Hands Free Google Assistant for Raspberry Pi

by LukeK1990

Hello and welcome to my first Instructable!

In this instructable I am going to show you what I consider to be the easiest way to install an all singing, all dancing Google Assistant on your Raspberry Pi. She is completely hands free with the OK Google command and she starts up automatically when you boot up your Pi. She's super cool and seriously easy to setup!

So how's this all possible?

A short while ago Google released a do-it-yourself AI kit with issue #57 of The Magpi. This made it extremely easy to create your own Google assistant however getting hold of the voice kit was little bit harder and in many places it sold out within hours.

Fortunately, Google made all the software available online complete with full instructions. This meant that we did not need a copy of The Magpi to take advantage of all that hard work.

Despite this, there doesn't appear to be any clear instructions online on utilizing the voice kit without a copy of the magazine or without the hardware that was shipped with it. Instead, most of the tutorials attempt to install everything from scratch often resulting in mess of code that's impossible to follow for non-coders like me.

And that's where this Instructable comes in. It is the outcome of 2 days hard work looking at hundreds of tutorials, videos and posts online. The result is a stable Google Assistant which **runs on startup** and is **voice activated** with the 'OK Google' hotword.

What's required?

To complete this Instructable successfully you will need the following:

- A Raspberry Pi 3 (with the usual microSD card and power cord).
- A basic speaker with a 3.5 mm aux connection
- A USB microphone
- A mouse and keyboard

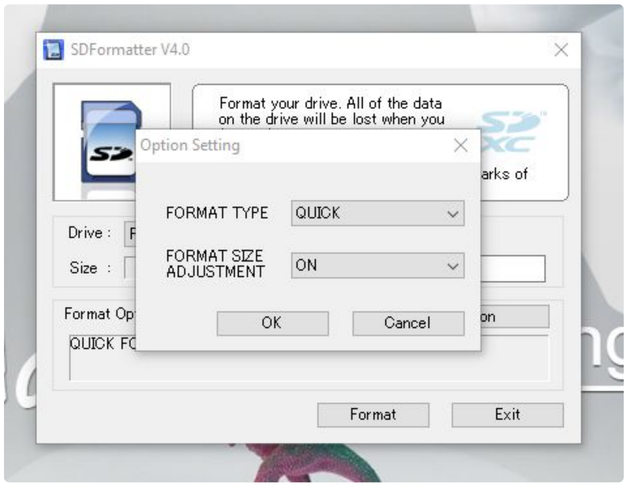
https://youtu.be/iMNe0-EDWBM

Step 1: Formatting the SD Card

The very first thing that we need to do is to format our SD card. Let’s use the SD Association's Formatting Tool (https://www.sdcard.org/downloads/formatter_4/) which is recommended by the official Raspberry Pi Foundation.

Once installed, launch the application and click ‘**Option**’ You need to change the option for ‘**FORMAT SIZE ADJUSTMENT**’ to ‘**ON**’.

Now click ‘**OK**’ and double check that we are formatting the correct drive, then click ‘**Format**’. This shouldn’t take too long, just wait for the confirmation that the drive was formatted successfully before you move onto the next step.



FORMAT TYPE QUICK

FORMAT SIZE ADJUSTMENT ON

Step 2: Preparing the SD Card

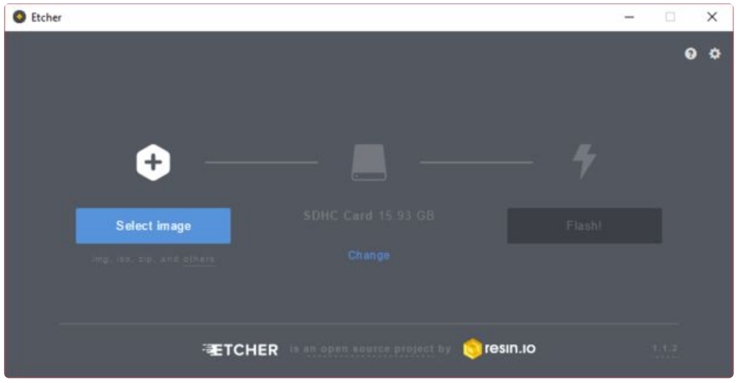
Next we need to download the Voice Kit microSD card image for the Raspberry Pi. You can download the image directly from Google’s AIY Project (https://dl.google.com/dl/aiyprojects/voice/aiyprojects-latest.img.xz) site.

In order to transfer the image that we just downloaded onto our SD card we are going to use a program called Etcher.io (https://etcher.io/). It’s free, open source and does not require installation.

Once you’ve downloaded Etcher, run the program and you should see a screen like the one above. It can take a minute or two to load so if it doesn’t load right away be patient.

Click ‘**Select image**’ and navigate to the voice kit image that we just downloaded (aiyprojects-2017-05-03.img). Once selected double check that you are writing the contents onto the correct disk. Assuming that we have the correct disk selected then click ‘**Flash!**’.

It can take around 20 minutes or more to write the image to your SD card so feel free to go and make yourself a nice cup of tea and I will see you back here in a bit!

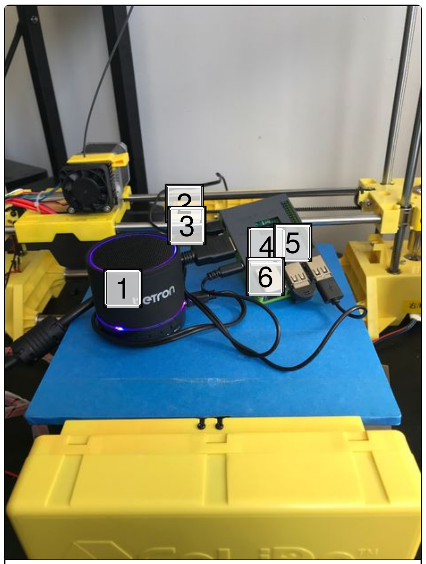

Etcher
Select image
img, iso, zip, and others
SDHC Card 15.93 GB
Change
Flash!
ETCHER is an open source project by resin.io
1.1.2

Step 3: Power Up That Pi!

