Your own Raspberry Pi Google Assistant

Janne Spijkervet
Dec 25, 2018 · 5 min read

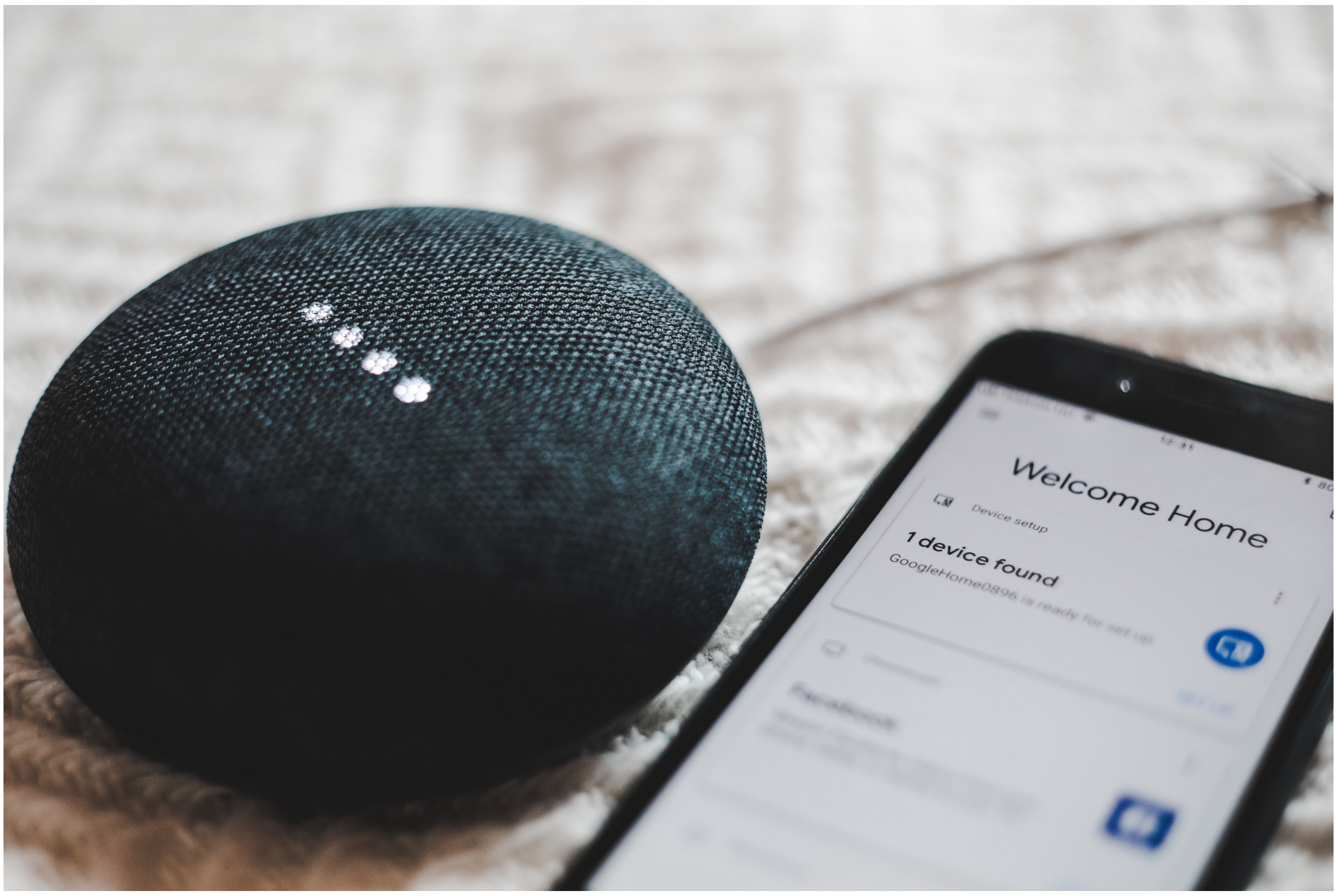


Photo by Bence ▲ Boros on Unsplash

With the holiday season around the corner — there is some more time to do a DIY project. I'll be trying to make my own home a little smarter, using a Raspberry Pi as the main home controller. The Google Home and its smaller brother are beautiful devices with many amazing features.

