Add Rhasspy local voice control   
to Home Assistant

Using a Raspberry Pi Zero with reSpeaker 2-mic HAT

# Introduction

It seems obvious to me now that using a voice assistant \*means\* multiple Satellite units distributed around the house (why have only one place where commands can be given?); and logically these satellites should be relatively cheap, with the bulk of the cpu-intensive work done on one base station.

I’ll start by assuming that

1. you already know what Home Automation is about, and have Home Assistant installed and running;
2. you have realised that speaking commands is much more convenient than reaching for a screen or app;
3. you don’t want any company listen to your private conversations, or to to hold your home hostage if their policies or T&C changes;
4. you want to speak voice commands in several rooms through the house ... so the compute-intensive processing is done at a base station; and multiple inexpensive low-powered satellite devices (microphone, speaker and Wi-Fi) distributed around the house.
5. You are familiar with PC operation, and have a working terminal program (with SSH);
6. You are familiar with RasPi console (we will not be using a GUI);
7. You have a moderate level of expertise and intelligence, because explaining every little thing at idiot level would make this document too tedious to write ;-)

Over the past year I have gone from a Home Assistant newbie - to a working HA with Rhasspy and 3 Satellites configuration (i.e. I can give commands in three different rooms). There were learning curves for HA, Rhasspy stand-alone, Rhasspy Base/satellite, External MQTT, HermesLEDcontrol, HA Intents, and node-RED. That is a lot for a new user to get their head around, even someone with 30 years programming experience !

This document tries to fast-track you to a working, extensible HA voice assistant; hopefully without all the experimenting, dead ends, lessons and a whole lot of learning curve I went through. I do however want to pass on enough of the distilled lessons to give you understanding of how the pieces fit together.

|  |
| --- |
| There are many *many* combinations of hardware and software which can do the same job. For this tutorial I have assembled a combination of components which work together for me.  HA and Rhasspy are particularly designed to be flexible in terms of which modules are used – so this document must NOT be misinterpreted as being the only (or even the best) way that Rhasspy can be configured. Where feasible I have tried to pass on my understanding of the issues involved.  IMPORTANT: every linux kernel update can break respeaker drivers and/or the LED control, so DO NOT perform updates once you have the satellite working.  Hardware required:   * desktop or other PC running linux, Windows, or MacOS (mainly for web browser to control the other devices). * I assume you already have Home Assistant installed and running on a computer, which we will refer to as the Base machine. * A Raspberry Pi Zero W (or Pi Zero W 2 or Pi 3A+) and reSpeaker 2-mic HAT (or adafruit Voice Bonnet), to be dedicated as a satellite. * A MicroSD card (class 2 or 10) with 8GB or more capacity. The software we are installing uses about 3GB, the rest is used by statistics, log files, backups, etc.   This tutorial was written October 2021 and uses Home Assistant 2021.11, Mosquitto 6.0.1, and Rhasspy 2.5.11. Things can change fairly quickly with all the work being done on HA, Rhasspy and the other packages these use … so be aware that it may be out-of-date when you read it – hopefully only in minor details. |

**Table of Contents**

[Introduction 1](#__RefHeading___Toc1199_1910159651)

[General comments 3](#__RefHeading___Toc4450_4015176539)

[Terminology / concept overviews 3](#__RefHeading___Toc1203_1910159651)

[PC 4](#__RefHeading___Toc7554_518319112)

[Terminal / Secure Shell (SSH) 4](#__RefHeading___Toc6989_5183191121)

[Home Assistant 4](#__RefHeading___Toc7556_518319112)

[Rhasspy 4](#__RefHeading___Toc7558_518319112)

[Mosquitto / MQTT 5](#__RefHeading___Toc7560_518319112)

[HermesLEDControl 5](#__RefHeading___Toc6974_518319112)

[hostname 5](#__RefHeading___Toc7562_518319112)

[Base station computer 6](#__RefHeading___Toc7564_518319112)

[Satellite computer 6](#__RefHeading___Toc7566_518319112)

[Processor board 6](#__RefHeading___Toc6966_518319112)

[Sound cards 6](#__RefHeading___Toc6968_518319112)

[Cases 7](#__RefHeading___Toc6970_518319112)

[Satellite Site-IDs 7](#__RefHeading___Toc7568_518319112)

[Preparation 8](#__RefHeading___Toc1205_1910159651)

[Setup Base station 9](#__RefHeading___Toc1207_1910159651)

[MQTT Broker 9](#__RefHeading___Toc1522_4015176539)

[Install Rhasspy Add-on to Home Assistant 9](#__RefHeading___Toc4433_4015176539)

[Configure Rhasspy Base 10](#__RefHeading___Toc1524_4015176539)

[Setup Satellite 11](#__RefHeading___Toc1211_1910159651)

[On your PC 11](#__RefHeading___Toc7570_518319112)

[On your Satellite Raspberry Pi 12](#__RefHeading___Toc1874_177331566)

[Install drivers and test 12](#__RefHeading___Toc1876_177331566)

[Install Rhasspy 13](#__RefHeading___Toc1878_177331566)

[On your PC 13](#__RefHeading___Toc1880_177331566)

[Test microphone and speaker 16](#__RefHeading___Toc1882_177331566)

[Troubleshoot Rhasspy mic problems 18](#__RefHeading___Toc1217_1910159651)

[Install HermesLedControl and test 19](#__RefHeading___Toc1213_1910159651)

[debug 19](#__RefHeading___Toc1492_4004906539)

[Run Rhasspy as a service 21](#__RefHeading___Toc6991_518319112)

[Setting up Rhasspy voice Assistant 23](#__RefHeading___Toc801_4015176539)

[1 – telling Rhasspy what commands to expect 23](#__RefHeading___Toc1494_4004906539)

[Intent 24](#__RefHeading___Toc4437_4015176539)

[Groups 24](#__RefHeading___Toc4439_4015176539)

[Alternatives 24](#__RefHeading___Toc4441_4015176539)

[Optional words 24](#__RefHeading___Toc4443_4015176539)

[Tags / Entities 25](#__RefHeading___Toc4445_4015176539)

[2 - Getting the Intents from the Satellite to Host 26](#__RefHeading___Toc1496_4004906539)

[3 – Action the intents within Home Assistant 28](#__RefHeading___Toc1498_4004906539)

[And that’s it ! 28](#__RefHeading___Toc15573_2589493998)

[Next steps / history 29](#__RefHeading___Toc4447_4015176539)

[Change HLC patterns 29](#__RefHeading___Toc7574_518319112)

[Troubleshooting: Logs and places to check: 30](#__RefHeading___Toc6956_518319112)

[On the Satellite: 30](#__RefHeading___Toc6958_518319112)

[On the Base station: 30](#__RefHeading___Toc6960_518319112)

# General comments

* Rhasspy and HA seem to be designed to be all things to all people, and they appear to do it well, supporting an impressive number of hardware and software components. The down side is that there is no one correct path for a beginner to follow to a completed working system. Each hardware and software component has it’s own trade-offs.
* While Free/Open Source Software has some great advantages, being beginner-friendly (or even user-friendly) is rarely one of them. Developers are spending their own time unpaid (thank you so much guys), and quite reasonably they would rather be implementing the next great feature, than writing user-level documentation or beginners tutorials.
* Everything is in the documentation … but it can be hard to find the information you want; and often harder to understand what it means. Be prepared for a lot of learning curve.   
  In practice, there are quite a few concepts and technologies that a beginner has to get their head around, in order to get this project running.
* HA and Rhasspy currently benefit from an extremely active developer community; resulting in releases monthly - however this means that things do change fairly quickly – sometimes breaking an installation that was working.
* For small and cheap deployment as a dedicated Voice Assistant satellite, a RasPi Zero with a respeaker 2-mic (or equivalent) HAT is a popular combination; hence this tutorial.
* To make the most of the RasPi zero’s limited resources I want to avoid any unnecessary processing tasks – specifically the x-windows GUI and Docker – so I use Raspberry Pi OS Lite version (aka Debian install method).