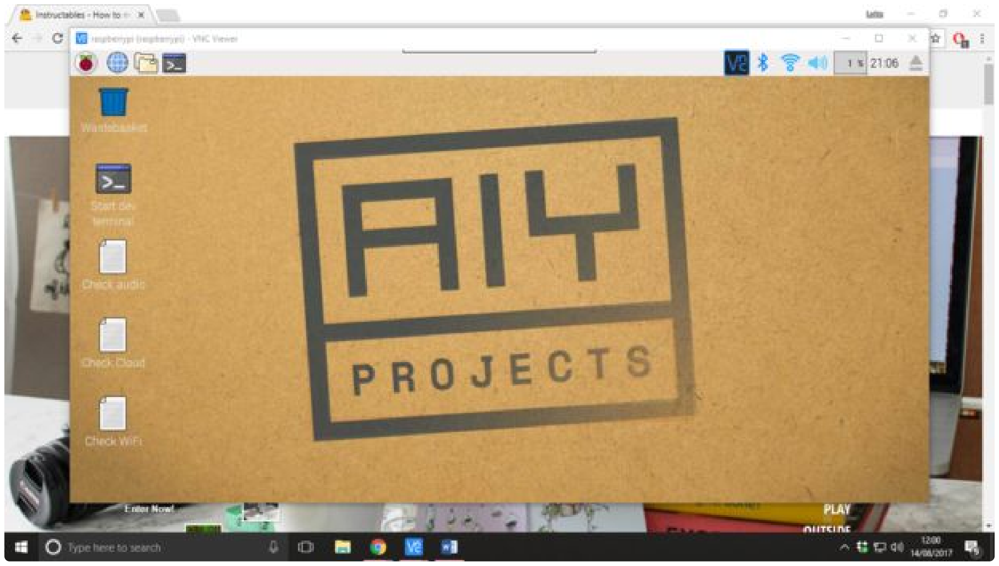
As soon as the SD card is ready we can put the microSD card into our Raspberry Pi. At this point we also need to connect our power lead, HDMI cable, keyboard, mouse, monitor, speaker and USB microphone. Hopefully your setup should look something like mine.

I am using a wireless keyboard and mouse so don’t freak out if you end up with a couple of extra cables with your own setup!

With the power cable plugged in allow your Raspberry Pi to boot up and you will soon be presented with the standard Pixel desktop.



1. Aux Speaker
2. Raspberry Pi Power Lead
3. HDMI Cable (Connects your PI to a monitor)
4. USB Microphone
5. USB Power supply for my AUX Speaker
6. USB for my wireless mouse and keyboard - you cant really see it in this photo but it is there, promise!

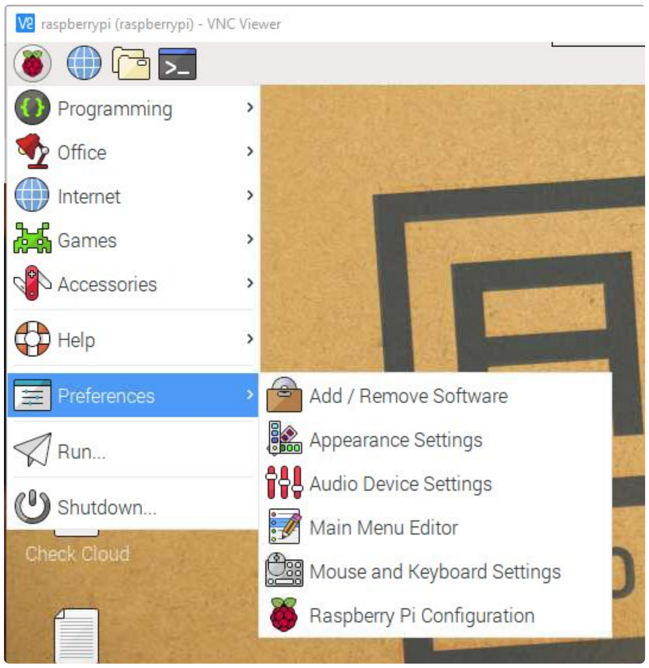


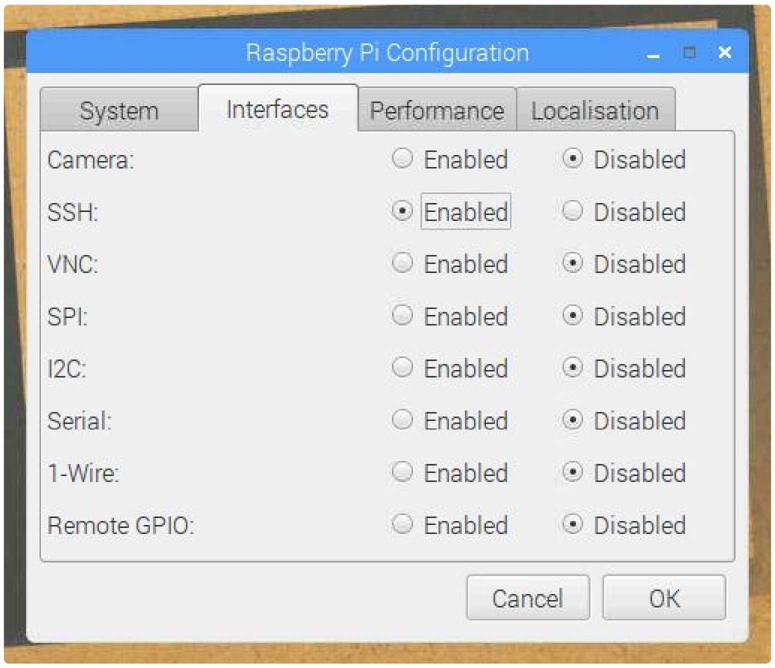
Step 4: Initial Configuration

Grab your cursor and navigate to the Raspberry Pi logo at the top left of your screen. From the dropdown select ‘**Preferences**’ and then ‘**Raspberry Pi Configuration**’.