In this article, I'll lay out the steps I've taken to install the Google Assistant on a Raspberry Pi. You will be able to have the following functionalities:

- Google Assistant (most features, except media playback on the Pi)
- Voice / Hotword activation: "Hey Google" or "Ok Google" and ask the question

- Initialized on boot with a service, so you do not have to manually start the program every reboot

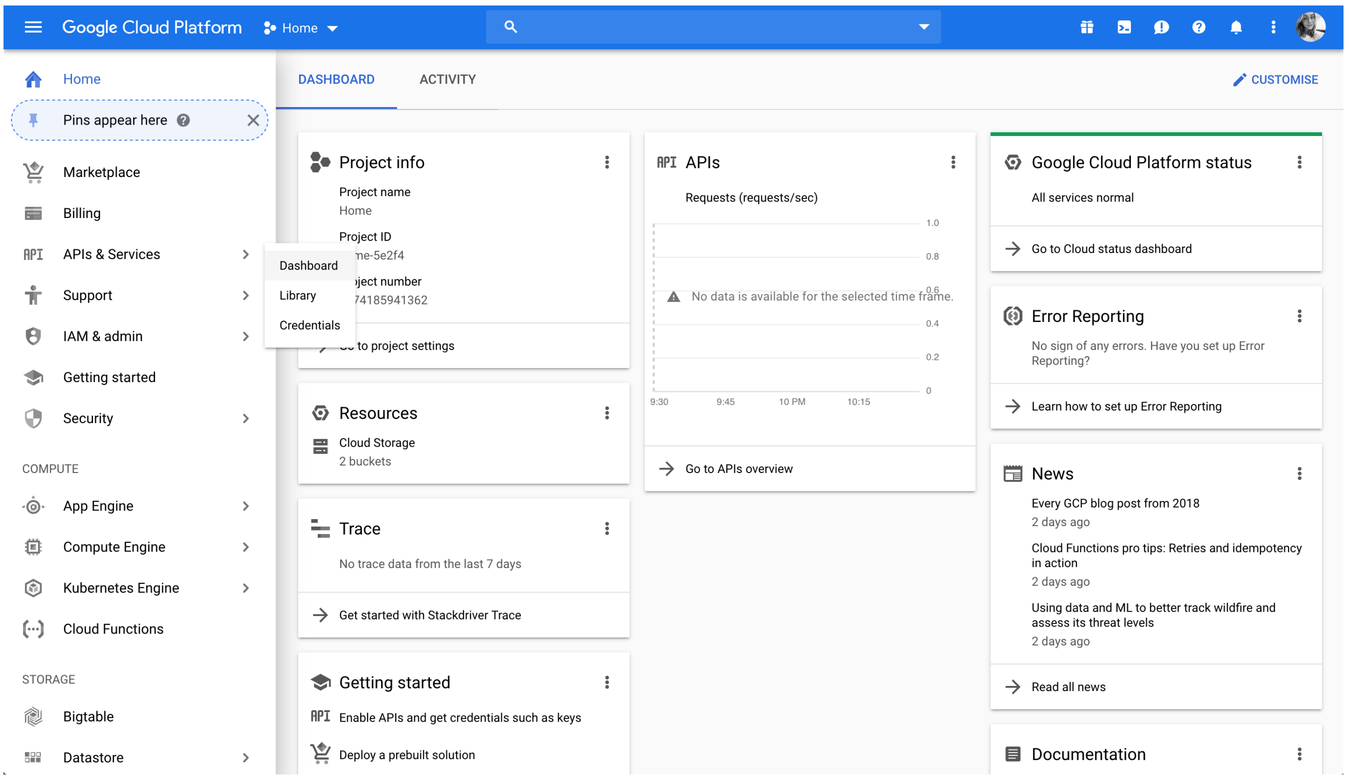
Step 0: What you will need

- Raspberry Pi 2 or 3
- Micro SD card (minimum 8GB) with Raspbian installed (or any other Debian-based distribution)
- USB microphone
- A speaker

