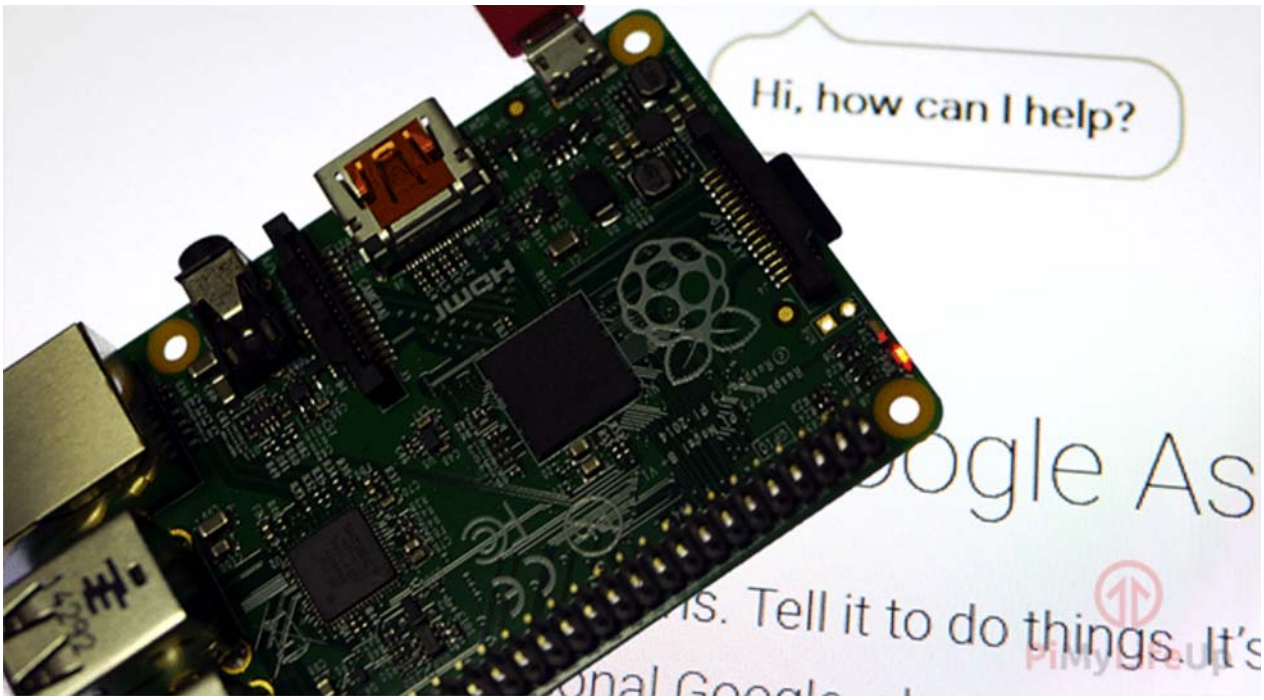
PiMyLifeUp

Building your own Raspberry Pi Google Assistant

by Gus Mar 25, 2018 Updated Aug 21, 2018 Beginner, IoT



This Raspberry Pi Google assistant project will walk you through on how to build and set up your very own Pi-powered Google Assistant. This assistant will actively listen to your voice and respond to your queries, all you need to say is “Ok Google” or “Hey Google” followed by your query.

To set up your own Google Assistant, we will be walking you through how to test your audio setup, signing up for the Google Assistant API and also show you how to download

and set up the actual Google Assistant examples. By the end of this tutorial, you should have a very capable assistant operating.

Setting up your very own Google Assistant on your Raspberry Pi all you will need is a USB Microphone and a set of speakers. Of course, you will also need all the default equipment that you need for getting your Raspberry Pi started.

Equipment List

The entire list of all the pieces of equipment that we used for this Raspberry Pi Google assistant tutorial is listed below.

Recommended:

- Raspberry Pi 2 or 3
- Micro SD Card
- Power Supply
- USB Microphone
- Speakers
- Ethernet Network Connection or Wifi dongle (The Pi 3 has WiFi inbuilt)

Optional:

- Raspberry Pi Case

Registering for the Google API

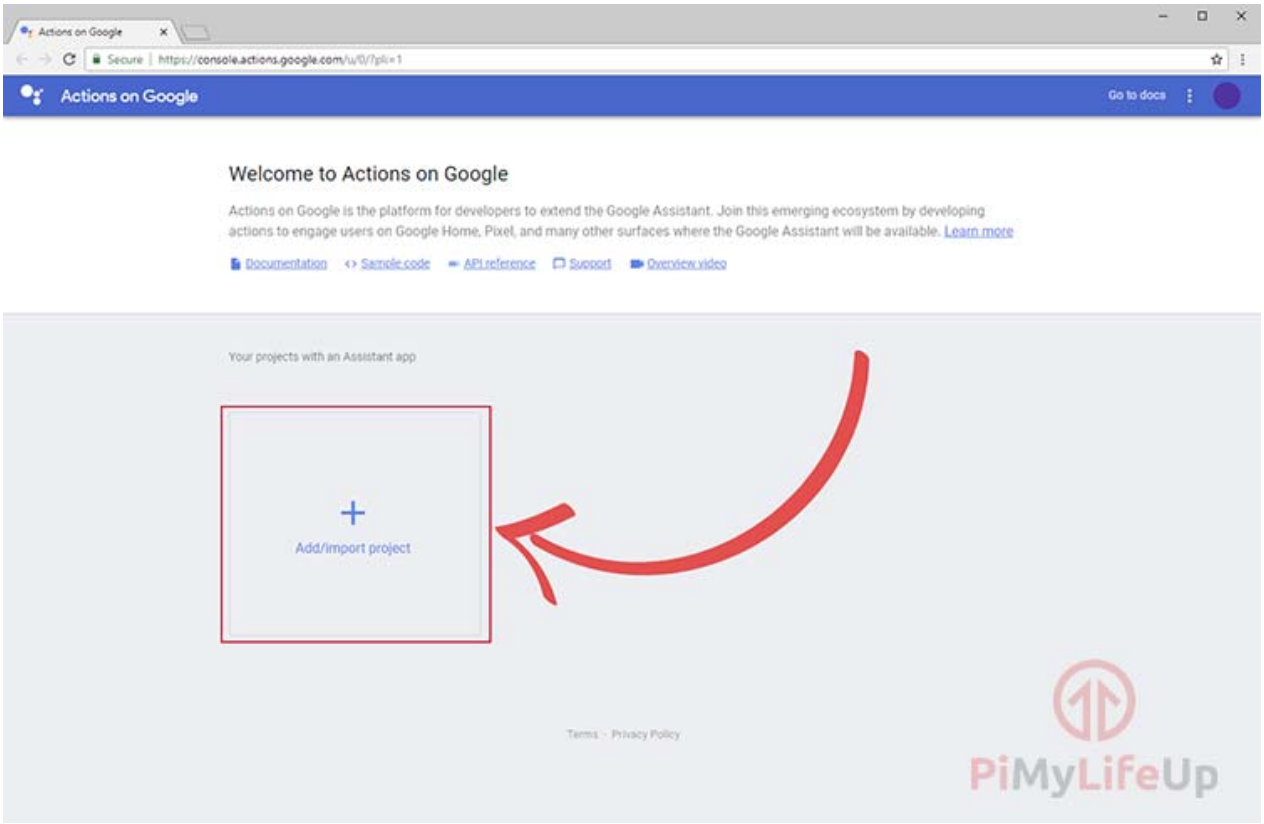
1. Before we get started with setting up the Google Assistant code on the Raspberry Pi itself, we must first register and set up a project on the Google Actions Console.