# Terminology / concept overviews

I find it helpful to start by defining key terms which are used by Rhasspy and Hermes, both to make it easier to understand this document – but also to get you used to using the same terms which others should be using reduce confusion.

### PC

Home Assistant and Rhasspy are controlled through their web pages, so you will need a standard PC computer (running linux, MacOS or Windows) with a web browser. You will also need a terminal program (described below) and the Raspberry Pi Imager or Etcher to format and load onto a microSD card.

### Terminal / Secure Shell (SSH)

Normally a keyboard, mouse and monitor screen directly connected to the computer are used to enter commands and view the results. This is often called the *console* terminal.

Linux also provides for *remote terminal*s – a program running on another computer, which relays the users input and the host computers output, as though it is directly connected. SSH (Secure Shell) is the most common method for this, and is available in most remote access programs. SSH is available in the Windows, macOS and linux Terminal programs. There are many alternatives such as telnet and puTTY.

SCP (SecureCopy) is part of SSH, and uses SSH to copy files to the remote system which is running SSH.

### Home Assistant

An open source project providing a Home Automation system which is modular, expandable and runs locally – without reliance on cloud servers or external service providers.

Useful references: https://www.home-assistant.io/docs/

<https://community.home-assistant.io/>

### Rhasspy

A modular open source Voice assistant project, which aims to provide interface similar to Google Home, Alexa, Siri – but operates locally without sending any data out to servers in the cloud.

The Rhasspy program runs on both the Satellites and Base station - the difference is in which modules are called at each stage. The Satellites hear the voice, and supervise the whole process … but they will request the base station (which is running on a more powerful computer) to do the compute-heavy tasks, such as translating the audio recording to text, interpreting the intention from the text, and translating any response from text to an audio recording.

Rhasspy coordinates the various steps required of a voice assistant:

|  | Service | Description |
| --- | --- | --- |
|  | Audio Input | Listens to the microphone |
|  | Wake Word | Listens for the “wake” word - similar to “Alexa” or “hey Google” |
|  | Speech Recognition | Converts voice commands into text. Eg “Turn the bedroom light on” |
|  | Intent Recognition | Recognises the users intention (called intents and slots) from text. Eg intent=switch\_light, device=bedroom\_light, action=on |
|  | Intent Handle | Sends recognised intents to other software to be actioned. Eg home Assistant turns off a light |
|  | Text to Speech | Translates text to an audio file. |
|  | Audio Output | Plays audio through a speaker. Eg “Bedroom light is on” |

At each step there are a number of (mostly interchangeable) options, allowing the system to be highly customised.

Useful references: <https://rhasspy.readthedocs.io/en/latest/>

<https://community.rhasspy.org/>

### Mosquitto / MQTT

Mosquitto is an app which provides a MQTT (Message Queue Telemetry Transport) server, used to communicate messages between all sorts of applications and devices.

Rhasspy automatically installs Mosquitto MQTT broker for messaging between the Rhasspy modules running on that machine, called “Internal MQTT”. Because it is self-contained, it is set up with fewer settings; and is suitable where all the Rhasspy modules are running on the same machine.

However a single External MQTT is preferred for communication between multiple satellite machines and/or multiple applications (eg driving the LEDs). In our case we will use Home Assistant’s MQTT Broker Add-on.

Useful references: https://www.home-assistant.io/docs/mqtt/

Both Internal or External MQTT options are discussed in [https://rhasspy.readthedocs.io/en/latest/tutorials/#server-with-satellites](https://rhasspy.readthedocs.io/en/latest/tutorials/" \l "server-with-satellites).

### HermesLEDControl

The HermesLEDcontrol (HLC) module listens to the MQTT messages, and display various patterns on the reSpeaker LEDs to give visual feedback to the user.

Useful references: <https://github.com/project-alice-assistant/HermesLedControl/wiki>

### hostname

All computers are identified on a network by its unique I*P Address,* like 192.168.1.100. These numbers can be hard for humans to remember, so a method of allocating a name to each computer is used, called a *hostname*.

On a RasPi, the hostname given to the machine is defined in raspi-config. To access the RasPi, follow the hostname with “.local” (eg: http://raspberrypi.local).

### Base station computer

In Rhasspy, the Base station is a computer also running Rhasspy, which will provide those compute-intensive services as required to the Satellites.

I assume you already have HA running on a computer which will be used as your base station; and we will install the HA Rhasspy add-on.

### Satellite computer

A low-power computer, such as a Raspberry Pi Zero W with reSpeaker 2-mic HAT with microphone and speakers. Several satellite units can be located throughout the house, providing a voice assistant service.

#### Processor board

The RasPi Zero is cheap and small, with only one USB port and no wired Ethernet connector. Fortunately we will use it “headless” (without keyboard or screen connected) through it’s built-in WiFi – so these are not actually limitations.

A RasPi Zero is certainly usable, however its slower processing speed results in a noticeable delay before you can speak your command – so if you are purchasing a RasPi to be a satellite, I strongly recommend spending a little more to get one of the newer RasPi Zero W 2’s or a RasPi 3A+. These both have a faster processor and similar I/O for not much more cost. This tutorial is the same for these boards.

If you already have a spare RasPi 3B laying around, these also make a great satellite – though their extra hardware and cost is not needed for a satellite.

#### Sound cards

There are several variations of the 2-mic HAT which are equivalent, all using the seeed reSpeaker driver.

|  |  |  |
| --- | --- | --- |
| Microphone card | On-board Speaker | other |
|
| Seeed ReSpeaker 4-mic HAT | None. Will have to use RasPi’s | 12 LEDs |
| Seeed ReSpeaker 2-mic HAT | Mono JST and headphone | 3 LEDs, button |
| Adafriut Voice Bonnet | Left and right JST, and headphone | 3 LEDs, button |
|  |  |  |
| USB microphone and speaker | USB sound card has 3.5mm sockets for mic and headphones |  |

Note that while these are popular and convenient, the reSpeaker driver does not appear to take advantage of multiple microphones, so we do not automatically get better quality of audio. For voice assistant use, a simple USB sound card with speakers and microphone can also give similar results.

The [Raspberry Pi IQaudio Codec Zero](https://core-electronics.com.au/raspberry-pi-iqaudio-codec-zero.html) uses a different chip to the reSpeaker devices, and has a different driver. I have not used this board, so cannot comment on it.