Next, go to ‘**Interfaces**’ and enable '**SSH**'.

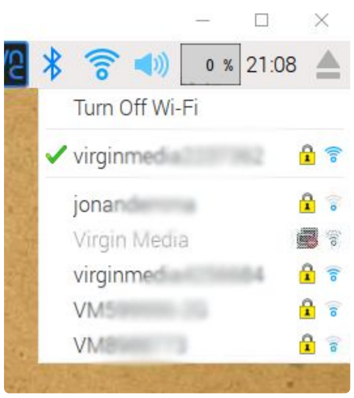
Now click on the WiFi logo at the top right of the screen and select your WiFi network. If your WiFi is password protected you will prompted to enter that in. The green tick confirms that we connected successfully and we’re ready to move onto the next step.




Raspberry Pi Configuration

System | Interfaces | Performance | Localisation

Camera:	Enabled	[✓ Disabled]
SSH:	[✓ Enabled]	Disabled
VNC:	Enabled	[✓ Disabled]
SPI:	Enabled	[✓ Disabled]
I2C:	Enabled	[✓ Disabled]
Serial:	Enabled	[✓ Disabled]
1-Wire:	Enabled	[✓ Disabled]
Remote GPIO:	Enabled	[✓ Disabled]



Step 5: Updating the Installation

This is about as advanced as this tutorial gets. We are going to use the command prompt to update the Google Assistant SDK, Project Kit & dependencies to ensure we have the latest versions. Do not panic if none of this makes much sense to you but it is essential that we do not skip this step. Just do exactly as this tutorial says being careful not to make any typos and it will all work out just fine.

So, panic over, let's begin! Double click on the desktop icon name '**Start dev terminal**'.

You should see a scary looking command window like the one above.

Next type the following commands into the terminal exactly as they appear below. There are 9 commands here and each should be entered separately in the order in which they appear. After typing each command hit '**Enter**' on your keyboard before moving onto the next one. Some of the commands will take a few seconds to complete so be patient waiting for each to finish before moving onto the next one.

cd ~/assistant-sdk-python

git checkout master

git pull origin master

cd ~/voice-recognizer-raspi

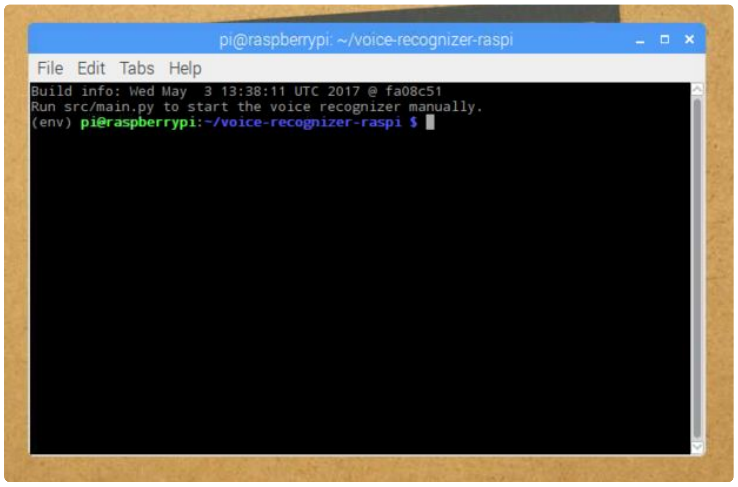
git checkout master

git pull origin master

cd ~/voice-recognizer-raspi

rm -rf env

scripts/install-deps.sh



```
Build info: Wed May  3 13:38:11 UTC 2017 @ fa08c51
Run src/main.py to start the voice recognizer manually.
(env) pi@raspberrypi:~/voice-recognizer-raspi $
```

Step 6: Preparing the Configuration Files

Next we need to backup our existing configurations files and bring over the newest versions that were just updated. Here's 4 more commands for you to do just that. These can be done in the same command window that we were just using. Once again, they must be done in this order and should be typed precisely as they appear below:

```
cp ~/.config/status-led.ini ~/.config/status-led.ini~
```

```
cp ~/.config/voice-recognizer.ini ~/.config/voice-recognizer.ini~
```

```
cp ~/voice-recognizer-raspi/config/status-led.ini.default ~/.config/status-led.ini
```

```
cp ~/voice-recognizer-raspi/config/voice-recognizer.ini.default ~/.config/voice-recognizer.ini
```

Step 7: Setting Up the Hotword

Awesome work so far! We are getting really close now so hang in there.

We now need to change the trigger for our Google AIY project kit so that it responds to our voice when we speak the words '**OK Google**'.

Type the following command into the command window:

```
nano ~/.config/voice-recognizer.ini
```

This will produce this even scarier window.

Within this new window, look for the following code:

# Select the trigger: gpio (default), clap, ok-google.

# trigger = clap

We need to change this code to:

# Select the trigger: gpio (default), clap, ok-google.

trigger = ok-google

If you use the arrow keys on your keyboard you will notice a curser appears. Using the arrow keys, bring the curser down to the line of text that we are trying to change. Using the backspace key on your keyboard delete the line of text that reads '**# trigger = clap' then type 'trigger = ok-google'**.

