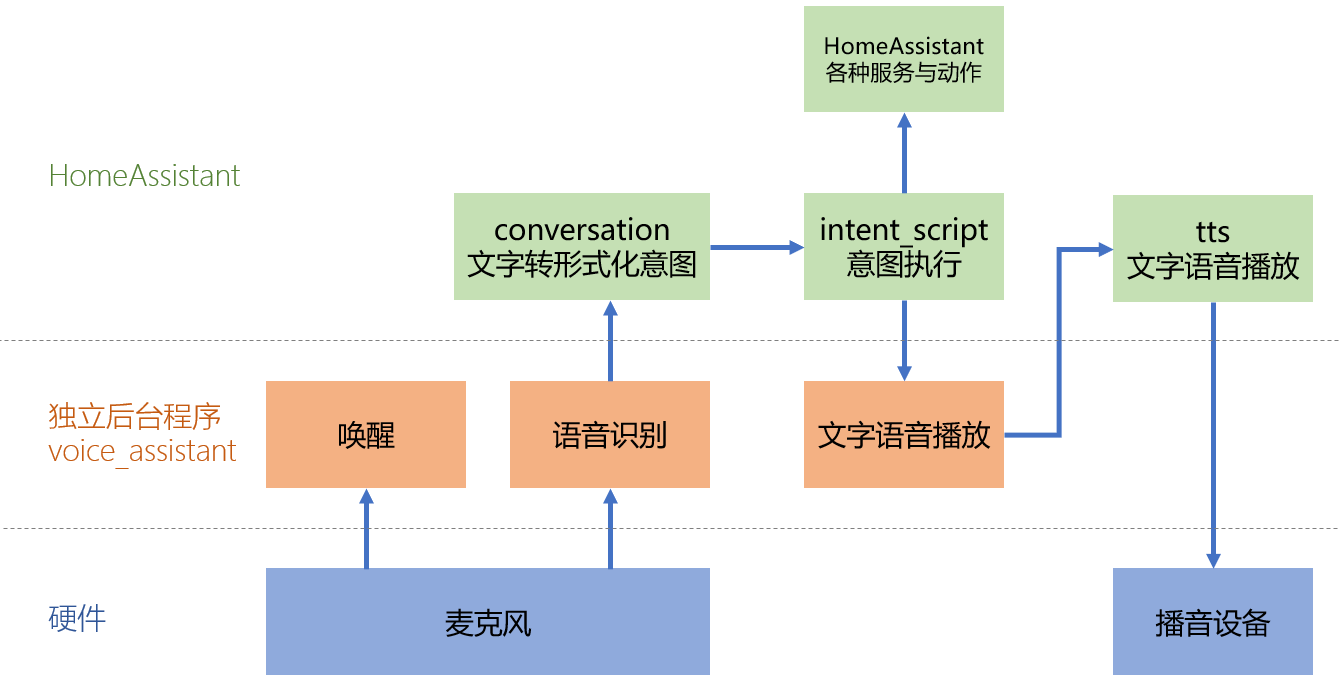
DIY智能音箱（1）——整体架构、硬件安装

【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 libpcre3 libpcre3-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.umdl

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/zhujisheng/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/>

