# GassistPi Documentation

***Release 6.0***

**Shivasiddharth**

**May 16, 2022**

# Contents


GassistPi

CHAPTER 1

# About this project

**Note:** Developments have been stopped and this project is no longer maintained.

In May of 2017, Google Released it's AIY Projects kit. Initially not many had access to it, so that is when I started modifying the Google Assistant SDK adding AIY like features to help out the ones left without the kit.

Every project requires a name and so I named it GassistPi ("G"oogle "Assist"ant on "Pi").

Fast forwarding to date, the project works not only on Pi boards but on a number of other platforms (checkout the supported platforms page for more details).

It has some interesting custom actions for both entertainment and home-automation needs. Primary language of coding is Python and the project has been structed in a way to allow even inexperienced programmers to modify existing codes and implement their own custom actions.

This is a project for the single board community community, driven by the community.

CHAPTER 2

# Features

**All features are applicable to all boards, unless and otherwise mentioned.**

1. Headless auto start on boot.
2. Voice control of GPIOs without IFTTT, api.ai, Actions SDK (Only for Raspberry Pi Boards - non OSMC).
3. Voice control of NodeMCU without IFTTT and MQTT.
4. Radio streaming.
5. Voice control of servo connected to RPi GPIO (Only for Raspberry Pi Boards - non OSMC).
6. Safe shutdown RPi using voice command.
7. Stream Music from YouTube.
8. Indicator lights for assistant listening and speaking events.
9. Startup audio and audio feedback for wakeword detection.
10. Pushbutton service to stop Music or Radio playback.
11. Parcel tracking using Aftership API.
12. RSS Feed streaming.
13. Control of Kodi or Kodi Integration.
14. Streaming music from Google Play Music.
15. Casting of YouTube Videos to Chromecast and Chromecast media control by voice.
16. Voice control of Radio/YouTube/Google Music volume levels.
17. Control Sonoff-Tasmota Devices.
18. Track Kickstarter campaigns.
19. Emulated Philips Hue HUB service and control of Emulated Hue Lights.
20. Search recipes and get push message of ingredients and link to recipe.
21. Remote control of Magic Mirror.

22. Play your Spotify playlist.
23. Custom wakeword activation for all Pi boards using Snowboy and Picovoice-Porcupine.
24. Mute microphones to prevent listening to Ok-Google hotword (Only Raspberry Pi Boards - non OSMC).
25. Create custom conversations.
26. Control of lights added to Domoticz.
27. Stream music from Gaana.com.
28. Stream your playlist from Deezer.
29. Custom actions in French, Italian, German, Dutch, Spanish and Swedish.
30. Send commands over MQTT to the Google Assistant (Only Armv7 boards).
31. Control Assistant using IR Remote (Only Raspberry Armv7 boards).
32. Send Voice Messages from the SBC to the Mobile using Pushbullet (Only Armv7 boards).
33. Send Clickatell SMS messages.
34. CES 2019 Like Live Translator or Interpreter (Needs Cloud Speech).
35. Control Demoticz, Sonoff devices from other assistant devices.
36. Run custom script by voice.
37. Sending voice messages from phone to the Raspberry Pi.

CHAPTER 3

# Before Starting

**Note:** **For Non-Issue Help and Interaction Use Gitter:** https://gitter.im/GassistPi/Lobby/

## 3.1 Supported Platforms

**Note:**

- Do not use the prebuilt AIY Image.
- Do not run the upgrade command.
- Skip the updation process when setting up a fresh copy of Raspbian OS.

**Any single board computer or machine running one of the following OS:**

- Armbian Buster and Bullseye
- Raspbian Buster and Bullseye
- OSMC Stretch
- Ubuntu Bionic

## 3.2 Getting Started

**Note:** **${USER}** will automatically take your username. No need to change that. Just copy pasting the commands on terminal will work.

Install git and clone the project:

```
sudo apt-get install git
git clone https://github.com/shivasiddharth/GassistPi
```

# CHAPTER 4

# Configuring audio

**Note:**

- Non-Raspberry Pi users and users using other setups, choose the USB-DAC option.
- The speaker-test command is used to initialize alsa, so please do not skip that.
- AIY-HAT users, please reboot the Pi at places mentioned, else it will lead to audio and taskbar issues.
- Those using any other DACs or HATs install the cards as per the manufacturer's guide and then you can use the USB-DAC config file after changing the hardware ids.
- Respeaker users, please do not use their official setup for this project.

## 4.1 Choose the audio configuration according to your setup

Non-Raspbian users install Alsa first:

```
sudo apt-get install alsa-utils
```

### 4.1.1 Users on Raspberry Pi OS Prior to Dec 2020 Release

#### USB DAC or USB Sound Card Users

Run the following in the terminal:

```
sudo apt-get update
cd /home/${USER}/
sudo chmod +x ./GassistPi/audio-drivers/USB-DAC/scripts/disable-onboard.sh
sudo ./GassistPi/audio-drivers/USB-DAC/scripts/disable-onboard.sh
sudo reboot
```

(continues on next page)

(continued from previous page)

```
cd /home/${USER}/
sudo chmod +x ./GassistPi/audio-drivers/USB-DAC/scripts/install-usb-dac.sh
sudo ./GassistPi/audio-drivers/USB-DAC/scripts/install-usb-dac.sh
speaker-test
```

### AIY-HAT Users

Run the following in the terminal:

```
sudo apt-get update
cd /home/${USER}/
sudo chmod +x ./GassistPi/audio-drivers/AIY-HAT/scripts/configure-driver.sh
sudo ./GassistPi/audio-drivers/AIY-HAT/scripts/configure-driver.sh
sudo reboot
cd /home/${USER}/
sudo chmod +x ./GassistPi/audio-drivers/AIY-HAT/scripts/install-alsa-config.sh
sudo ./GassistPi/audio-drivers/AIY-HAT/scripts/install-alsa-config.sh
speaker-test
```

### AIY-VOICE-BONNET Users

Run the following in the terminal:

```
sudo apt-get update
cd /home/${USER}/
sudo chmod +x ./GassistPi/audio-drivers/AIY-VOICE-BONNET/scripts/configure-driver.sh
sudo ./GassistPi/audio-drivers/AIY-VOICE-BONNET/scripts/configure-driver.sh
sudo reboot
cd /home/${USER}/
sudo chmod +x ./GassistPi/audio-drivers/AIY-VOICE-BONNET/scripts/install-alsa-config.
↪sh
sudo ./GassistPi/audio-drivers/AIY-VOICE-BONNET/scripts/install-alsa-config.sh
speaker-test
```

---

**Note:**

**Pi users using a release between May 2020 and Dec 2020:**

- Copy the USB-MIC-HDMI and USB-MIC-JACK folders from the /Extras/May2020 directory and paste them in the audio-drivers directory and then proceed with the instructions below.