There are other multi-microphone units (such as [https://wiki.seeedstudio.com/ReSpeaker-USB-Mic-Array](https://wiki.seeedstudio.com/ReSpeaker-USB-Mic-Array/)) with firmware providing features like Voice Activity Detection, Direction of Arrival, Beamforming, Noise Suppression, De-reverberation, Acoustic Echo Cancellation … but at a price.

#### Cases

Curiously I have only found a couple of 3D printer models for cases.

### Satellite Site-IDs

Rhasspy on the base station will only respond to MQTT messages from specified Satellites, so it is important that each Satellite is given a unique ID, and that all the Satellite IDs are listed at the Base station.

# Preparation

Rather than installing each component one at a time, I have chosen to minimise the swapping between devices, and so some things seem to be out of order. I suggest you tick off each step *as you do it*, so you don’t accidentally miss something that might not seem obvious.

There are several values which you can decide, which will be used throughout this document. I suggest you note down the vales for later reference:

* You will need to decide on unique Satellite IDs for the Base station and each Satellite. If you know where you’re going to place the satellite, you might use the location in the siteID, because, well it would be nice for Home Assistant to know that you’re speaking in the bedroom when you ask it to turn on the light ;-).   
  I called mine “base”, “sat-1”, “sat-2” because I have no imagination, and I was unsure which room I will end up putting them.
* Your values:   
    
   Base station ID: \_\_\_\_\_\_\_\_\_\_\_\_\_ Satellite(s): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* External MQTT requires a username and password. You can use an existing user, but I created a new user to be used by Rhasspy, which I called “rhasspy”. No imagination ;-)  
    
   Rhasspy MQTT username: \_\_\_\_\_\_\_\_\_\_\_ password: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* What do you wish to use as name (optional) and/or IP addresses for your machines:  
    
   base station hostname: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ IP: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
    
   Satellite hostname: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ IP: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
    
   Satellite hostname: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ IP: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Setup Base station

If you already have a base station with one or more satellites, you can skip this section … otherwise let us set up your base station.

You do already have a computer running HA to use as the Rhasspy base station, right ?

## MQTT Broker

External MQTT will need the Satellites to log into the MQTT Broker, by giving a username and password. You can use one of the existing HA users, but I chose to create a new user “rhasspy” to help distinguish between Rhasspy-related messages and messages from other users. .

In HA, select Configuration > People & Zones > Users, then click the [+ Add User] button, and add your new rhasspy user details.

In Home Assistant, select Configuration > Devices & Services > Integrations and click the [+ Add Integration] button. Select the MQTT integration and install.

Leave all the default values. Note that port number 1883 is used for normal MQTT. Do **not** add any user details into the logins section, as this will interfere with the built-in “homeassistant” user.

## Install Rhasspy Add-on to Home Assistant

If you are using Home Assistant OS and have not yet installed Rhasspy, do so now.

In Home Assistant, select Configuration > Add-On, Backup & Supervisor > Add-ons page, click the [Add-on store] button.

In the Add-On Store page, click the 3 dots in top right corner, select “Repositories” and add “<https://github.com/synesthesiam/hassio-addons>”. Refresh the web page, and you should now see a section titled “Synesthesiam Hass.IO Add-Ons” at the bottom of the page, from which you can select “Rhasspy Assistant”.

In the “Configuration” tab you can change the default language by changing the “en” under “profile name” to “fr”, “de”, or whichever language you wish. Scrolling down note the “Network Rhasspy web IU + API” is at port 12101. A base station probably doesn’t have microphone and speakers connected, and so the Audio Input and Output devices will have the value “Default”.

If you have changed any of the settings (probably the language), click [SAVE] and restart HA.

After restarting HA and starting Rhasspy, click on the [Log] tab and you will notice a lot of [DEBUBG] messages. You can click on [REFRESH] at the bottom to update the log display. After a while (could be a couple of minutes) messages will be added to the Log to indicate that the Rhasspy is ready to use:

[INFO:2021-10-16 23:04:51,033] rhasspyserver\_hermes: Started

[DEBUG:2021-10-16 23:04:51,034] rhasspyserver\_hermes: Starting web server at http://0.0.0.0:12101

Running on 0.0.0.0:12101 over http (CTRL + C to quit)

Click on the [Info] tab and then on the [OPEN WEB UI] button to configure Rhasspy. Alternatively you can simply browse to port 12101 of your Base station (e.g. [http://192.168.1.98:12101](http://192.168.1.98:12101/) ).

## Configure Rhasspy Base

In Rhasspy on the Base station (browse to port 12101 of your Base station e.g. [http://homeassistant.local:12101](http://192.168.1.98:12101/) ):

* Select Settings from the menu, or by clicking the cog wheels on the left of the page.
* set MQTT to “External” and entered the hostname or IP address of the computer which is running the Mosquitto server (my base station) and the 1883 port number from the previous step.
* Set Speech to Text to “Kaldi”, and add your Satellite SiteIds:
* Intent Recognition to “Fsticuffs”, and add your Satellite SiteIds: at the bottom of the section.
* Text to Speech to “Larynx”, and add your Satellite SiteIds: at the bottom of the section.
* Dialogue Management to “Rhasspy”, and add your Satellite SiteIds.
* Audio Recording, Wake Word, Audio Playing and Intent Handling should all be set to “Disabled” on the base station.

Note that Satellite Ids are important ! MQTT may be handling messages from/to lots of other modules, so this tells Rhasspy base which messages to listen to. A message with any other Satellite Id will be ignored.

We will come back to the Base station later, to setup the sentences, intents; and actions to be performed.

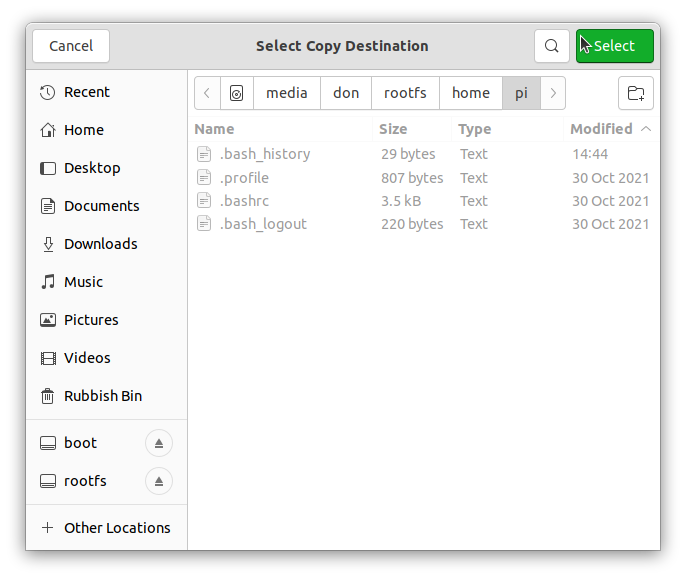
# Setup Satellite

IMPORTANT: every linux kernel update can break respeaker drivers and/or the LED control, so DO NOT perform updates once you have the mic working. In fact I will change this section to download a pre-configured IMG with a known working combination of components -=- albeit for a very specific hardware environment :-(

Refer: <https://community.rhasspy.org/t/how-im-integrating-between-rhasspy-and-home-assistant/3090/6>

## On your PC

I have not found a simple link to download the installation file direct to the RasPi, so we will download the latest versions of the programs we will be using to your main desktop computer, and then copy to your Satellite.