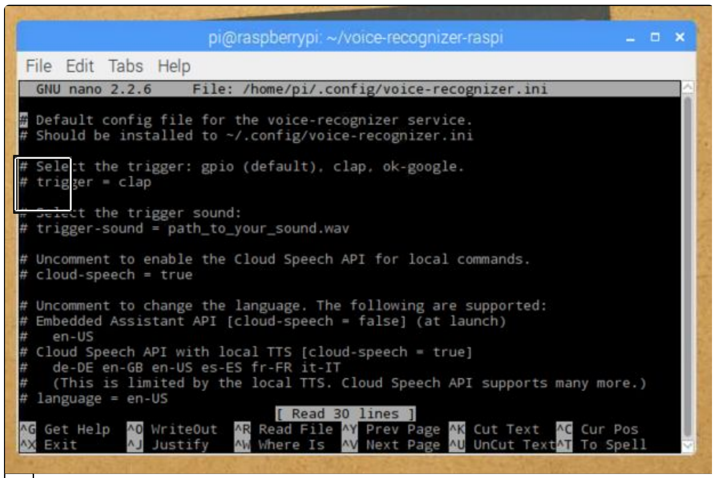
Notice that I have also removed the # symbol, it is important we do not include the # in this new line of text.

I have attached a before and after screenshot of what this should all look like (encase I lost you there).

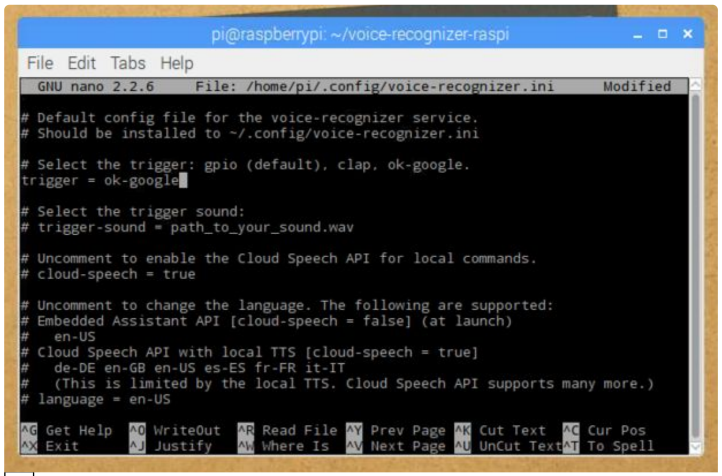
Assuming your window looks exactly like mine we can close and save the changes. Hold '**Ctrl**' on your keyboard and press '**X**' to close the window. We will then be prompted to save the changes we made, press '**Y**' and then hit '**Enter**' on your keyboard. The window will now close and the changes have been saved.

To ensure the changes have taken affect we need to restart the service. Type the following command into the terminal window and hit '**Enter**':

```
sudo systemctl restart voice-recognizer.service
```



```
  GNU nano 2.2.6      File: /home/pi/.config/voice-recognizer.ini

# Default config file for the voice-recognizer service.
# Should be installed to ~/.config/voice-recognizer.ini

# Select the trigger: gpio (default), clap, ok-google.
# trigger = clap

# Select the trigger sound:
# trigger-sound = path_to_your_sound.wav

# Uncomment to enable the Cloud Speech API for local commands.
# cloud-speech = true

# Uncomment to change the language. The following are supported:
# Embedded Assistant API [cloud-speech = false] (at launch)
#   en-US
# Cloud Speech API with local TTS [cloud-speech = true]
#   de-DE en-GB en-US es-ES fr-FR it-IT
#   (This is limited by the local TTS. Cloud Speech API supports many more.)
# language = en-US
                           [ Read 30 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```



```
  GNU nano 2.2.6      File: /home/pi/.config/voice-recognizer.ini        Modified

# Default config file for the voice-recognizer service.
# Should be installed to ~/.config/voice-recognizer.ini

# Select the trigger: gpio (default), clap, ok-google.
trigger = ok-google

# Select the trigger sound:
# trigger-sound = path_to_your_sound.wav

# Uncomment to enable the Cloud Speech API for local commands.
# cloud-speech = true

# Uncomment to change the language. The following are supported:
# Embedded Assistant API [cloud-speech = false] (at launch)
#   en-US
# Cloud Speech API with local TTS [cloud-speech = true]
#   de-DE en-GB en-US es-ES fr-FR it-IT
#   (This is limited by the local TTS. Cloud Speech API supports many more.)
# language = en-US

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

1. This is the line of text that we need to change: # trigger = clap

1. undefined
2. undefined
3. undefined
4. undefined
5. undefined
6. undefined
7. undefined
8. undefined
9. undefined
10. undefined
11. undefined
12. undefined
13. undefined
14. undefined
15. undefined
16. undefined
17. undefined
18. undefined
19. undefined
20. undefined
21. undefined
22. undefined
23. undefined
24. undefined
25. undefined
26. undefined
27. undefined
28. undefined
29. undefined
30. undefined
31. undefined

Step 8: Audio Configuration (Part 1)

Right now Google Assistant more or less alive and ready to serve.. Congratulations!

However, before you get too excited, you cannot hear each other. That’s because the Google AIY Project Image was configured to work with the hardware that was shipped with the kit. Since we are using a standard aux speaker and usb microphone we need to tweak some of the configuration.