---

### USB Mic and HDMI Users

Run the following in the terminal:

```
sudo apt-get update
cd /home/${USER}/
sudo chmod +x ./GassistPi/audio-drivers/USB-MIC-HDMI/scripts/configure.sh
sudo ./GassistPi/audio-drivers/USB-MIC-HDMI/scripts/configure.sh
sudo reboot
cd /home/${USER}/
```

(continues on next page)

(continued from previous page)

```
sudo chmod +x ./GassistPi/audio-drivers/USB-MIC-HDMI/scripts/install-usb-mic-hdmi.sh
sudo ./GassistPi/audio-drivers/USB-MIC-HDMI/scripts/install-usb-mic-hdmi.sh
speaker-test
```

### USB Mic and Audio Jack Users

Run the following in the terminal:

```
sudo apt-get update
cd /home/${USER}/
sudo chmod +x ./GassistPi/audio-drivers/USB-MIC-JACK/scripts/usb-mic-onboard-jack.sh
sudo ./GassistPi/audio-drivers/USB-MIC-JACK/scripts/usb-mic-onboard-jack.sh
speaker-test
```

### Custom Voice HAT Users

Run the following in the terminal:

```
sudo apt-get update
cd /home/${USER}/
sudo chmod +x ./GassistPi/audio-drivers/CUSTOM-VOICE-HAT/scripts/install-i2s.sh
sudo ./GassistPi/audio-drivers/CUSTOM-VOICE-HAT/scripts/install-i2s.sh
sudo reboot
cd /home/${USER}/
sudo chmod +x ./GassistPi/audio-drivers/CUSTOM-VOICE-HAT/scripts/custom-voice-hat.sh
sudo ./GassistPi/audio-drivers/CUSTOM-VOICE-HAT/scripts/custom-voice-hat.sh
speaker-test
```

### Respeaker/Waveshare/Raspiaudio 2 Mic HAT Users

Run the following in the terminal:

```
sudo apt-get update
cd /home/${USER}/
git clone https://github.com/shivasiddharth/WM8960-Audio-HAT
cd ./WM8960-Audio-HAT/
sudo ./install.sh
```

Before restarting, run:

```
sudo nano /etc/pulse/default.pa
```

In that, add the following lines and save:

```
load-module module-alsa-source device = hw: 0,0
load-module module-alsa-sink
```

Now, you can reboot:

```
sudo reboot
speaker-test
```

### 4.1.2 Users on Raspberry Pi OS Dec 2020 Release or After

#### USB DAC or USB Sound Card or USB Mic Users

**From Dec 2020 release, USB audio devices are plug and play.**

1. Insert your USB device.
2. Right click on the audio/speaker icon on the bar at the top.
3. Select your Audio Input and Audio Output device.

#### AIY-HAT Users

Run the following in the terminal:

```
sudo apt-get update
cd /home/${USER}/
sudo chmod +x ./GassistPi/audio-drivers/AIY-HAT/scripts/configure-driver.sh
sudo ./GassistPi/audio-drivers/AIY-HAT/scripts/configure-driver.sh
sudo reboot
speaker-test
```

#### AIY-VOICE-BONNET Users

Run the following in the terminal:

```
sudo apt-get update
cd /home/${USER}/
sudo chmod +x ./GassistPi/audio-drivers/AIY-VOICE-BONNET/scripts/configure-driver.sh
sudo ./GassistPi/audio-drivers/AIY-VOICE-BONNET/scripts/configure-driver.sh
sudo reboot
speaker-test
```

#### Custom Voice HAT Users

Run the following in the terminal:

```
sudo apt-get update
cd /home/${USER}/
sudo chmod +x ./GassistPi/audio-drivers/CUSTOM-VOICE-HAT/scripts/install-i2s.sh
sudo ./GassistPi/audio-drivers/CUSTOM-VOICE-HAT/scripts/install-i2s.sh
sudo reboot
speaker-test
```

#### Respeaker/Waveshare/Raspiaudio 2 Mic HAT Users

Run the following in the terminal:

```
sudo apt-get update
cd /home/${USER}/
git clone https://github.com/shivasiddharth/WM8960-Audio-HAT
```

(continues on next page)

(continued from previous page)

```
cd ./WM8960-Audio-HAT/
sudo ./install.sh
```

Before restarting, run:

```
sudo nano /etc/pulse/default.pa
```

In that, add the following lines and save:

```
load-module module-alsa-source device = hw: 0,0
load-module module-alsa-sink
```

Now, you can reboot:

```
sudo reboot
speaker-test
```

# CHAPTER 5

## Installing Google Assistant

1. Follow the instructions here to configure a developer project and account settings.

   Then follow this guide to register the device and obtain the credentials.json file. Refer to the video below for step by step guidelines.

2. Place the credentials.json file in/home/${USER}/ directory.

**Note:** Do not rename the credentials file.

3. Use the one-line installer for installing Google Assistant.

   3.1 Change directory:

```
cd /home/${USER}/
```

   3.2 Make the installer Executable:

```
sudo chmod +x ./GassistPi/scripts/gassist-installer.sh
```

   3.3 Execute the installer:

```
sudo  ./GassistPi/scripts/gassist-installer.sh
```

**Note: When Prompted, enter your Google Cloud console Project-Id, A name for your Assistant and the Full path to the credentials file, including the json extension.**

4. Copy the google assistant authentication link from terminal and authorize using your google account.
5. Copy the authorization code from browser onto the terminal and press enter.
6. Assistant installation is done now.

