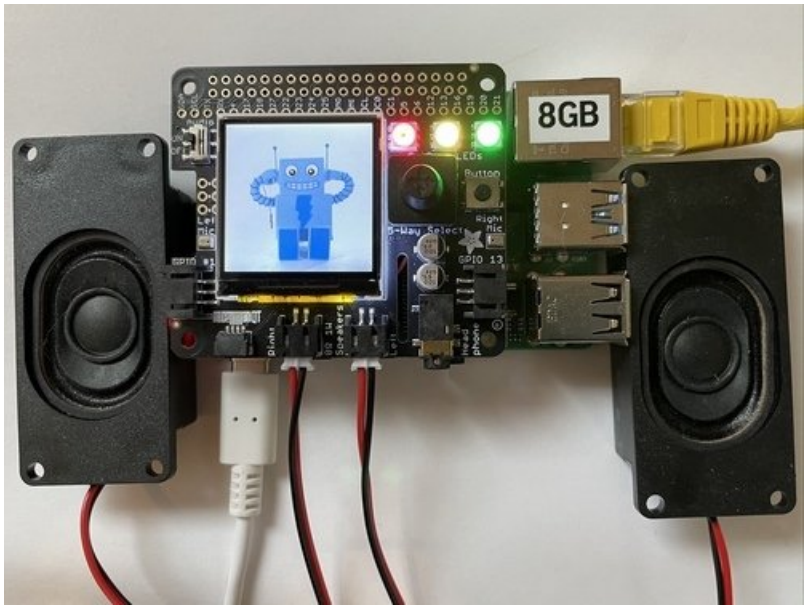
adafruit learning system

Python Edge Speech Recognition with Voice2JSON

Created by Melissa LeBlanc-Williams



Last updated on 2021-08-04 02:28:54 PM EDT

Guide Contents

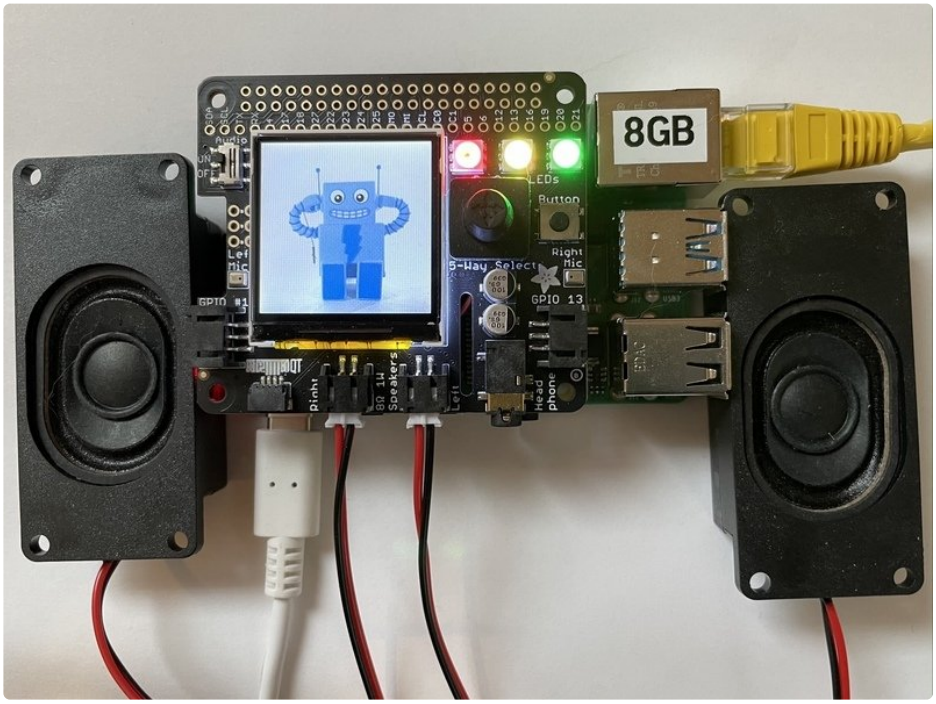
Guide Contents 2
Overview 3
- Parts 3

Raspberry Pi Setup 5
- Set your Timezone 5
- Install Voice2JSON 6
- Speech Synthesis Library 7
- Install a Profile 8
- Latest Pillow Library 8
- CircuitPython Libraries 9

Configuring Custom Commands 10
- Train the Profile 10

Demo Code 12
- Use 14
- Demo Code Walkthrough 14

Overview



Many of the reliable speech recognition systems today such as Amazon Alexa or Google assistant connect to the internet and remote servers to process the speech data. However, with Voice2JSON (https://adafru.it/Tcn), you can have your speech recognition data processed right on your Raspberry Pi This is called edge detection.