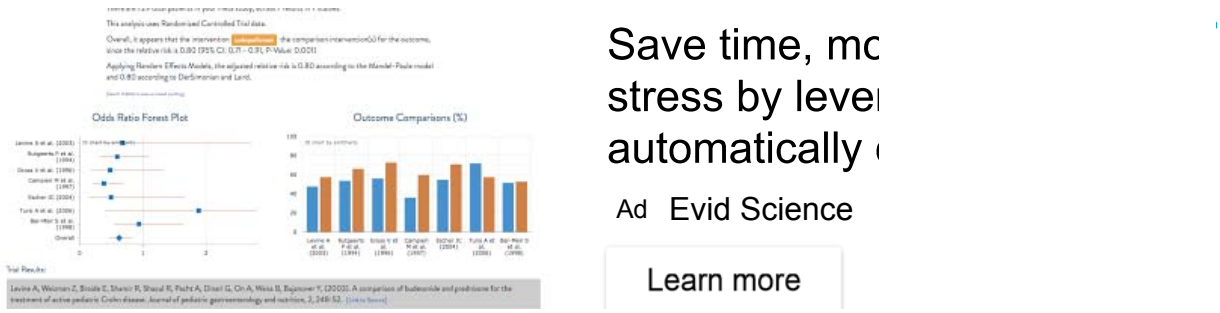
With your Google account ready to go to the Google Console Actions dashboard, or just go to the URL below which will take you there.

```
https://console.actions.google.com
```

2. Once you have logged into your account, you will be greeted with the following screen.

On here you will want to click the "**Add/Import project**" button as shown in our screenshot below.

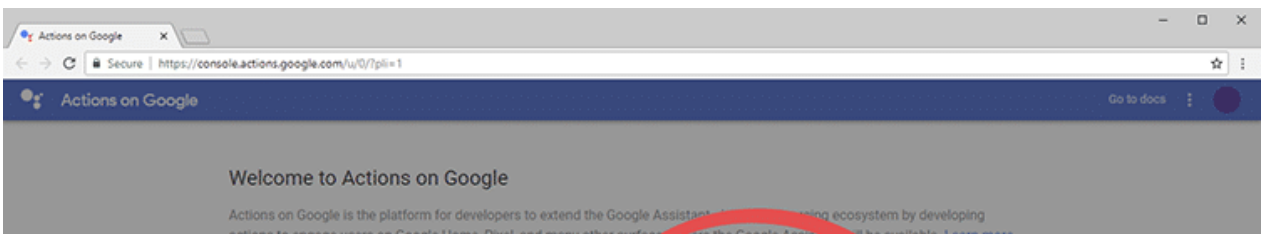


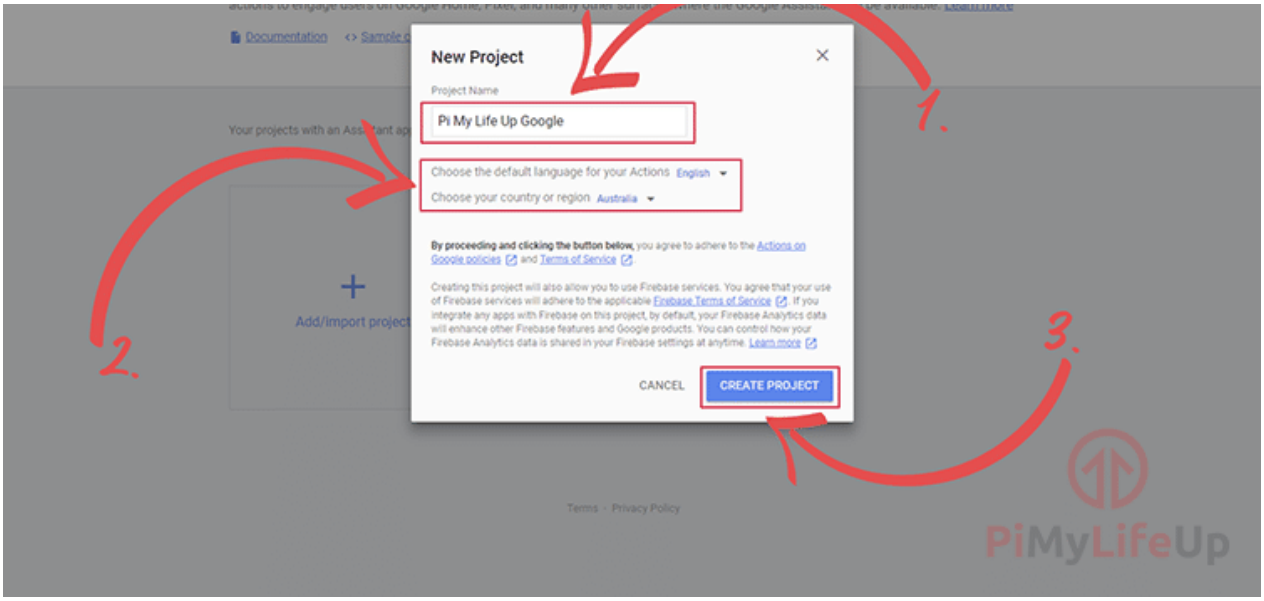


3. On this next screen, you will be asked to enter a "**Project Name**" (**1.**)

In addition to a project name you need to set both your country and your language as shown in the screenshot (**2.**)

Once you have set the Project Name and chosen your language and country, click the "**Create Project**" (**3.**) button.