**Note:** At the first start of the assistant, the volume may be low. Issue **Hey Google, Set volume to maximum** to set the volume to maximum.

CHAPTER 6

# Headless Google Assistant

## 6.1 Setting up Google Assistant to auto start on boot

1. Open the service files in the /GassistPi/systemd/ directory and verify your project and model ids and save the file.
2. Change directory:

```
cd /home/${USER}
```

3. Make the service installer executable:

```
sudo chmod +x ./GassistPi/scripts/service-installer.sh
```

4. Run the service installer:

```
sudo ./GassistPi/scripts/service-installer.sh
```

5. Enable the service:

```
sudo systemctl enable gassistpi.service
```

6. Start the service:

```
sudo systemctl start gassistpi.service
```

## 6.2 Steps to manually start the assistant

At any point of time, if you wish to manually start the assistant:

1. Stop the service if it is already running:

```
sudo systemctl stop gassistpi.service
```

2. **Ok-Google Hotword/Pi4/Pi3/Pi2/Armv7 users** Open a terminal and execute the following:

```
/home/${USER}/env/bin/python -u /home/${USER}/GassistPi/src/main.py  --
↪project_id 'replace this with the project id' --device_model_id 'replace␣
↪this with the model id'
```

3. **Pushbutton/Pi Zero/Pi B+ and other Non-Armv7 users** Open a terminal and execute the following:

```
/home/${USER}/env/bin/python -u /home/${USER}/GassistPi/src/pushbutton.py --
↪project-id 'replace this with your project id' --device-model-id 'replace␣
↪this with the model id'
```

Insert your Project Id and Model Id in quotes in the mentioned places

## 6.3 Disabling assistant's auto start

At any point of time, if you wish to stop the auto start of the assistant:

Open a terminal and execute the following:

```
sudo systemctl stop gassistpi.service
sudo systemctl disable gassistpi.service
```

CHAPTER 7

# Updating the project

1. Change directory:

```
cd /home/${USER}/
```

2. Make the update script executable:

```
sudo chmod +x /home/${USER}/GassistPi/scripts/update.sh
```

4. Run the update script:

```
sudo /home/${USER}/GassistPi/scripts/update.sh
```

5. If there is an update available, the project will be updated else the script will make a smooth exit.
6. If the Project is updated, reconfigure the **config.yaml** file.

**Note:** The update script will make a backup of your existing project folder before updating eg. GassistPi.bak-xxxx-xx-xx**

# CHAPTER 8

Guide to use the customizations/features

## 8.1 Enabling or Disabling Custom Actions

Major custom actions have been provided with a control key or switch in the **config.yaml**. Set it to **Enabled** to enable the custom actions and set it to **Disabled** to disable them.

## 8.2 Using Custom Actions in Non-English Languages

1. Languages supported: French, Italian, Spanish, Dutch, German and Swedish.
2. In the **config.yaml** file, under the **Languages and Choice** option set your desired language.
3. Use the Translated versions of the English syntaxes given for all the custom actions.
4. You can change the keywords/trigger words for the custom actions in the keywords file.

## 8.3 Controlling Sonoff-Tasmota, Domoticz devices from Google Home

1. This has been implemented using Adafruit_IO.
2. Create an acount and a feed in adafruit.io website.
3. Enter those details in the config.yaml file.
4. Register or login into IFTTT and create an applet to send commands from google assistant to adafruit_io feed.
5. For controlling domoticz and sonoff devices, the adafruit.io command should match the syntaxes for the respective custom actions.

## 8.4 Using the Interpreter Mode

**Note:** Google has the interpreter feature enabled in the cloud. That is, you can use the interpreter without this customization hack. You can leave this feature disabled in the config.yaml.

**Note:** This uses GOOGLE CLOUD SPEECH API. Free usage is limited to 60MINS/MONTH.

1. Go to the projects page on your Google Cloud Console.
2. Select your project from the list.
3. On the left top corner, click on the hamburger icon or three horizontal stacked lines.
4. From the **API and services** option, select library and in the search bar type **speech**, select **Cloud Speech API** and click on "ENABLE".
5. You will be prompted to create a billing account if you already have not created one. Follow the onscreen instructions to create a billing account and then Enable the API.
6. Create a service account and generate credentials.
7. Copy the downloaded the JSON key and place it /home/pi/ directory **DO NOT RENAME**.
8. Enter the path to the Key along with the key name Eg: /home/pi/xxxx.json in the config.yaml file in the "Google_Cloud_Speech_Credentials_Path" field under "Speechtotext". You can use one key for Cloud Speech and Cloud Text to Speech, but should enter the same path seperately in config.yaml

**Command Syntax:**

To start the interpreter:

```
Hey Google, Start __Your-Desired-Language__ interpreter.
```

To stop the interpreter:

```
Hey Google, Stop interpreter.
```

## 8.5 Using Google Cloud Text to Speech

**Note:** GOOGLE CLOUD TEXT TO SPEECH API has limited free access. Once the quota is exceeded, the program will automatically switch to gTTS.

1. Go to the projects page on your Google Cloud Console.
2. Select your project from the list.
3. On the left top corner, click on the hamburger icon or three horizontal stacked lines.
4. "From the API and services" option, select library and in the search bar type text, select "Cloud Text-to-Speech API" and click on "ENABLE".
5. In the API window, click on "Credentials" and then on "+ Create Credential".

6. In the "Add credentials to your project" window, in step-1 under "Which API are you using?" drop down choose "Cloud Text-to-Speech API" and down below choose "No, I'm not using them". Then click on "What credentials do I need?"
7. In step-2 give your service account a name and on the right in the "Role" drop down choose Project–>Owner and under "Key Type" select "JSON" and click "Continue".
8. Copy the downloaded key and place it /home/pi/ directory DO NOT RENAME.
9. Enter the path to the Key along with the key name Eg: /home/pi/xxxx.json in the config.yaml file in the **Google_Cloud_TTS_Credentials_Path** field.

