Raspberry Pi Google AI Assistant Using Push-to-talk Feature

### Course Name:

### EE-629A Internet of Things

### Professor:

### Prof. Kevin Lu

### Project By:

### Rida Zainab

### Raveena Mehta

### Monami Mukhopadhyay

## 

### Abstract:

The main aim of this project is to build a Raspberry Pi voice-controlled AI Assistant. It is the age of AI, and we all are swarmed away with the number of mesmerizing open-source AI projects getting in the market every day. They are one of the driving factors for business growth.

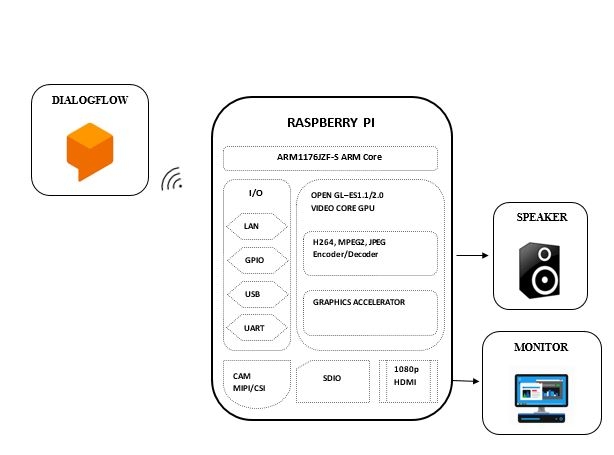
Google provides the API for using its voice service, which is open source. Using Google voice service, we can play music, get information about weather, book tickets and many more. We just need to ask. In this project, we build and configure our own Raspberry Pi-powered Google Assistant.

### Pre-requisites:

Our approach requires the below equipment list:

|  |  |
| --- | --- |
| * Raspberry Pi 3 Model | C:\Users\Monami\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\3E47A551.tmp |
| * Micro SD card with Raspbian installed (or any other Debian-based distribution) | C:\Users\Monami\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\AFA4872D.tmp |
| * USB microphone | C:\Users\Monami\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B122CEC3.tmp |
| * Speaker with 3.5mm connector | Image result for speaker with 3.5mm jack |

The architecture is better shown on the diagram below:

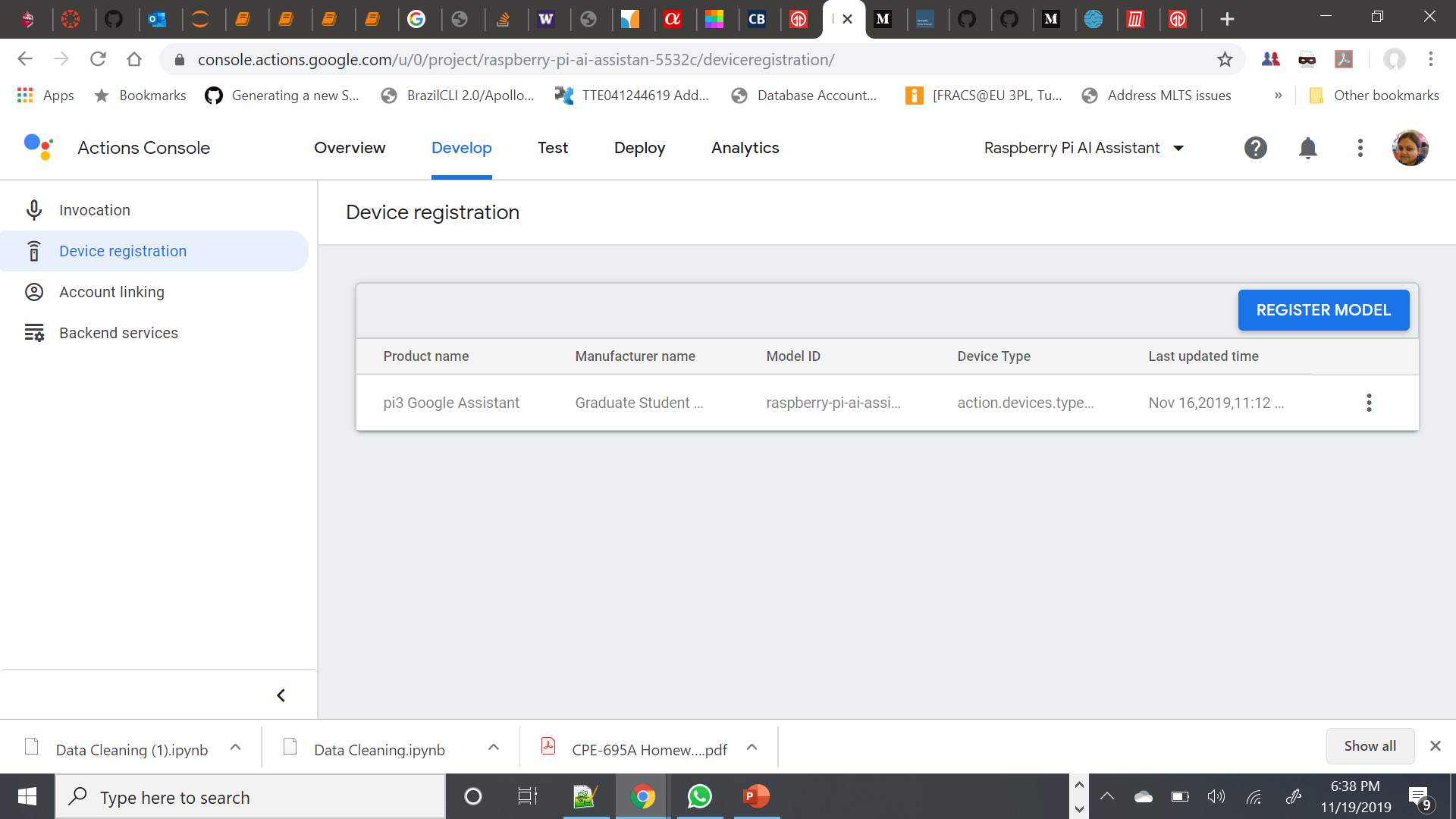


Block Diagram of the Raspberry Pi Google AI Assistant

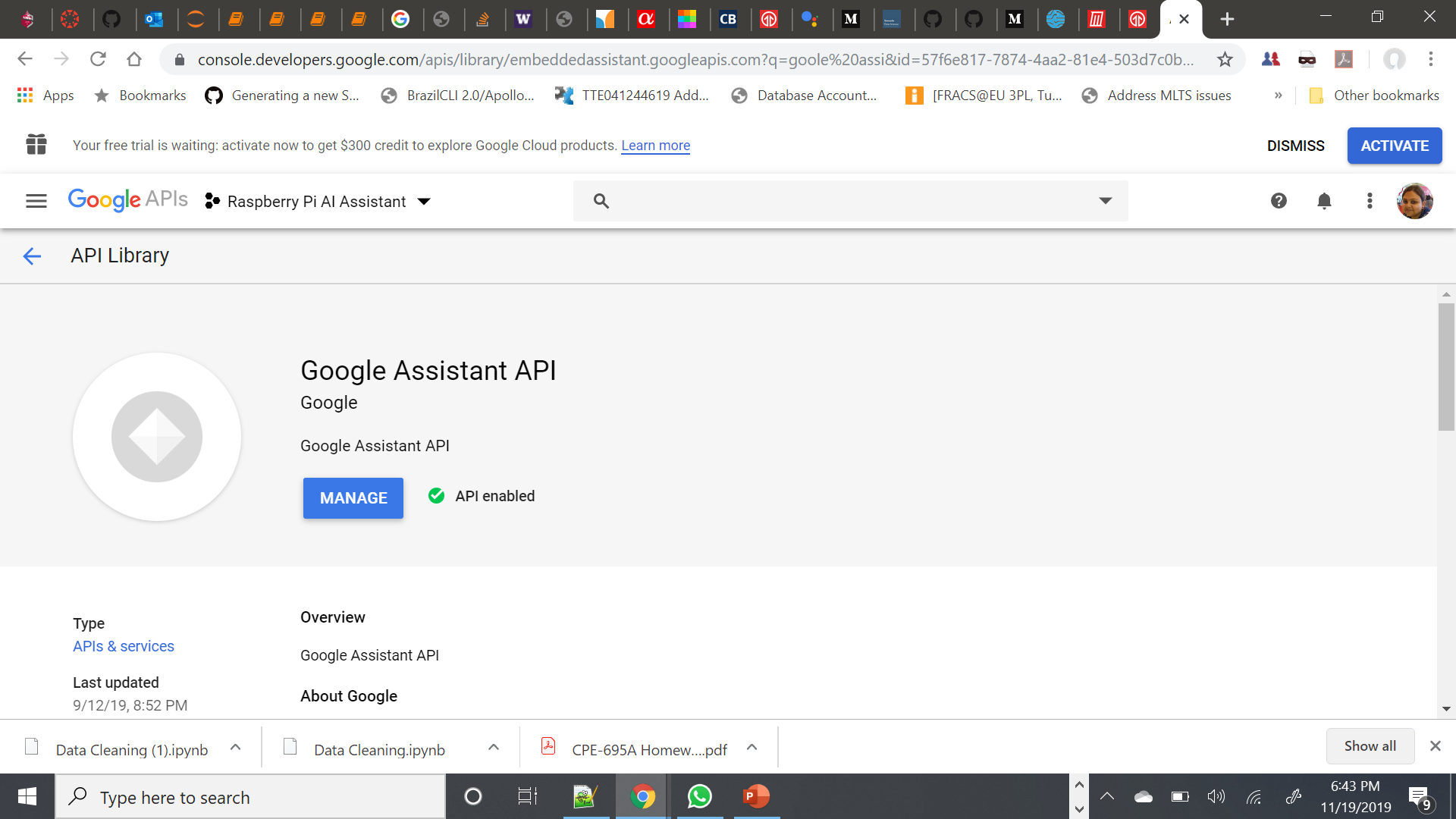
### Steps (Configuration & Setup):

