

Building Google Assistant with raspberry Pi

Installation Manual

- The Google Assistant is a virtual assistant powered by artificial intelligence and developed by Google that is primarily available on mobile and smart home devices.
- The Google Assistant can engage in two-way conversations.

Hardware Requirements

1. Raspberry Pi Model A/B/B+
2. SD Card
3. Ethernet Cable / Wi-Fi
4. Power Supply
5. USB Microphone
6. Speakers

- **Software Requirements**

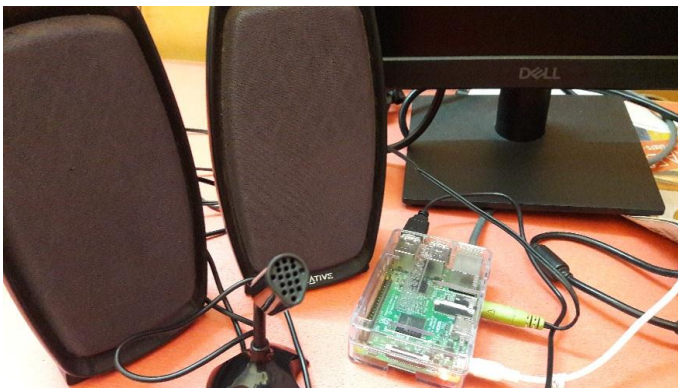
1. Raspbian Stretch OS
2. Google Account

Configure and Test Audio

- Attach USB Microphone and Speakers to Raspberry Pi.
- After connection, test it to ensure audio is working or not.

Follow below tutorial to configure and test audio

https://drive.google.com/open?id=1gexNOv_zK1jqRpeyCn7JKU6Tpud2Vn82



Note that...

- If recording and playback are working, then you are done configuring audio.
- If not, check that the microphone and speaker are properly connected.
- If this is not the issue, then try a different microphone or speaker.

Step 1: Registering the Google API

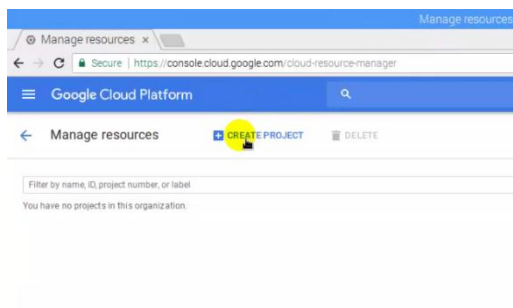
1. Before we get started with setting up the Google Assistant code on the Raspberry Pi itself, we must first register and setup OAuth access to Google's Assistant API. To do this, you will need to have a Google account already.

Once you have your Google account ready, go to the following web address.

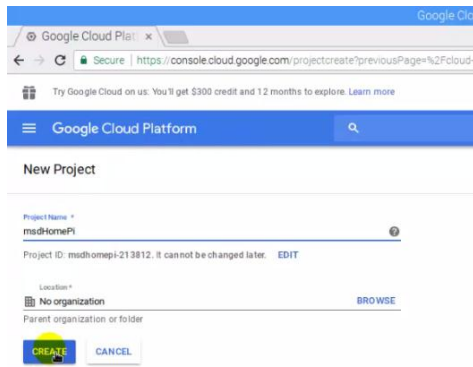
<https://console.cloud.google.com/project>



2. Once you have logged into your account, a dialog box appears, in which select your **Country** and for the two radial boxes, we selected 'No' to wanting email updates and 'Yes' to agree to their terms of service.



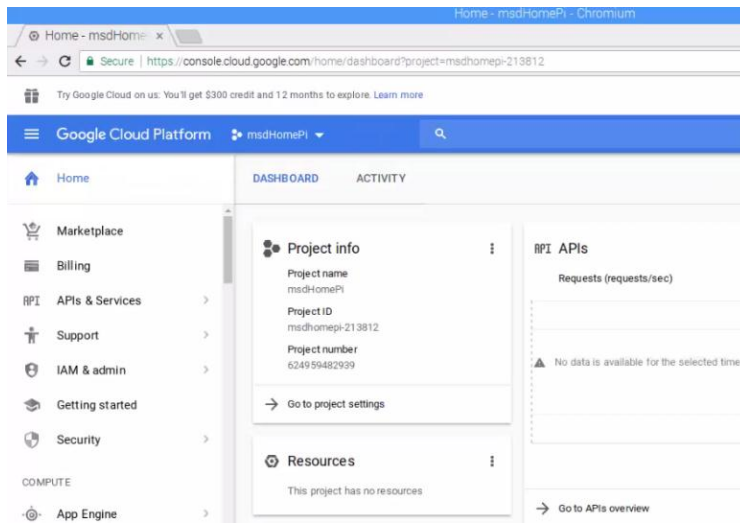
3. Click on **Create Project** Link.