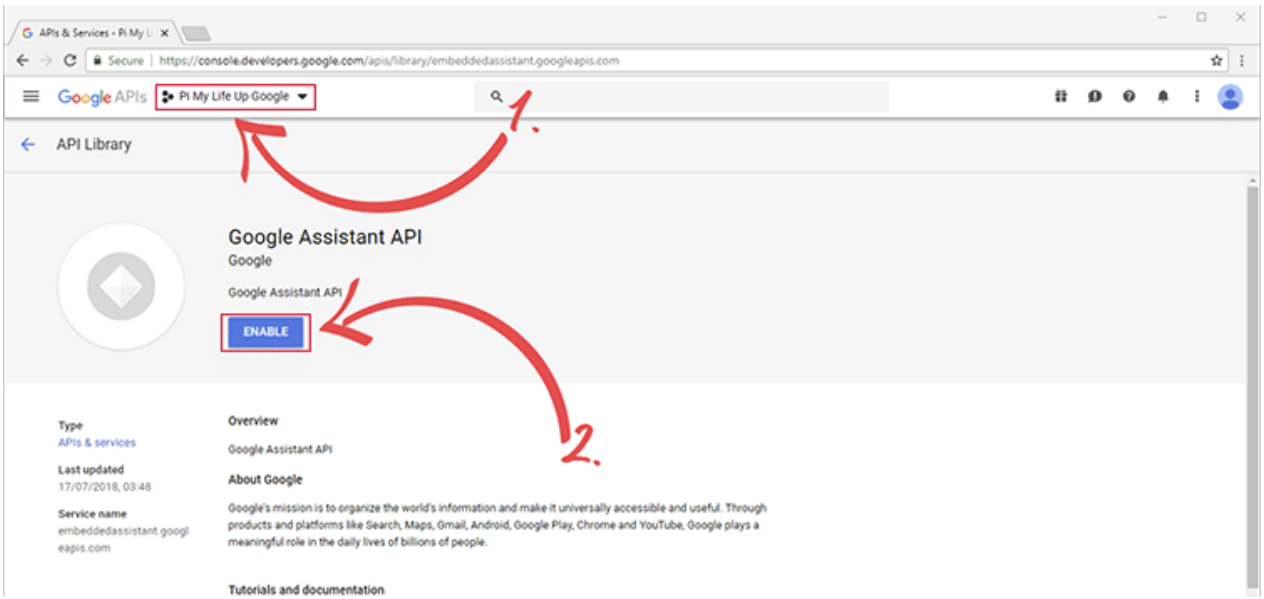




4. In a **new** tab, go to the Google developers console and enable the Google Embedded Assistant API.

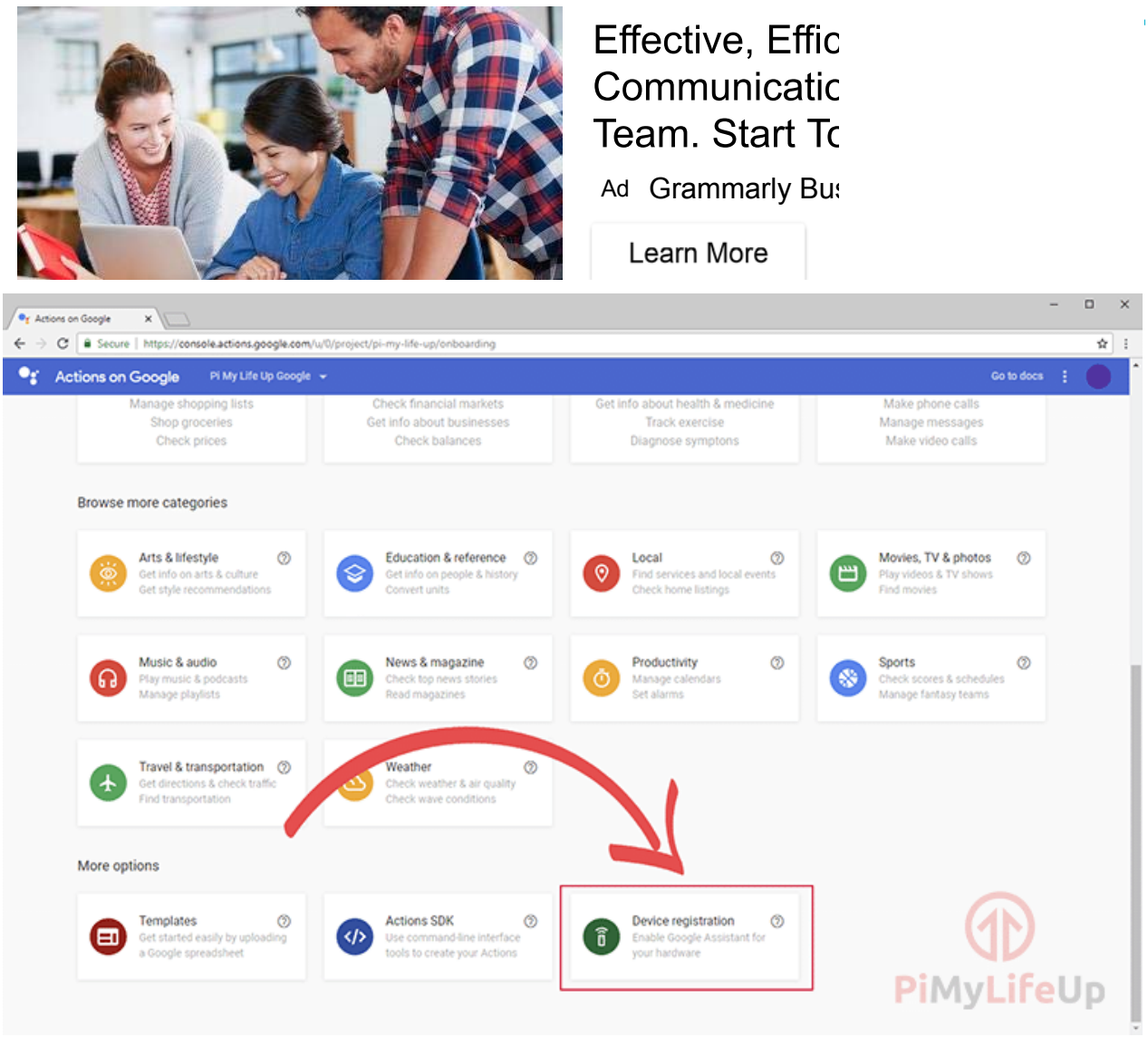
Now before you go ahead and press the “**Enable**” button make sure that you have your project selected (**1.**)

Once you are sure you have your current project selected, click the “**Enable**” button (**2.**)

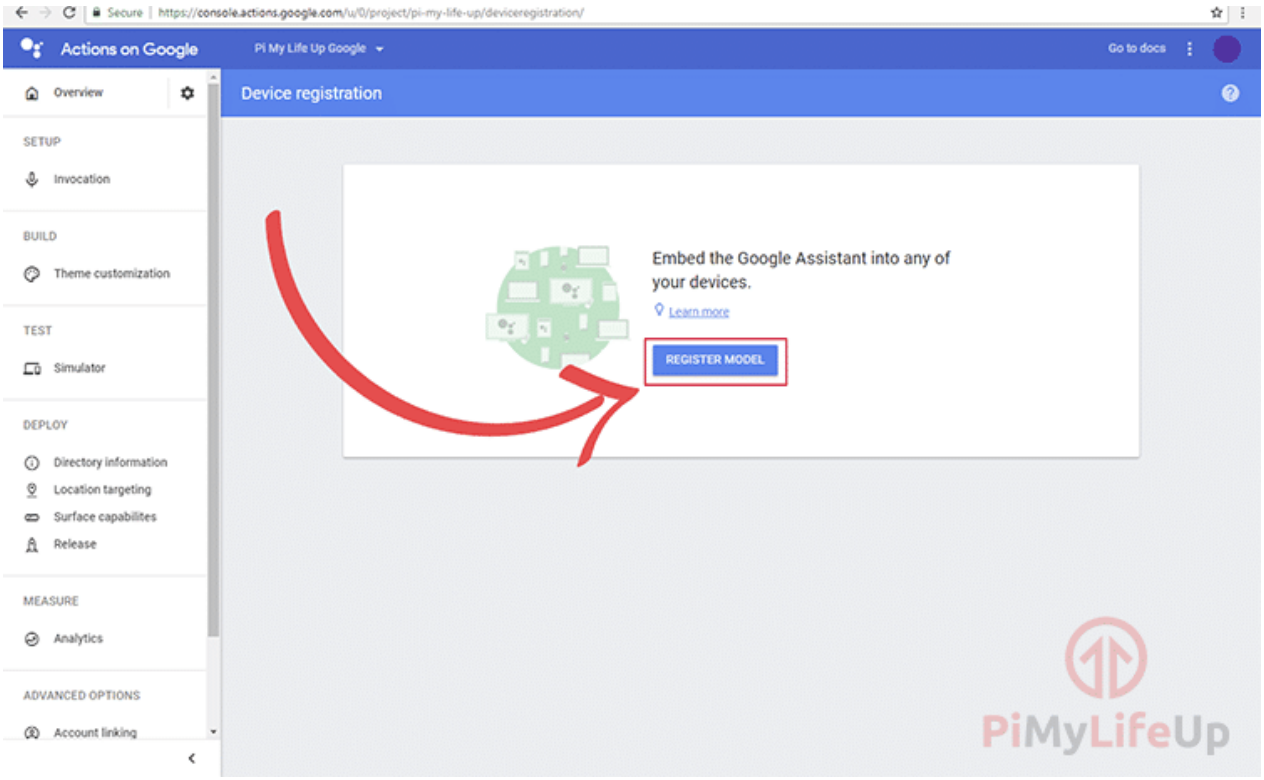


5. Now back in the other tab where you created the project, scroll down to the bottom of the screen.

You should see a box with the text "**Device Registration**" on it as we have shown in the screenshot below. Click it to continue.