Once again we will use the same command window, this time type:

sudo leafpad /boot/config.txt

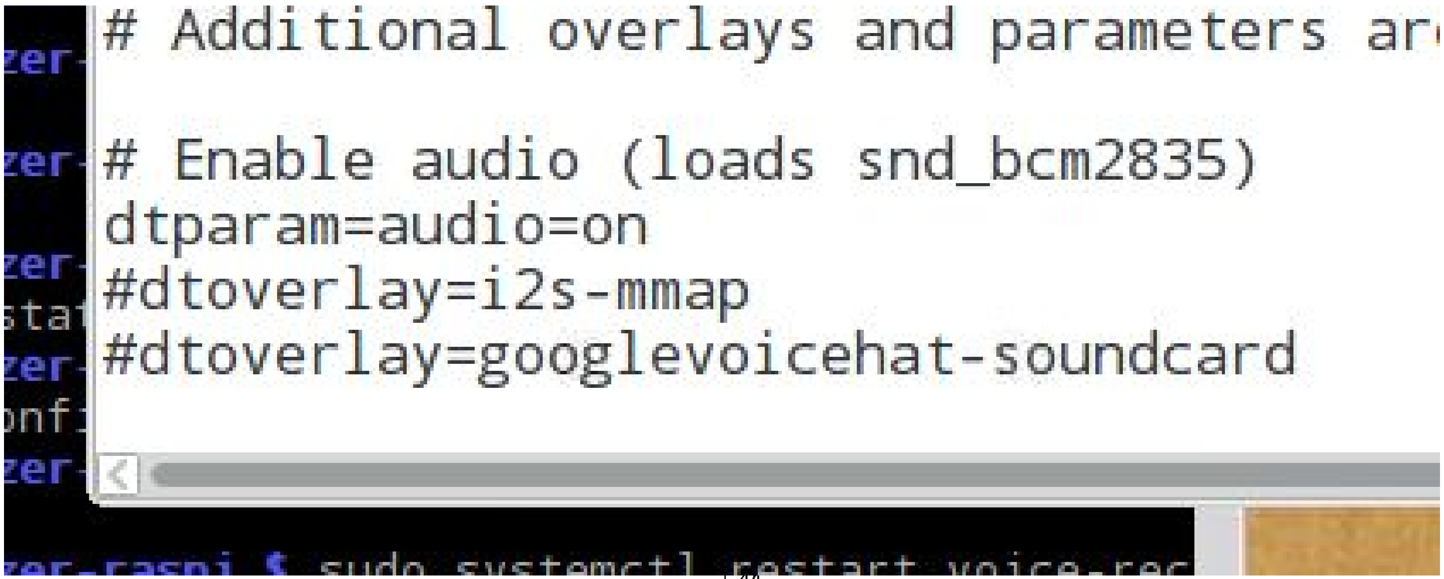
This will open a text window. Scroll to the very bottom of the document and remove the # in front the the line dtparam=audio=on and insert a # in front of the two lines below it.

After you have made these changes it should exactly like this:

# Enable audio (loads snd_bcm2835)
dtparam=audio=on
#dtoverlay=i2s-mmap
#dtoverlay=googlevoicehat-soundcard

I have also attached a screenshot to show you what this will look like.

Go to ‘**File**’ then click ‘**Save**. You can now close the document.



```
# Additional overlays and parameters ar

# Enable audio (loads snd_bcm2835)
dtparam=audio=on
#dtoverlay=i2s-mmap
#dtoverlay=googlevoicehat-soundcard
```

```
sudo systemctl restart voice-rec
```

44. undefined
45. undefined
46. undefined
47. undefined
48. undefined
49. undefined
50. undefined
51. undefined
52. undefined
53. undefined
54. undefined
55. undefined
56. undefined
57. undefined
58. undefined
59. undefined
60. undefined
61. undefined
62. undefined
63. undefined
64. undefined
65. undefined
66. undefined
67. undefined
68. undefined
69. undefined
70. undefined
71. undefined
72. undefined
73. undefined

Step 9: Audio Configuration (Part 2)

Back in the command window again, type:

```
sudo leafpad /etc/asound.conf
```

When you hit ‘**Enter**’ a new text document will open. This time delete ALL the text within the document and replace it with the following:

```
pcm.!default {
type asym
capture.pcm "mic"
playback.pcm "speaker"
}
pcm.mic {
type plug
slave {
pcm "hw:1,0"
}
}
pcm.speaker {
type plug
slave {
pcm "hw:0,0"
}
}
```

Again I have attached a screenshot showing you what this will look like.

Once again save and close the document.

Now it is time to reboot your Raspberry Pi. Click on the Raspberry Pi logo at the top left of your screen and click on ‘**Shutdown**’ then ‘**Reboot**’.

After you have rebooted the Pi we have just one more tweak to make. Double click on the ‘**Start dev terminal**’ icon once again and type the following:

```
leafpad /home/pi/voice-recognizer-raspi/checkpoints/check_audio.py
```

In this final document you need to locate the line of code that reads:

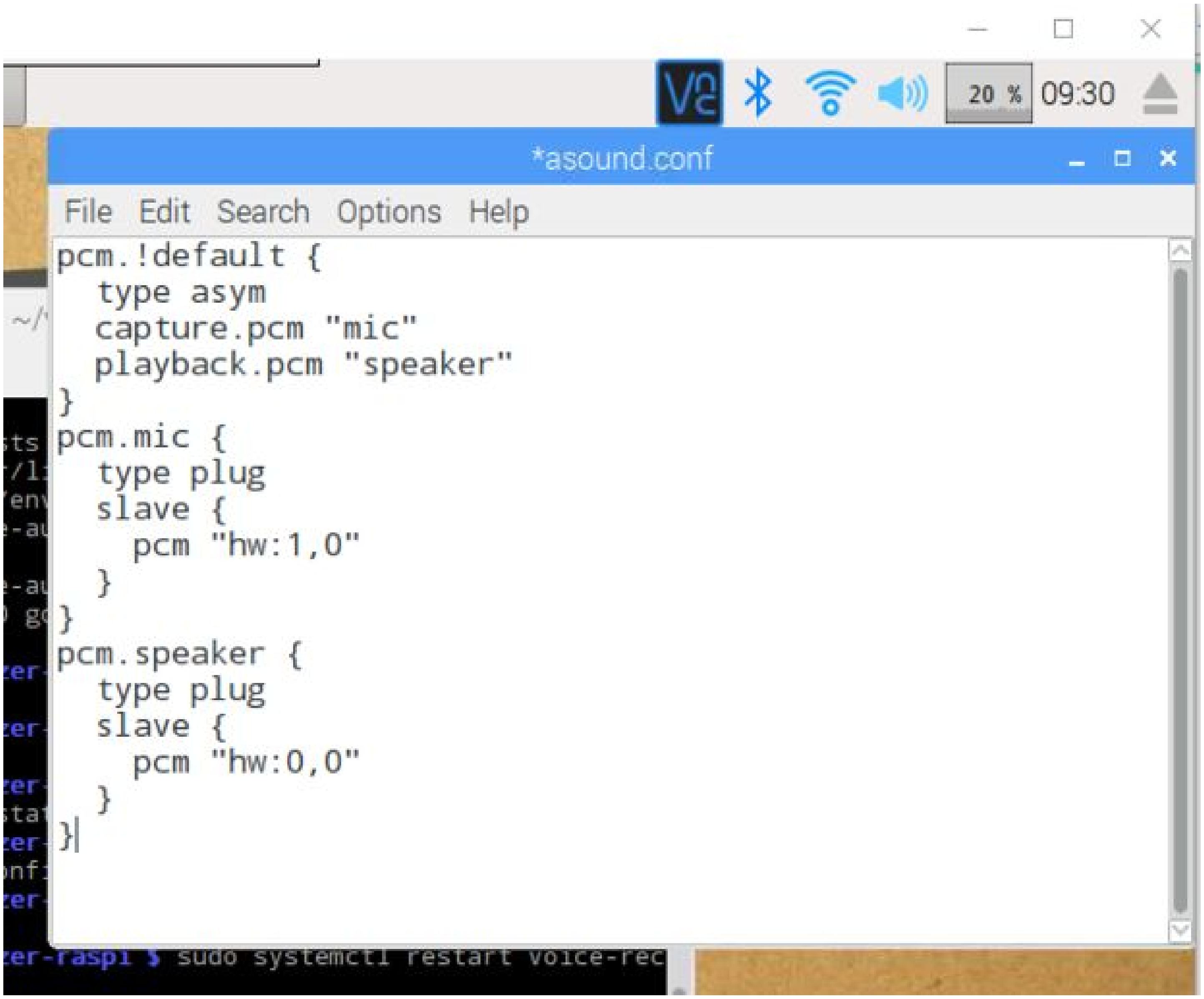
```
VOICEHAT_ID = ‘googlevoicehat’
```

Change this to:

```
VOICEHAT_ID = ‘bcm2835’
```

Once you have made these changes, just as we did before, save then close this document.

74. undefined
75. undefined
76. undefined
77. undefined
78. undefined
79. undefined
80. undefined
81. undefined
82. undefined
83. undefined
84. undefined
85. undefined
86. undefined
87. undefined
88. undefined
89. undefined
90. undefined
91. undefined
92. undefined
93. undefined
94. undefined
95. undefined
96. undefined
97. undefined
98. undefined
99. undefined
100. undefined
101. undefined
102. undefined
103. undefined
104. undefined
105. undefined
106. undefined
107. undefined
108. undefined
109. undefined
110. undefined
111. undefined
112. undefined
113. undefined
114. undefined
115.



```
pcm.!default {
  type asym
  capture.pcm "mic"
  playback.pcm "speaker"
}
pcm.mic {
  type plug
  slave {
    pcm "hw:1,0"
  }
}
pcm.speaker {
  type plug
  slave {
    pcm "hw:0,0"
  }
}
```

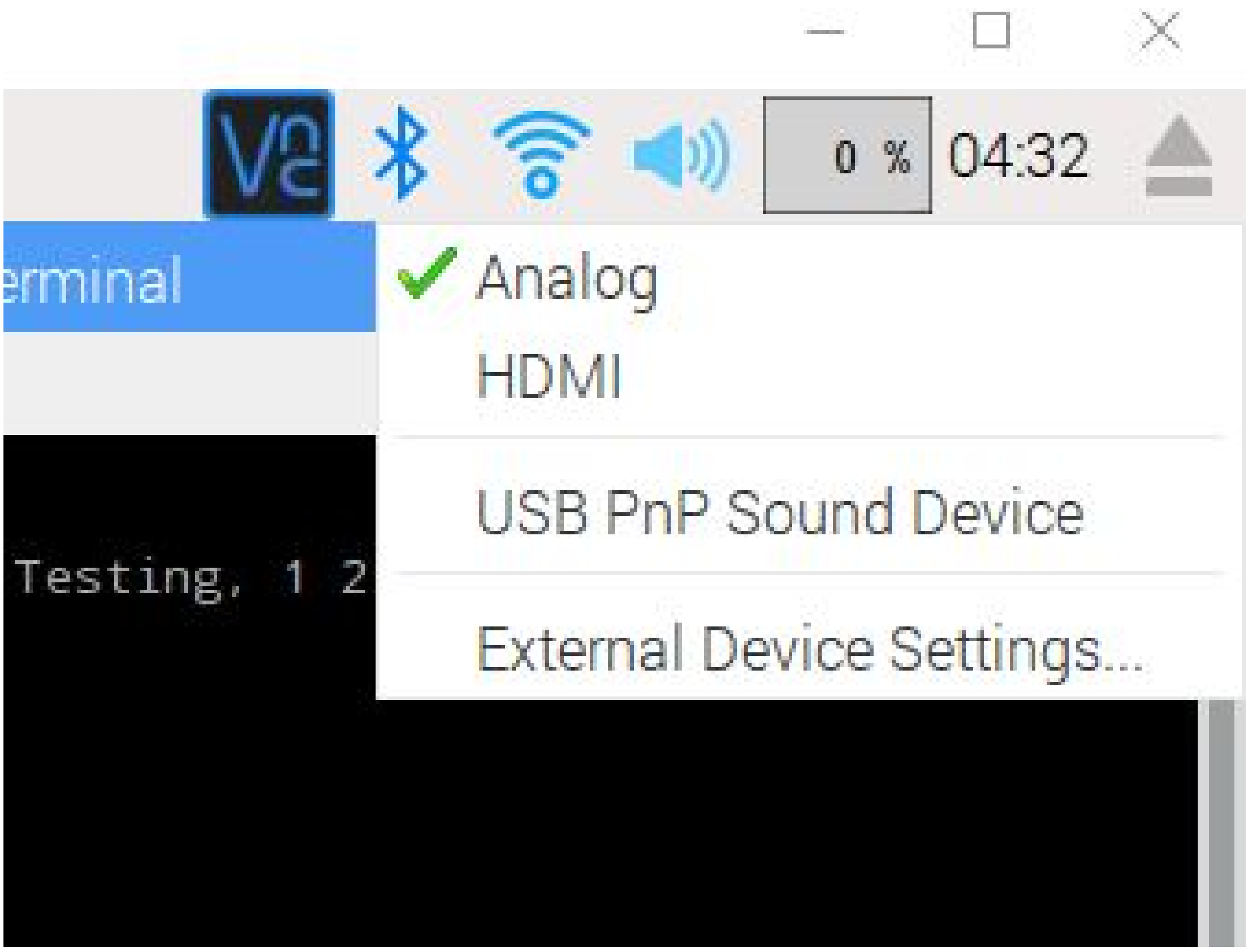
```
sudo systemctl restart voice-rec
```