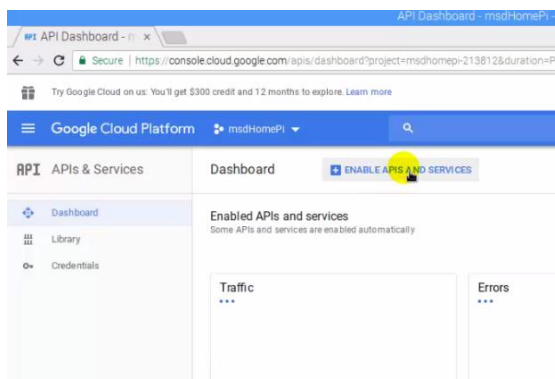
4. Enter **Project Name** and Click on **Create**

The project creation process can take some time. You should receive a notification in the top right-hand corner of the screen when it's complete. If it doesn't automatically appear after some time, try refreshing the page.

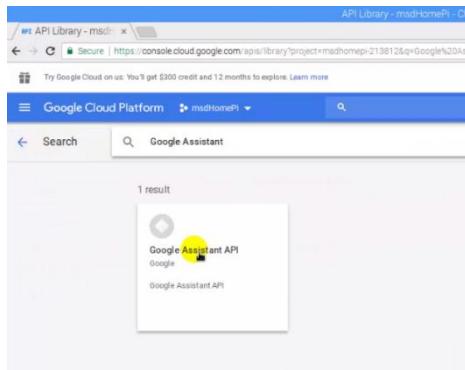
5. Once it has appeared, click the project name.



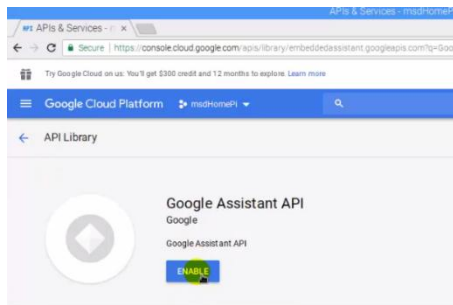
6. On the side menu, select API's & Services



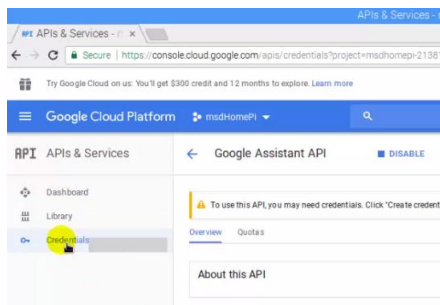
7. In Dashboard Option, Click on **Enable API's & Services**



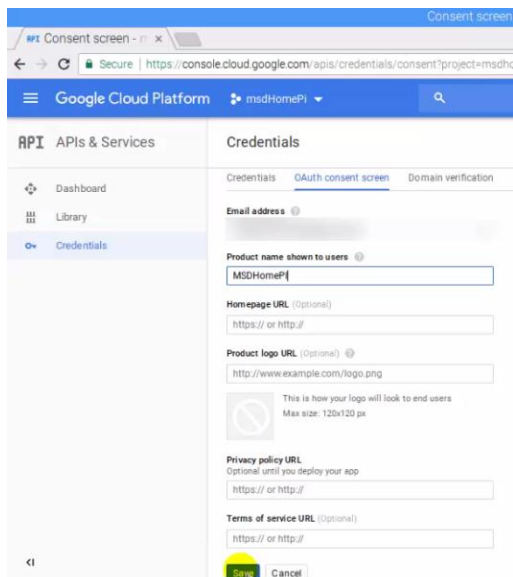
8. Search for **Google Assistant API** in Library Section



9. Click on **Enable** Button



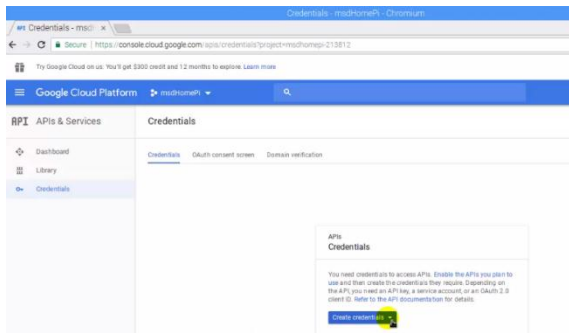
10. Now in APIs & Services Page, Select Credentials Option.