### Registering for the Google API

We need to go to <https://console.actions.google.com> to register and set up project.



Next, we need to go to <http://console.developers.google.com> enable the Google Embedded Assistant API.



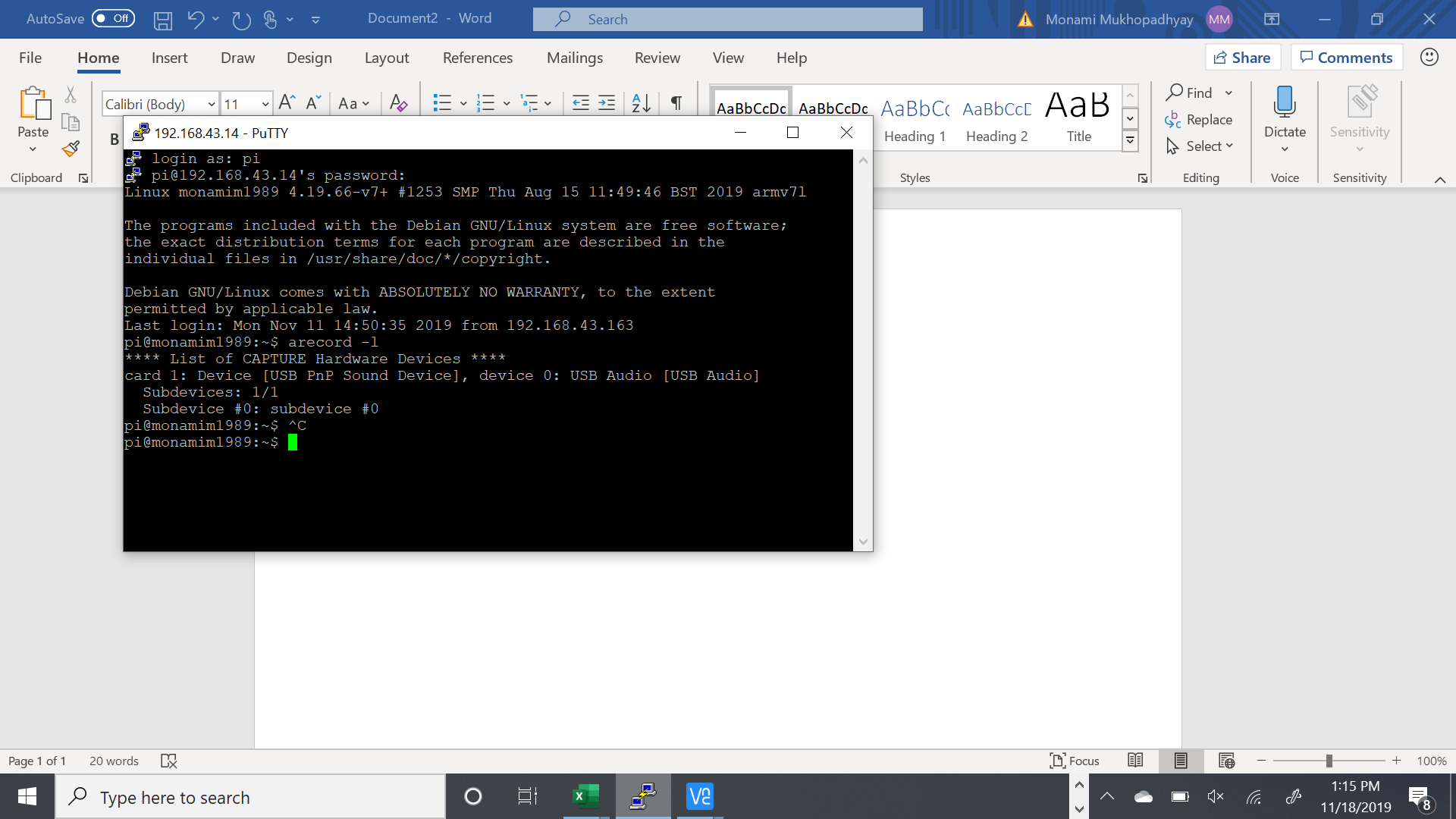
We also need to do Raspberry Pi device registration on Google platform. Once we have registered the model, we see the “Download credentials” screen. This screen provides a credentials file which we need for our Raspberry Pi 3 based Google Assistant to talk with the server. We need to save this credentials json file.