---

**Note:** You can use one key for Cloud Speech and Cloud Text to Speech, but should enter the same path seperately in config.yaml

---

## 8.6 Adding Custom Search API and Generating API Key

1. Go to the projects page on your Google Cloud Console.
2. Select your project from the list.
3. On the left top corner, click on the hamburger icon or three horizontal stacked lines.
4. Move your mouse pointer over **API and services** and choose **Credentials**.
5. Click on create credentials and select API Key and choose close. Make a note of the created API Key and enter it in the config.yaml script at the indicated location.
6. "From the API and services" option, select library and in the search bar type **search**, select **Custom Search API** API and click on "ENABLE".
7. In the API window, click on "All API Credentials" and in the drop down, make sure to have a tick (check mark) against the API Key that you just generated.

## 8.7 Adding YouTube Data API and Generating API Key

1. Go to the projects page on your Google Cloud Console.
2. Select your project from the list.
3. On the left top corner, click on the hamburger icon or three horizontal stacked lines.
4. Move your mouse pointer over **API and services** and choose **Credentials**.
5. Click on create credentials and select API Key and choose close. Make a note of the created API Key and enter it in the config.yaml script at the indicated location.
6. "From the API and services" option, select library and in the search bar type **youtube**, select **YouTube Data API v3** API and click on "ENABLE".
7. In the API window, click on "All API Credentials" and in the drop down, make sure to have a tick (check mark) against the API Key that you just generated.

---

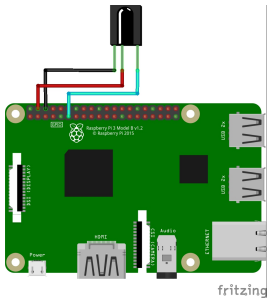
**Note:** If a custom action uses both Custom Search and YouTube API, you need to enable both the APIs but only one API KEY needs to be generated.

---

**Note:** The same API key can be used for all the associated custom actions.

## 8.8 Controlling Assistant or Sending Preset Commands Using IR Remote

1. Connect the IR Receiver according to the wiring diagram given below.




**Note:** The diagram given is for GPIO 17, if you are using another GPIO, please make the suitable changes to the connection.

2. Run the sample IR receiver script to get the codes for your desired buttons:

```
python /home/${USER}/GassistPi/Extras/IR-Sensor.py
```

3. In the config.yaml under IR, list your codes and corresponding queries/actions. The number of queries should match the number of codes listed.
4. If you want to execute the custom actions like Spotify, YouTube playback, Domoticz Control etc, prefix the word custom.

   Eg:

```
custom Play God's Plan from Youtube
custom Turn On _Domoticz-device-name__
custom Play all the songs from Google Music
```

5. If you are sending a command to be processed by google assistant, there is no need to prefix custom.

   Eg:

```
what is the time
what can you do for me
```

   **Video for reference:**

## 8.9 Sending Commands or Queries to Google Assistant Over MQTT

1. Set up your desired MQTT broker. If you are setting up Raspberry Pi as a MQTT broker, follow the guide below.

2. Enter the MQTT broker credentials and subscription topic in the provided config.yaml file.
3. Set the **MQTT_Control** to **Enabled**.
4. Now, you can send queries or commands to google assistant over MQTT.
5. If you are sending a command for custom actions, prefix custom in the payload.

   Eg:

```
custom Play God's Plan from Youtube
custom Turn On __Domoticz-device-name__
custom Play all the songs from Google Music
```

6. If you are sending a command to be processed by google assistant, there is no need to prefix custom.

   Eg:

```
what is the time
what can you do for me
```

7. To turn on/off microphone just send the simple command mute.

   Eg:

```
mute
```

   **For more details on the how to use this feature, refer to the video below:**

## 8.10 Streaming Music from Deezer

---

**Note:**

- As a roundabout approach, I have programmed the assistant to get the playlist details using Deezer API and then fetch those tracks from YouTube.
- This feature uses a YouTube Data API v3.
- Click here for guidelines to add YouTube Data API to the project and to generate the required key.

---

1. Add your Deezer user number in the config.yaml under the **Deezer:** and **User_id**.
2. In the config.yaml, under **Google_cloud_api_key:** replace **ENTER-YOUR-GOOGLE-CLOUD-API-KEY-HERE** with the key from Google Cloud Console.

**Command Syntax:**

To play the playlists added to your Deezer account:

```
Hey Google, Play playlist __playlist-number__ from Deezer.
```

Example:

```
Hey Google, Play __playlist 1__ from Deezer
```

## 8.11 Streaming Music from Gaana.com

**Note:**

- As a roundabout approach, I have programmed the assistant to get the playlist details using Deezer API and then fetch those tracks from YouTube.
- This feature uses both YouTube Data API v3 and Custom Search API.
- Click here for guidelines to add YouTube Data API to the project and to generate the required key.
- Click here for guidelines to add Custom Search API to the project and to generate the required key.

1. Add your playlists in the config.yaml under **Gaana: and Playlists:**.
2. In the config.yaml, under **Google_cloud_api_key:** replace **ENTER-YOUR-GOOGLE-CLOUD-API-KEY-HERE** with the key from Google Cloud Console.

**Command Syntax:**

1. To play the playlists added in config.yaml file:

```
Hey Google, Play playlist __playlist-number__ from Gaana.com
```

   Example:

```
Hey Google, Play __playlist 1__ from Gaana.com
```

2. To play other playlists:

```
  Hey Google, Play __user-playlist-query__ from Gaana.com

Example::

  Hey Google, Play __Bollywood top 50__ from Gaana.com
```

## 8.12 Controlling Domoticz Devices

**Note:** As of today, you can control lights and switches only, more controls will be added in the future.

1. In the config.yaml file under **Domoticz:** change **Domoticz_Control:** from **Disabled** to **Enabled**.
2. List the device names and the ids that you want to control in the config.yaml file. The names should be the same as the ones in the domoticz server.