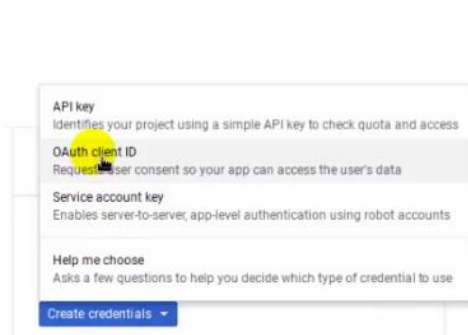
11. Switch to the **oauth consent screen**

Enter name in **Project Name** shown to users

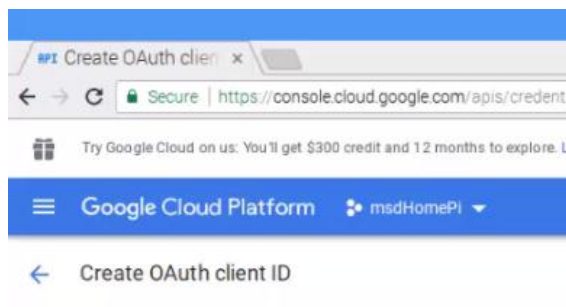
Click on **Save**



12. Click on **Create Credentials** button



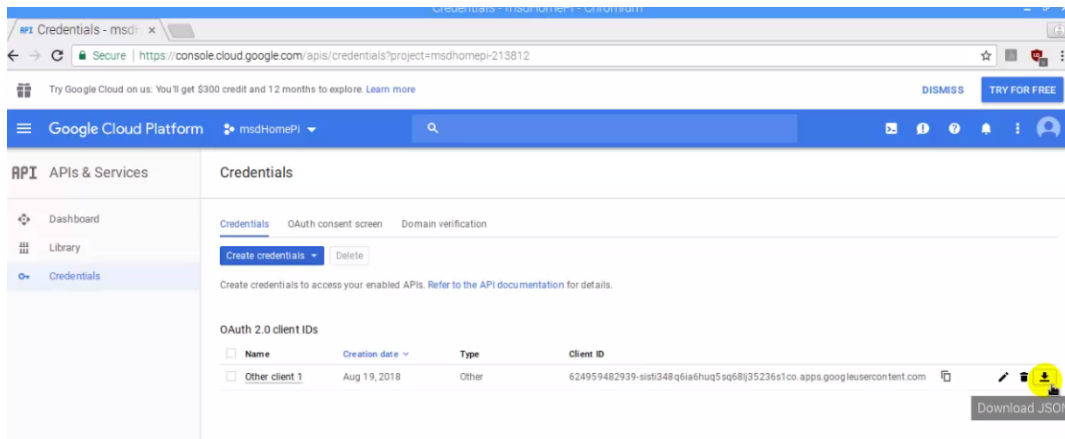
13. In the drop-down menu,
Click on **OAuth Client ID**



14. Set the **Application Type to Other**,
You can also change the name for client ID.
Or keep default as it is.

Click on **Create** button.

15. Now Download the **Credentials file** for our newly created **oauth credential**.



Downloaded file will be saved under **/home/pi/Downloads** folder.