6. You will now be taken to the following screen, click the "**Register Model**" button to continue.



7. On this screen, you need to set a "**Product Name**", "**Manufacturer name**" and set a "**Device Type**" (**1.**)

Below you can see the data that we entered into it, it doesn't hugely matter what you set here, but all three boxes do need to be set for you to be able to register your model.

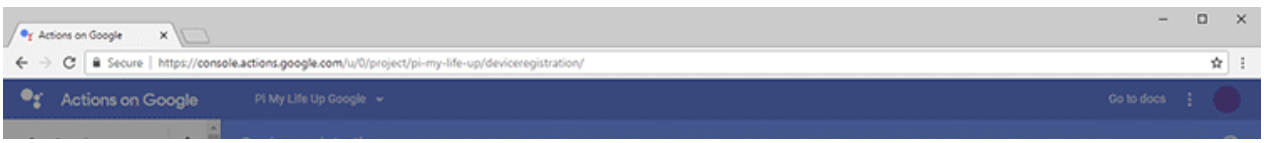
For the "**Product Name**" we just set this as a simple descriptor of what we are using this for, which in this tutorials case is simply a "**pi3 Google Assistant**".

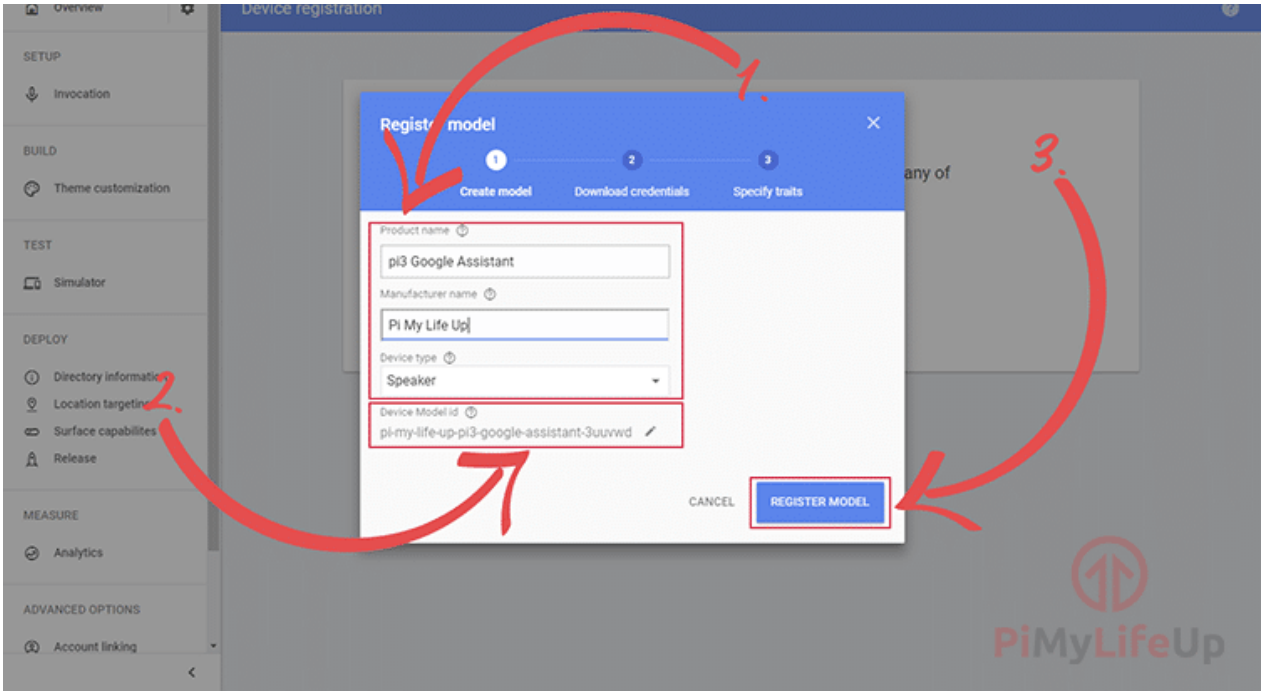
"**Manufacturer name**" doesn't hugely matter as we have no intention of this being a widely used device, so we just set this to our websites name "**Pi My Life Up**".

Lastly, we set the "**Device Type**" as "**Speaker**" as we felt it matched best what we intend on using the Google Assistant API for on our Raspberry Pi.

Make sure you write down then "**Device Model ID**" (**2.**) as you will need this later in the tutorial

Finally, once everything is set, and you have written down the "**Device Model ID**" press the "**Register Model**" (**3.**) button to continue.





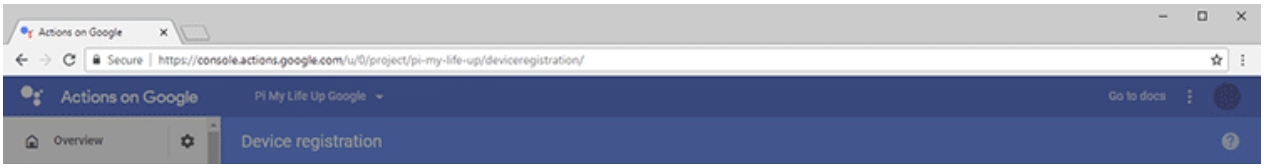
8. Now that you have registered the model you will now be taken to the "**Download credentials**" screen.