**Command Syntax:**

1. To On/Off/Toggle:

```
Hey Google, Turn On/Turn Off/Toggle  __Name of your light__
```

   Example:

```
Hey Google, Turn On __Bedroom Lamp__
```

2. To Change Brightness (between 0 and 100):

```
Hey Google, Set  __Name of your light__ brightness to __desired value__
```

Example:

```
Hey Google, Set __Bedroom lamp__ brightness to __5__
```

3. To Change Colour (refer to the list of available colors):

```
Hey Google, Set  _Name of your light_ color to __desired color__
Hey Google, Change  __Name of your light__ to __desired color__ color
```

Example:

```
Hey Google, Set __Bedroom lamp__ color to __red__
Hey Google, Change __Bedroom lamp__ to __red__ color
```

## 8.13 Custom Conversations

1. Customize the assistant's reply to a specific question.
2. Add the list of questions and answers in config.yaml under the **Conversation**: option.
3. **There must be only one question, but corresponding answers can be as many**.
4. Sample questions and answers has been provided, please follow the same pattern.

## 8.14 Custom Wakeword Activation

1. You can choose to either Enable or Disable the custom wakeword activation in the config.yaml file.
2. In the config.yaml file, under Wakewords, change the **"Custom_Wakeword"** to **'Enabled'** if you want to use the custom wakeword or set it to 'Disabled' if you dont want to use the custom wakeword option.
3. You have a choice between Snowboy and Picovoice for the custom wakeword engine.
4. For Snowboy, change **"Wakeword_Engine"** to **Snowboy** and for Picovoice, change **"Wakeword_Engine"** to **Picovoice**.
5. For changes to take effect, you need to restart the assistant. Changing status while an instance of assistant is already running will not cause any change.
6. Create your custom snowboy model here. Add the models to **/GassistPi/src/resources/snowboy_models** directory.
7. Sample Snowboy and Picovoice models have been provided and placed in the /GassistPi/src/resources/ folder. Set your desired models by setting their paths in the config.yaml file.
8. To disable the default **"Ok Google"** hotword, set the Ok_Google option to **"Disabled"**.

---

**Note:** If you turn off the default **Ok Google** wakeword/hotword, everytime you invoke the assistant using the custom wakeword, you will get a prompt for the Mic being turned Off and On.

---

9. Users using pushbutton.py or Pi Zero users have an option between using custom wakeword and GPIO trigerring. If custom wakeword is enabled, then GPIO trigger will not work. To enable GPIO triggering, set custom wakeword to 'Disabled'.

## 8.15 Playing Spotify Playlist

---

**Note:**

- Spotify API currently only supports playback in a web browser, but DRM content is being blocked in the Raspberry Pi.
- As a roundabout approach, I have programmed the assistant to get the playlist details using Spotipy API and then fetch those tracks from YouTube. This custom program has a better accuracy than spotify playlist playback using mpsyt.
- This feature uses a YouTube Data API v3.
- Click here for guidelines to add YouTube Data API to the project and to generate the required key.

---

1. Click here and register for a spotify developer account, if you already don't have one.
2. In the developer's dashboard, choose **CREATE A CLIENT ID**. In the pop-up window provide the requested details.
3. Set the Redirect URIs to http://localhost:8888
4. Click on the new app created and copy the **CLIENT ID** and **CLIENT SECRET**. Paste it in the config.yaml file in the indicated space.
5. Access spotify here and copy the username to be entered in config.yaml.

**Command Syntax:**

To play your playlist:

```
Hey Google, Play __user-playlist-query__  from Spotify
```

Example:

```
Hey Google, Play __Workout playlist__ from Spotify
Hey Google, Play __Top Dance Numbers__ from Spotify
```

---

**Note:** If your playlist name does not have the word **playlist** do not use that in the query.

---

## 8.16 Tracking Kickstarter Campaigns

A custom Google search engine for Kickstarter has been used. This requires an API to be added to your existing project.

Click here for guidelines to add Custom Search API to the project and to generate the required key.

**Command Syntax:**

To track a Kickstarter campaign:

```
Hey Google, Track __your-desired-campaign__ Kickstarter campaign
Hey Google, What is the status of __your-desired-campaign__ Kickstarter campaign
```

Example:

```
Hey Google, Track __Mycroft 2__ Kickstarter campaign
Hey Google, What is the status of __Mycroft 2__ Kickstarter campaign
```

## 8.17 Emulated Philips Hue Control

1. Credits for the Emulated Hue to Marius Motea.
2. Follow the guidelines given in the diyHue's documents to setup the Emulated Hue Service.
3. Download sketches for your NodeMCU/Wemos/ESP Devices from here.
4. If the Hue config file is not in the default location, change the path to the Hue config file in the following lines of main.py script.

```
if os.path.isfile('/opt/hue-emulator/config.json'):
      with open('/opt/hue-emulator/config.json', 'r') as config
```

**Command Syntax:**

1. To turn On/Off lights:

```
Hey Google, Turn __Hue-Light-Name__ On/Off
```

2. To change light color:

```
Hey Google, Change __Hue-Light-Name__ colour to __Required-Colour__
```

3. To change brightness:

```
Hey Google, Change __Hue-Light-Name brightness__ to __Required-Brightness-Level__
```

## 8.18 Sending Voice Messages from Pi to Phone/Tablet

**Note:** For pushing voice messages, the GassistPi uses Pushbullet API.

1. Download and install pushbullet app on your tablet/mobile device.
2. Visit www.pushbullet.com register for new account or sign in with your existing account.
3. Choose Settings–>Account and then choose **Create access token**.
4. Copy this token and paste in config.yaml under **Pushbullet** and **Pushbullet_API_KEY**.

**Command Syntax:**

To send message:

```
Hey Google, Send message
```

## 8.19 Send SMS via Clickatell

1. Create a free account with clickatell.com.
2. Sign in to Clickatell SMS Platform.
3. Create sms integration.
4. Add your generated clickatell api number in config.yaml at the given point.