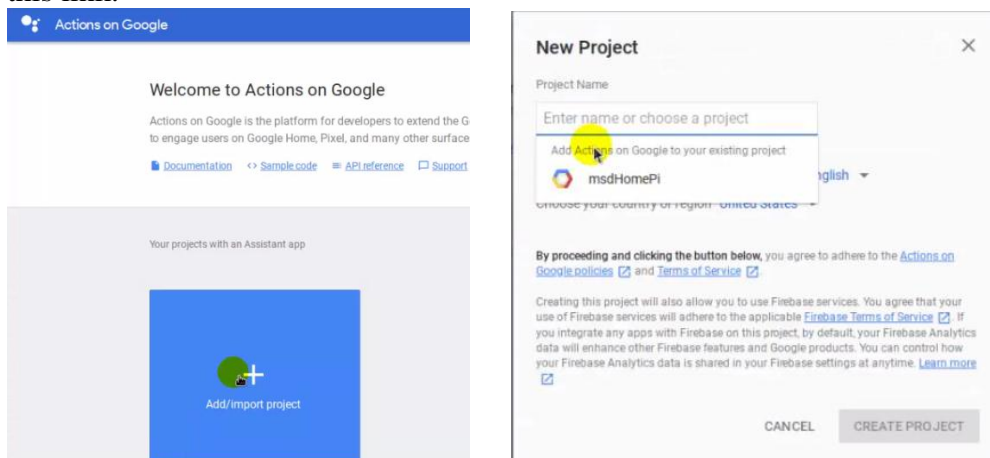
Step 2: Account Settings

Now, go to the URL **<https://myaccount.google.com/activitycontrols>**, and activate the following activity controls to ensure that the Google Assistant API works correctly.

- **Web & App Activity**
- **Location History**
- **Device Information**
- **Voice & Audio Activity**

Step 3: Register the Device Model

1. Now open **<https://www.console.actions.google.com>**
Select your **Project** (created in previous step), **Or** Even you can also create your new project with this link.



New Project ✕

Project Name

Choose the default language for your Actions: **English** ▾

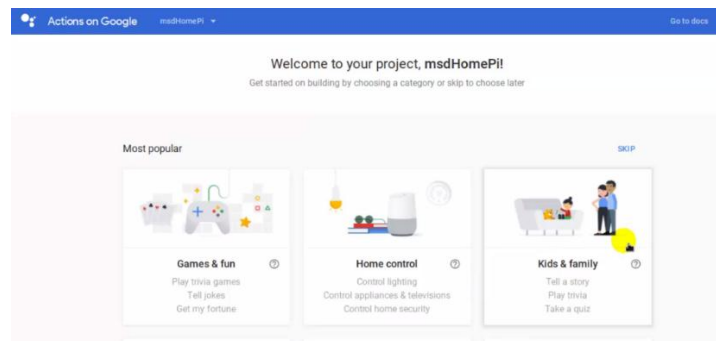
Choose your country or region: **United States** ▾

By proceeding and clicking the button below, you agree to adhere to the [Actions on Google policies](#) and [Terms of Service](#).

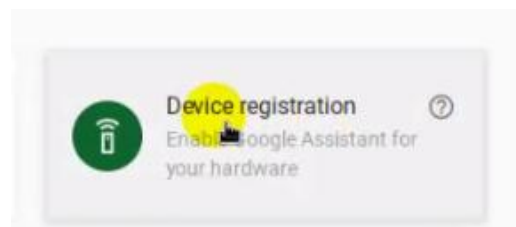
Creating this project will also allow you to use Firebase services. You agree that your use of Firebase services will adhere to the applicable [Firebase Terms of Service](#). If you integrate any apps with Firebase on this project, by default, your Firebase Analytics data will enhance other Firebase features and Google products. You can control how your Firebase Analytics data is shared in your Firebase settings at anytime. [Learn more](#)

☐

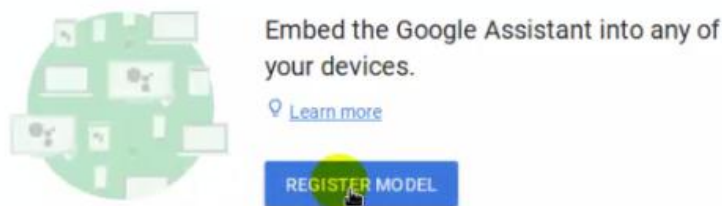
CANCEL **IMPORT PROJECT**



2. Scroll Down to get **Device Registration Option**, Click on **Device Registration**.



3. Click the **Register Model** button to continue.



Register model ✕

1 Create model 2 Download credentials 3 Specify traits

Product name ⓘ

Manufacturer name ⓘ

Device type ⓘ

Device Model id ⓘ
 msdhomepi-213812-msd-pi-p1jgi7 ✎

CANCEL **REGISTER MODEL**

4. Set **Product Name** as a simple descriptor

Set **Manufacturer Name** as anything (because it doesn't hugely matter)

Set the **Device Type** as **Speaker** (as it matches with our current project)

Note down this **Device Model Id** for later use.