Voice2JSON (https://adafru.it/Tcn) works on top of other existing speech recognition dictionaries. This guide will be using a profile based on PocketSphinx because of its large vocabulary. PocketSphinx is maintained by Carnegie Mellon University.

Voice2JSON (https://adafru.it/Tcn) is a speech recognition tool for listening to speech and translating that to an intent. This allows your program to easily respond to the intent instead of worrying about the specific syntax of what was spoken. For instance, if somebody says "set light to green" instead of "change light to green" then it will still work.

While Voice2JSON (https://adafru.it/Tcn) was not designed to be run in Python, this guide does it anyways by launching the executable with subprocess commands and monitoring the JSON output. By leveraging Blinka, the CircuitPython compatibility layer, some powerful things can be done with the Adafruit BrainCraft HAT.

Parts

Your browser does not support the video tag.
Adafruit BrainCraft HAT - Machine Learning for Raspberry Pi 4

The idea behind the BrainCraft HAT is that you'd be able to "craft brains" for Machine Learning on the EDGE, with Microcontrollers & Microcomputers. On ASK...

Out of Stock

Out of Stock

Raspberry Pi 4 Model B - 2 GB RAM

The Raspberry Pi 4 Model B is the newest Raspberry Pi computer made, and the Pi Foundation knows you can always make a good thing better! And what could make the Pi 4 better...

Out of Stock

Out of Stock

Stereo Enclosed Speaker Set - 3W 4 Ohm

Listen up! This set of two 2.8" x 1.2" speakers are the perfect addition to any audio project where you need 4 ohm impedance and 3W or less of power. We particularly like...

$7.50

In Stock

Add to Cart

Official Raspberry Pi Power Supply 5.1V 3A with USB C

The official Raspberry Pi USB-C power supply is here! And of course, we have 'em in classic Adafruit black! Super fast with just the right amount of cable length to get your Pi 4...

$7.95

In Stock

Add to Cart

Raspberry Pi Setup

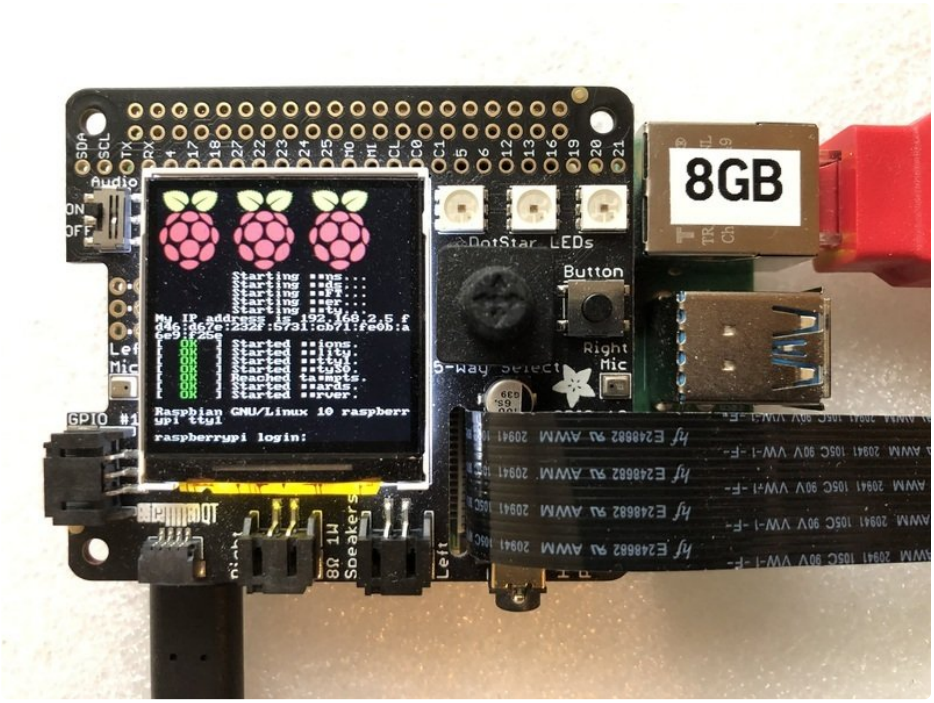
> Make sure you are using the Lite version of Raspberry Pi OS. The desktop version has had some issues with Google Voice and the audio driver.