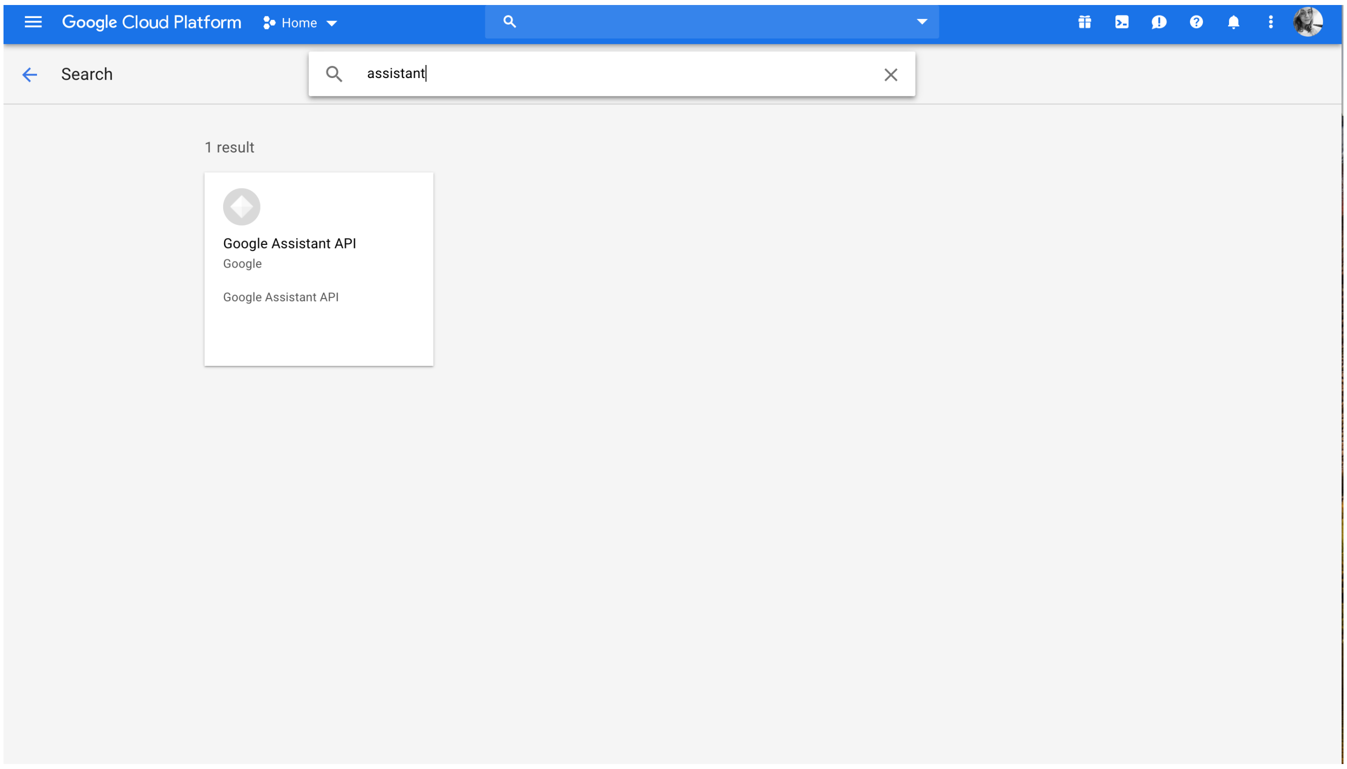
Step 1: Registering your Device

1. Go to `https://console.actions.google.com` to register your project on the Google Actions Console.
2. Next, go to `https://console.cloud.google.com` to enable the Google Assistant API.

> Make sure to select your newly created project in the selection panel next to the Cloud Platform logo!