1. On your desktop computer browse to [https://rhasspy.readthedocs.io/en/latest/installation/#debian](https://rhasspy.readthedocs.io/en/latest/installation/" \l "debian), and click on “armel” in the line  
    [armel](https://github.com/rhasspy/rhasspy/releases/latest/download/rhasspy_armel.deb) - Raspberry Pi 0, 1
2. Download the Raspberry Pi Imager from <https://www.raspberrypi.org/software/operating-systems/>
3. Run Raspi-imager, select “Raspberry Pi OS Lite (32-bit)”; click [Storage] button and choose your SD card.
4. Press <Ctrl-Shift-X> to pop-up Advances options dialog box. Set hostname; enable SSH and set password for ‘pi’ user; configure Wi-Fi SSID, password and wifi country; and save.
5. Click the [Write] button.
6. When the file has finished copying, copy the rhasspy\_armel.deb file downloaded in step 1 from your PCs Downloads directory to the /home/pi/ directory on the rootfs partition of the microSD card.
7. Eject the boot or rootfs partitions.

## On your Satellite Raspberry Pi

1. Place the microSD card in your RaspPI and turn the power on. Be patient because the first time it takes a couple of minutes for the RasPi Zero to start.
2. If you have HDMI attached you will see many messages scroll past and a reboot – this is normal. Watch out for the scrolling to stop with a “raspberrypi login:“ message.   
     
   If you have no monitor and keyboard connected, use the terminal program on your PC. Connect by SSH to the hostname or IP address of your Satellite machine. You may need to follow instructions at [https://www.raspberrypi.org/documentation/computers/remote-access.html#how-to-find-your-ip-address](https://www.raspberrypi.org/documentation/computers/remote-access.html" \l "how-to-find-your-ip-address)  
   Note that it can take several minutes for the first power on before the RasPi can accept SSH connections.

### Install drivers and test

1. Turn off wi-fi power saving to reduce drop-outs  
    sudo iw dev wlan0 set power\_save off
2. Seeed stopped updating their driver a couple of years ago, so install HinTak’s updated respeaker driver

sudo apt install git  
git clone <https://github.com/Hintak/seeed-voicecard.git>  
cd seeed-voicecard  
sudo ./install.sh

This will take several minutes, and may include reinstalling parts of the OS kernel. It will probably end with the instruction displayed tor reboot the Satellite machine.

sudo reboot

1. We can perform a test of the mic and speakerarecord -f cd -Dhw:1 test.wav  
   say a few words, then press [Ctrl-C]. Now test withaplay -Dhw:1 test.wav

If this test doesn’t work, do arecord -l and aplay -l, and check whether there is a device “1”. Device numbers can change with each reboot if you add or remove a device (eg a HDMI monitor that can play sound).   
Change device number as necessary and try again.

1. We can adjust the audio parameters  
    alsamixer  
   Press [F6] and change to the seeed sound card.  
   Press [F5] to view both microphone and speaker controls (since reSpeaker 2-mic has both on the same card)  
   use the keyboard arrow keys to alter every setting which shows in the red down to approx 60-70  
   Press [Esc] to exit  
   In you wish, you can go back to step 5) and test again.
2. Test the LEDs

git clone <https://github.com/respeaker/mic_hat.git>  
cd mic\_hat  
python3 interfaces/pixels.py

The three LEDS should turn on and perform a test pattern. Press [Ctrl-C] to stop the program.

1. You can also test the button by

python3 interfaces/button.py

The program checks the button regularly and prints “on” if the button is pressed and “off” if the button is not pressed. Type [Ctrl-C] to stop the program.

### Install Rhasspy

1. \*\*\* temporary fix (refer <https://github.com/rhasspy/rhasspy/issues/259>)

sudo apt-get install libffi6

1. install Rhasspy,

cd ~  
sudo apt install ./rhasspy\_armel.deb

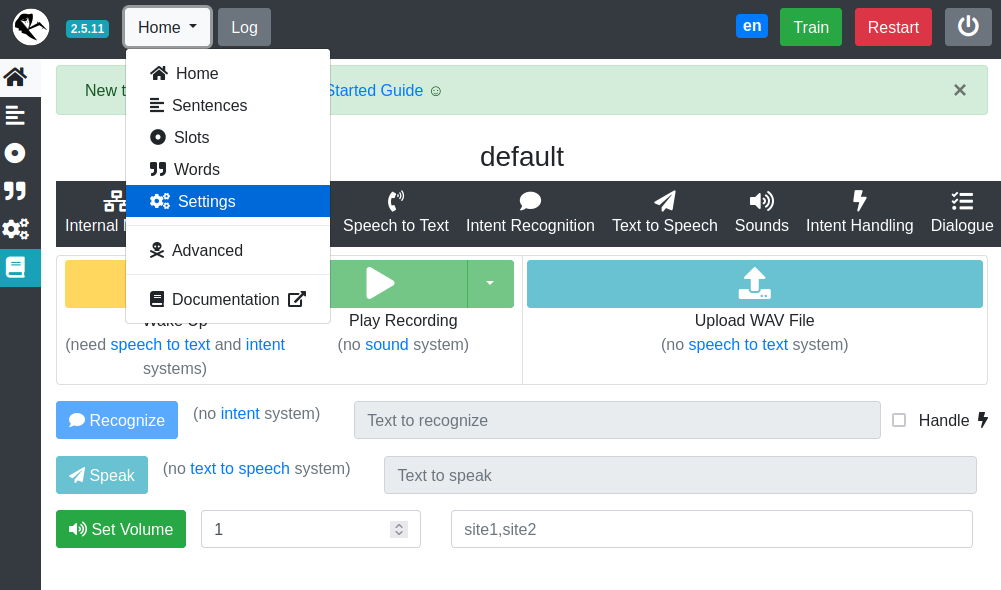
1. configure and test. Note that the –Profile en indicates English language to be used, and should be changed to suit yourself, eg “fr” for French.

rhasspy --profile en

You should see a load of messages (most starting “DEBUG:”) until after a minute or so ...

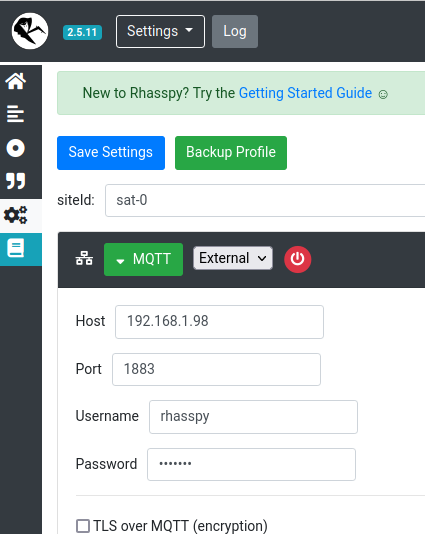
Running on 0.0.0.0:12101 over http (CTRL + C to quit)