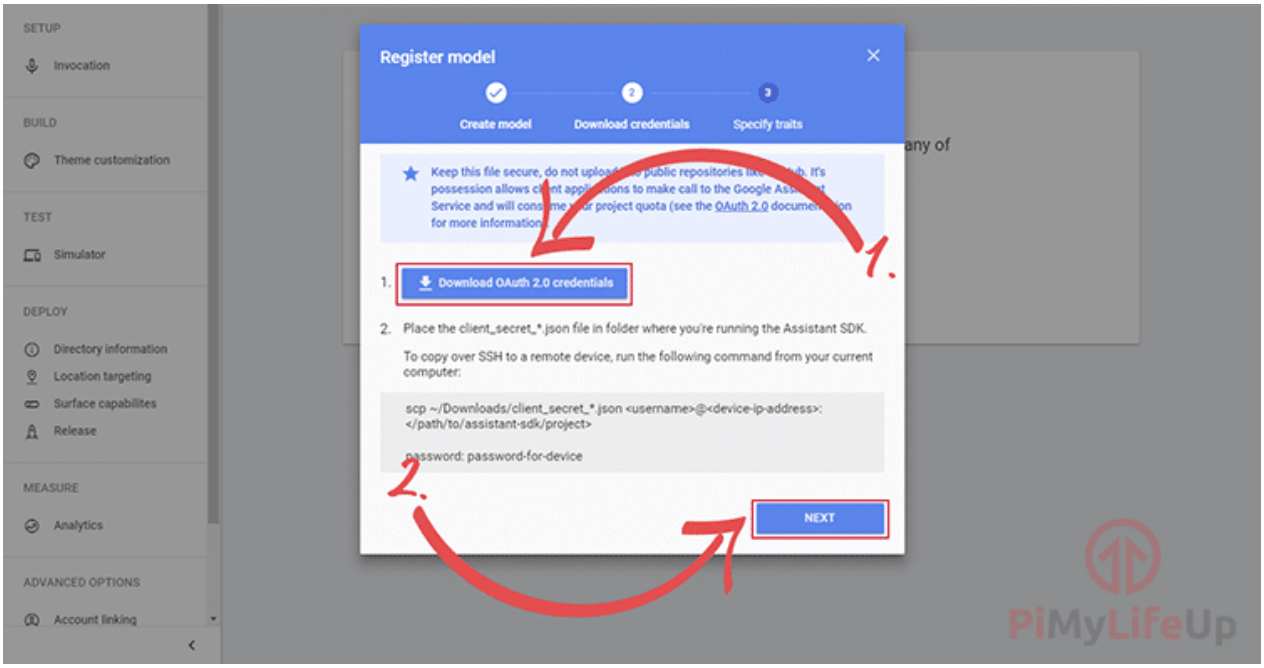
This screen is crucial as the provided credentials file is what we need for our Raspberry Pi 3 based Google Assistant to talk with the server.

To get this credentials file click the "**Download OAuth 2.0 credentials**" (**1.**) button as shown on the screenshot below.

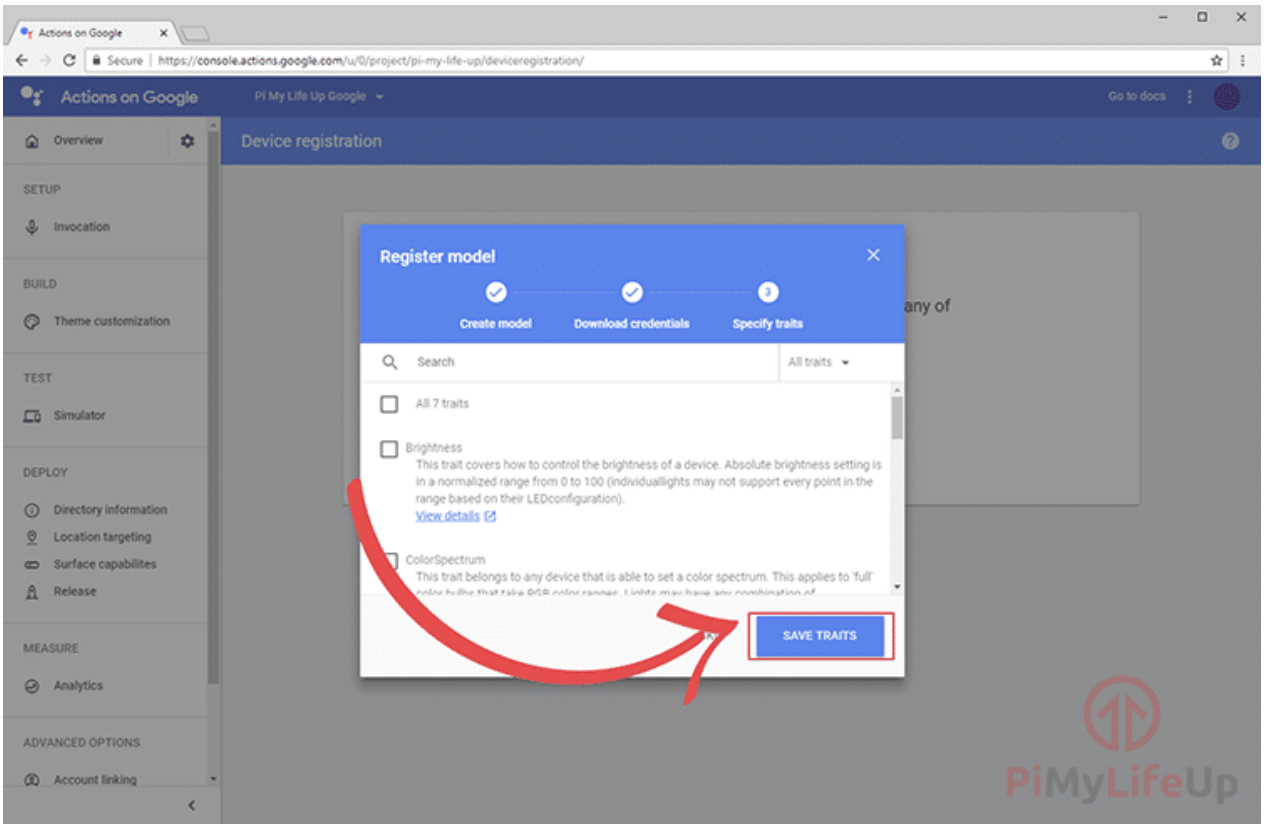
Keep this somewhere safe, as we will the text inside the file to the Raspberry Pi. (Of course, unless you downloaded it directly to your Pi)

Once you have the credentials safely stored on your on your computer or Raspberry Pi, you need to click the "**Next**" (**2.**) button.