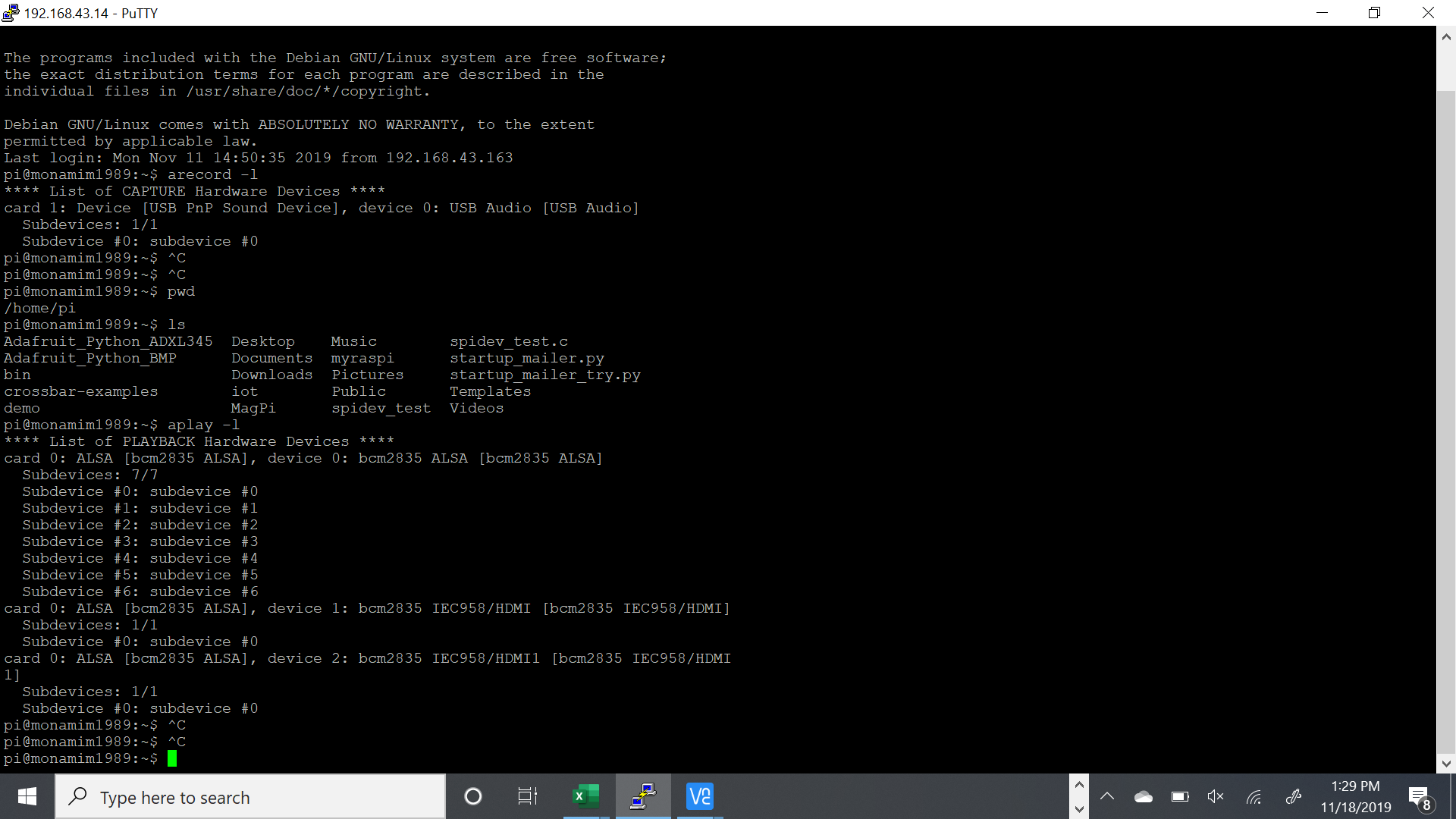
We also need to configure the OAuth consent screen, without this Google will not let us authorize our Raspberry Pi Google Assistant device.

### Setting up Audio for Google Assistant

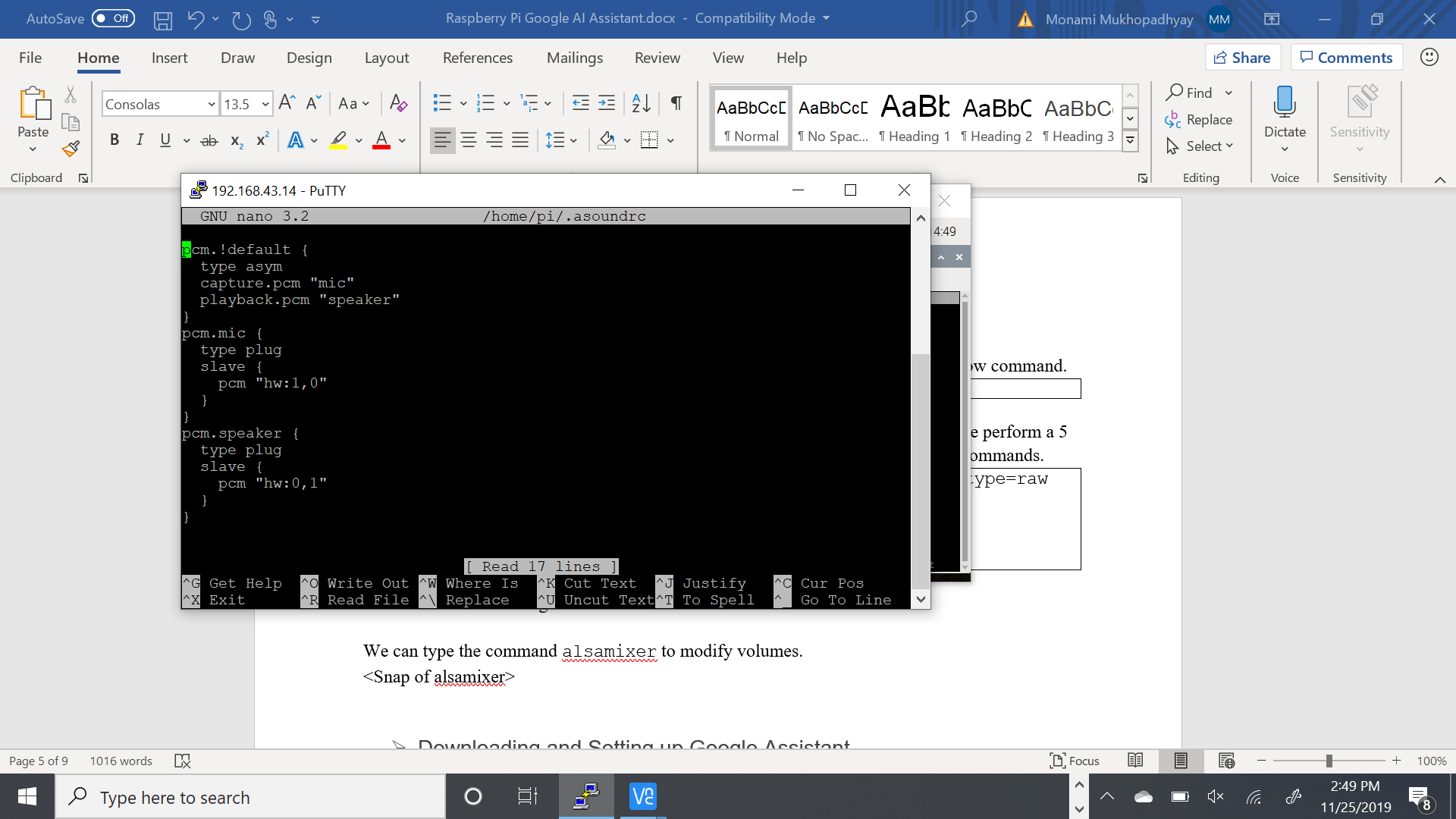
We note down the device and card number for both USB Microphone and Speaker using below command.

|  |
| --- |
| **arecord -l aplay -l** |





Using this information, we need to edit a file called asoundrc.



Once, the configuration is done, we check if speaker is working properly using below command.