First to setup all of the packages on the Raspberry Pi. If you haven't done so already, take a look at the Adafruit BrainCraft HAT - Easy Machine Learning for Raspberry Pi (https://adafru.it/NLE) guide if you are using the BrainCraft HAT.

This will take you through all the steps needed to get the Raspberry Pi updated and the BrainCraft HAT all set up to the point needed to continue. However, skip the display Module Setup portion since you will be using Python to draw to the display.

> Skip the Display Driver installation for now so you can control this through Python. If you already have it installed, you can run it without parameters and choose the Uninstall option to remove it.

Be sure you have some speakers hooked up to the BrainCraft HAT, either through the JST ports on the front or the headphone jack. You will need these later for the speech synthesis.

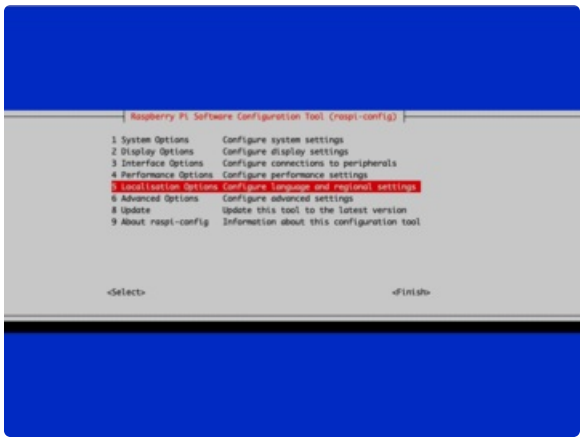


Set your Timezone

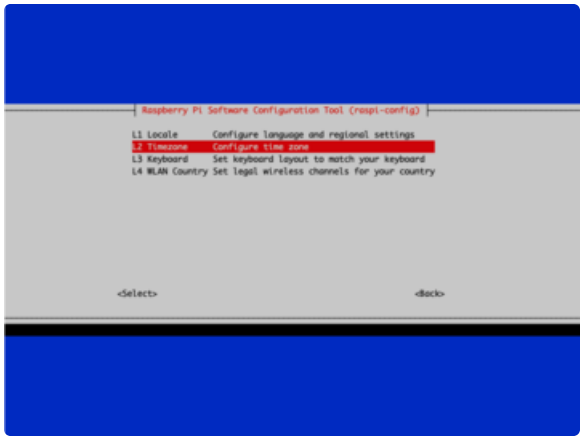
If you haven't done so already, be sure your timezone is set correctly. A freshly setup Raspberry Pi is

usually set to GMT by default. You can change it by typing:

```
sudo raspi-config
```



Select **Localisation Options.**



The select **Timezone**. This will take you to a section where you can select your Timezone. The organization is a bit unusual. For instance, you were in the US Pacific Timezone, you would select **US** and **Pacific Ocean**.

This will ensure that it is able to tell you the correct time. You can find more information about using **raspi-config** in the official documentation (https://adafru.it/jsD).