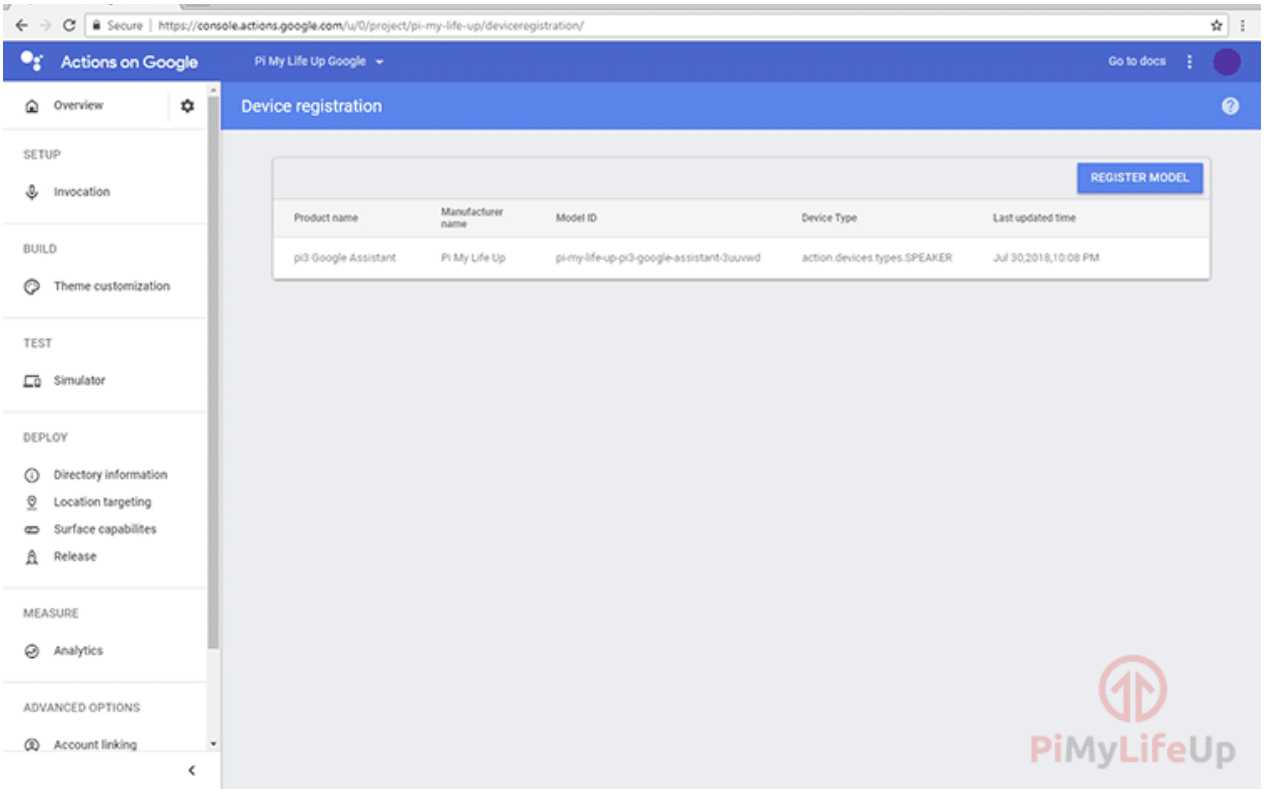




9. Finally, you can specify any traits that you might need, in our case we don't need any of these so we just clicked the "**Save Traits**" button as shown below.



10. Once everything is done, you should be shown on this screen. We now only have one last thing we need to do before we can set up the Google Assistant on the Raspberry Pi itself.



11. Finally, we need to go to the URL displayed below, on here you will need to activate the following activity controls to ensure that the Google Assistant API works correctly.

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

```
https://myaccount.google.com/activitycontrols
```

Setting up your Audio for Google Assistant

1. Now that we have set up an account on the Google Actions Console we must configure the audio for it. The Google Assistant SDK that we will be using has some strict requirements for it to work correctly.

To get started with setting up the audio on the Raspberry Pi we must first obtain the card and device numbers for our various inputs and outputs. Our steps below will show you have to get the correct numbers for these devices.

1a. Locate your USB microphone by utilizing the following command. Write down both the card number and the device number for it.

```
arecord -l
```

1b. Now to locate your speaker, we will be utilizing the following command. Again write down the card number and device number.

Note that the Raspberry Pi's **3.5mm-jack** is typically labeled as **Analog** or **bcm2835 ALSA**, with the **HDMI output** being identified as **bcm2835 IEC958/HDMI**.

```
aplay -l
```

2. Now that we have grabbed our device and card numbers for both the microphone and our audio output we need to create a file named **.asoundrc** in the pi users home directory.

The **.asoundrc** file helps by defining which audio devices that the audio driver should be utilizing.

We can create this file by running the following command within the terminal.

```
nano /home/pi/.asoundrc
```

3. Within this file enter the following lines.

Make sure that you replace **<card number>** and with **<device number>** their respective values that you retrieved during **Step 1**.

```
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>"
  }
}
```

4. With the lines in, and changes made. Save by pressing **CTRL + X** then **Y** and finally **ENTER**.

Downloading and setting up Google Assistant

1. Before we get into all the hard work of setting up our Raspberry Pi Google Assistant and setting up the required API .we will first test to ensure our audio is working.

At this stage, you must have your USB microphone and speakers attached to your Raspberry Pi.

Once you are sure both are connected to the Raspberry Pi, we can test to make sure that the speakers are working correctly by running the following command.

```
speaker-test -t wav
```

You should hear sound from your speakers. This sound will be a person speaking.

If you do not hear anything coming from your speaker's double check they are plugged in correctly and are turned up.

2. Now, let's test our microphone by making a recording, to do this we will run the following command on your Raspberry Pi.

This command will make a short 5-second recording.

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

If you receive an error when running this command make sure that you have your microphone plugged in, this command will only succeed if it can successfully listen to your microphone.

3. With our recording done we can now run the following command to read in our raw

output file and play it back to our speakers.

Doing this will allow you to test the playback volume and also listen to the recording volume.

Doing this is a crucial task as you don't want your Raspberry Pi picking up every little noise but you also don't want it being able to barely hear you when you say "**Ok Google**".

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

4. If you find the playback volume or recording volume is either too high or too low, then you can run the following command to launch the mixer.

This command will allow you to tweak the output volumes for your various output devices. From our tests, we recommend you use a level of at least 70, utilize the command in **Step 1** of this section to check the volume levels.

```
alsamixer
```

Once you have confirmed that your microphone and speakers are working correctly, you can move onto setting up your very own Raspberry Pi Google Assistant!

Downloading and setting up Google Assistant

1. With the Google Assistant API now configured and set up there are a few things we need to do.

Please note from this part onwards you will be required to complete the tutorial on the Raspbian desktop directly and not over SSH, this is since you will have to use the built-in web browser.

Let's first update the Raspberry Pi's package list by running the following command.

```
sudo apt-get update
```

2. Once the Raspberry Pi has finished updating, we can then proceed with setting up

everything we need for running the Google Assistant API.

On your Raspberry Pi, we will be creating a file where we will store the credentials we downloaded earlier on our computer.

To do this run the following two commands to create a folder and begin writing our credential file.

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

3. Within this file, we need to copy the contents of the credentials file that we downloaded to your computer. You can open the **.json** file in any text editor and press **CTRL + A** then **CTRL + C** to copy the contents.

Now in your SSH window, **right click** and click **"Paste"**.

4. Once you have copied the contents of your credentials over to our nano session, we can then save the file by pressing **Ctrl + X** then **Y** and then finally hitting **Enter**.

5. Now with the credentials file now saved safely to our Raspberry Pi we will start installing some of the dependencies we rely on.

Run the following command to install Python3 and the Python 3 Virtual Environment to our Raspberry Pi.

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

6. We can now enable python3 as our virtual environment variable by running the following command on our Raspberry Pi.

```
python3 -m venv env
```

7. With that now enabled we can go ahead and ensure that we have installed the latest versions of pip and the **setuptools**. To do this, we will run the following command on the Raspberry Pi.

```
env/bin/python -m pip install --upgrade pip setuptools --upgrade
```

8. To get into this new Python environment that we have set up we should run the following command in terminal.

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

9. Now that we have all the packages we need to install the Google Assistant Library, to do this we will run the following command to utilize pip to install the latest version of the Python package.

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

Authorizing your Raspberry Pi for the Google Assistant

1. Now that we have set up all the prerequisites to running the Google Assistant software on our Raspberry Pi we can finally complete the last few steps to authorize the system.

To do this, we must first install the Google authorization tool to our Raspberry Pi. This package will allow us to authenticate our device and give ourselves the rights to be able to make Google Assistant queries for your Google Account.

Run the following command on the Raspberry Pi to install the Python authorization tool.

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

2. With the Google Authentication library now installed, we need to run it. To do this, we will be running the following command on our Raspberry Pi.

This command will generate a URL you will need to go to in your web browser so be prepared.

```
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
```

3. You will now be presented with the text "**Please visit this URL to authorize this application:**" followed by a very long URL. Make sure you copy this URL entirely to your web browser to open it.

4. 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.

Afterward, you should be presented with a screen with the text "**Please copy this code, switch to your application and paste it there**" followed by a long authentication code.