3. Select the **API & Services** tab and search for and enable the Google Asssitant API.

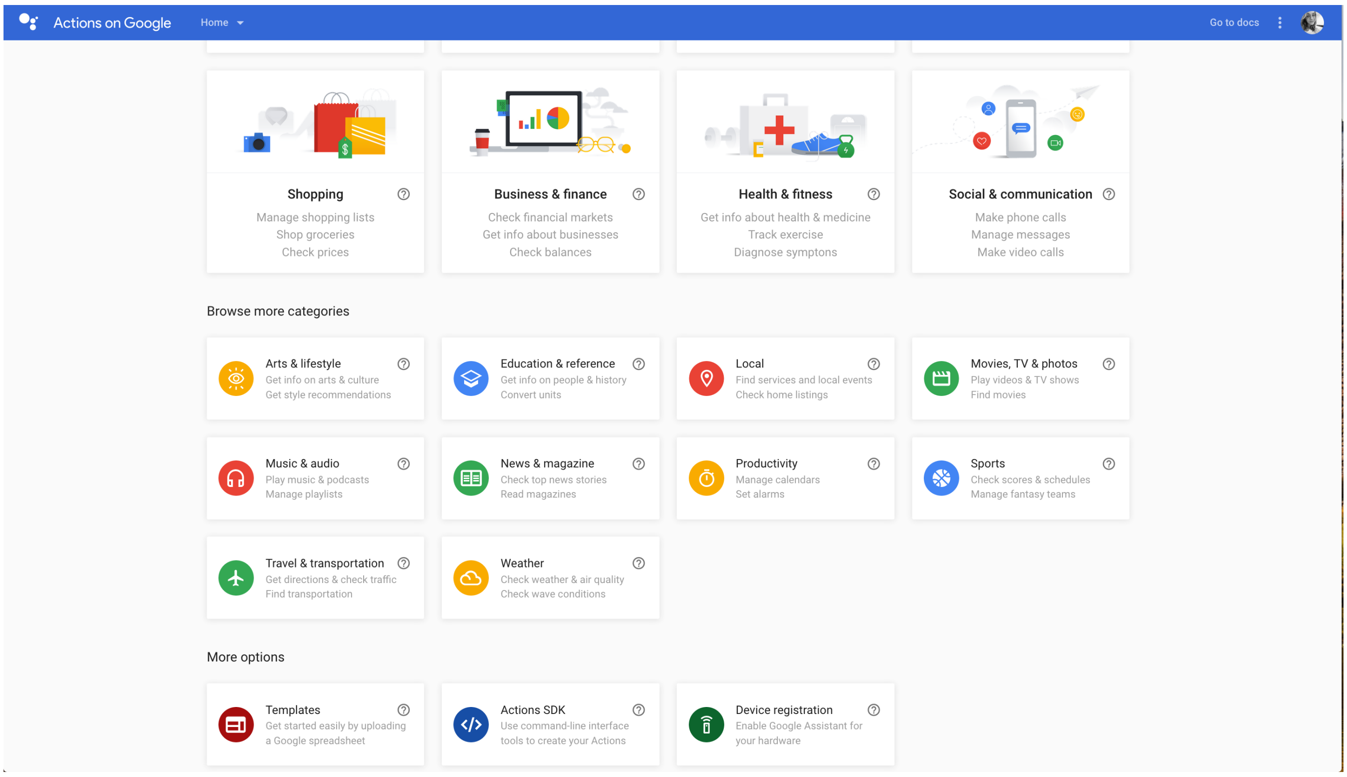


1 result

Google Assistant API
Google
Google Assistant API

4. Make sure to enable all settings in https://myaccount.google.com/activitycontrols so the API works correctly.

5. Go back to `https://console.actions.google.com` to register your Raspberry Pi device in the bottom right corner of the applications panel.