|  |
| --- |
| **speaker-test -t wav** |

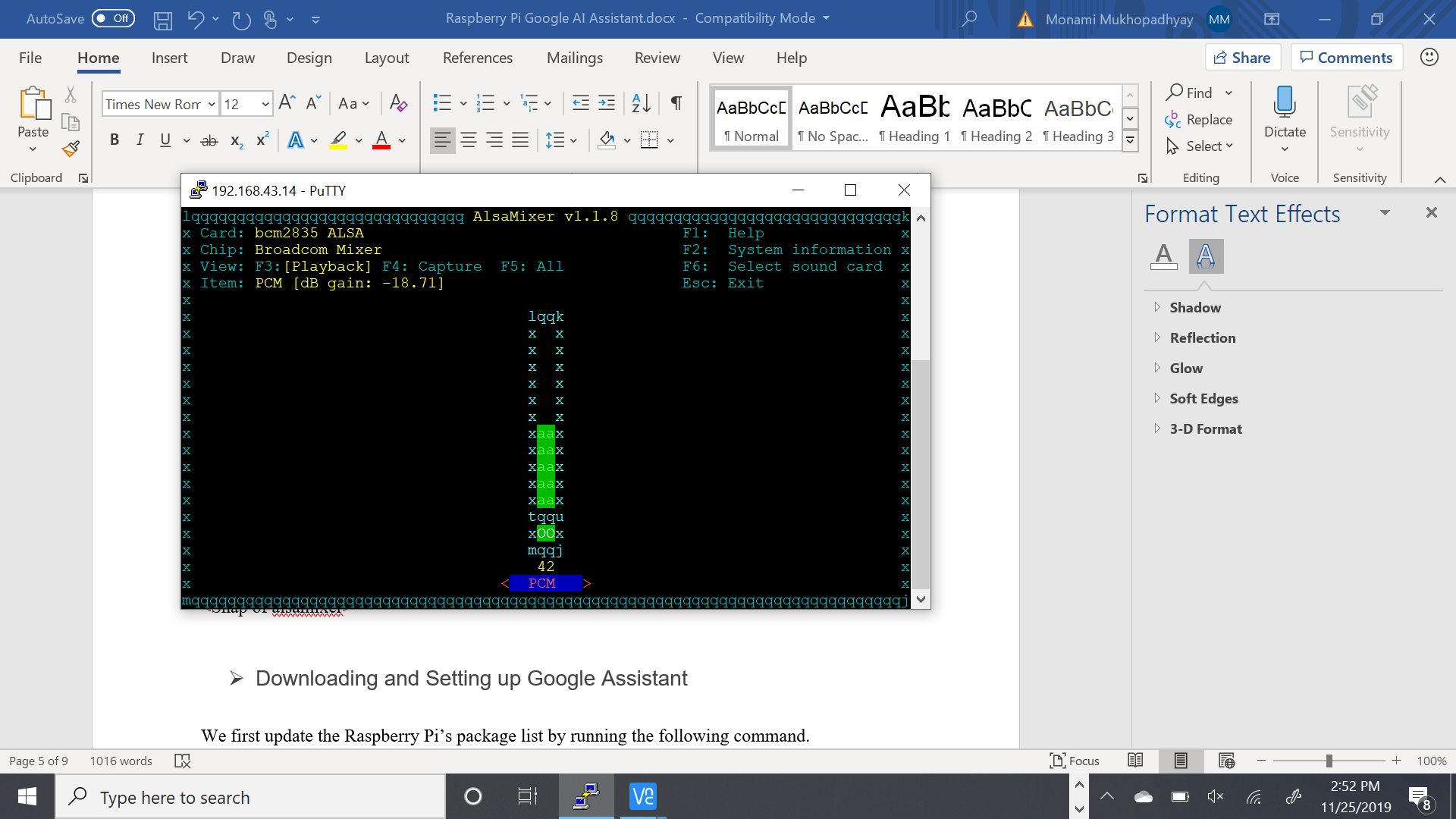


Now, we can check if both speaker and microphone work together well. For this, we perform a 5 sec voice recording, and then playback the recording via speaker. We need below commands.

|  |
| --- |
| **arecord --format=S16\_LE --duration=5 --rate=16000 --file-type=raw out.raw**  **aplay --format=S16\_LE --rate=16000 out.raw** |



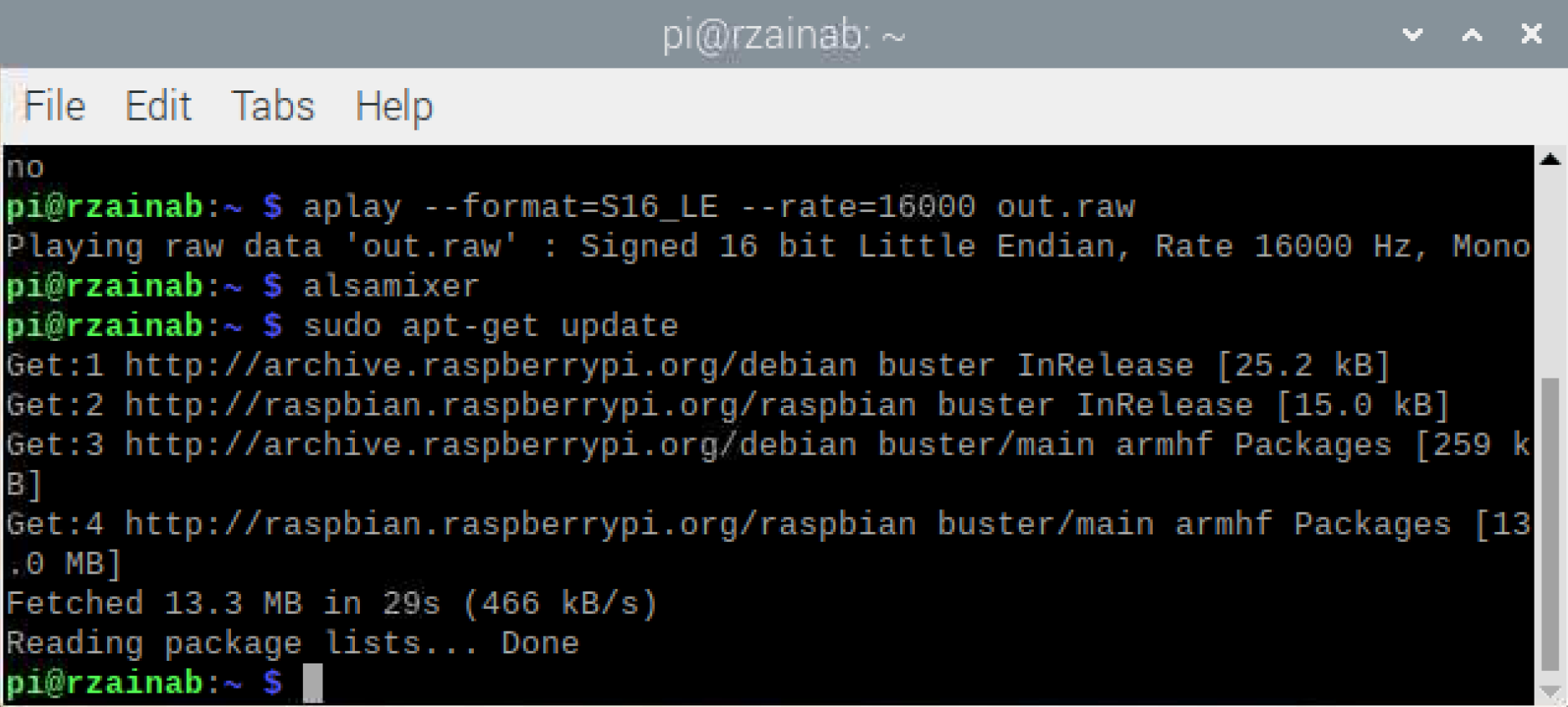
We can type the command alsamixer to modify volumes.