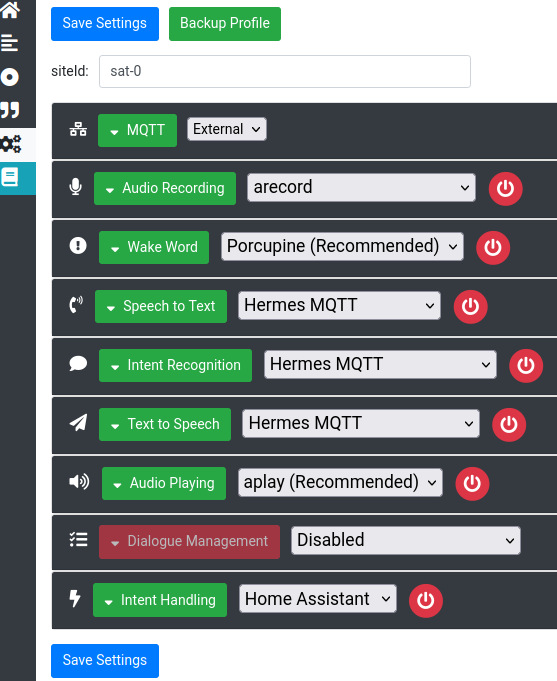
From here on there will be an awful lot of messages coming up on the Satellite’s terminal – most starting “[DEBUG:”



#### On your PC

1. Back on your desktop computer, open your web browser and visit http://<IP\_ADDRESS>:12101 where <IP\_ADDRESS> is the hostname or IP address of your Satellite RasPi. You should see the Rhasspy web interface.
2. Either click on the cogs icon on the left of the screen or select “Settings” from the menu.



1. Change the “siteId:” from **default** to something meaningful. Each satellite needs a unique siteID to allow the shared MQTT server to know where a request has come from, and hence where the result should be returned to.
2. Next to the black [MQTT] button select “External”, then click on the [MQTT] button (which has now turned green) to display the MQTT options.
3. In the Host field, enter the machine name or IP address of the Base system, and check that the Port number matches the MQTT port number set on the Base station.
4. Click on the blue [Save Settings] button.   
     
   A window will pop-up confirming “Settings saved” and asking whether to “Restart Rhasspy ?” Click [OK] to each of these restart requests in this section.
5. To get started, enable the following services and click "Save Settings":

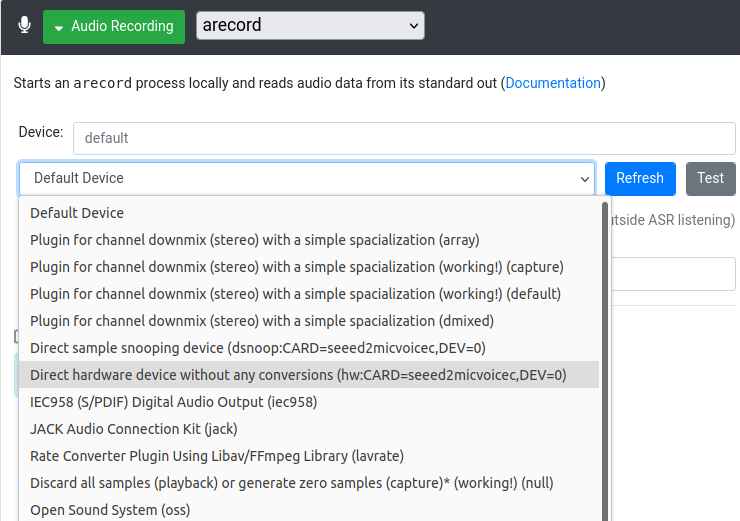
Troubleshooting: The worst case in these setup steps is that Rhasspy does not boot properly. If so, using Terminal on the Rhasspy Satellite computer you can press [Ctrl-C] to stop Rhasspy, enter

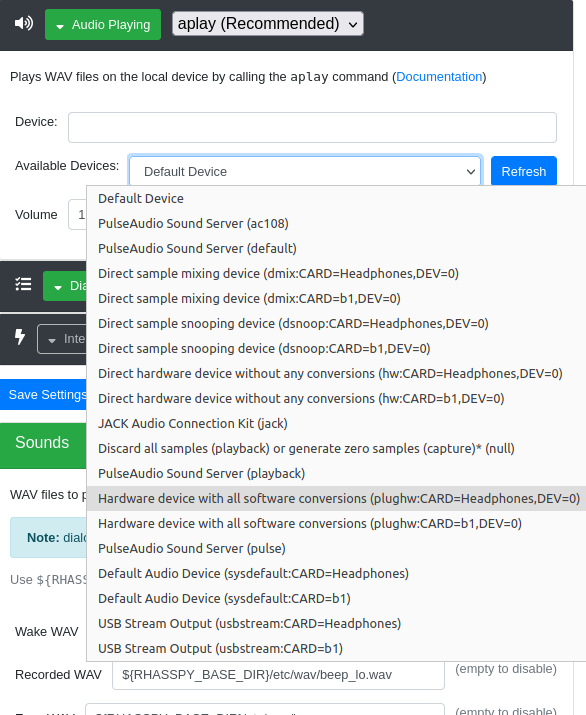
rm /home/pi/.config/rhasspy/profiles/en/profile.json

then go back to step 11)

* Audio Recording (arecord)
* Wake Word (Porcupine)
* Speech to Text (Hermes MQTT)
* Intent Recognition (Hermes MQTT)
* Text to Speech (Hermes MQTT)
* Audio Playing (aplay)
* Leave Dialogue Management Disabled – we do not need it on the satellite.
* Intent Handling (Home Assistant)

1. Click [Save Settings] and confirm to Restart Rhasspy.



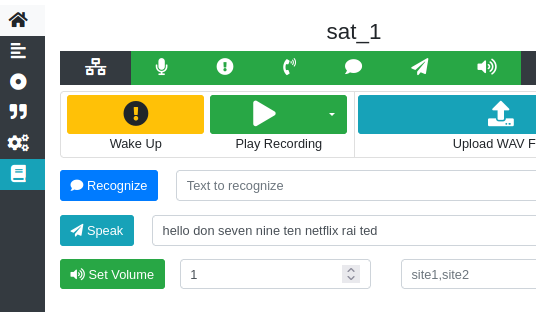
1. Click the green [Audio Recording] button to show the microphone controls.
2. Click the grey [Test] button to get Rhasspy to test each device to guess whether it’s working or not.   
     
   Click the blue "Refresh" button; then choose the “hw:CARD=seed2micvoice”option from the Device drop-down list.   
     
   Note the “(working!)” next to those options which Rasspy think are working.
3. For “port”enter 12203, and in the Output siteId, enter the SiteId value you entered at step 2 above.   
   You can tick the “Audio Statistics” box to show values … which should change in real time as you speak.
4. Click on the green [Wake Word] button to show the options, and in “UDP Audio (Input)” enter 12203.
5. If you go down to the green [Audio Playing] button you should find that the drop-down list of Available Devices has also been populated.   
     
   Select one, then click the blue [Save Settings] button and restart Rhasspy.
6. click the blue [Save Settings] button and restart Rhasspy.

#### Test microphone and speaker

This is important – there’s no point proceeding until Rhasspy can hear and respond to us.

1. Go back to the Home page (icon of a house on left of screen, or “Home” in the menu).   
     
   Speak the word “porcupine”. Hopefully you will hear a chirp sound from the headphones or speaker to indicate that (a) the microphone has heard you, (b) the “Porcupine” Wake Word module has recognised its name, and (c) the speaker has played a sound.   
     