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

If the authentication was accepted you should see the following line appear on your command line:
"**credentials saved: /home/pi/.config/google-oauthlib-tool/credentials.json**"

5. Now with the authentication credentials now saved, Google still requires us to verify ourselves through a popup display.

At the time of writing this requires us to launch Chromium with CORS disabled otherwise you will run into errors, and the device will not be authorized.

To do this make sure you have all instance of Chromium on your Raspberry Pi closed, open up a **new terminal session** and enter the command below.

This command will launch the browser with the CORS security disabled, don't browse the internet with this disabled.

```
chromium-browser --disable-web-security --user-data-dir "/home/pi/"
```

Once the browser opens, go back to your other terminal session as that is already set up to run the Google Assistant samples.

6. Now with the authentication credentials now saved, Google still requires us to agree

to some stuff.

With this command make sure you replace **<projectid>** with the id of your project.

If you don't know what your Project ID is you can go to Actions Console on Google, click the project you created, then click the Cog in the top left-hand corner then "**Project Settings**".

Also, make sure you replace **<deviceid>** with the device ID that you obtained earlier in the tutorial.

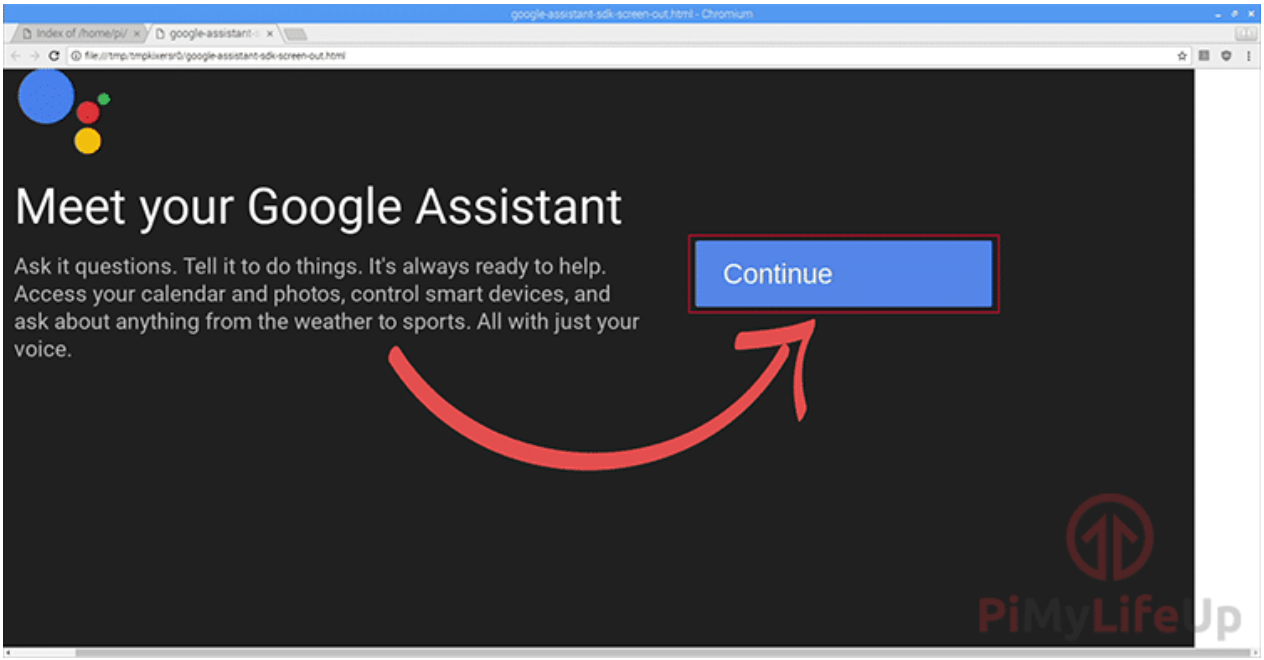
If you have lost your device ID can find it again by going to the Console Actions website, clicking the project you created then going to "**Device registration**"

For our first run of the googlesamples code we must run it with **–display**, this will load up a new tab in the Chromium browser we launched earlier.

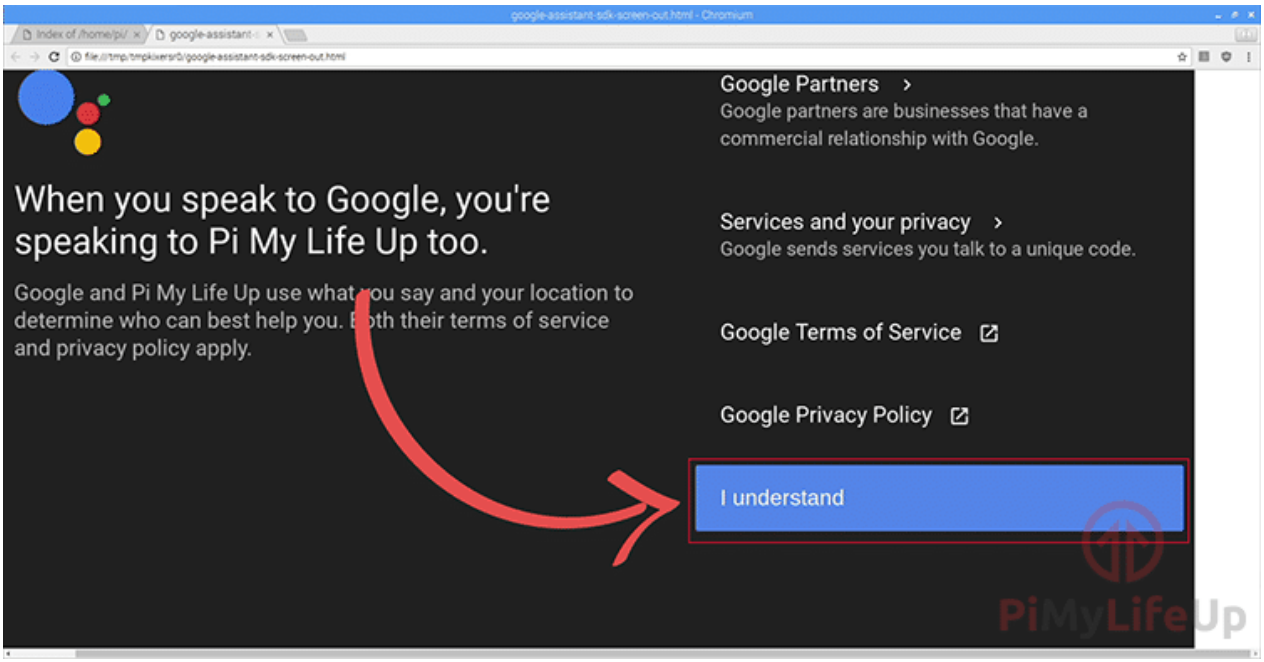
```
googlesamples-assistant-pushtotalk --project-id <projectid> --device-model
```

7. After running the push to talk sample, press "**Enter**" to trigger it and ask it any question.

Upon asking your first question, you will be shown the screen below, begin by clicking the "**Continue**".