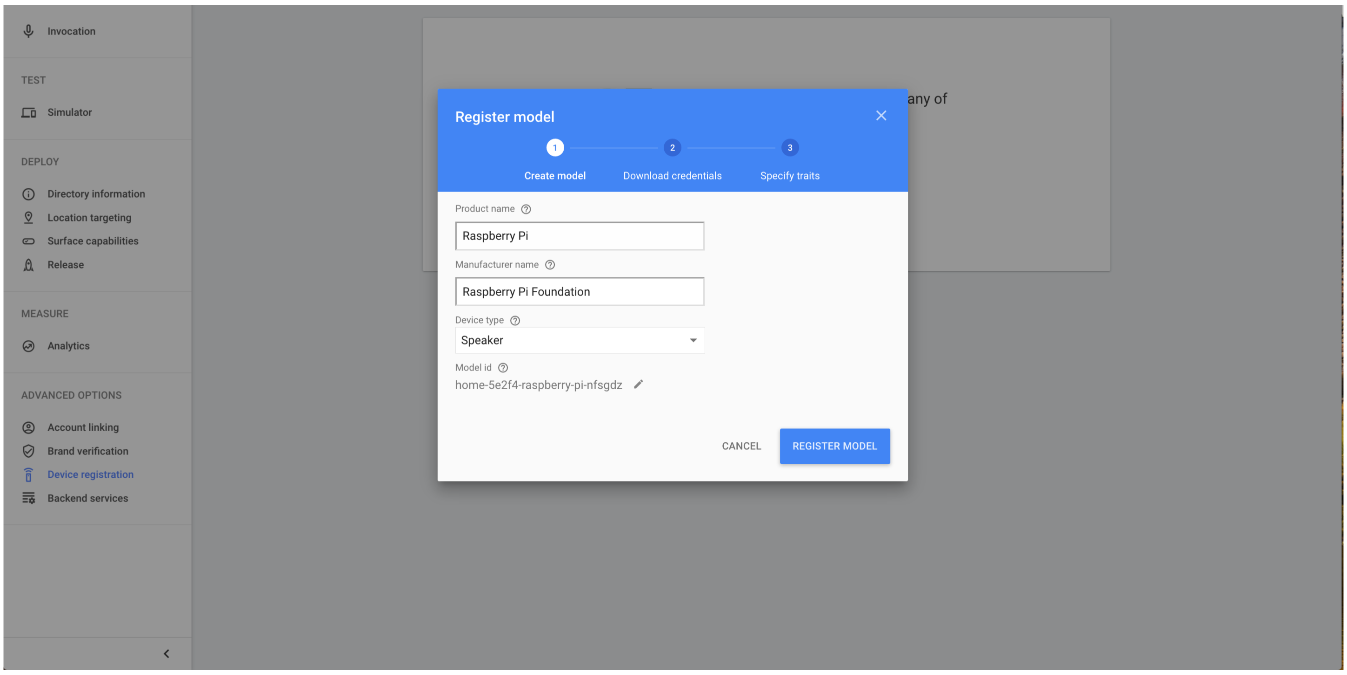
Shopping — Manage shopping lists, Shop groceries, Check prices
Business & finance — Check financial markets, Get info about businesses, Check balances
Health & fitness — Get info about health & medicine, Track exercise, Diagnose symptoms
Social & communication — Make phone calls, Manage messages, Make video calls

Browse more categories

Arts & lifestyle — Get info on arts & culture, Get style recommendations
Education & reference — Get info on people & history, Convert units
Local — Find services and local events, Check home listings
Movies, TV & photos — Play videos & TV shows, Find movies
Music & audio — Play music & podcasts, Manage playlists
News & magazine — Check top news stories, Read magazines
Productivity — Manage calendars, Set alarms
Sports — Check scores & schedules, Manage fantasy teams
Travel & transportation — Get directions & check traffic, Find transportation
Weather — Check weather & air quality, Check wave conditions

More options

Templates — Get started easily by uploading a Google spreadsheet
Actions SDK — Use command-line interface tools to create your Actions
Device registration — Enable Google Assistant for your hardware



Product name: Raspberry Pi
Manufacturer name: Raspberry Pi Foundation
Device type: Speaker
Model id: home-5e2f4-raspberry-pi-nfsgdz

6. Download the credentials file so we can use them later on. We do not need to setup any of the traits.

Step 2: Audio setup

1. Make note of the cards / device numbers of your audio devices by using:

```
arecord -l
aplay -l
```

2. Use the information to edit the `asoundrc` file:

```
nano /home/pi/.asoundrc

pcm.!default {
  type asym
  capture.pcm "mic"
  playback.pcm "speaker"
}
pcm.mic {
  type plug
  slave {
    pcm "hw:<card number>,<device number>"
  }
}
pcm.speaker {
  type plug
  slave {
    pcm "hw:<card number>,<device number>"
    }
  }
```

3. You can then test your devices using:

```
speaker-test -t wav

arecord --format=S16_LE --duration=5 --rate=16000 --file-type=raw
out.raw

aplay --format=S16_LE --rate=16000 out.raw

alsamixer
```

Step 3: Installing the Google Assistant Library

1. Set up the directories where we will install our virtual environment to:

```
mkdir ~/googleassistant
nano ~/googleassistant/credentials.json
```

2. We will be using a virtual environment so our global environment does not become cluttered. Install **venv** with:

```
sudo apt-get install python3-dev python3-venv
```

3. Initialize the environment, install the latest version of **pip** and activate the environment with:

```
python3 -m venv env && env/bin/python -m pip install --upgrade pip
setuptools --upgrade && source env/bin/activate`
```

4. Now we can install the **Google Assistant Library for Python**:

```
python -m pip install --upgrade google-assistant-library google-
assistant-sdk[samples]
```