   If you already have your Rhasspy Base fully configured (i.e. this isn’t your first Satellite), you can try giving Porcupine a command now. With luck you should see the command recognised.   
     
   All good, and you can proceed to the next section.

Not the end if it didn’t all “just work”. Now we test speaker and microphone separately.

1. Click the yellow [Wake Up] button to start Rhasspy recording a possible command.   
     
   You should hear a chirp from the speaker or headphones, and see a pop-up telling you that Rhasspy is “Listening for Command”. The chirp indicates the Audio Playing device is working – so if you didn’t hear the chirp, you may need to go back to step 20) and select a different Audio Playing device.
2. You can also test the speaker by typing a phrase onto the text box next to the [Speak] button, then clicking [Speak].
3. Testing the microphone is not so obvious. to test the mic, just wake up the system and talk. Even if no intent is recognized, the play button next to the wake up button should play it, and if the speakers don’t work, you can always download it and play it on your pc.
4. Mic testing via audio input statistics: If you are not interested in how the mic sounds and just want to know if it works at all, there is an audio statistic button in the audio recording section of the settings. Just activate and look if it will turn out numbers, if so, it has some kind of sound, it might be just noise thought. If you speak or otherwise make sounds near the mic the numbers should reflect that, if they do, your mic is recording your environment. It might do so perfectly clear, or with lots of noise, this is not the way to tell but that is what the main gui is for.
5. After the chirp, the microphone listens until it hears 30 seconds of silence (assuming you have stopped talking).

Try swapping between the “hw:” and “plughw:” versions.

Expand the "Audio Recording" section by clicking the green button. You should see a drop down list with available microphones. If you have trouble recording audio, try choosing a specific device instead of using the default (make sure to "Save Settings").

Clicking the blue "Refresh" button will query PyAudio again for this list. The "Test" button next to "Refresh" will attempt to record audio from each device and guess if it's working or not. The text "working!" will show up next to working microphones in the list.

If you continue to have problems with speakers or microphone, stop now and see the appendix at the end of this document: Troubleshoot Rhasspy mic problems

Manually add “Port: 1883” to mqtt section in Rhasspys ~/.config/rhasspy/profiles/en/profile.json

"mqtt": {

"enabled": "true",

"host": "192.168.1.98",

"port": "1883",

"username": "rhasspy"

"password": "redated",

"site\_id": "sat-2",

},

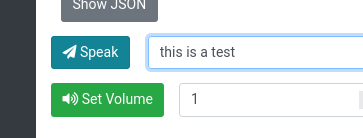
Change [Intent Handling] to “Remote HTTP”

### Troubleshoot Rhasspy mic problems

Assuming that :

1. machine has microphone and speakers connected
2. drivers installed and tested at the OS (eg on linux by by arecord | aplay)
3. installed Rhasspy, and configured with
   1. MQTT = internal, Audio Recording=arecord and Audio Playing=aplay

Problems fall into several main categories:

1. speaker not working.
   * most easily checked from OS by using arecord and aplay, and checking alsamixer settings.
   * This is easily checked on the satellite’s web page by entering some text and clicking the [Speak] button.
   * Check that speaker is plugged in and powered on (if appropriate).
   * Check that devices are still correct. Note that connecting or disconnecting an HDMI monitor can change the speaker device numbering.
   * Try changing between the hw: and plughw: devices ???
2. wakeword not recognised. When you speak the wakeword, no beep is heard.
   * Try again in a minute. If using a RasPi Zero, make that 3-5 minutes.
3. words not recognised is province of the “Speech to Text” settings. This can be performed on the satellite, or passed by MQTT or remote HTTP to a another instance of Rhasspy
   * Check the syntax of the Sentences.ini file (via the Rhasspy Sentences menu option)

## Install HermesLedControl and test

On the Satellite Terminal…

1. If rhasspy is still running, press <CTRL-C> to stop Rhasspy. Press <Enter> key when you see the “Shutting down core” message.
2. Download the automatic downloading tool. Do not use master unless you know what you are doing!

wget https://gist.githubusercontent.com/Psychokiller1888/a9826f92c5a3c5d03f34d182fda1ce4c/raw/cbb53252dd55dc4e9f5f6064a493f0981cf133fb/hlc\_download.sh

Run it

1. enter the following commands:

sudo chmod +x hlc\_download.sh

sudo ./hlc\_download.sh

Sudo is required to install as we download the missing packages and create a log directory

latest RasPi OS comes with python 3.9, so another work-around is required.

Other work-around required. Other tutorials are required to follow.

* https://kiboost.github.io/jeedom\_docs/other/Rhasspy/HermesLedControl/

You will be asked a series of questions. Accept the default paths, choose the relevant multi-choice options. Choose any pattern (we can change later). Do NOT enable DoA.

|  |  |
| --- | --- |
| What assistant engine are you using? | 2) rhasspy |
| What's the path to your assistant config file? Path: (/.config/rhasspy/profiles/en/profile.json) |  |
| What device do you wish to control with SLC? | 2) respeaker42icArray |
| What pattern do you want to use? | 1) google |
| Where should the configuration be saved to? Path: /home/don/.config/hermesLedControl |  |
| Do you want to enable DoA now? | 2) no |

Ideally (on the RasPi Zero, because it is much slower to boot), we want to see:

1. When Rhasspy is running (like turning on the Power LED)
2. When wakeword is detected, and Rhasspy is listening
3. Intent is NOT recognised
4. Intent recognised (and being passed to Home Assistant to action)

add "import logging" to the top of /home/pi/hermesLedControl\_v2.0.15/models/LedPattern.py

sudo systemctl daemon-reload

sudo systemctl enable hermesledcontrol

### debug

Start the service: sudo systemctl start hermesledcontrol  
Stop the service: sudo systemctl stop hermesledcontrol  
Restart the service: sudo systemctl restart hermesledcontrol  
Show service log: journalctl -u hermesledcontrol.service

You can change the pattern by stopping the service, editing and starting the service

sudo nano /etc/systemd/system/hermesledcontrol.service

ExecStart=/home/pi/hermesLedControl\_v2.0.3/venv/bin/python3 main.py --engine=rhasspy --pathToConfig=/home/pi/.config/rhasspy/profiles/fr/profile.json --hardware=respeaker2 --pattern=kiboost

## Run Rhasspy as a service

This is great, but currently if we reboot the Satellite (or the power goes off) we have to connect to the satellite through SSH and manually re-start rhasspy. Instead we can run Rhasspy as a service.

There is an issue with running as a service - rhasspy’s shutdown and reboot operations require us to enter our sudo password at the terminal ... which no longer exists. The satellite cannot shutdown cleanly.