**Command Syntax:**

To send sms:

```
Hey Google, Send clickatell message to __Person-name__
```

## 8.20 Get Recipe Messaged to Mobile/Tablet

---

**Note:** This feature uses the Pushbullet API for sending the recipes. Please first setup the Pushbullet feature as given here.

---

**GassistPi uses Edamam for getting recipe details/info. To use this feature:**

1. Click here to visit the developers' porta for Edamam.
2. Signup as a developer/login with your existing account.
3. In the Menubar at the top, Click on Dashboard–>Applications–>Create a new applicatiuon–>Recipe Search API and then create a new application.
4. Copy the application id and application key and paste it in the actions.py script under the getrecipe function.

---

**Note:** While copying the application key, do not copy the "—"

---

**Command Syntax:**

To get recipes:

```
Hey Google, Get ingredients for __Required-Item__
```

## 8.21 Control Magic Mirror by Voice

1. You can control either Magic Mirror running on another Pi or Magic Mirror running on the same pi as GassistPi.
2. As a prerequisite, you should have the remote control module installed in the Pi running Magic Mirror.
3. Enter the Ip address of Pi running Magic Mirror in the config.yaml against the variable **mmmip** declared.

**Command Syntax:**

1. To show/hide weather module:

```
Hey Google, Show/Hide Weather on Magic Mirror
```

2. To turn magic mirror display on/off:

```
Hey Google, Turn Magic Mirror display on/off
```

3. To power off/reboot/restart Magic Mirror:

```
Hey Google, Power off/Reboot/Restart Magic Mirror
```

## 8.22 Indicators for Google Assistant's Listening and Speaking Events

**Note:** Default GPIOs for the indicators are BCM GPIO05 and GPIO06. If you want to change the pins, change it in the config.yaml file.

Connect LEDs with colours of your choice to GPIO05 for Listening and GPIO06 for Speaking Events.

## 8.23 Pushbutton to Stop Music/Radio Playback and for Muting the Microphone

**Note:** Default GPIO for the pushbutton is BCM GPIO23. If you want to change the pins, change it in the config.yaml file.

Connect a pushbutton between GPIO23 and Ground. Single press mutes microphone and double press stops music streaming.

## 8.24 Voice Control of GPIOs, Servo Motor and Safe Shutdown of Pi

**Note:**

- This feature uses a combination of two keywords to prevent false positives. These can be changed in the keywords.yaml file.
- The default main keyword is **trigger**.
- The default keyword for servo motor control is **servo**.
- The default keyword for safe shutdown is **shutdown**.
- Names for devices connected to the GPIOs should be assigned in the config.yaml file.

**Command Syntax:**

1. To turn device connected to GPIO on/off:

```
Hey Google, **Trigger** turn __Device-Name__ on/off
```

2. To turn servo motor (example by 90 degrees):

```
Hey Google, **Trigger** **servo** 90
```

3. To power off the Pi:

```
Hey Google, **Trigger Shutdown**
```

**Check out the demo in the video below:**

## 8.25 Voice Control of NodeMCU

NodeMCU control has been implemented in two ways:

1. Control of NodeMCU running a webserver.
2. Control of NodeMCU running Sonoff-Tasmota Firmware.

### 8.25.1 Controlling NodeMCU Running Webserver

1. Download the Arduino IDE code for Nodemcu from here.
2. Add the wifi credentials, make the desired changes and upload the Arduino code onto the NodeMCU and get the IP address from the serial monitor.
3. Add the NodeMCU's IP address in the config.yaml under the **ESP** and **IP**.
4. Set device names in the config.yaml under **ESP** and **devicename**.
5. The default keyword for NodeMCU control is **Wireless**. You can change this in the keywords.yaml file.

**Command Syntax:**

To turn the device or ESP Pin on/off:

```
Hey Google, **Wireless** turn __Device-Name__ on/off
```

### 8.25.2 Controlling NodeMCU Running Sonoff-Tasmota Firmware

1. Download the Tasmota firmware from this link.

---

**Note:** This is an old firmware. You can upgrade to the latest firmware from this one.

---

2. Upload the firmware properly. You can use this video for reference.
3. Set your Tasmota details in the under **Tasmota_devicelist** in config.yaml file.

**Command Syntax:**

To turn the Tasmota device on/off:

```
Hey Google, Turn __Tasmota-Device-Name__ on/off
```

## 8.26 Casting YouTube Videos to Chromecast and Chromecast Control

**Note:**

- This feature uses a YouTube Data API v3. Click here for guidelines to add YouTube Data API to the project and to generate the required key.
- The default keyword is **Chromecast**. You can change that in the main.py script at `# if 'chromecast'.lower() in str(usrcmd).lower():`.
- The IP address of Chromecast can be set in the actions.py file at:

```
def chromecast_control(action):
  # Chromecast declarations
  # Do not rename/change "TV" its a variable
  TV = pychromecast.Chromecast("192.168.1.13") #Change ip to match the ip-address␣
↪of your Chromecast
```

- Google has permitted the Chromecast control on its own. So this feature has been disabled. It can be enabled, if Google disables chromecast control in the future.

**Command Syntax:**

1. To Play Video on Chromecast:

```
Hey Google, Play __Desired-Video__ on Chromecast
```

2. To Stop Playback:

```
Hey Google, Stop Chromecast
```

3. To Change volume:

```
Hey Google, Chromecast Volume Up/Down
```

## 8.27 Controlling Media or Music Streaming by Voice

You can change volume, play, pause, change music/radio. Command syntaxes have been given below with the keyword within * symbols.