### Downloading and Setting up Google Assistant

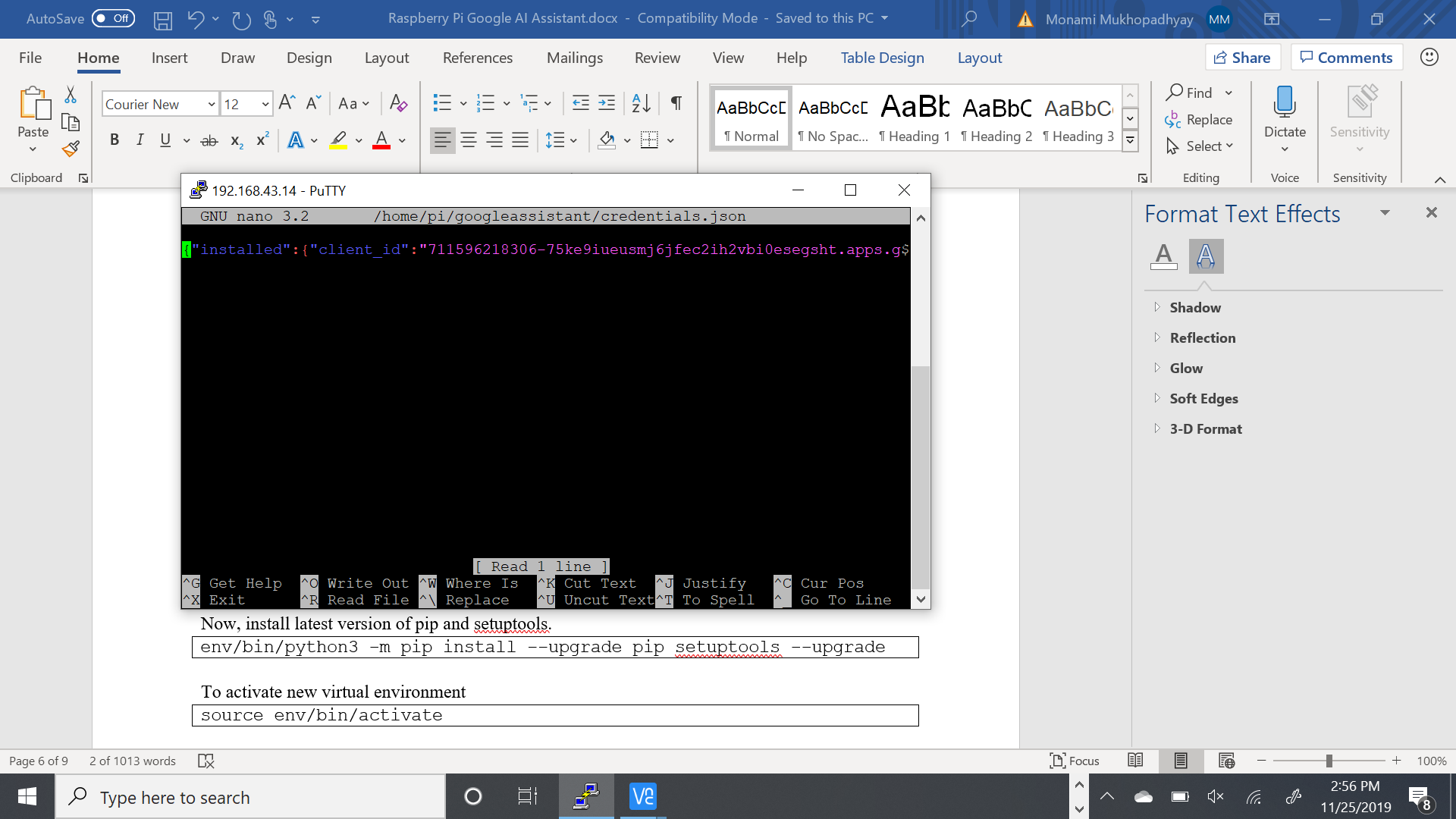
We first update the Raspberry Pi’s package list by running the following command.

|  |
| --- |
| **sudo apt-get update** |



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

|  |
| --- |
| **mkdir ~/googleassistant**  **nano ~/googleassistant/credentials.json** |



Next, we 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 libssl-dev libffi-dev libportaudio2** |

We can now enable Python3 on our virtual environment variable using this command.

|  |
| --- |
| **python3 -m venv env** |

Now, install latest version of pip and setuptools.

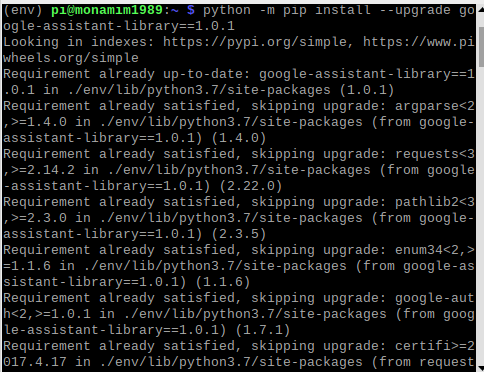
|  |
| --- |
| **env/bin/python3 -m pip install --upgrade pip setuptools --upgrade** |

To activate new virtual environment

|  |
| --- |
| **source env/bin/activate** |

Now we can install the Google Assistant Library for Python

|  |
| --- |
| **python3 -m pip install --upgrade google-assistant-library**  **python3 -m pip install --upgrade google-assistant-sdk[samples]** |



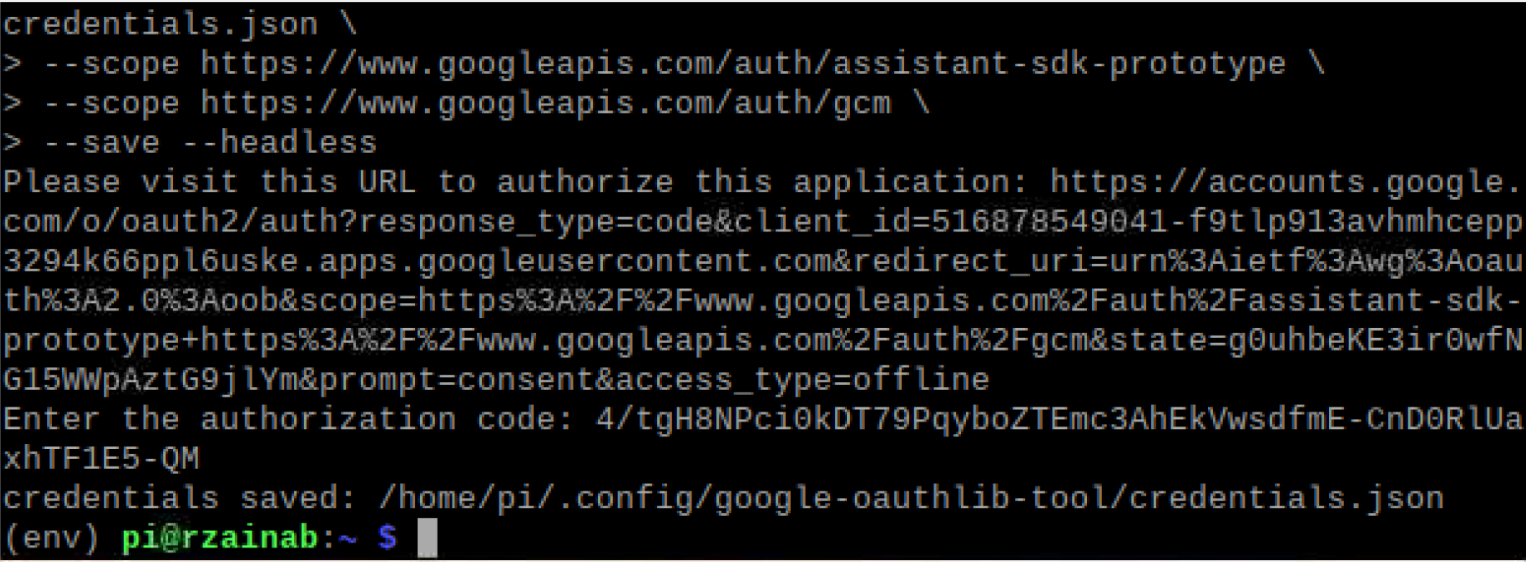
### Authorizing the Pi for Google Assistant

For this, we need to install the authorization tool so that we can authorize the Google Assistant API.