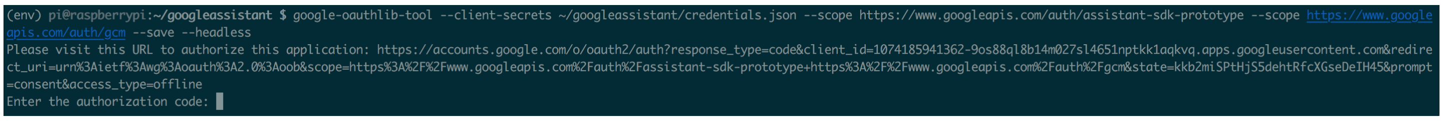
Step 4: Authorizing the Pi for the Google Assistant

1. Let’s install the authorization tool so we can authorize with the Google Assistant API we just enabled for our application:

```
python -m pip install --upgrade google-auth-oauthlib[tool]
```

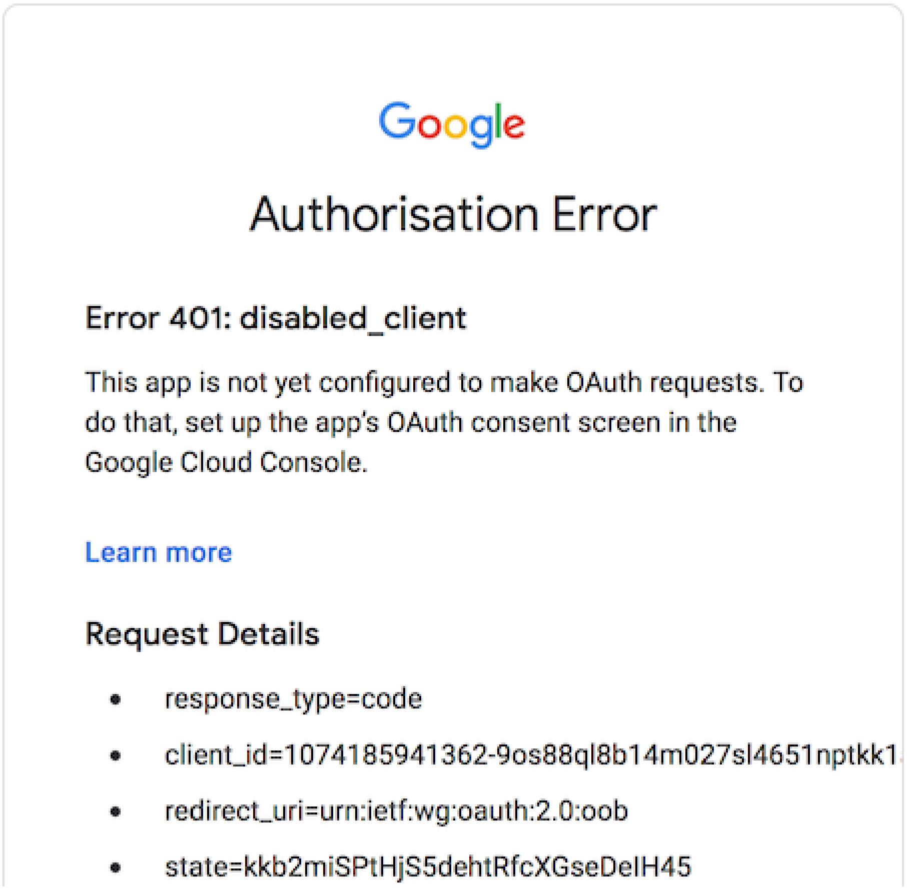
2. The following will give us an authorization url:

```
google-oauthlib-tool --client-secrets
~/googleassistant/credentials.json \
--scope https://www.googleapis.com/auth/assistant-sdk-prototype \
--scope https://www.googleapis.com/auth/gcm \
--save --headless
```



```
(env) pi@raspberrypi:~/googleassistant $ google-oauthlib-tool --client-secrets ~/googleassistant/credentials.json --scope https://www.googleapis.com/auth/assistant-sdk-prototype --scope https://www.google
apis.com/auth/gcm --save --headless
Please visit this URL to authorize this application: https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=1074185941362-9os88ql8b14m027sl4651nptkk1aqkvq.apps.googleusercontent.com&redire
ct_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fassistant-sdk-prototype+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fgcm&state=kkb2miSPtHjS5dehtRfcXGseDeIH45&prompt
=consent&access_type=offline
Enter the authorization code: 
```