Click on **Register Model**

Click on **Next**, then Click on **Save Traits**.

Device registration				
				REGISTER MODEL
Product name	Manufacturer name	Model ID	Device Type	Last updated time
MSD Pi	MSD Gurukul	msdhomepi-213812-msd-pi-p1jgi7	action.devices.types.SPEAKER	Aug 19, 2018, 12:20 PM

Step 4: Install the SDK and Sample Code on Raspberry PI

1. Update Raspberry Pi

```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo apt-get update

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo apt-get upgrade
```

2. Create a directory, and then Copy credential file (downloaded in previous step) to your directory (with new name credentials.json)

```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ mkdir googleassistant
pi@raspberrypi:~ $ cp ~/Downloads/client_secret_624959482939-sisti348q6ia6huq5sq68lj35236s1co.apps.googleusercontent.com.json ~/googleassistant/credentials.json
pi@raspberrypi:~ $ ls googleassistant/
credentials.json
```

3. Install Python3 and the Python 3 Virtual Environment to our Raspberry Pi.

`$ sudo apt-get install python3-dev python3-venv` #for Python 3

```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo apt-get install python3-dev python3-venv
```


4. Install the package's system dependencies

`$ sudo apt-get install portaudio19-dev libffi-dev libssl-dev`

```
pi@raspberrypi:~ $ sudo apt-get install portaudio19-dev libffi-dev libssl-dev
```


5. Enable python3 as our virtual environment variable

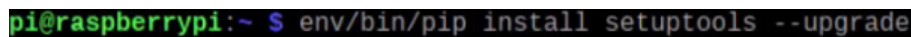
```
$ python3 -m venv env
```



```
pi@raspberrypi:~ $ python3 -m venv env
```

6. Install the latest versions of pip and the setuptools.

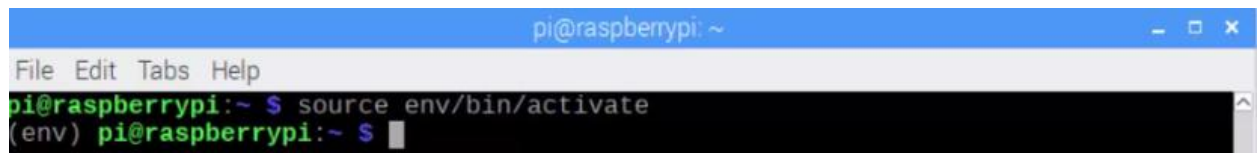
```
$ env/bin/pip install setuptools --upgrade
```



```
pi@raspberrypi:~ $ env/bin/pip install setuptools --upgrade
```

7. Get into this new Python environment that we have set up

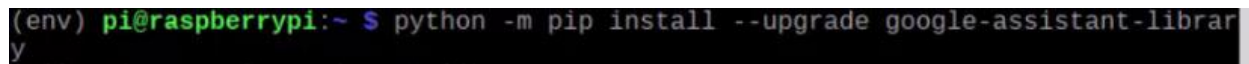
```
$ source env/bin/activate
```



```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ source env/bin/activate
(env) pi@raspberrypi:~ $
```

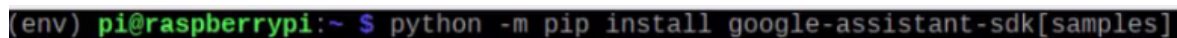
8. Now Install all the packages that we need to install the Google Assistant Library

```
$ python -m pip install --upgrade google-assistant-library
```



```
(env) pi@raspberrypi:~ $ python -m pip install --upgrade google-assistant-library
```

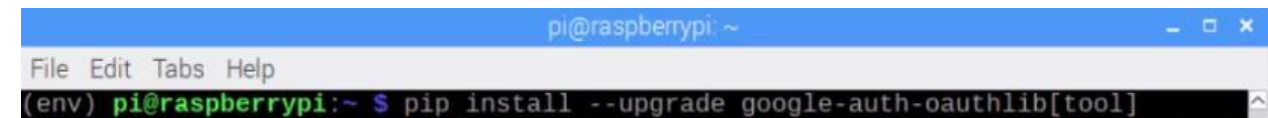
```
$ python -m pip install --upgrade google-assistant-sdk[samples]
```



```
(env) pi@raspberrypi:~ $ python -m pip install google-assistant-sdk[samples]
```