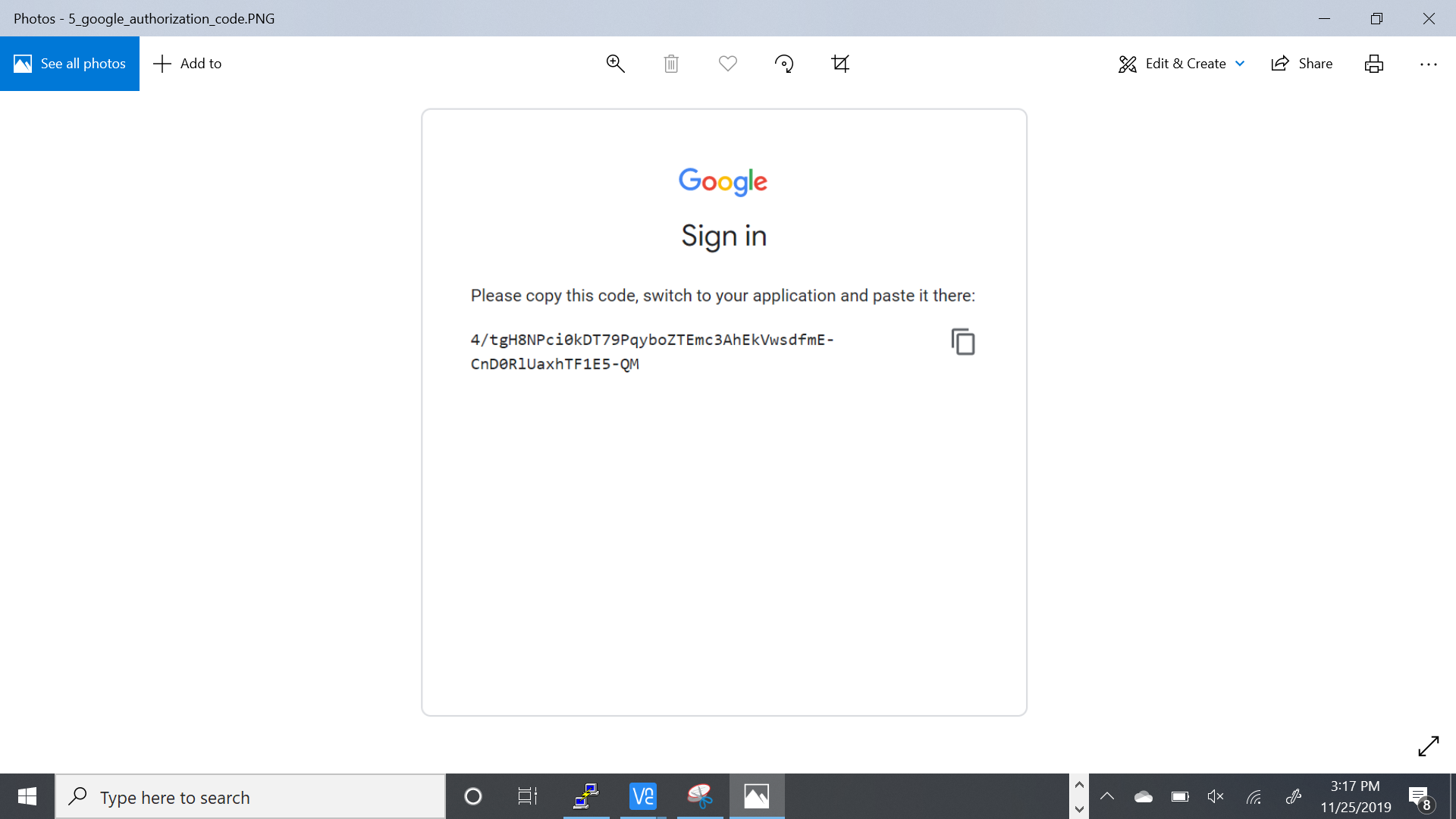
|  |
| --- |
| **python -m pip install --upgrade google-auth-oauthlib[tool]** |

Once the authentication library is installed, we need to run following command on Raspberry Pi. This step is very crucial.

|  |
| --- |
| **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** |



This command will generate a URL. We need to copy this URL and open in web browser. From here, copy the authentication code and paste on the Pi terminal and press enter.



If authentication is accepted, we can see something like this.

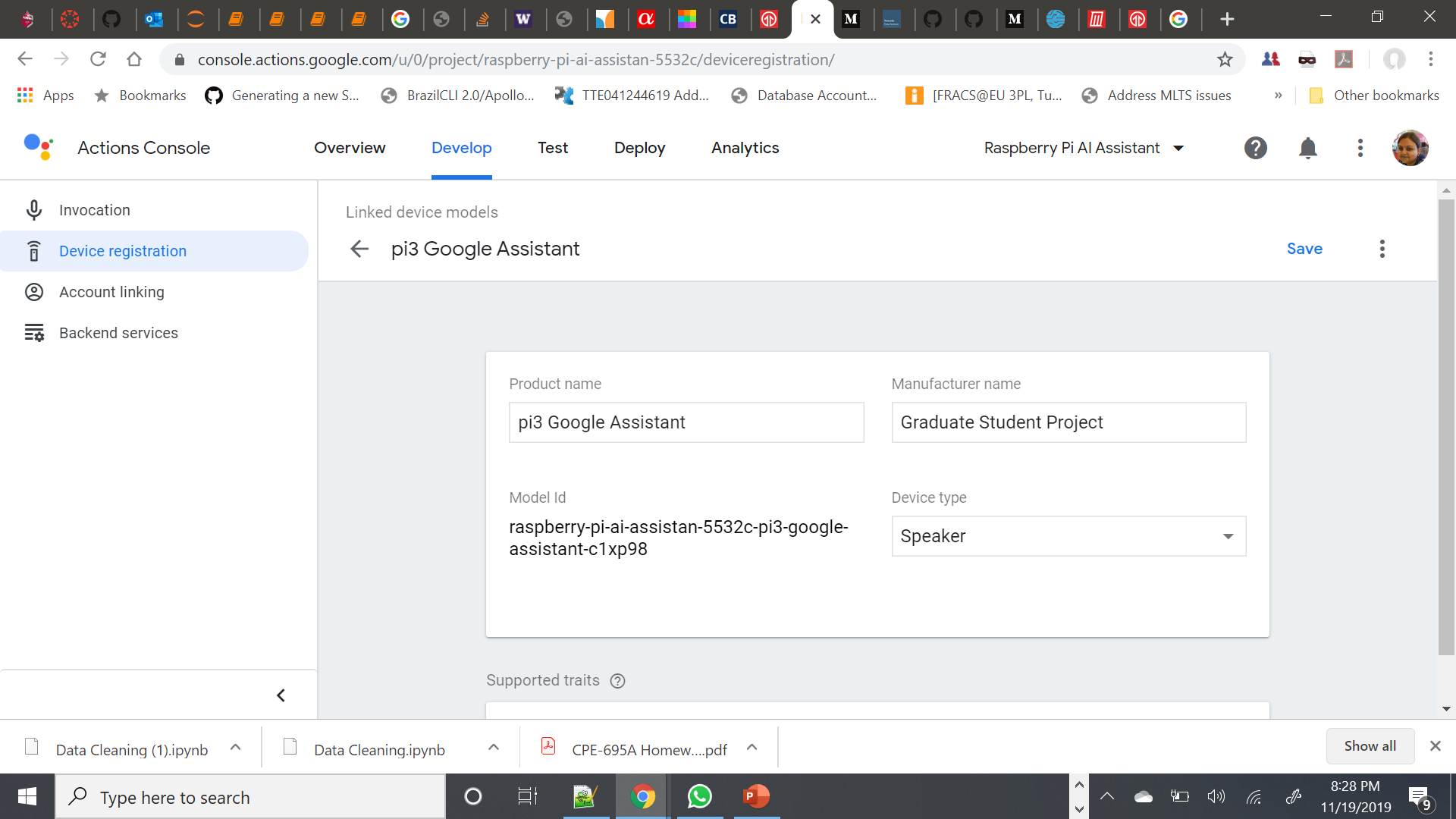
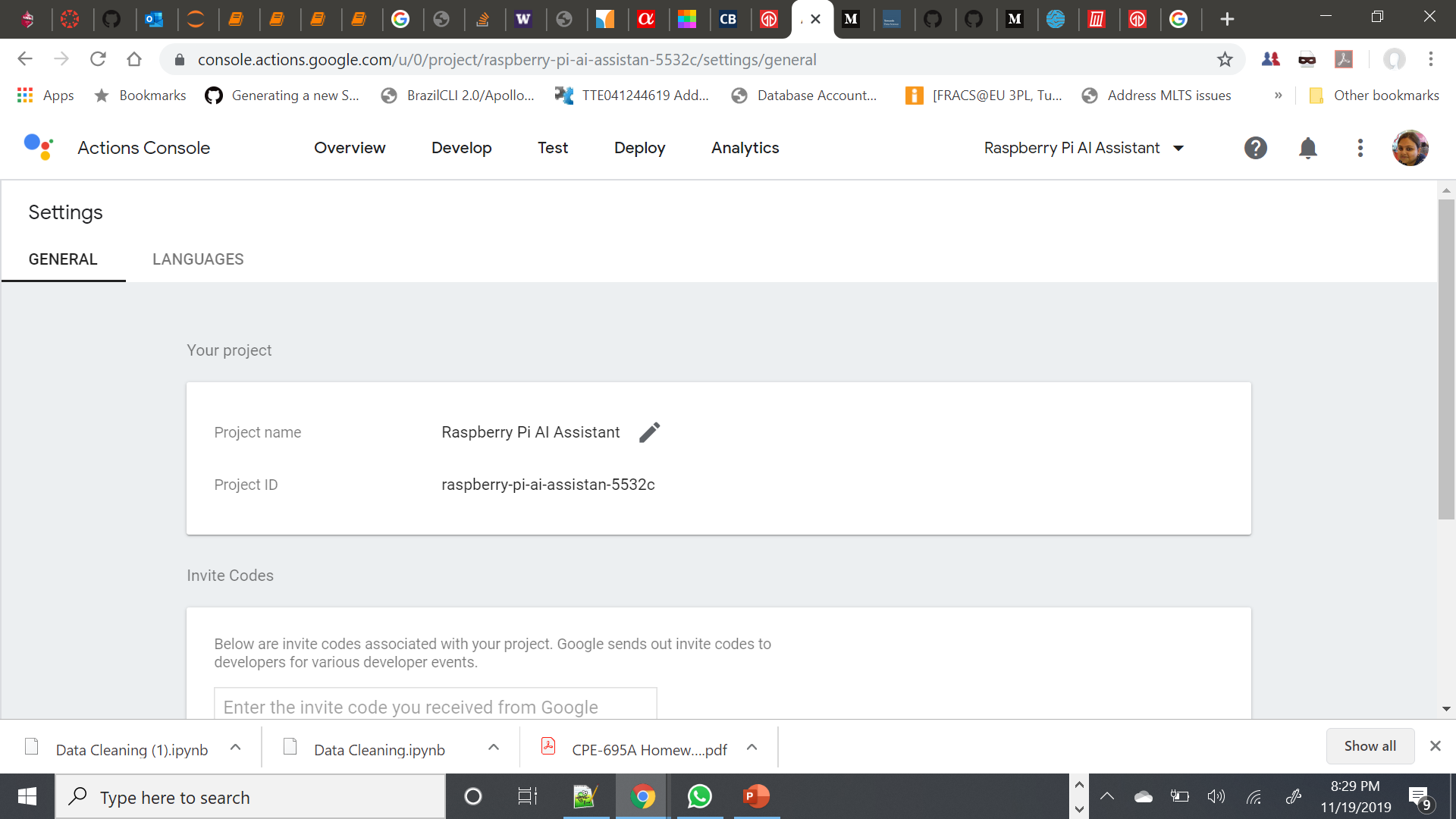
**“credentials saved: /home/pi/.config/google-oauthlib-tool/credentials.json”**