3. You may get the following OAuth error:



Google

Authorisation Error

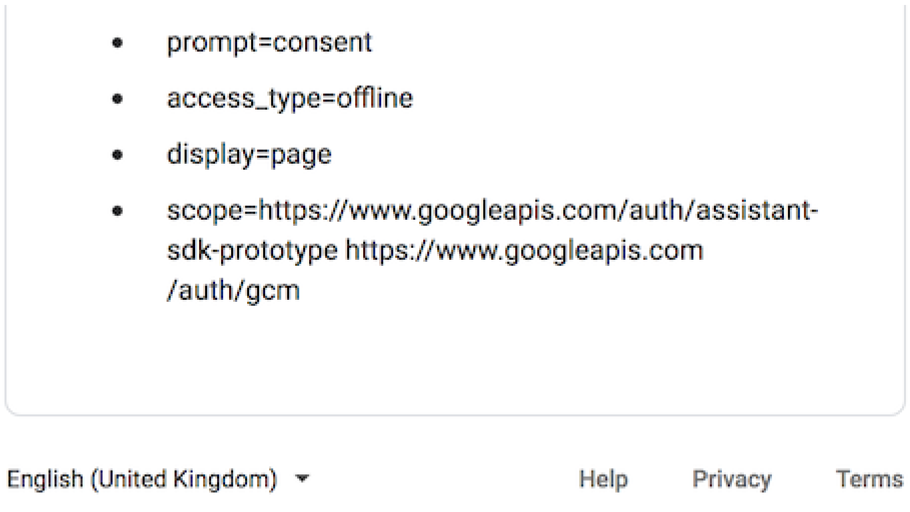
Error 401: disabled_client

This app is not yet configured to make OAuth requests. To do that, set up the app's OAuth consent screen in the Google Cloud Console.

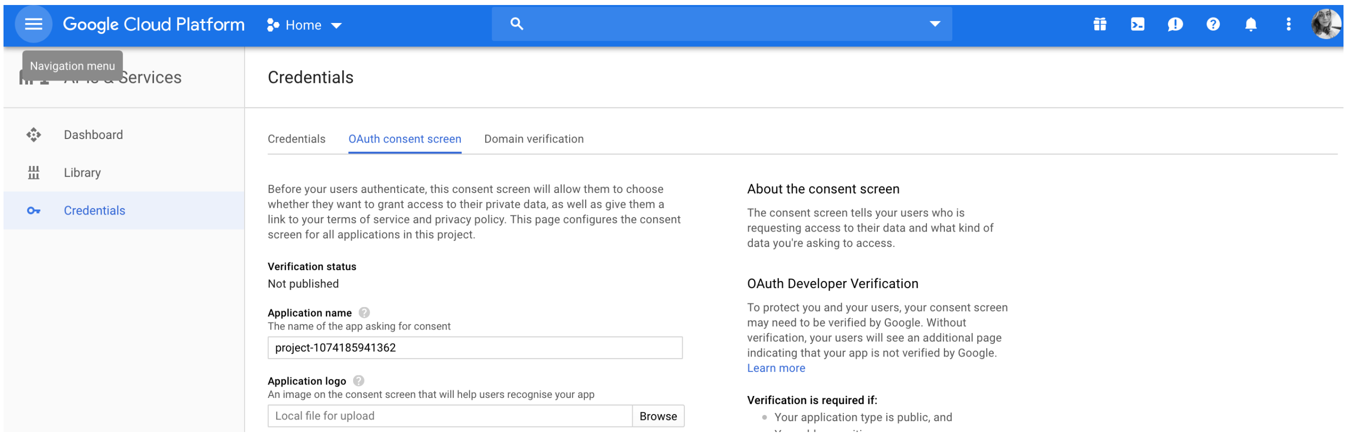
Learn more

Request Details

- response_type=code
- client_id=1074185941362-9os88ql8b14m027sl4651nptkk1
- redirect_uri=urn:ietf:wg:oauth:2.0:oob
- state=kkb2miSPtHjS5dehtRfcXGseDelH45


- prompt=consent
- access_type=offline
- display=page
- scope=https://www.googleapis.com/auth/assistant-sdk-prototype https://www.googleapis.com/auth/gcm

4. To solve this, we can setup the consent screen for the application by going to `Credentials -> OAuth Consent Screen` on the **Google Cloud Platform** *(Again, make sure to select your application!)*



Before your users authenticate, this consent screen will allow them to choose whether they want to grant access to their private data, as well as give them a link to your terms of service and privacy policy. This page configures the consent screen for all applications in this project.