Install Voice2JSON

Installing Voice2JSON is fairly straightforward. First you need to install some prerequisites by running the following command:

```
sudo apt-get install libasound2 libasound2-data libasound2-plugins
```

```
pi@raspberrypi:~ $ sudo apt-get install libasound2 libasound2-data libasound2-plugins
Reading package lists... Done
Building dependency tree
Reading state information... Done
libasound2 is already the newest version (1.1.8-1+rpt1).
libasound2 set to manually installed.
libasound2-data is already the newest version (1.1.8-1+rpt1).
libasound2-data set to manually installed.
The following package was automatically installed and is no longer required:
  python-colorzero
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  fontconfig fontconfig-config fonts-dejavu-core libaom0 libasyncns0 libavcodec58 libavresample4 libavutil56 libcairo2
  libdrm-amdgpu1 libdrm-common libdrm-nouveau2 libdrm-radeon1 libdrm2 libflac8 libfontconfig1 libgdk-pixbuf2.0-0 libgd
  libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libgsm1 libharfbuzz0b libice6 libjack-ja
  libopenjp2-7 libopus0 libpango-1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0 libpixman-1-0 libpulse0 librsvg2-2 librsv
  libsm6 libsnappy1v5 libsndfile1 libsoxr0 libspeex1 libswresample3 libthai-data libthai0 libtheora0 libtiff5 libtwola
  libvdpau1 libvorbis0a libvorbisenc2 libvpx5 libwavpack1 libwebp6 libwebpmux3 libx11-xcb1 libx264-155 libx265-165 libx
  libxcb-present0 libxcb-render0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxdamage1 libxfixes3 libxi6 libxrender1 libx
  libzvbi-common libzvbi0 mesa-va-drivers mesa-vdpau-drivers va-driver-all vdpau-driver-all x11-common
Suggested packages:
  jackd2 opus-tools pulseaudio librsvg2-bin lm-sensors speex
The following NEW packages will be installed:
  fontconfig fontconfig-config fonts-dejavu-core libaom0 libasound2-plugins libasyncns0 libavcodec58 libavresample4 li
  libdatrie1 libdrm-amdgpu1 libdrm-common libdrm-nouveau2 libdrm-radeon1 libdrm2 libflac8 libfontconfig1 libgdk-pixbuf
  libgl1 libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libgsm1 libharfbuzz0b libice6 lib
  libogg0 libopenjp2-7 libopus0 libpango-1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0 libpixman-1-0 libpulse0 librsvg2-
  libshine3 libsm6 libsnappy1v5 libsndfile1 libsoxr0 libspeex1 libswresample3 libthai-data libthai0 libtheora0 libtiff
  libvdpau-va-gl1 libvdpau1 libvorbis0a libvorbisenc2 libvpx5 libwavpack1 libwebp6 libwebpmux3 libx11-xcb1 libx264-155
  libxcb-glx0 libxcb-present0 libxcb-render0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxdamage1 libxfixes3 libxi6 lib
  libxxf86vm1 libzvbi-common libzvbi0 mesa-va-drivers mesa-vdpau-drivers va-driver-all vdpau-driver-all x11-common
0 upgraded, 98 newly installed, 0 to remove and 0 not upgraded.
```

Next verify you are on the armhf architecture by typing:

```
dpkg-architecture | grep DEB_BUILD_ARCH=
```

```
pi@raspberrypi:~ $ dpkg-architecture | grep DEB_BUILD_ARCH=
DEB_BUILD_ARCH=armhf
```

Next download the package with **wget** and install the Voice2JSON file:

```
wget https://github.com/synesthesiam/voice2json/releases/download/v2.0/voice2json_2.0_armhf.deb
sudo apt install ./voice2json_2.0_armhf.deb
```

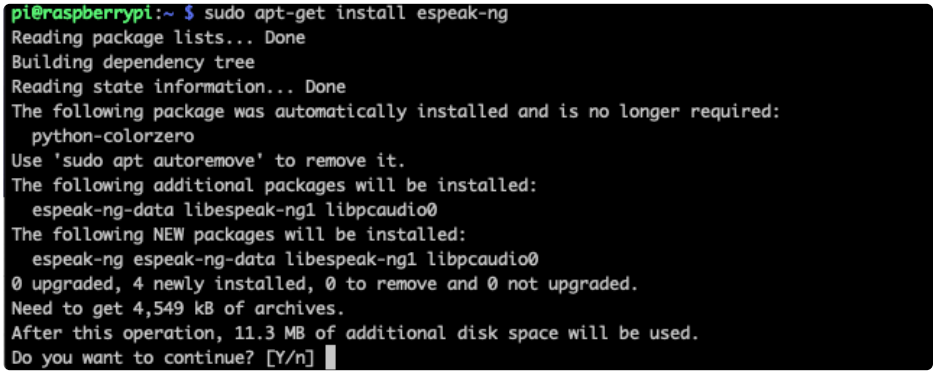
```
pi@raspberrypi:~ $ sudo apt install ./voice2json_2.0_armhf.deb
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'voice2json' instead of './voice2json_2.0_armhf.deb'
The following package was automatically installed and is no longer required:
  python-colorzero
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  espeak espeak-data jq libatlas3-base libespeak1 libgfortran5 libjq1 libltdl7 libonig5 li
  libsox-fmt-base libsox3 libvorbisfile3 sox
Suggested packages:
  libsox-fmt-all
The following NEW packages will be installed:
  espeak espeak-data jq libatlas3-base libespeak1 libgfortran5 libjq1 libltdl7 libonig5 li
  libsox-fmt-base libsox3 libvorbisfile3 sox voice2json
0 upgraded, 19 newly installed, 0 to remove and 0 not upgraded.
Need to get 5,169 kB/97.7 MB of archives.
```

