How To Integrate A Google Assistant On The Raspberry Pi

Introduction

A simple Google Assistant Python example program for the Raspberry Pi.

This applications note is based on the most excellent article and code by PJ Evans, published in The MagPi magazine issue 84 and the code here:

- https://www.raspberrypi.org/blog/build-demolition-mans-verbal-morality-ticketing-machine/
- https://github.com/mrpjevans/stenographer

It handles:

- Local audio playout
- Audio control – volume, stop etc.
- Local command handling – Time, Date etc.
- Google Assistant via keywords “Smart Assistant”, “Ok Google”, “Hey Google” etc.

The code uses:

Google Cloud Speech-to-Text:

- https://cloud.google.com/speech-to-text/

Google Assistant SDK:

- https://developers.google.com/assistant/sdk/guides/service/python/

Requirements

- Raspberry Pi - This application was tested on a B3.
- A USB microphone to complement the on-board audio output **
- Loudspeakers to hear the audio output
- https://github.com/xmos/vocalfusion_3510_avs_setup
- Python 3 (>= v3.5)

** An alternative to a basic USB microphone is the excellent XVF3510 from XMOS, which includes echo cancellation, noise suppression and more. https://www.xmos.com/products/voice/kits . The XMOS XVF3510 gives a much more professional result than a regular USB microphone. For example better performance in a noisy environment and, due to the built in echo canceller, speech played out by the device does not get transcribed into commands.

Initialization

To install the required packages, perform the following installation steps: Note: some of the apt and pip3 installation processes will report errors. Rerun the command and it should work fine.

```
    sudo apt update; sudo apt upgrade -y
sudo apt install -y alsa-tools alsa-utils python3-pyaudio portaudio19-dev python3-all-dev python3-dev sox lame libsox-fmt-all mpg321
    sudo pip3 install 'google-cloud-speech>=0.36.0'
    sudo pip3 install gtts pyaudio
```

Audio I/O Testing

When connecting the USB microphone it is useful to test that it works correctly. There are severl applications that can help with testing audio I/O, including aplay and arecord, but two of the most useful are VLC media player and Audacity.

VLC is a very flexible and powerful media player and Audacity is a lightweight Digital Audio Workstation (DAW).

These can both be installed using the following command:

```
sudo apt install vlc audacity
```

To test the microphone, open up audacity. Ensure the microphone is selected in the appropriate pull-down menu at the top and hit the record button. You should be able to record audio from the microphone and play it back through the loudspeakers.

Setting Up Google Speech To Text

Follow the steps here, starting at the section “Create your Google project”:

- https://www.raspberrypi.org/blog/build-demolition-mans-verbal-morality-ticketing-machine/

Ensure that you copy the .json file and set an environment variable that the libraries will use to locate your credentials JSON and replace [FILE_NAME] with the actual name of the JSON file):

```
export GOOGLE_APPLICATION_CREDENTIALS="/home/pi/[FILE_NAME].json"
```

Note: the path must be an absolute path, i.e. not starting with ‘home’ (~).

Setting Up Google Assistant

Follow the steps here:

- https://willhaley.com/blog/google-assistant-sdk-hello-world/

Starting at the section “Application”. Note, it is not necessary to create a new Python environment unless you wish to.

Here is a summary:

```
sudo apt-get install -y portaudio19-dev libffi-dev libssl-dev
pip3 install google-assistant-library google-auth-oauthlib[tool] google-assistant-sdk[samples]
```

Before authenticating (google-oauthlib-tool) you will need to add ".local/bin/" to the path by adding the following line to the end of .bashrc:

```
export PATH=$PATH:/home/pi/.local/bin/
```

To configure the audio interface, add the following line to the end of .bashrc:

```
export PA_ALSA_PLUGHW=1
```

Now reboot the Raspberry Pi.

This application supports two methods for interacting with Google Assistant:

- 1/ SEND_AUDIO_REQUEST = 1

- Processed command strings are converted to audio (using gTTS) and sent to Google. The audio response is then played out.
- 2/ SEND_AUDIO_REQUEST = 0
- Processed command strings are sent to Google (using my_textinput.py). The text response is then converted to audio (using gTTS) and played out.

** Note, as of testing in May 2020, mode SEND_AUDIO_REQUEST == 1 does not work. Testing shows that this might be due to the audio generated by gTTS not being compatible with the Google Speech API. It is recommended that text mode is used, i.e. SEND_AUDIO_REQUEST = 0.

Audio mode uses the standard googlesamples-assistant-pushtotalk example to send the request to Google. To enable this mode you need to authenticate using the following command:

```
python3 /home/pi/.pyenv/versions/3.7.6/lib/python3.7/site-packages/googlesamples/assistant/grpc/pushtotalk.py --device-model-id your-model-id --project-id your-project-id
```

** Note, the pushtotalk.py file will move directory, with changes in Python version. This folder is correct at May 2020.

After this, you can record a request into a .wav file (e.g. myrequest.wav) and send it to Google using the following command:

```
googlesamples-assistant-pushtotalk -i myrequest.wav
```

To use text mode, it is necessary to modify the provided assistant_textinput.py at the line starting "with SampleTextAssistant(..." to add your Device Model ID and Device ID:

```
with SampleTextAssistant("en-GB", "your-device-model-id", "your-device-id", display,
```

Then copy the modified file to the googlesamples assistant folder:

```
cp assistant_textinput.py /home/pi/.local/lib/python3.7/site-packages/googlesamples/assistant/grpc
```

Executing The Application

The application source code is located in assistant.py. If this is made executable then it can be executed on the command line using:

```
./assistant.py
```

Now a range of voice commands can be tried including:

::

Play music What's the date What's the time Ok Assistant who is Barack Obama Ok Assistant what is the weather in London

To enable the assistant to play local Music, just list a number of local music files in "songlist.lst". These songs are then played by using the command "Play Music". The code to implement this command is at line #239:

```
subprocess.Popen(shlex.split("mpg321 --list songlist.lst -Z --loop 0"), stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL)
```

Use the following command to auto-generate the songlist.txt file with songs stored in your Music folder:

```
sudo apt install -y mlocate; sudo updatedb; locate .mp3 | grep /home/pi/Music > songlist.lst
```

Note: When executing the python script you will see a whole list of errors related to ALSA/Portaudio. These can be ignored or you can revert the Linux kernel back to an older kernel that does not exhibit these issues:

```
sudo rpi-update a08ece3d48c3c40bf1b501772af9933249c11c5b; sudo reboot
```

Utility Applications

A utility “listDevices.py” is provided to print the list of audio devices available in Python.