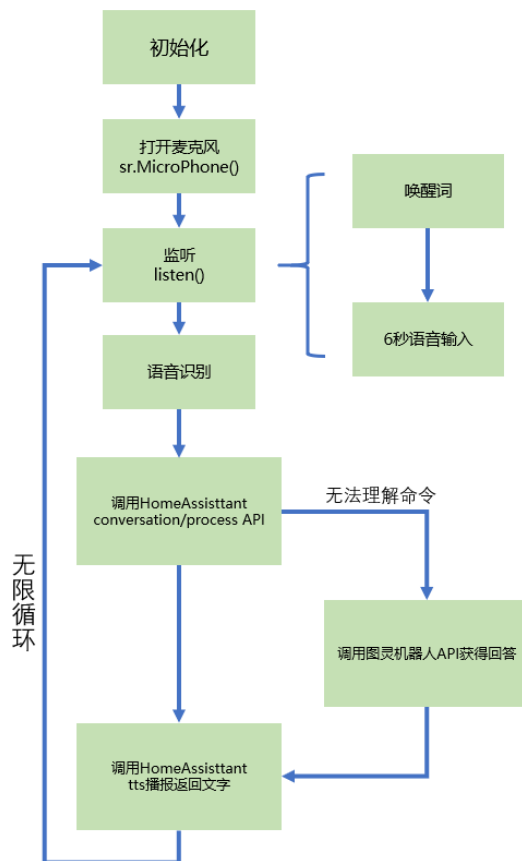
最后一课——积木构建智慧空间

【参考】

- 系统结构



- 程序逻辑



- 完整带注释的 voice_assistant.py 程序（你需要修改其中着色部分）

```
#!/usr/bin/env python3

# speech_recognition 库: https://github.com/Uberi/speech_recognition
import speech_recognition as sr

# 这两个 python 程序简单封装了访问 HA 和图灵机器人的 API
from ha_cli import ha_cli
```

```

from ais_cli import tuling123

# 标准的 python 记录日志的库
import logging

# 设置日志格式为“[日志级别] 日志时间 日志内容”
logging.basicConfig(format='[% (levelname)s] % (asctime)s % (message)s', level=logging.INFO)

# 访问图灵机器人的 user_id 和 api_key, 你需要自己申请获得 (免费)
tuling_user_id = '407535'
tuling_api_key = '8bee51f9bca64295a5c7f34b2eeb81ed'

# 访问 HomeAssistant 的 token, 你需要在你的 HomeAssistant 的 WEB 前端生成
ha_token='eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOiJlNTA1ODY4MDIsIm1zcyI6IjRlODRjNjJiYTl3NzQxNzA5ZjgyYkZlZWYyZmJgZmIiwiZmxhZjoxODY1OTQ2ODAyfQ.YK5X47ucA5Ug2NlRNa0VE7Z6bZruwthIdPzFAVT7eYA'

# snowboy 项目的位置, 以及你所使用的唤醒词模型文件。
# 模型文件可以设置一个或多个, 你也可以生成自己的唤醒词
snowboy_location = '/home/pi/voice_assistant/sb/'
snowboy_models = ['/home/pi/voice_assistant/sb/models/snowboy.umdl',
                  '/home/pi/voice_assistant/sb/models/smart_mirror.umdl']
snowboy_config = (snowboy_location, snowboy_models)

# import snowboy 库
# 如此引入, 是因为 snowboydecoder.py 并不在当前目录, 也不在系统目录中
import sys
sys.path.append(snowboy_location)
import snowboydecoder
sys.path.pop()

# 构建 speech_recognition 的 Recognizer 类
r = sr.Recognizer()

# 构建访问 HomeAssistant 和图灵机器人的客户端类
ha = ha_cli(token=ha_token)
tuling = tuling123(user_id=tuling_user_id, api_key=tuling_api_key)

# 打开麦克风, 采样率 16000
with sr.Microphone(sample_rate=16000) as source:
    # 无限循环不退出, 不断进行唤醒、识别、动作.....
    while True:
        # try 防止碰到异常整个程序退出
        try:
            logging.info("开始监听.....")

            # 监听, 使用 snowboy 唤醒后, 监听 6 秒, 获得指令的语音
            audio = r.listen(source,
                             phrase_time_limit=6,
                             snowboy_configuration=snowboy_config,
                             hot_word_callback=snowboydecoder.play_audio_file
                             )

            logging.info("开始识别.....")

            # 播放监听结束提示音
            snowboydecoder.play_audio_file(fname=snowboy_location+'resources/dong.wav')

            # 使用 google 的语音识别服务
            result = r.recognize_google_cn(audio, language='zh-CN')
            # 你也可以选择使用微软的语音识别服务
            #result = r.recognize_azure(audio, language='zh-CN', key='7a393a3b7954490dab750a490b264f27',
            location='westus')

            #import re
            #result = re.sub(r'[^\w\s]', '', result)
        except sr.UnknownValueError:
            result = ''
        except Exception as e:
            logging.warning("识别错误: {0}".format(e))
            continue

```

```

logging.info("识别结果: " + result)

# 防止 HomeAssistant 连接不上或其它错误时，整个程序退出
try:
    # 通过 HomeAssistant 的 API，将语音识别结果发送给 conversation 组件
    speech = ha.process(result)

    # 如果 conversation 未定义对应指令
    if speech == "Sorry, I didn't understand that":
        # 将语音识别结果发送给图灵机器人，获得回答
        speech = tuling.command(result)

    # 调用 HomeAssistant 的 persistent_notification.create 服务，在 HA 前端显示通知消息
    ha.note(message=result)

    # 调用 HomeAssistant 的 baidu_say，播放指令应答信息
    # 如果没有配置 tts.baidu，可以使用'google_say'
    ha.speak(speech, tts='baidu_say')

except Exception as e:
    logging.error("与 HomeAssistant 通讯失败: {0}".format(e))
    continue

```

● 加入服务

生成文件（需 root 权限）：/etc/systemd/system/voice_assistant.service

内容为：

```

[Unit]
Description=Voice Assistant
After=network.target

[Service]
Type=simple
User=pi
ExecStart=/home/pi/voice_assistant/voice_assistant.py

[Install]
WantedBy=multi-user.target

```

● 设置快捷命令

```

echo sudo systemctl enable voice_assistant > ~/bin/va-autostart
echo sudo systemctl disable voice_assistant > ~/bin/va-deautostart
echo sudo systemctl restart voice_assistant > ~/bin/va-restart
echo sudo systemctl start voice_assistant > ~/bin/va-start
echo sudo systemctl stop voice_assistant > ~/bin/va-stop
echo sudo journalctl -fu voice_assistant > ~/bin/va-log
chmod +x ~/bin/va-*

```