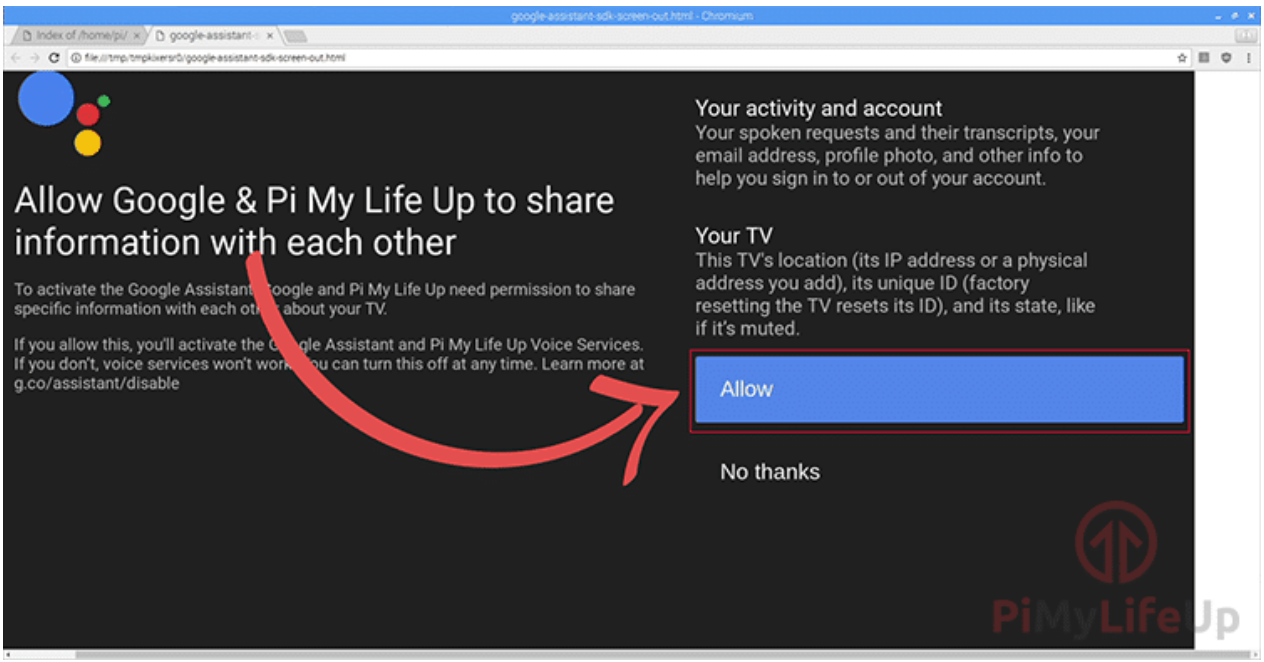


8. After selecting "**Continue**" you will now be asked if you agree to a variety of different Google policies, to continue you must click the "**I understand**" button.



9. Finally, you will be asked to allow Google and your project the right to be able to share information with each other. Without this, the Google Assistant project will not function correctly.

To continue on you must click the "**Allow**" button as showcased below.



10. With that now done we can now use the push to talk Google Assistant sample and

hear a response.

This time when you press the "**Enter**" in the terminal and speak an action such as "**What is the time**" you should hear a verbal response and another tab will be automatically opened displaying the action you just called.

Don't worry you can disable the tab behavior by removing the **–display** argument on the command. We only needed this to get up the authorization screen.

If you don't hear any response from the Google Assistant but a new tab does open that contains your results, then you should go back to the "**Setting up your Audio for Google Assistant**" section and the "**Testing your Audio for Google Assistant**" section.

In the next section, we will show you how you can rerun the Google Assistant without having to go searching through all the steps again, make sure you keep your "**Project ID**" and "**Device ID**" handy as we still need those to utilize the samples.

▶ Using the Google Assistant on the Raspberry Pi

1. Now that we have finally fully authorized our Raspberry Pi we will now walk you through the steps on how to run the Google Assistant software without having to go through the entire tutorial.

To begin with whenever you start a new terminal session you will need to put it back into the environment that we set up the Google Assistant software in.

To do this, run the following command on your Raspberry Pi.

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

2. You will know whether this has worked correctly by seeing **(env)** appear at the front of each line.

With **(env)** appearing it means you can once again make calls to the Google Assistant samples.

To start up the push to talk sample you will need to run the following command, lucky for us, 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.

```
googlesamples-assistant-pushtotalk
```

3. In addition to the push to talk sample, there is also the hotword sample code. This sample code will work by listening for certain phrases to trigger it. By default, this will listen for "**Ok Google**" followed by the command that you want to issue to it.

Starting the hotword sample is very much like the push to talk sample. However, you will be required to enter an invalid device id for it to work currently.

This incorrect device ID could be as simple as "**abcd**".

```
googlesamples-assistant-hotword --device-model-id <deviceid>
```

4. Hopefully, by now you will have successfully set up your Raspberry Pi with the Google Assistant software.

If you run into audio issues, we recommend you go back to our "**Setting up your Audio for Google Assistant**" section and the "**Testing your Audio for Google Assistant**" section.

Getting the Google Assistant to start on start up

1. Now that you have got your Google Assistant up and running 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.

To begin, we will write our bash script that will execute the hotword sample. We will be calling this bash script from our service file.

```
nano /home/pi/start_assistant.sh
```

2. To this file, you will want to add the following lines.

The first line helps define what should be utilized to run this file, so when the command line interprets it, it will know that it is required to make use of bash.

The second line we use **source** to read and execute commands from the activate file, this will put us into the python environment that we created earlier in the tutorial.

Finally, we run the hotwords Google Assistant sample. Again we are setting an incorrect device id as at the time of writing, using a correct one will break the sample.

```
#!/bin/bash

source /home/pi/env/bin/activate
googlesamples-assistant-hotword --device-model-id abcd
```

3. Once you have added those lines to the file, save it by pressing **CTRL + X** then **Y** and finally **ENTER**.

4. Now that we have created our bash script we can move onto creating our service file for our Google Assistant by running the following command on our Raspberry Pi.

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

5. Within this file enter the following lines.

These lines tell the operating system how to handle our service, what user to run it with and if it relies on a particular thing such as that the network is available.

```
[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
```

6. With those lines added to the service file, save it by pressing **CTRL + X** then **Y** and finally **ENTER**.

7. Now that we have created the service we need to enable it so that it will actually start on startup.

To do this we just need to use systemctl, type in the following command to enable our new service.

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

8. With the service now enabled let's try starting it, this will allow us to make sure that everything is working as intended.

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

9. You should now be able to use the "**Ok Google**" hotword to talk and interact with the Google Assistant.

Try it now to ensure the service is working as intended.

If for some reason it is not working you can utilize the command below to display the last output from the Google Assistant sample, keep a lookout for any errors.

```
sudo systemctl status assistant.service
```

10. Hopefully, at this point, you will now have your Google Assistant successfully set up to start on boot up.

Love personal assistants, then you might also want to check out our tutorial on setting up Alexa on the Raspberry Pi. It's pretty easy to set ip and quite a bit of fun once you get it working.

If you hear a response, you have successfully set up your very own Google Assistant on your Raspberry Pi. If you want to provide feedback or discuss this Raspberry Pi Google

assistant project, then head to our forums.

Share

Google AIY Kits now Include the Raspberry Pi Zero WH

Raspberry Pi Alexa: Build your own Amazon Echo

Setup your own Raspberry Pi AirPlay Receiver

An Amazing DIY Raspberry Pi Music Player

Raspberry Pi Chromium: Learn How to Install Chromium OS

Raspberry Pi Wireless Access Point

Comments

Sorry comments are closed! For help, feedback and more, head over to the forums.

Follow us on social

Search for Tutorials

Search