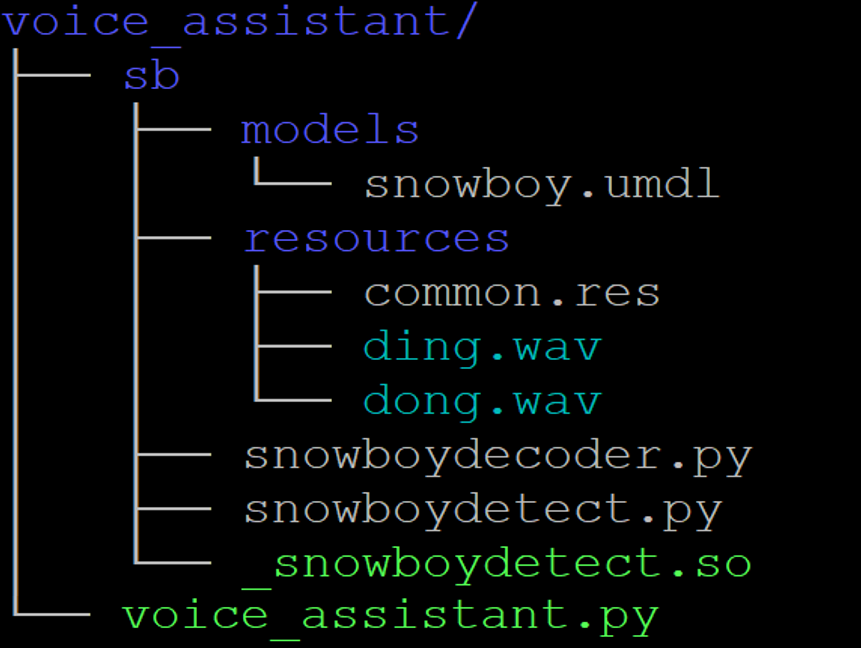
DIY智能音箱（3）——完成主程序架构

【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



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
   1. 引入ha\_cli库
   2. 获得HA的访问token
   3. 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:

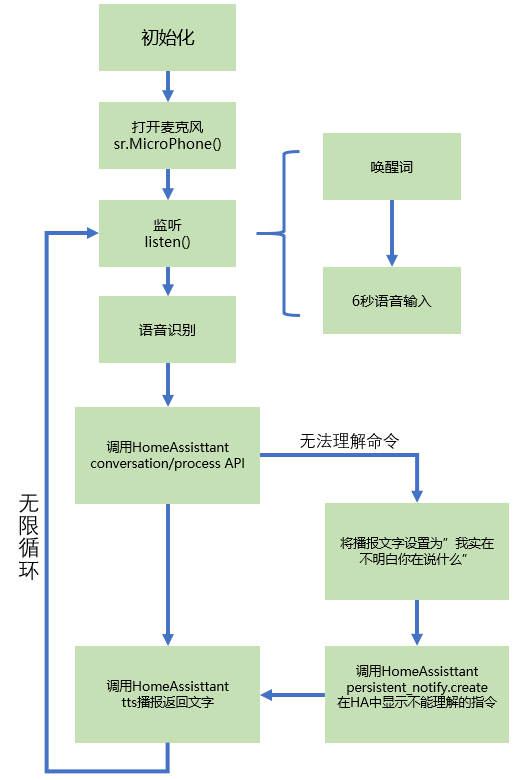
speech:

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.eyJpc3MiOiJhNWI2OTZhNGI4NjM0ZjU3YTlhMmExOTYzNjBjNmEwNSIsImV4cCI6MTg2NTY2NTYxMiwiaWF0IjoxNTUwMzA1NjEyfQ.H\_4vjF-Ta0HbNVw7bNe8pWh9n9BwQb1lfFR67XeX\_pI'

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()

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
   1. 定义查询温度意图的命令文本与执行内容
   2. 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: YaEF9KGD6WvoXpvGMZxtX3Qj