9. Install the Python authorization tool on Raspberry Pi

```
$ pip install --upgrade google-auth-oauthlib[tool]
```



```
pi@raspberrypi: ~
File Edit Tabs Help
(env) pi@raspberrypi:~ $ pip install --upgrade google-auth-oauthlib[tool]
```

10. To run the installed Google Authentication library, use following command.

```
$ google-oauthlib-tool --client-secrets /path/to/client_secret_client-id.json --scope
https://www.googleapis.com/auth/assistant-sdk-prototype --scope
https://www.googleapis.com/auth/gcm --save --headless
```

```
pi@raspberrypi: ~  
File Edit Tabs Help  
(env) pi@raspberrypi:~ $ google-oauthlib-tool --client-secrets ~/googleassistant  
/credentials.json --scope https://www.googleapis.com/auth/assistant-sdk-prototyp  
e --scope https://www.googleapis.com/auth/gcm --save --headless
```

This command will generate a URL, Right Click on above URL, **Copy it**.

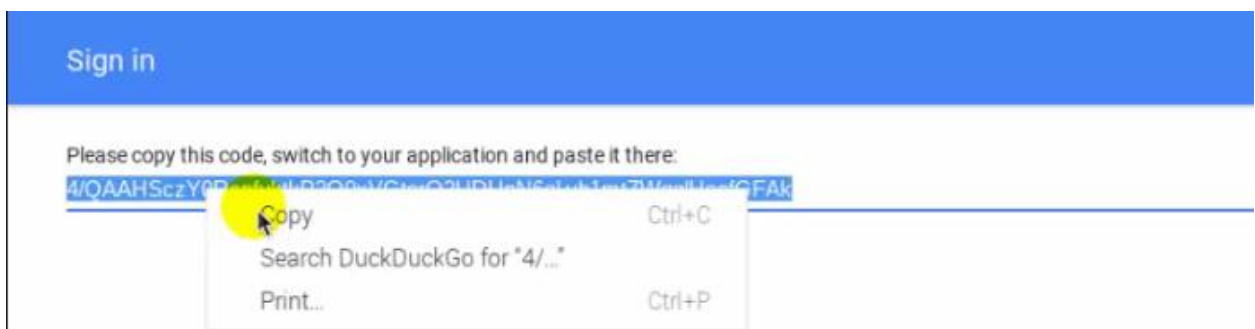
```
pi@raspberrypi: ~  
File Edit Tabs Help  
(env) pi@raspberrypi:~ $ google-oauthlib-tool --client-secrets ~/googleassistant  
/credentials.json --scope https://www.googleapis.com/auth/assistant-sdk-prototyp  
e --scope https://www.googleapis.com/auth/gcm --save --headless  
Please visit this URL to authorize this application: https://accounts.google.com  
/o/oauth2/auth?response_type=code&client_id=624959482939-sisti348q6ia6huq5sq68lj  
35236s1co.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Aawg%3Aoauth%3A2.0%  
3Aaob&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fassistant-sdk-prototype+ht  
tps%3A%2F%2Fwww.googleapis.com%2Fauth%2Fgcm&state=WxNd1BSnLy7uPFPhdKoXcTW7NQdXnQ  
&prompt=consent&access_type=offline  
Enter the authorization code: █
```

You will need to go to in your **web browser**, **Paste above URL in address bar**.

On this screen login to your Google account, if you have multiple accounts make sure you select the one you set up your API key with.