**Command Syntax:**

1. To Pause:

```
Hey Google, **Pause Music**
```

2. To Resume:

```
Hey Google, **Resume Music**
```

3. To Increase/Decrease volume:

```
Hey Google, Increase/Decrease **Music Volume** by __A-Number-Between-0-100__
```

4. To Set Volume:

```
Hey Google, Set/Change **Music Volume** to __A-Number-Between-0-100__
```

5. To Set Volume to Maximum/Minimum:

```
Hey Google, Set/Change **Music Volume** to maximum/minimum
```

6. To Change to Previous Track:

```
Hey Google, **Play Previous**
Hey Google, **Play Previous** Song
Hey Google, **Play Previous** Track
```

7. To Change to Next Track:

```
Hey Google, **Play Next**
Hey Google, **Play Next** Song
Hey Google, **Play Next** Track
```

## 8.28 Music Streaming from YouTube

1. Music playback from YouTube is facilitated by the YouTube Data API v3.
2. Click here for guidelines to add YouTube Data API to the project and to generate the required key.

**Command Syntax:**

1. To Play a Song:

```
Hey Google, Play __Desired-Song-Name__ from YouTube
```

2. To Play and Song and Related 10 Songs:

```
Hey Google, Autoplay __Desired-Song-Name__ from YouTube
```

3. To Play a Song from a Playlist:

```
Hey Google, Play __Desired-Playlist-Name__ from YouTube
```

4. To Play 10 Songs from a Playlist:

```
Hey Google, Autoplay __Desired-Playlist-Name__ from YouTube
```

5. To Play a Song from a Channel:

```
Hey Google, Play __Desired-Channel-Name__ from YouTube
```

6. To Play 10 Songs from a Channel:

```
Hey Google, Autoplay __Desired-Channel-Name__ from YouTube
```

**Note:** Depending upon the internet and the system/board specification, the features that involve 10 tracks will take some time to fetch the track list.

## 8.29 Music Streaming from Google Music

..note:: Due to Google shutting down the Google Music service/app, the parts of codes pertaining to this feature have been commented out.

The music streaming from Google Music is facilitated by using gmusicapi.

First you need to authorize the api to access your Google Music data. Run the following script to authorize prior to starting the assistant.

```
/home/${USER}/env/bin/python -u /home/${USER}/GassistPi/Extras/gmusicauth.py
```

**Command Syntax:**

1. To Play All Songs in Loop:

   ```
   Hey Google, Play All the songs from Google Music
   ```

2. To Play a Playlist:

   ```
   Hey Google, Play songs from the first playlist from Google Music
   ```

**Note:** Playlists are sorted by date created, if you have multiple playlists, use a similar syntax replacing first with second, third etc

3. To Play Songs by a Particular Artist:

   ```
   Hey Google, Play songs by artist __YOUR-DESIRED-ARTIST__ from Google Music
   ```

4. To Play Songs from a Particular Album:

   ```
   Hey Google, Play songs from album __YOUR-DESIRED-ALBUM-NAME__ from Google Music
   ```

## 8.30 Run Custom Scripts

1. By default, running custom scripts is disabled.
2. To enable it, change **Script_Control:** in config.yaml from **Disabled** to **Enabled**.

**Note:** The number of script names mentioned should match the number of script commands in config.yaml.

**Command Syntax:**

To Run a Script:

```
Hey Google, Run Script __Desired-Script-Name__
```

## 8.31 Playing Radio Channels

Radio station names and the associated links are to be set in the config.yaml file

**Note:** The number of radio station names mentioned should match the number of radio station links in config.yaml.

**Command Syntax:**

To Play a Radio Station:

```
Hey Google, Play Radio __Desired-Radio-Station-Name__
```

## 8.32 Tracking Parcels

1. Regsiter for a free account with Aftership.
2. Generate an API number and add parcels to the tracking list.
3. Add the API number in the actions.py file

For a better understanding follow the attached youtube video.

**Command Syntax:**

To Track Parcels:

```
Hey Google, Where is my parcel

Hey Google, Track my parcel
```

## 8.33 RSS Feeds Streaming

**Note:** **numfeeds** variable within the feed function in actions.py file is the feed limit. Certain RSS feeds can have upto 60 items and **numfeeds** variable limits the number of items to stream. The default value has been set to 10, which if you want can change.

**Command Syntax:**

To Play Feeds:

```
Hey Google, Top Tech News please

Hey Google, Top Sports News please

Hey Google, Top World News please
```

**Note:** You can interrupt the feed playback using the Stop Pushbutton

## 8.34 KODI Control

**Note:** By default, the KODI control is disabled. To enable, in the config.yaml, under kodi, change control option from **'Disabled'** to **'Enabled'**.

1. Music playback from YouTube is facilitated by the YouTube Data API v3.
2. Click here for guidelines to add YouTube Data API to the project and to generate the required key.
3. Enable HTTP COntrol in KODI i. Settings → Services → Control → Allow remote control via HTTP. ii. Set the port number to 8080, username to kodi and password to kodi (username and password should be in lowercase).
4. For Kodi to play the YouTube video, you need to add and enable the YouTube Plugin on Kodi.