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

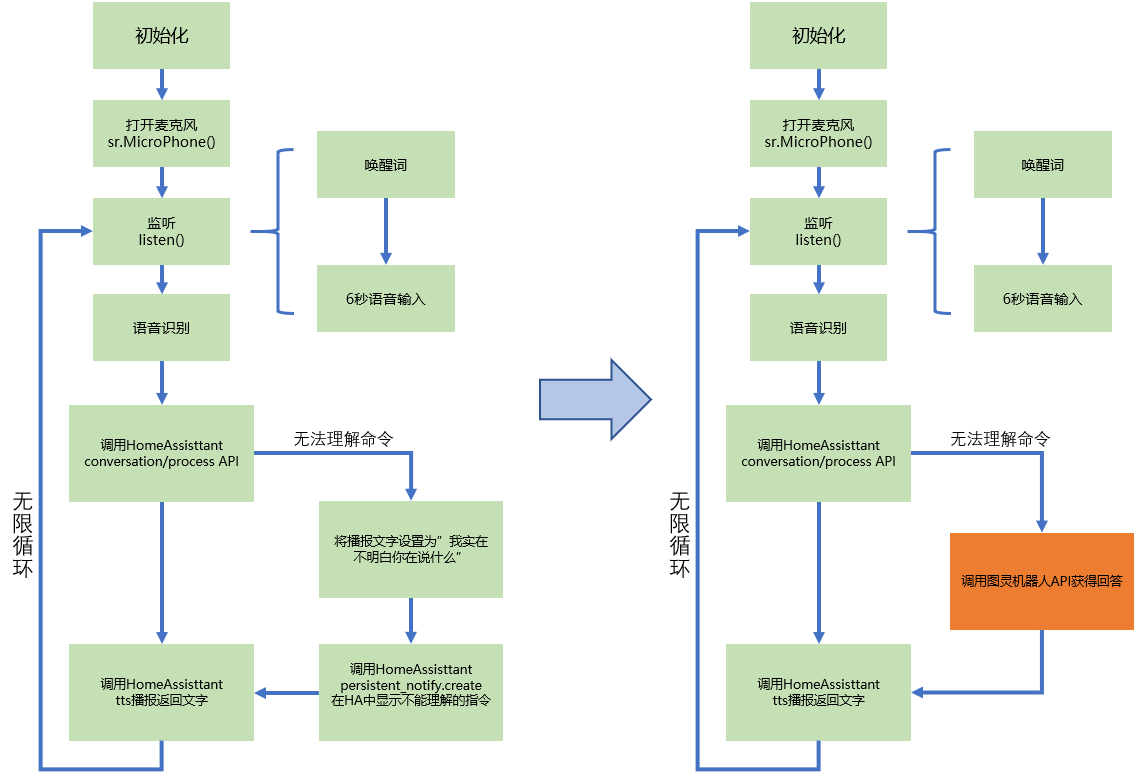
完善（2）——准确回答任意问题

【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='eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJhNWI2OTZhNGI4NjM0ZjU3YTlhMmExOTYzNjBjNmEwNSIsImV4cCI6MTg2NTY2NTYxMiwiaWF0IjoxNTUwMzA1NjEyfQ.H\_4vjF-Ta0HbNVw7bNe8pWh9n9BwQb1lfFR67XeX\_pI'

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()

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
   1. 安装sox
   2. 录制三个wav音频文件
   3. 下载并修改制作程序，制作模型文件
   4. 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/Uberi/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