Click on **Allow**

11. **Copy the authentication code and paste it back into your terminal and press enter.**



```
pi@raspberrypi: ~  
File Edit Tabs Help  
(env) pi@raspberrypi:~ $ google-oauthlib-tool --client-secrets ~/googleassistant  
/credentials.json --scope https://www.googleapis.com/auth/assistant-sdk-prototyp  
e --scope https://www.googleapis.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=624959482939-sisti348q6ia6huq5sq68lj  
35236s1co.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Aawg%3Aoauth%3A2.0%  
3Aaob&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fassistant-sdk-prototype+ht  
tps%3A%2F%2Fwww.googleapis.com%2Fauth%2Fgcm&state=WxNd1BSnLy7uPFPhdKoXcTW7NQdXnQ  
&prompt=consent&access_type=offline  
Enter the authorization code: 4/QAAHSczY0RcpfuktkP200xVGtor02UDUnN6oLyb1mtZWqq1H  
eofGFAk█
```

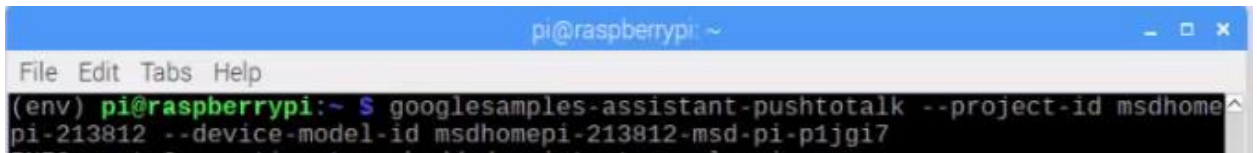
12. if the authentication was accepted, you received a message **credentials saved**

```
Enter the authorization code: 4/QAAHSczY0RcpfuktkP200xVGtorO2UDUnN6oLyb1mtZWqqlH
eofGFAk
credentials saved: /home/pi/.config/google-oauthlib-tool/credentials.json
```

Step 5: Run the Sample Code

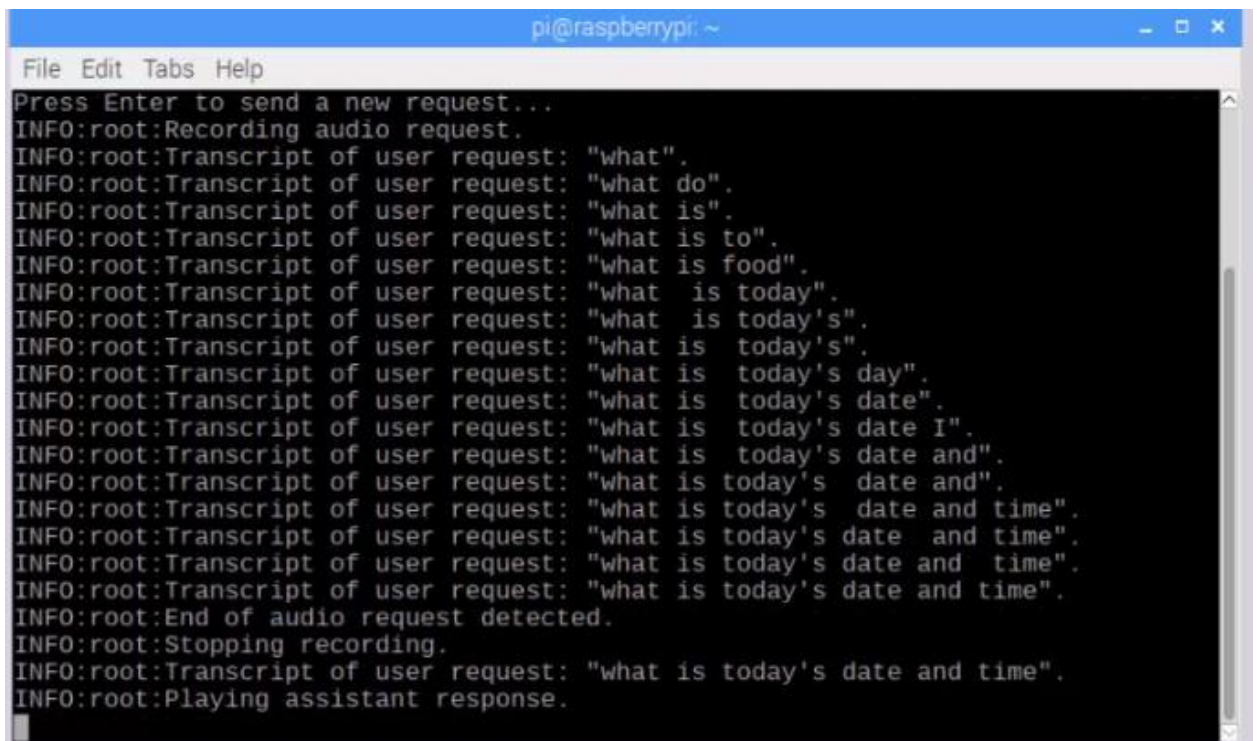
Finally, we have finished setting up everything we need to run the Google Assistant sample on our Raspberry Pi. Run the software by running the following command on your Raspberry Pi.

```
$ googlesamples-assistant-pushtotalk --project_id my-dev-project --device_model_id
my-model-id
```