Speech Synthesis Library

Voice2Json is also capable of making use of speech synthesis, so it's helpful to have the espeak library

installed:

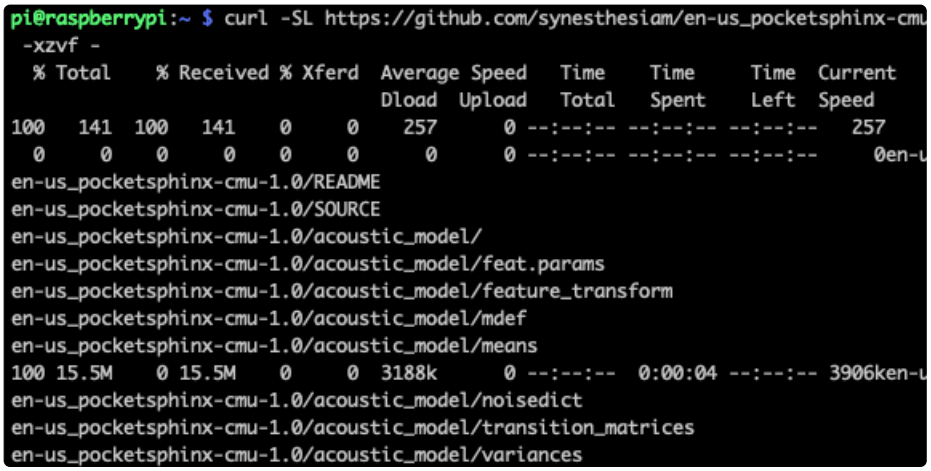
```
sudo apt-get install espeak-ng
```



Install a Profile

Voice2JSON uses profiles in order to combine a language with a speech recognition engine. The profile is not included as part of the package installation, so you will need to install that separately. Though there are many additional profiles, this setup installs the US English/PocketSphinx profile using the following commands:

```
mkdir -p ~/.config/voice2json
curl -SL https://github.com/synesthesiam/en-us_pocketsphinx-cmu/archive/v1.0.tar.gz | tar -C
~/.config/voice2json --skip-old-files --strip-components=1 -xzvf -
```



Latest Pillow Library

The demo project uses `displayio` which uses Pillow, or the Python Imaging Library, underneath. To get the latest version of Pillow, you can install by upgrading to the latest PIP and then installing Pillow with the following commands:

```
python3 -m pip install --upgrade pip
python3 -m pip install --upgrade Pillow
```

CircuitPython Libraries

A few CircuitPython libraries are needed for this project. These can be easily installed through PIP using the following command:

```
python3 -m pip install adafruit-circuitpython-st7789 adafruit-circuitpython-dotstar
```

After that finishes, you should be ready to configure your setup.

Configuring Custom Commands

To configure custom commands, you will need to edit the **sentences.ini** file located inside the `~/.config/voice2json` folder. You can find out a lot more information from the official documentation (https://adafru.it/Tco). The demo code will only be covering a subset of the available features.

Go ahead and make sure your **sentences.ini** file looks like the following:

```
[GetTime]
what is the time
what time is it
tell me the time

[ChangeLightColor]
light_name = (left | middle | right) {lightname}
color = (red | green | blue | yellow | orange | purple | white | off) {color}

set [the] <light_name> light [to] <color>
make [the] <light_name> light <color>

[DisplayPicture]
category = ((cat | adafruit) {category})
type = (picture | image | photo)
display [(a | an)] <category> <type>
show [me] [(a | an)] <category> <type>
find [me] [(a | an)] <category> <type>
```

There are 3 different intents here that can be communicated in a few different ways each. For the **GetTime** intent, there are a few ways of wording it that will yield the same result.