Step 10: Testing the Audio

On the desktop there is a file called ‘**Check audio**’. Double click on this and follow the prompts to ensure that both the speak and microphone is working.

If you followed this Instructable correctly there should be no problems. However if you cannot hear anything, double check that the volume is turned up and that your Raspberry Pi is using ‘**Analog**’ for sound output. You can do this by right-clicking on the sound icon at the top of the screen. ‘**Analog**’ should be ticked just like the example in the screenshot.

Assuming you passed the audio check, we can move onto the next step.


0 %
04:32
erminal
Analog
HDMI
USB PnP Sound Device
External Device Settings...

```
Testing, 1 2
```

Step 11: Connecting to the Cloud

Before Google Assistant will give us answers to life's burning questions we need to connect her to Google's Cloud Services.

This is easy to do but if you haven't been in the cloud before then it may seem a bit daunting at first.

Here's what we need to do:

1) On the Raspberry Pi open up the Chrome internet browser and go to the Cloud Console: https://console.cloud.google.com/

2) Sign in with an existing Google account or sign up if you do not have one.

3) Create a new project and give it a name. I called mine '**Google Pi**'

4) Using the search bar start typing '**Google Assistant**' and you should see the '**Google Assistant API**'. Click on it and then when the next page loads click '**Enable**' to activate the API.

5) Go to '**API Manager**' then '**Credentials**' and create an '**OAuth 2.0 client**'.

6) Click '**Create credentials**' and select '**OAuth client ID**'. If you have never been in the cloud before then you will now be prompted to configure your consent screen. You'll need to name your app, I called mine '**Raspberry Pi**'. All other fields can be left blank.

7) In the Credentials list, find your new credentials and click the download icon on the right.

8) The chrome browser will now download a small JSON file with all your credentials stored safely inside. Find this file and rename it to '**assistant.json**' then move it to /home/pi/assistant.json.

9) Finally, go to the Activity Controls page: https://myaccount.google.com/activitycontrols and turn on the following services: Web and app activity, Location history, Device information, Voice and audio activity. Be sure to log in with the same Google account as before!

If you got stuck at any point during this stage, do not freak out, Google has done an excellent job at documenting this process with screenshots for each step over on the Google AIY Kit website (https://aiyprojects.withgoogle.com/voice/#users-guide-1-1--connect-to-google-cloud-platform).

Step 12: Final Testing

If everything was setup correctly in the cloud we are now ready to talk to Google. Using the '**Start dev terminal**' command window again, type the following:

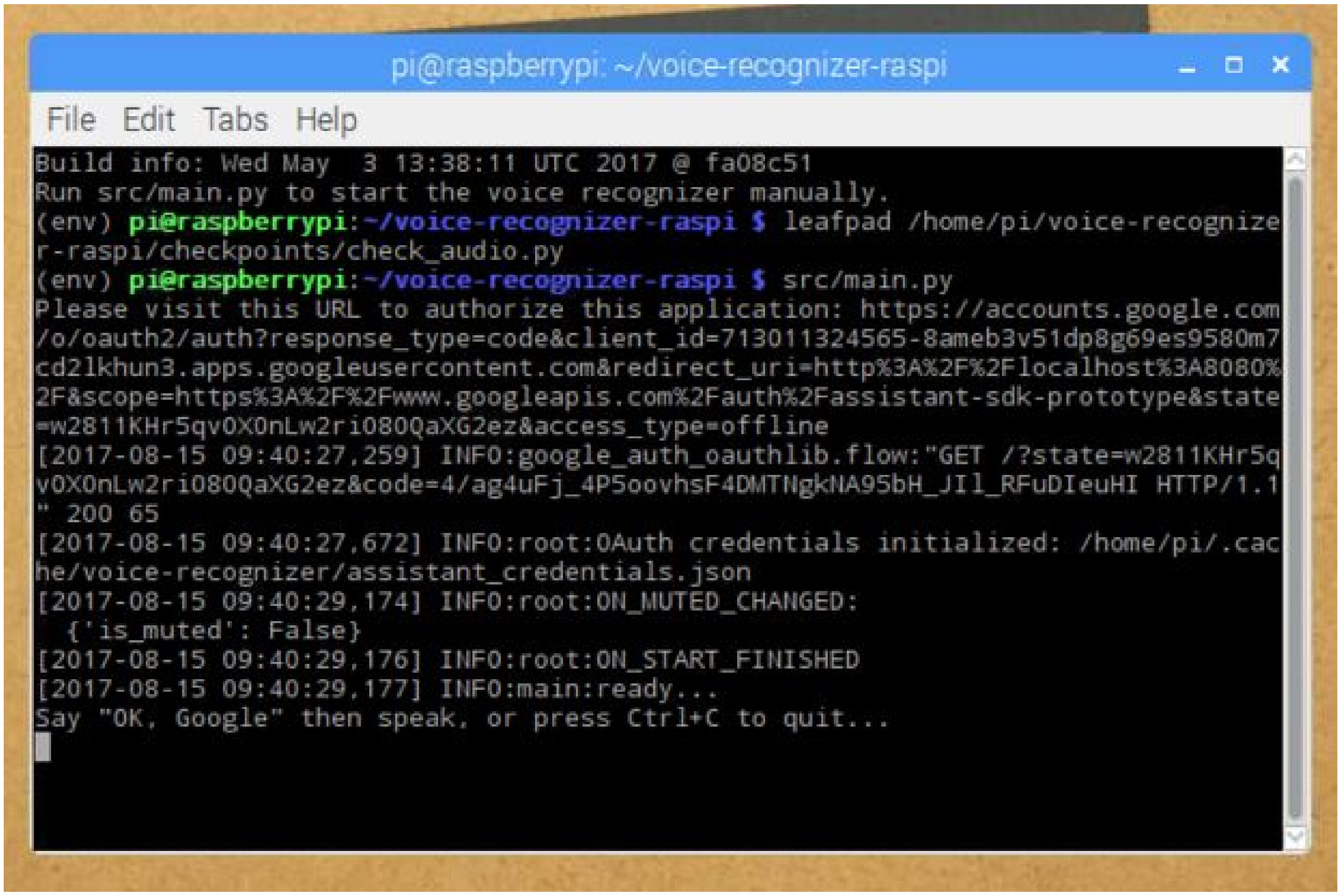
```
src/main.py
```

This will wake up our assistant but since this is the first time that we are connecting to Google's Services, a web browser will open and you will need to login to Google to give permission for the Raspberry Pi to access the Google Assistant API. Again making sure that you use the same Google account logins as you did before.

Once you have logged in successfully and granted permission you will be prompted to close the window. The command window will now look like the screenshot attached confirming that everything was setup correctly.

Go ahead, ask a question, she's listening!

Before you get too excited though, we are not quite finished. When you have finished playing, close the window, to do this just use the white cross at the top right of the window.



```
Build info: Wed May  3 13:38:11 UTC 2017 @ fa08c51
Run src/main.py to start the voice recognizer manually.
(env) pi@raspberrypi:~/voice-recognizer-raspi $ leafpad /home/pi/voice-recognize
r-raspi/checkpoints/check_audio.py
(env) pi@raspberrypi:~/voice-recognizer-raspi $ src/main.py
Please visit this URL to authorize this application: https://accounts.google.com
/o/oauth2/auth?response_type=code&client_id=713011324565-8ameb3v51dp8g69es9580m7
cd2lkhun3.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8080%
2F&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fassistant-sdk-prototype&state
=w2811KHr5qv0X0nLw2ri080QaXG2ez&access_type=offline
[2017-08-15 09:40:27,259] INFO:google_auth_oauthlib.flow:"GET /?state=w2811KHr5q
v0X0nLw2ri080QaXG2ez&code=4/ag4uFj_4P5oovhsF4DMTNgkNA95bH_JIl_RFuDIeuHI HTTP/1.1
" 200 65
[2017-08-15 09:40:27,672] INFO:root:OAuth credentials initialized: /home/pi/.cac
he/voice-recognizer/assistant_credentials.json
[2017-08-15 09:40:29,174] INFO:root:ON_MUTED_CHANGED:
  {'is_muted': False}
[2017-08-15 09:40:29,176] INFO:root:ON_START_FINISHED
[2017-08-15 09:40:29,177] INFO:main:ready...
Say "OK, Google" then speak, or press Ctrl+C to quit...
```

Step 13: Setting Up Google Assistant on Startup

I promised you that our Google Assistant would startup automatically when we power up the Raspberry Pi. To do this, open a fresh command window using the '**Start dev terminal**' icon on the desktop. Type the following line of code into your command window and hit '**Enter**' on your keyboard:

```
sudo systemctl enable voice-recognizer
```

We just configured auto startup of our Google Assistant with one line of code.. How easy was that!!

Step 14: The Finish Line

Now that you have completed all the steps go ahead and reboot your Raspberry Pi. If you have followed all these instructions carefully then Google Assistant should be running in the background when the Pi loads up. Give it a try, say OK Google to wake her and ask her anything you like!

I really hope you liked this Instructable. It is a result of 2 days hard work and lots of online reading. I am definitely not a coder so I have tried to find the easiest and most logical way to get a working Google Assistant onto a Raspberry Pi and I believe this is it.

If you have any questions or suggestions with regards to this Instructable please let me know in the comments below. I would also love to hear how yours turned out.

I read a lot of tutorials, blog posts and forum entries but most of my success with this project came from 2 posts online that were both trying to accomplish similar tasks: http://eduncan911.com/stem/embedded/aiy-google-assistant-upgrades-may-2017.html (http://eduncan911.com/stem/embedded/aiy-google-assistant-upgrades-may-2017.html) and http://www.androidauthority.com/build-google-assistant-raspberry-pi-770296/ (http://www.androidauthority.com/build-google-assistant-raspberry-pi-770296/). This Instructable is a result of figuring out how to put the two together in an easy and straight forward Instructable!