A terminal window titled 'pi@raspberrypi: ~' showing the command: `(env) pi@raspberrypi:~ $ googlesamples-assistant-pushtotalk --project-id msdhomepi-213812 --device-model-id msdhomepi-213812-msd-pi-p1jgi7`

Use **Project ID** and **Model ID** created in previous steps.



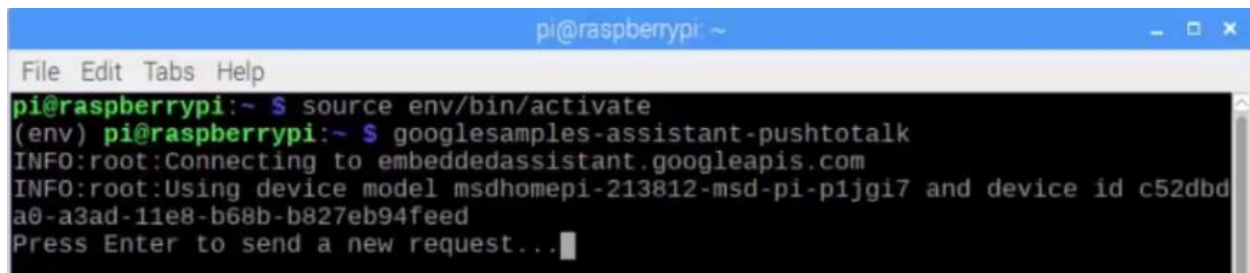
A terminal window titled 'pi@raspberrypi: ~' showing the output of the Google Assistant sample. It starts with 'Press Enter to send a new request...' and 'INFO:root:Recording audio request.' followed by a series of transcripts for the user request: 'what', 'what do', 'what is', 'what is to', 'what is food', 'what is today', 'what is today's', 'what is today's', 'what is today's day', 'what is today's date', 'what is today's date I', 'what is today's date and', 'what is today's date and', 'what is today's date and time', 'what is today's date and time', 'what is today's date and time'. It then shows 'INFO:root:End of audio request detected.', 'INFO:root:Stopping recording.', 'INFO:root:Transcript of user request: "what is today's date and time".', and finally 'INFO:root:Playing assistant response.'

Using the Google Assistant on the Raspberry Pi

- Now that we have finally fully authorized our Raspberry Pi as Google Assistant.
- So for **next time login**, if you want to **run the Google Assistant software without having to go through the entire tutorial, follow above steps**
- Start a new terminal session and set it into virtual environment.(**source env/bin/activate**)
- To start up the push to talk sample, run the **googlesamples-assistant-pushtotalk** command, this time **we do not need** to write in the **project id or the device id** as these were cached when we first utilized the push to talk tool.

```
$ source env/bin/activate
```

```
$ googlesamples-assistant-pushtotalk
```



```
pi@raspberrypi ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ source env/bin/activate  
(env) pi@raspberrypi:~ $ googlesamples-assistant-pushtotalk  
INFO:root:Connecting to embeddedassistant.googleapis.com  
INFO:root:Using device model msdhomepi-213812-msd-pi-p1jgi7 and device id c52dbd  
a0-a3ad-11e8-b68b-b827eb94feed  
Press Enter to send a new request...
```

Getting the Google Assistant to start up on boot

Up to this time, you have got your Google Assistant up and running

Now you will likely want to get it to start up on boot rather than having to go to the effort of entering the commands every time.

The easiest way to achieve this is to create a service for it. This service will allow the Google assistant to run in the background on the Raspberry Pi and easily allow us to retain control over it.

Please follow below tutorial

<https://drive.google.com/open?id=1dYSHka8jSAOaCrhGaKNQ28IA9GHaO8BY>

That's all!!!

Thank you....