For the **ChangeLightColor** intent, this uses a few list variables to limit the responses to a manageable set. It also uses some optional words to give more flexibility to the phrasing.

For the **DisplayPicture** intent, this is similar to the previous intent, except with **type**, it is not returning value since it doesn't matter for the demo. It just makes it easier to expand the vocabulary without lots of extra typing.

Train the Profile

Before you can run the demo, you will need to train your profile. You can do that by running the following command:

```
voice2json train-profile
```

```
pi@raspberrypi:~ $ voice2json train-profile
WARNING:rhasspynlu.g2p:Missing word 'adafruit'
Training completed in 4.5462816839999505 second(s)
```

It usually only takes about 5 seconds on a Raspberry Pi 4. You can ignore the warning about the missing word. It just means that it didn't exist in the dictionary, but it seems to have no trouble recognizing it.

Demo Code

To download code and libraries to your Raspberry Pi, click the **Download Project Bundle** button below to get the code and other project files as a zip file.

```
import os
import subprocess
import random
import json
import re
from datetime import datetime
import board
import displayio
import adafruit_dotstar
from adafruit_st7789 import ST7789

IMAGE_FOLDER = "images"

listen_command = "/usr/bin/voice2json transcribe-stream | /usr/bin/voice2json recognize-intent"
speak_command = "/usr/bin/voice2json speak-sentence '{}'"
pattern = re.compile(r'(?<!^)(?=[A-Z])')

dots = adafruit_dotstar.DotStar(board.D6, board.D5, 3, brightness=0.2,
pixel_order=adafruit_dotstar.RBG)
dots.fill(0)

colors = {
    'red': 0xff0000,
    'green': 0x00ff00,
    'blue': 0x0000ff,
    'yellow': 0xffff00,
    'orange': 0xff8800,
    'purple': 0x8800ff,
    'white': 0xffffff,
    'off': 0
}

lights = ['left', 'middle', 'right']

def get_time():
    now = datetime.now()
    speak("The time is {}".format(now.strftime("%-I:%M %p")))

def display_picture(category):
    path = os.getcwd() + "/" + IMAGE_FOLDER + "/" + category
    print("Showing a random image from {}".format(category))
    load_image(path + "/" + get_random_file(path))

def get_random_file(folder):
    filenames = []
    for item in os.listdir(folder):
        if os.path.isfile(os.path.join(folder, item)) and item.endswith((".jpg", ".bmp",
".gif")):
```

```
            filenames.append(item)
    if len(filenames):
        return filenames[random.randrange(len(filenames))]
    return None

def load_image(path):
    "Load an image from the path"
    if len(splash):
        splash.pop()
    # CircuitPython 6 & 7 compatible
    bitmap = displayio.OnDiskBitmap(open(path, "rb"))
    sprite = displayio.TileGrid(bitmap, pixel_shader=getattr(bitmap, 'pixel_shader',
displayio.ColorConverter()))

    # # CircuitPython 7+ compatible
    # bitmap = displayio.OnDiskBitmap(path)
    # sprite = displayio.TileGrid(bitmap, pixel_shader=bitmap.pixel_shader)

    splash.append(sprite)

def change_light_color(lightname, color):
    dotstar_number = lights.index(lightname)
    dots[dotstar_number] = colors[color]
    print("Setting Dotstar {} to 0x{:06X}".format(dotstar_number, colors[color]))

def speak(sentence):
    for output_line in shell_command(speak_command.format(sentence)):
        print(output_line, end='')

def shell_command(cmd):
    popen = subprocess.Popen(cmd, stdout=subprocess.PIPE, shell=True, universal_newlines=True)
    for stdout_line in iter(popen.stdout.readline, ""):
        yield stdout_line
    popen.stdout.close()
    return_code = popen.wait()
    if return_code:
        raise subprocess.CalledProcessError(return_code, cmd)

def process_output(line):
    data = json.loads(line)
    if not data['timeout'] and data['intent']['name']:
        func_name = pattern.sub('_', data['intent']['name']).lower()
        if func_name in globals():
            globals()[func_name](**data['slots'])

displayio.release_displays()
spi = board.SPI()
tft_cs = board.CE0
tft_dc = board.D25
tft_lite = board.D26

display_bus = displayio.FourWire(spi, command=tft_dc, chip_select=tft_cs)

display = ST7789(
    display_bus,
    width=240,
    height=240,
```

```
    rowstart=80,
    rotation=180,
    backlight_pin=tft_lite,
)

splash = displayio.Group()
display.show(splash)

for output_line in shell_command(listen_command):
    process_output(output_line)
```