Given that this is a dedicated voice assistant satellite and not connected to the interweb, I can probably put up with authorising the user pi to have root privileges, so that the sudo password will not be required.

sudo visudo

at end of the file, paste

pi ALL = NOPASSWD: ALL

2. setup the service, by

sudo nano /etc/systemd/system/rhasspy.service

[Unit]

Description=Rhasspy Autostart

After=network-online.target

[Service]

Type=simple

User=pi

WorkingDirectory=/home/pi

ExecStart=/bin/bash -lc '/usr/bin/rhasspy --profile en 2>&1 | cat'

StandardOutput=syslog

StandardError=syslog

SyslogIdentifier=rhasspy

Restart=on-failure

RestartSec=30

[Install]

WantedBy=multi-user.target

sudo systemctl enable rhasspy

sudo systemctl daemon-reload

sudo systemctl start rhasspy

To check for errors (display end of the log for the rhasspy.service):

journalctl -e -u rhasspy.service

3. Linux system services handle stdout differently from command line. If we had used "ExecStart=rhasspy --profile en" above, we would have got "spawnerr: unknown error making dispatchers for 'microphone': ENXIO" error. See `[Sherlock - Offline Voice Assistant project](https://ip-team4.intia.de/pages/knowledge/systemd.html)` for a description

And, so far, this seems to be working for me.

--------- alternatives -----------------

Here is my file /lib/systemd/system/rhasspy.service

[Unit]

Description=Rhasspy Service

After=syslog.target network.target

[Service]

Type=simple

ExecStart=/bin/bash -c '/usr/bin/rhasspy --profile fr 2>&1 | cat'

RestartSec=1

Restart=on-failure

StandardOutput=syslog

StandardError=syslog

SyslogIdentifier=rhasspy

[Install]

WantedBy=multi-user.target

------------------

In https://community.rhasspy.org/t/rhasspy-as-a-service-without-docker/3174/2

imo running any service with root privileges is not recommended, so here’s my rhasspy.service used on a headless debian (x64) machine:

[Unit]

Description=Rhasspy Service

After=syslog.target network.target mosquitto.service

[Service]

Type=simple

# for command, see https://github.com/rhasspy/rhasspy/issues/42#issuecomment-711472505

ExecStart=/bin/bash -c 'rhasspy -p de --user-profiles /opt/rhasspy/profiles 2>&1 | cat'

WorkingDirectory=/opt/rhasspy

User=rhasspy

Group=audio

RestartSec=10

Restart=on-failure

StandardOutput=syslog

StandardError=syslog

SyslogIdentifier=rhasspy

[Install]

WantedBy=multi-user.target

User rhasspy and working dir have to be added manually, obviously, and user rhasspy is added in group audio (despite the fact no direct audio hardware is used).  
Suggestions to improve it are welcome, as this also is some kind of copy/paste solution…

# Setting up Rhasspy voice Assistant

You should have:

* Rhasspy installed as a Base station (on a RasPi with Home Assistant and MQTT)
* Rhasspy installed as a satellite
* both installed with External MQTT
* have tested that words spoken at the satellite are recognised (converted to text)
* for this tutorial I have selected Fsticuffs for [Intent Recognition] in Rhasspy on the Base station. Fsticuffs will recognize only those sentences that Rhasspy was [trained on](http://homeassistant.local:12101/docs/training/" \l "sentencesini), and is documented at [http://homeassistant.local:12101/docs/intent-recognition/#fsticuffs](http://homeassistant.local:12101/docs/intent-recognition/" \l "fsticuffs). Other modules are available, which will work differently.

Before we get stuck in, lets explain some of the terminology used by Rhasspy.

|  |  |
| --- | --- |
| term | My explanation ;-) |
| intent | A command to be performed by HA |
| slots | variable |
| alternatives |  |
| fuzzy | Voice recognition isn’t 100% perfect. Often the result is “I’m 85% sure you said ‘Turn on the kitchen light’” |

TLDR;

* sentences.ini in Rhasspy on the base station tells rhasspy the syntax of the commands you will give (what words and in what order). It identifies the name of the *intent* and any *tags* (parameters)
* In HA’s configuration.yaml you add the matching intent name and the instructions to be performed
* Since the satellites are all set to “Hermes MQTT”, training only needs to be done on the Base machine.

## 1 – telling Rhasspy what commands to expect

First step is telling Rhasspy what sentences to expect (i.e. what commands you want to give). For this, in Rhasspy on the Base station machine, select “Sentences”. You will see some initial default sentences or some built-in intents (commands):

### Intent

Each *intent* is indicated by the square brackets starting in column 1 (e.g. [GetTime]).

It is followed by one or more voice commands that you might say. For instance, to command Rhasspy to say the time you might say “What time is it” or “Tell me the time”.

The intent name is sent to HA to tell it which command to perform.

### Groups

You can *group* multiple words together using (parentheses) like:

turn on the (living room lamp)

Groups (sometimes called sequences) can be tagged and substituted like single words. They may also contain [alternatives](https://rhasspy.readthedocs.io/en/latest/training/" \l "alternatives).

### Alternatives

A set of items where only one is matched at a time is (specified | like | this). For example, GetTemperature will match “how hot is it” or “how cold is it”.

### Optional words

Within a sentence template, you can specify *optional* word(s) by surrounding them [with brackets]. For example:

[an] example sentence [with] some optional words

will match:

* an example sentence with some optional words
* example sentence with some optional words
* an example sentence some optional words
* example sentence some optional words

### Tags / Entities

Named entities are marked in your sentence templates with {tags}. The name of the {entity} is between the curly braces, while the (value of the){entity} comes immediately before:

[SetLightColor]

set the light to (red | green | blue){color}

Speaking “set the light to blue” will pass to HA the intent name [SetLightColour] with the entity {color} containing the value “blue”.

A more complex example would be:

sentences.ini:

[ChangeLightState]

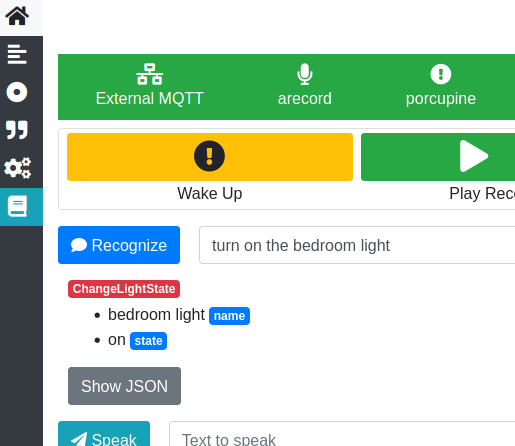
light\_name = (living room light | study light | bedroom light) {name}