**Command Syntax:**

| Command Syntax | What it does |
|---|---|
| Hey Google, Shuffle my songs on kodi | Shuffles all the songs added to the kodi library |
| Hey Google, Play songs from _Album **name**_ on kodi | Plays all the songs under the mentioned Album name |
| Hey Google, Play songs by, _Artist **name**_ on kodi | Plays all the songs rendered by the mentioned artist |
| Hey Google, Play _Song **name**_ song on kodi | Plays the requested song, if it has been added to the library |
| Hey Google, Play _Movie **name**_ movie on kodi | Plays the requested movie, if it has been added to the library |
| Hey Google, From YouTube, Play _Youtube **Video**_ on kodi | Fetches the YouTube video and plays it on kodi |
| Hey Google, What is playing? on kodi | Tells you by voice as to what is currently playing |
| Hey Google, Repeat this or Repeat one on kodi | Repeats the current track playing |
| Hey Google, Repeat all on kodi | Changes repeat mode to all |
| Hey Google, Repeat off on kodi | Turns off Repeat |
| Hey Google, Turn Shuffle On on kodi | Turns on shuffle mode |
| Hey Google, Turn Shuffle Off on kodi | Turns off shuffle mode |
| Hey Google, Play Next on kodi | Plays the next track |
| Hey Google, Play Previous on kodi | Plays the previous track |
| Hey Google, Scroll a bit forward on kodi | Fast forwards a movie/music by a small amount |
| Hey Google, Scroll forward on kodi | Fast forwards a movie/track by a large margin |
| Hey Google, Scroll a bit backward on kodi | Rewinds a movie/track by a small amount |
| Hey Google, Scroll backward on kodi | Rewinds a movie/track by a large margin |
| Hey Google, Set volume _Between 0 and **100**_ on kodi | Sets the volume to the mentioned number |
| Hey Google, Get volume on kodi | Tells you the current volume level by voice |
| Hey Google, Toggle mute on kodi | Either mutes or unmutes, depending on mute status |
| Hey Google, Pause on kodi | Pauses the current video/track |
| Hey Google, Resume on kodi | Resumes playing the video/track |
| Hey Google, Stop on kodi | Stops playing and closes the player |
| Hey Goolge, goto _Home_ on kodi | Opens the appropriate menu or window mentioned |
| Hey Goolge, goto _Settings_ on kodi | Opens the settings menu or window |
| Hey Goolge, goto _Videos_ on kodi | Opens the videos menu or window |
| Hey Goolge, goto _Weather_ on kodi | Opens the weather menu or window |
| Hey Google, goto _Music_ on kodi | Opens the music menu or window |
| Hey Google, Move Up on kodi | Moves selection pointer up |
| Hey Google, Move Down on kodi | Moves selection pointer down |
| Hey Google, Move Left on kodi | Moves selection pointer left |
| Hey Google, Move Right on kodi | Moves selection pointer right |
| Hey Google, Move Back on kodi | Goes back, equivalent to esc key |

Continued on next page

Table 1 – continued from previous page

| Command Syntax | What it does |
|---|---|
| Hey Google, Move Select on kodi | Makes a selection, equivalent to enter key |

CHAPTER 9

# List of Raspberry Pi GPIOs used in the project

| GPIO Number in BCM | Purpose |
|---|---|
| 25 | Assistant activity indicator for AIY Kits |
| 23 | Pushbutton to stop music/radio AIY and others |
| 05 and 06 | Google assistant listening and responding |
| 22 | Pushbutton trigger for gRPC API. |
| | Connect a pushbutton between GPIO 22 and GRND for manually triggering. |
| 12,13,24 | Voice control of devices connected to GPIO |
| 27 | Voice control of servo |
| 17 | IR Sensor for preset commands |

**Note:** Some HATS may use GPIOs 18, 19, 20, 21 for I2S audio please refer to the manufacturer's pinouts

CHAPTER 10

# Google Home Like Indicator

1. This feature requires an Arduino or NodeMCU to be connected to the Pi.
2. Sketches and wiring diagram have been provided in the Neopixel Indicator folder.
3. Change the Pin numbers in the given sketch according to your board and upload it.
4. Follow the given circuit diagram for connecting the arduino and the pi.

# CHAPTER 11

## List of available colors for home automation projects

| | | COLOURS LIST | | | |
|---|---|---|---|---|---|
| 'Almond' | 'Antique Brass' | 'Apricot' | 'Aquamarine' | 'Asparagus' | 'Atomic Tangerine' |
| 'Banana Mania' | 'Beaver' | 'Bittersweet' | 'Black' | 'Blizzard Blue' | 'Just Blue' |
| 'Blue Bell' | 'Blue Gray' | 'Blue Green' | 'Blue Violet' | 'Blush' | 'Brick Red' |
| 'Brown' | 'Burnt Orange' | 'Burnt Sienna' | 'Cadet Blue' | 'Canary' | 'Caribbean Green' |
| 'Carnation Pink' | 'Cerise' | 'Cerulean' | 'Chestnut' | 'Copper' | 'Cornflower' |
| 'Cotton Candy' | 'Dandelion' | 'Denim' | 'Desert Sand' | 'Eggplant' | 'Electric Lime' |
| 'Fern' | 'Forest Green' | 'Fuchsia' | 'Fuzzy Wuzzy' | 'Gold' | 'Goldenrod' |
| 'Granny Smith Apple' | 'Gray' | 'Just Green' | 'Green Blue' | 'Green Yellow' | 'Hot Magenta' |
| 'Inchworm' | 'Indigo' | 'Jazzberry Jam' | 'Jungle Green' | 'Laser Lemon' | 'Lavender' |
| 'Lemon Yellow' | 'Macaroni and Cheese' | 'Magenta' | 'Magic Mint' | 'Mahogany' | 'Maize' |
| 'Manatee' | 'Mango Tango' | 'Maroon' | 'Mauvelous' | 'Melon' | 'Midnight Blue' |
| 'Mountain Meadow' | 'Mulberry' | 'Navy Blue' | 'Neon Carrot' | 'Olive Green' | 'Orange' |
| 'Orange Red' | 'Orange Yellow' | 'Orchid' | 'Outer Space' | 'Outrageous Orange' | 'Pacific Blue' |
| 'Peach' | 'Periwinkle' | 'Piggy Pink' | 'Pine Green' | 'Pink Flamingo' | 'Pink Sherbet' |
| 'Plum' | 'Purple Heart' | "Purple Mountain's Majesty" | 'Purple Pizzazz' | 'Radical Red' | 'Raw Sienna' |
| 'Raw Umber' | 'Razzle Dazzle Rose' | 'Razzmatazz' | 'Just Red' | 'Red Orange' | 'Red Violet' |
| 'Robin's Egg Blue' | 'Royal Purple' | 'Salmon' | 'Scarlet' | 'Screamin' Green' | 'Sea Green' |
| 'Sepia' | 'Shadow' | 'Shamrock' | 'Shocking Pink' | 'Silver' | 'Sky Blue' |
| 'Spring Green' | 'Sunglow' | 'Sunset Orange' | 'Tan' | 'Teal Blue' | 'Thistle' |
| 'Tickle Me Pink' | 'Timberwolf' | 'Tropical Rain Forest' | 'Tumbleweed' | 'Turquoise Blue' | 'Unmellow Yellow' |