If you haven't already done so, be sure to copy **sentences.ini** over to `~/.config/voice2json` folder that was created during the Profile Installation step in the Raspberry Pi Setup.

The **lib** folder can be ignored since you should have already installed the necessary libraries during the Raspberry Pi Setup.

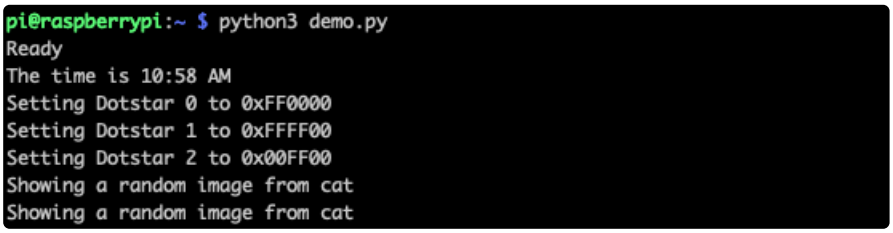
The **demo.py** file and **images** folder should be uploaded to the home folder of your Raspberry Pi. Once you have everything in place, start it up by typing:

```
python3 demo.py
```

If it doesn't work at first, make sure the Microphone On/Off switch is in the On position.

Use

Wait until it says **Ready** in the terminal and then you can begin speaking. You can ask it to tell you the time, to display a **cat** or **adafruit** image, or to set the left, middle, or right lights to **red**, **orange**, **yellow**, **green**, **blue**, **purple**, **white**, or **off**.



Here's a video of the demo in action:

Demo Code Walkthrough

Next we're going to go over the demo code section by section. First, you'll notice quite a few imports. Most of these are standard python imports, but here's the purpose of the different imports:

- **os** and **random** are used to get a list of files and folders for automatically finding images and randomly select one.
- **subprocess** is used to to run the Voice2JSON file from within Python
- **json** is used to decode the output from Voice2JSON
- **re** is the regular expression library and is used to convert the Intent names to proper Python function names to make adding new ones easier
- **datetime** is used for getting the current Date and Time
- **displayio** is the Display library which was rewritten to run with Blinka
- The remaining libraries are CircuitPython libraries for accessing various BrainCraft accessories

```
import os
import subprocess
import random
import json
import re
from datetime import datetime
import board
import displayio
import adafruit_dotstar
from adafruit_st7789 import ST7789
```

Next are a few settings including the name of the images folder, the Voice2JSON commands for listening and speaking, and a pre-compiled regular expression pattern to speed things up a bit.

```
IMAGE_FOLDER = "images"

listen_command = "/usr/bin/voice2json transcribe-stream | /usr/bin/voice2json recognize-intent"
speak_command = "/usr/bin/voice2json speak-sentence '{}'"
pattern = re.compile(r'(?<!^)(?=[A-Z])')
```

After that, the DotStars are initialized and set to off.

```
dots = adafruit_dotstar.DotStar(board.D6, board.D5, 3, brightness=0.2,
pixel_order=adafruit_dotstar.RBG)
dots.fill(0)
```

Then there are a couple of data structures used to provide meaning to the recognized values. The colors will translate the name to the actual value displayed on the DotStar. Altering these values would change the color displayed.

The lights list is just to give positional information to the name. Altering these values would change the order that it thinks the DotStars are in.

```
colors = {
    'red': 0xff0000,
    'green': 0x00ff00,
    'blue': 0x0000ff,
    'yellow': 0xffff00,
    'orange': 0xff8800,
    'purple': 0x8800ff,
    'white': 0xffffff,
    'off': 0
}

lights = ['left', 'middle', 'right']
```

Next is the `get_time()` function. It really just reads the current time, formats it, and uses the speak function which we will get into more detail further down.

```
def get_time():
    now = datetime.now()
    speak("The time is {}".format(now.strftime("%-I:%M %p")))
```

Next up are the picture display functions. The `display_picture()` function is the main handler and starts off by getting the full path to the image folder and passing it into `get_random_file()`.