Now, we need to obtain and note down our project id and device-model-id. We can check project id from ‘Project Settings’ inside <https://console.actions.google.com> and we can get device-model-id from ‘Device Registration’ inside <https://console.developers.google.com>

Our project details are as below:

Project id - raspberry-pi-ai-assistan-5532c

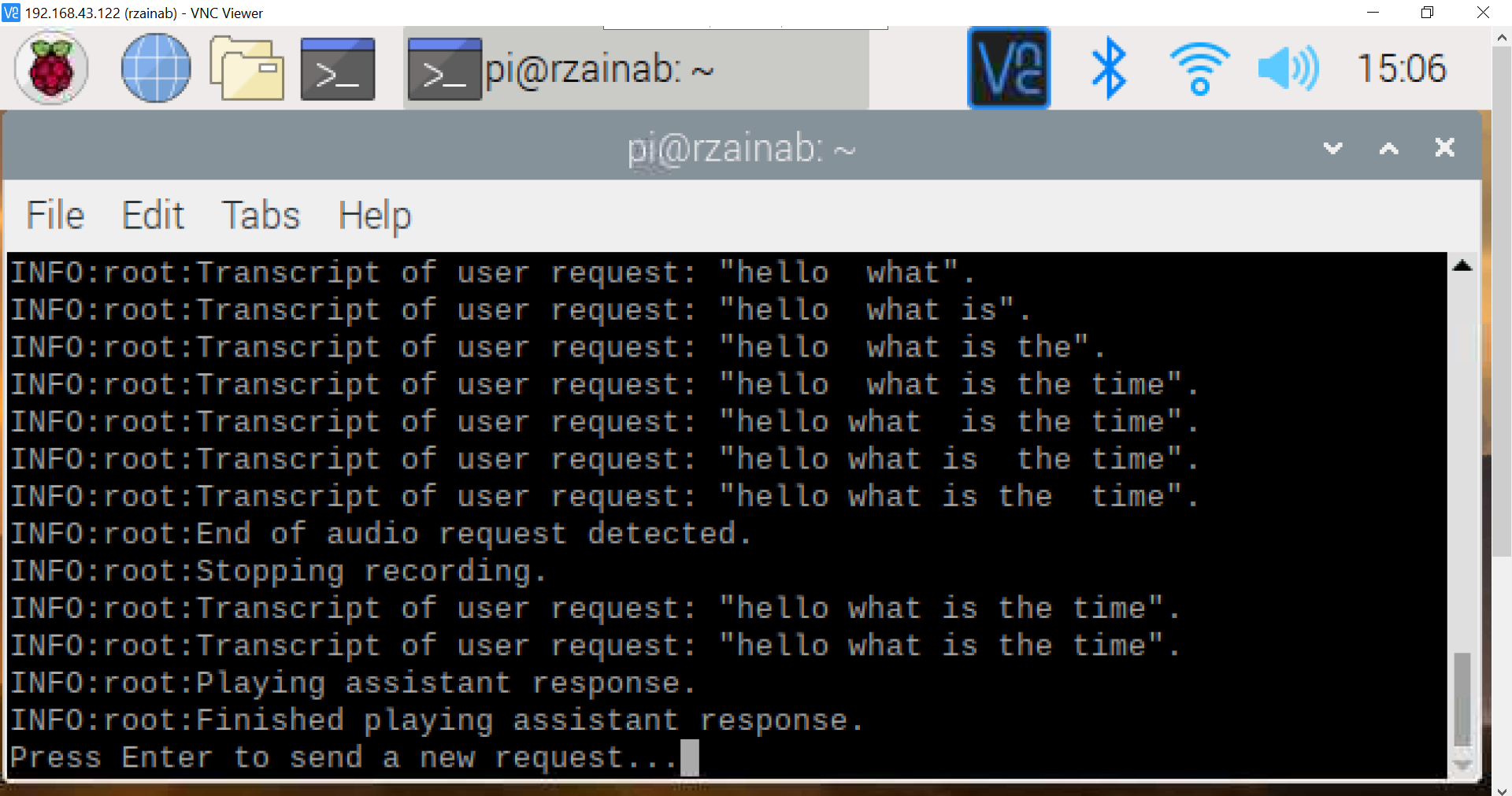
Device-id - raspberry-pi-ai-assistan-5532c-pi3-google-assistant-c1xp98



Using these data, we can type the following command to talk to Google Assistant.

|  |
| --- |
| **googlesamples-assistant-pushtotalk --project-id <projectid> --device-model-id <deviceid>** |

We press ENTER and speak any action. For example – “What is the time and date now?” We will receive a verbal response from assistant.



### Steps (Run Google Assistant on Raspberry Pi):

### Using Google Assistant on Raspberry Pi

This is the process which we can follow to run the Google Assistant on Pi after all the earlier mentioned configurations are completed.

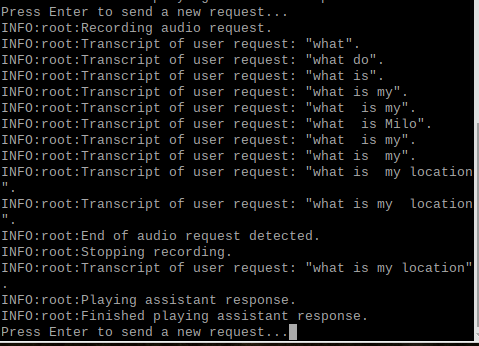
We can enter below command after starting a new terminal session. This is to activate the virtual environment.

|  |
| --- |
| **source env/bin/activate** |



To start up the push to talk sample, we will just need to run the following command.