light\_state = (on | off) {state}

turn <light\_state> [the] <light\_name>

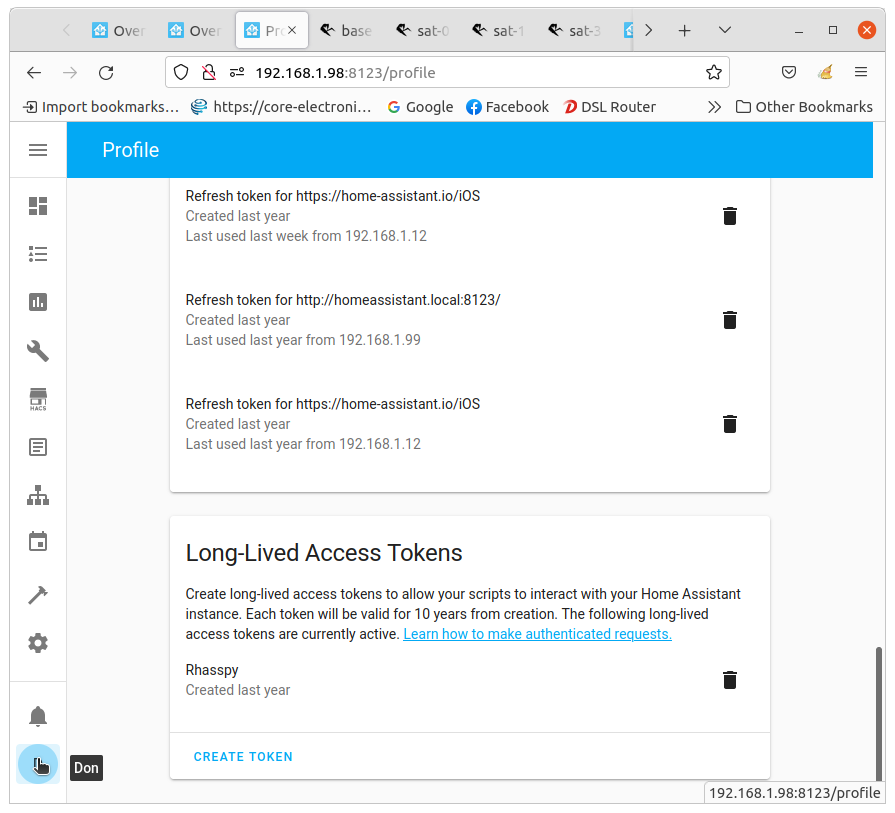
turn [the] <light\_name> <light\_state>

Speaking “turn on the bedroom light” will display on the Rhasspy Home screen of both the satellite and base station:

Rhasspy determines that the “ChangeLightState” intent is to be called with slot “name” having a value of “bedroom light” and slot “value” having a value of “on”.

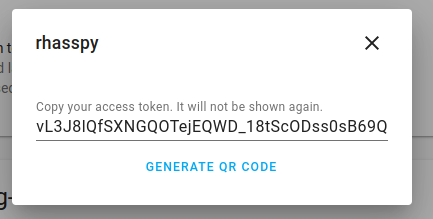
Important: Every time you make changes to the sentnces.ini you will have to [Save Sentences] and then click the green [train] button to update – before you can test the change. Yes this gets a bit tedious – but nowhere near as bad as having to make the same change on each Satellite.

## 2 - Getting the Intents from the Satellite to Host

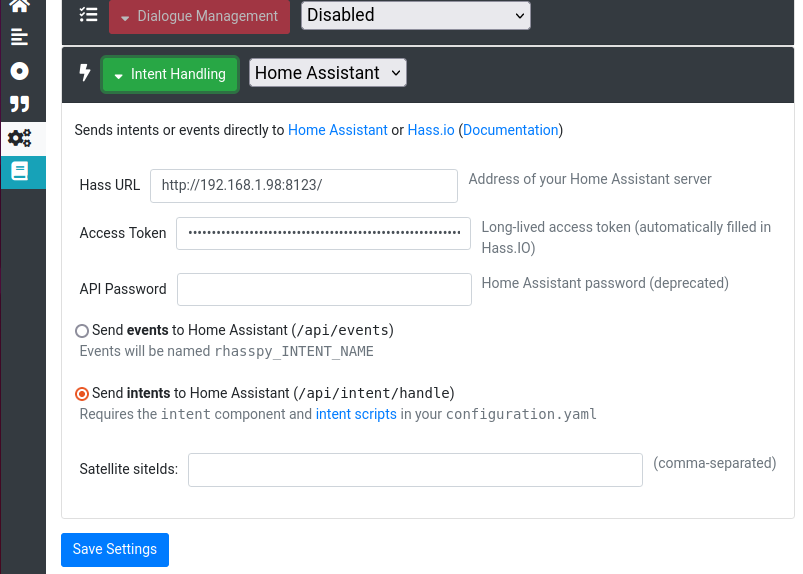


You may need to use a Long-lived Access Token. This is done at the very bottom of your profile page in Home Assistant

Click [Create Token], give it a name (I called mine “Rhasspy”) and a long sequence of characters (well beyond the left and right sides of the window) will be shown.



Click your mouse anywhere in the line of code, press the [Home] key to move the cursor to the start of the code, press [Shift-End] to highlight to the end of the code, and [Ctrl-C] to copy the code to clipboard.

In Rhasspy on the Satellite, set [Intent Handling] to “Home Assistant”, [Save], add the URL for Home Assistant – the “http://” protocol, hostname or IP address of the Home Assistant machine, and port number :8123.

Go to the browser window with your Satellite and paste that long code above into the “Access Token” field and click [Save Settings].

I am using intents in HA, so selected the “send intents”… option.

OK, to be more specific, Rhasspy calls the HASS URL and passes the JSON file which was generated in the first step.

## 3 – Action the intents within Home Assistant

Now to Home Assistant and we will add the intents into configuration.yaml so that Home Assistant will action them.

There are several ways to edit the .yaml files, and I have installed the “File editor” Add-on from Configuration > Add-ons.

I have added into the configuration.yaml page:

intent:

intent\_script: !include intents.yaml

The keywords “intent:” and “intent\_script:” are required. In my case I have chosen to place the actual intents in a separate file to make them easier to edit; but you can simply list the intents directly after “intent\_script:”.

In my new intents.yaml file I have:

#

# intents.yaml - the actions to be performed by Rhasspy voice commands

#

GetTime:

speech:

text: The current time is {{ now().strftime("%H %M") }}

LightState:

speech:

text: Turning {{ name }} {{state }}

action:

- service: light.turn\_{{ state }}

target:

entity\_id: light.{{ name | replace(" ","\_") }}

## And that’s it !

When the words “Porcupine, turn on the study light” are spoken, Rhasspy determines that the “LightState” intent is to be called with slot “name” having a value of “study light” and slot “value” having a value of “on” … as seen in the first image above.

This intent is passed (in a JSON file) to Home Assistant, where the intent name is looked up in the configuration.yaml (or intents.yaml) file.

Intent name “LightState” matches with “LightState:”, and parameters are substituted, so that Home Assistant effectively runs:

LightState:

speech:

text: Turning study light on

action:

- service: light.turn\_on

target:

entity\_id: light.study\_light

# Next steps / history

Having got it running, make a backup. Make two – a backup of your configuration, but also copy the microSD card onto another microSD card. Yes. Memory cards don’t have moving parts, but they do still go faulty over time. If your satellite starts playing up (especially if you haven’t changed anything at the base station), simply swap microSD cards and check if the problem is resolved.

WARNING: every update of Operating System kernel can break respeaker drivers. For best user experience I would suggest you ask users to exclude kernel updates in their package manager once they have the mic working; and only undo this if they want to upgrade the kernel and have either checked if their branch is still compatible or they uninstalled it before the upgrade to install the new branch.

## Change HLC patterns

# Troubleshooting: Logs and places to check:

### On the Satellite:

* + Console log (if running rhasspy from the command line)
  + From the Rhasspy web page, click on the grey [Log] button on the black line at top of the web page. Note that this has latest entry at the top – and is only a small subset of the console messages.

### On the Base station:

1. home Assistant not actioning the intent.
   * HA Config Logs shows HA errors. The full log at the bottom may provide more detail.