Verification status
Not published

Application name
The name of the app asking for consent
project-1074185941362

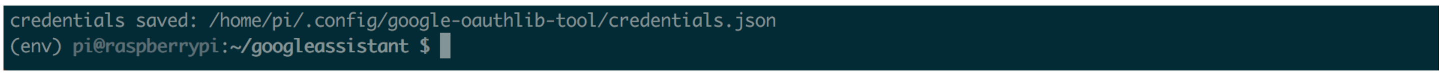
Application logo
An image on the consent screen that will help users recognise your app
Local file for upload

About the consent screen
The consent screen tells your users who is requesting access to their data and what kind of data you're asking to access.

OAuth Developer Verification
To protect you and your users, your consent screen may need to be verified by Google. Without verification, your users will see an additional page indicating that your app is not verified by Google.
Learn more

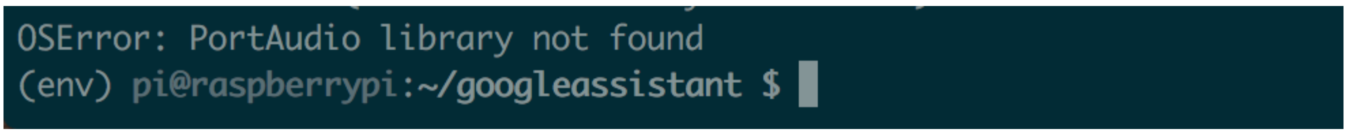
Verification is required if:
- Your application type is public, and

5. When filling in the authorization code, you will see:


```
credentials saved: /home/pi/.config/google-oauthlib-tool/credentials.json
(env) pi@raspberrypi:~/googleassistant $ 
```

Step 5: PortAudio errors

1. You may encounter the following PortAudio error:


```
OSError: PortAudio library not found
(env) pi@raspberrypi:~/googleassistant $ 
```

2. Install the dependency:

```
sudo apt-get install libportaudio2
```

3. To disable dropouts and delays in the audio signal, comment out the following in `/etc/pulse/default.pa` :

4. Also run PulseAudio system-wide to avoid issues with the Google Assistant daemon:

```
sudo nano /etc/systemd/system/pulseaudio.service
```

5. Add the following lines:

```
[Unit]
Description=PulseAudio Sound Server in system-wide mode
[Service]
Type=forking
PIDFile=/var/run/pulse/pid
ExecStart=/usr/bin/pulseaudio --system --disallow-exit=1 \
--disable-shm=1 --fail=1 --daemonize

[Install]
WantedBy=multi-user.target
```

6. Enable the service and add the *pi* user to the *pulse-access* group:

```
sudo systemctl --system enable pulseaudio.service
sudo adduser pi pulse-access
```

7. Disable the following module to avoid audio delays in `/etc/pulse/default.pa`

```
#load-module module-suspend-on-idle
```

```
sudo nano /etc/systemd/system/assistant.service
```

Step 6: Turn the Assistant into a service

1. Create a file `~/start_assistant.sh` , using the device model ID obtained from the Google Action platform:

```
#!/bin/bash
source /home/pi/googleassistant/env/bin/activate
googlesamples-assistant-hotword --device-model-id
<your_device_model_id>
```

2. Make the script executable:

```
chmod +x start_assistant.sh
```

3. Create a service file in `/etc/systemd/system/assistant.service`

```
[Unit]
Description=Google Assistant
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
ExecStart=/bin/bash /home/pi/start_assistant.sh
Restart=on-abort
User=pi
Group=pi

[Install]
WantedBy=multi-user.target
```

4. Enable and start the service:

```
sudo systemctl enable assistant.service
sudo systemctl start assistant.service
```

Finalizing

Now you can enjoy the Google Assistant on your Raspberry Pi by querying it using the hotwords “Hey Google” or “Ok Google”. You can also register the device in the Google Assistant app on iOS or Android.

You can also make your own custom actions using Google Actions, e.g. opening garage doors, controlling your coffee machine, changing the lights. You name it :) — I look

) forward to hear about what home automation projects you are doing!

Raspberry Pi Technology Artificial Intelligence Programming Home