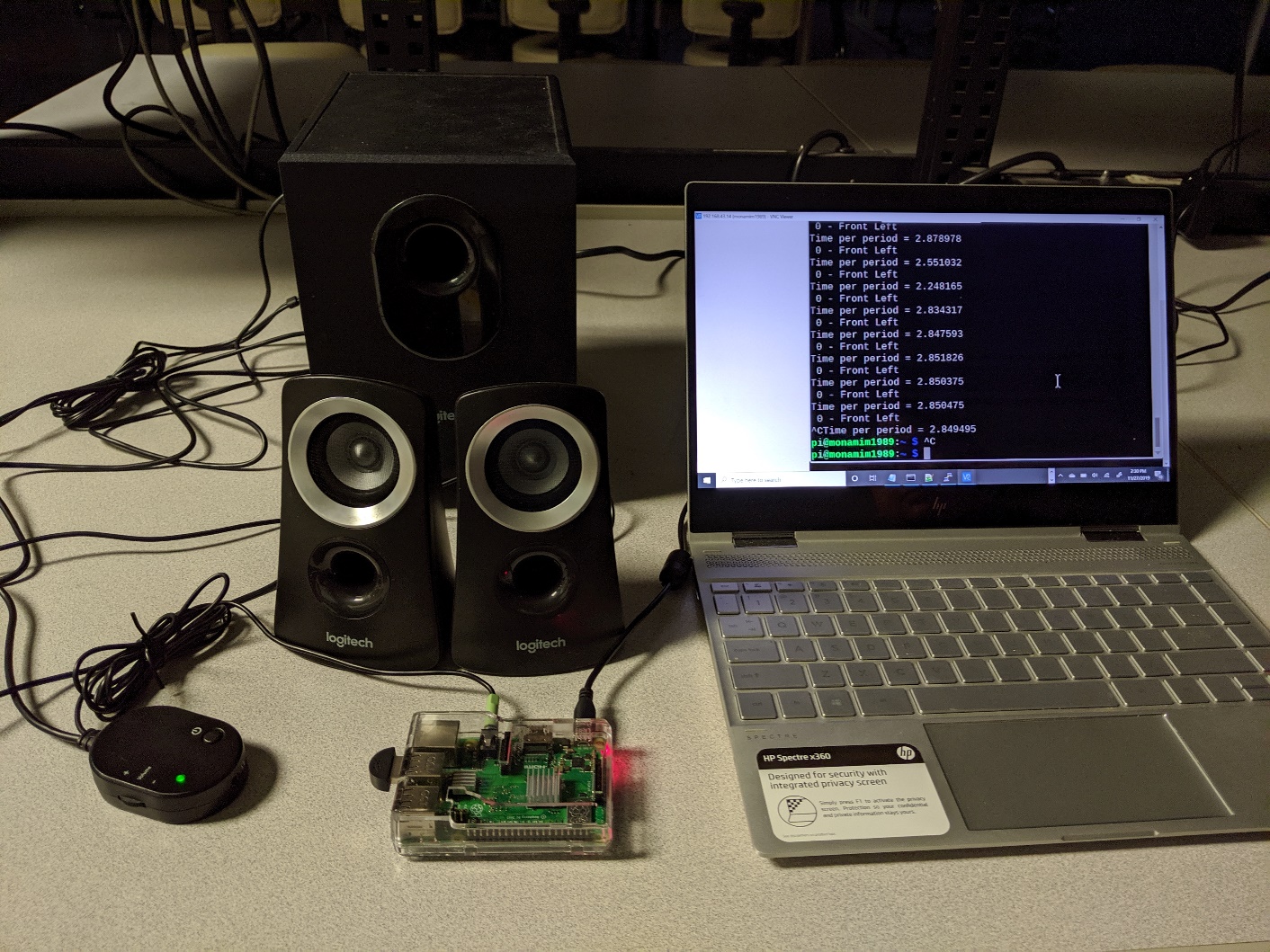
|  |
| --- |
| **googlesamples-assistant-pushtotalk** |



We press ENTER and ask any question. For example – “What is the time and date now?” We will receive a verbal response from assistant.

### Demo:

We show the complete demonstration of Google Assistant using Raspberry Pi in the below video. We ask couple of questions to the Assistant and it responds back correctly.



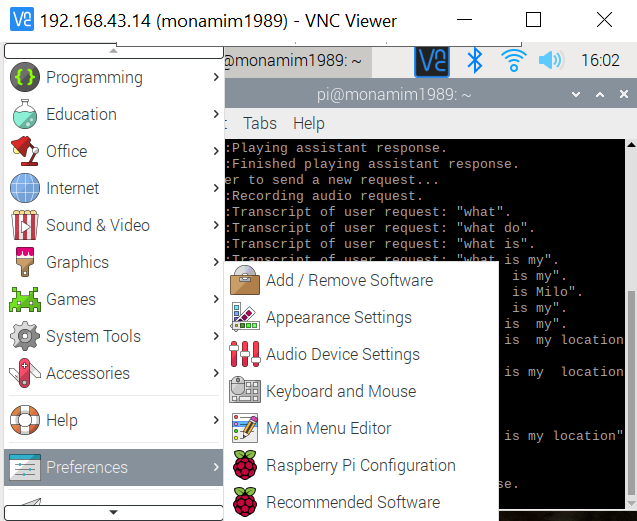
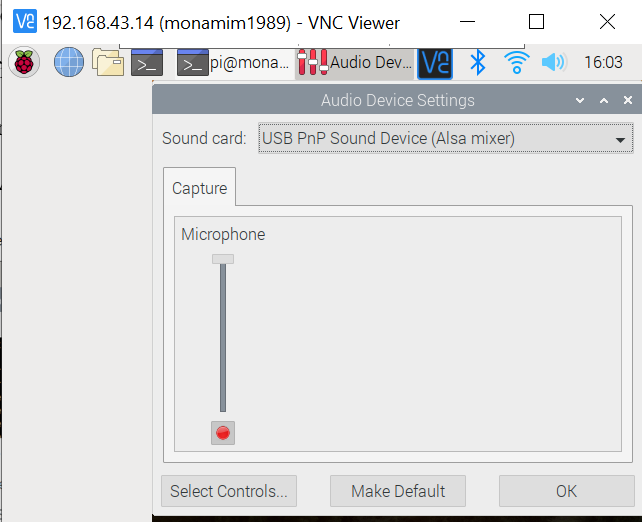


### Result

The Google AI Assistant that we build using Raspberry Pi functions as expected. It can successfully respond to all queries. It can understand the context of our query and react in an informed or smart way. This is important as it gives voice control a lot more power and moves it on from only reacting to specific phrases or commands.

### Issues Faced:

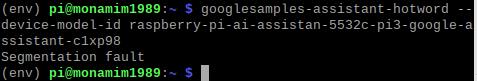
1. While working on this project, we initially faced the issue of configuration error for speaker and microphone using card number and device number. The microphone was not able to receive our voice. At first, we used command alsamixer to modify volumes. Even after that, the voice was not detected by microphone. Then we did further research and modified the Raspberry Pi Audio Device Settings. We increased the microphone capture to maximum value. This is how we resolved this issue.

1. Once, almost all the libraries were installed and credential settings were completed, we tried using the hotword ‘Ok Google’ voice-trigger feature of Google Assistant. The command used for this was:

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

But unfortunately, we received an error for this command called “Segmentation fault” as shown below-



After extensive research, we found that this error occurred because google-assistant-library was deprecated in 2019 because of which the Google Assistant Service no longer supports hotword detection “Ok Google” and “Hey Google”.

This error does not occur for push-to-talk feature. So, for our project we used this feature to trigger the Google Assistant. This command functions perfectly fine.

**googlesamples-assistant-pushtotalk**

### Contribution:

Each of the members had unique contributions towards the project. Rida Zainab is responsible for preliminary research, gathering information, external materials to refer to, project evaluation and code testing. Raveena Mehta is responsible for compiling all the modules, code testing, result extraction and documentation. Monami Mukhopadhyay is responsible for code testing, result extraction, project compilation, documentation, gathering and implementing references and other resources.

### References:

Kevinwlu. (n.d.). kevinwlu/iot. Retrieved from <https://github.com/kevinwlu/iot>.