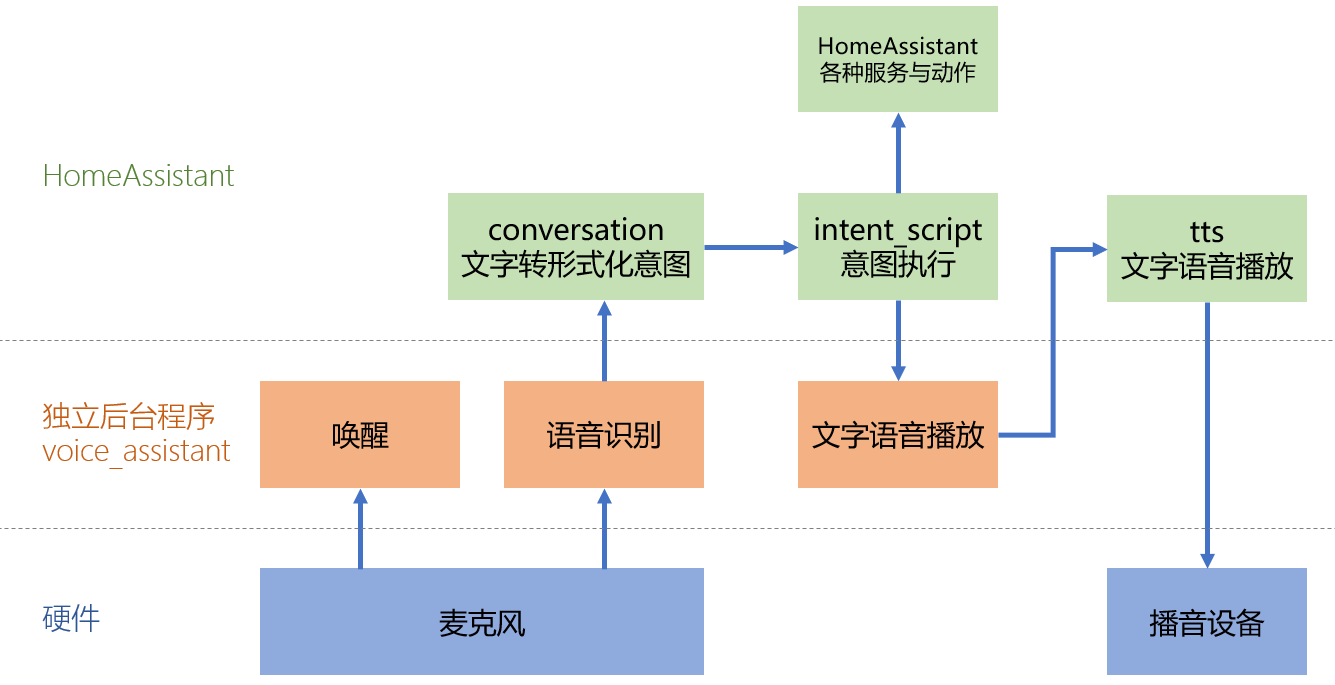
logging.basicConfig(format='[%(levelname)s] %(asctime)s %(message)s', level=logging.INFO)

# 使用logging.info(或者warning、error等，详见logging库的说明)替代print

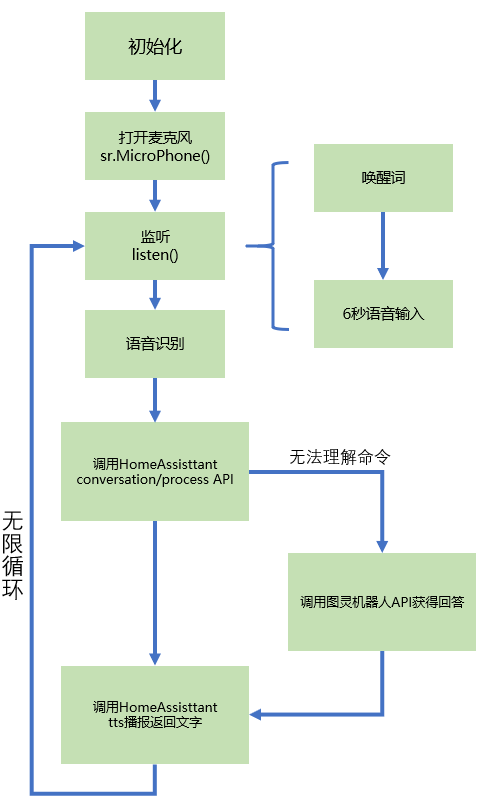
最后一课——积木构建智慧空间

【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 = '0bee51f9bca64295a5c7f34b2eeb81ed'

# 访问HomeAssistant的token，你需要在你的HomeAssitant的WEB前端生成

ha\_token='eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1NTA1ODY4MDIsImlzcyI6IjRlODRjNjJiYTI3NzQxNzA5ZjgyYWMzYWYzMjgzNzJmIiwiZXhwIjoxODY1OTQ2ODAyfQ.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-\*