The `get_random_file()` function does exactly what it sounds like, it randomly returns a file inside the specified path. It starts off by getting all files and filters them down to image files. Then it selects one at random and returns it.

The `load_image()` function will clear any existing sprite from the main splash group and then create a new sprite from the image located at the specified path and display it on the screen.

```
def display_picture(category):
    path = os.getcwd() + "/" + IMAGE_FOLDER + "/" + category
    print("Showing a random image from {}".format(category))
    load_image(path + "/" + get_random_file(path))

def get_random_file(folder):
    filenames = []
    for item in os.listdir(folder):
        if os.path.isfile(os.path.join(folder, item)) and item.endswith((".jpg", ".bmp",
".gif")):
            filenames.append(item)
    if len(filenames):
        return filenames[random.randrange(len(filenames))]
    return None

def load_image(path):
    "Load an image from the path"
    if len(splash):
        splash.pop()
    # CircuitPython 6 & 7 compatible
    bitmap = displayio.OnDiskBitmap(open(path, "rb"))
    sprite = displayio.TileGrid(bitmap, pixel_shader=getattr(bitmap, 'pixel_shader',
displayio.ColorConverter()))

    # # CircuitPython 7+ compatible
    # bitmap = displayio.OnDiskBitmap(path)
    # sprite = displayio.TileGrid(bitmap, pixel_shader=bitmap.pixel_shader)

    splash.append(sprite)
```

The `change_light_color()` function is the handler for changing lights. It will change the light at the specified position to the specified color. It starts off by figuring out the DotStar index by looking up the position name, then sets the DotStar at that index to the corresponding color value.

```
def change_light_color(lightname, color):
    dotstar_number = lights.index(lightname)
    dots[dotstar_number] = colors[color]
    print("Setting Dotstar {} to 0x{:06X}".format(dotstar_number, colors[color]))
```

Finally, there's the `speak()` function, which simply takes the value of `speak_command`, substitutes in the specified text, and runs the command using the `shell_command()` function.

```
def speak(sentence):
    for output_line in shell_command(speak_command.format(sentence)):
        print(output_line, end='')
```

The `shell_command()` function is where a lot of the magic happens. It is responsible for running the given command in a subprocess and returning any output. It will keep running and yielding output until the subprocess has completely finished running.

```
def shell_command(cmd):
    popen = subprocess.Popen(cmd, stdout=subprocess.PIPE, shell=True, universal_newlines=True)
    for stdout_line in iter(popen.stdout.readline, ""):
        yield stdout_line
    popen.stdout.close()
    return_code = popen.wait()
    if return_code:
        raise subprocess.CalledProcessError(return_code, cmd)
```

The `process_output()` function is the main JSON processing function. It starts off by decoding the JSON and making sure to only process if a timeout hadn't occurred, which happens regularly from Voice2JSON when no speech is detected.

If it detects genuine recognition, it will take the given **Intent** name and convert it to a Python function. All of the Intent names are defined in the **sentences.ini** file. It will make sure the function has been defined by looking in globals() and call it if it has.

The slots are the parameters that are defined in the **sentences.ini** file in curly braces. The `**` operator (double asterisk) is used for keyword argument unpacking and will pass the parameters in as function arguments automatically. The parameter and function argument names must match.

```
def process_output(line):
    data = json.loads(line)
    if not data['timeout'] and data['intent']['name']:
        func_name = pattern.sub('_', data['intent']['name']).lower()
        if func_name in globals():
            globals()[func_name](**data['slots'])
```

The next bit of code is standard displayio setup for the display on the BrainCraft HAT.

```
displayio.release_displays()
spi = board.SPI()
tft_cs = board.CE0
tft_dc = board.D25
tft_lite = board.D26

display_bus = displayio.FourWire(spi, command=tft_dc, chip_select=tft_cs)

display = ST7789(
    display_bus,
    width=240,
    height=240,
    rowstart=80,
    rotation=180,
    backlight_pin=tft_lite,
)

splash = displayio.Group()
display.show(splash)
```

Finally, is the code to run the main `listen_command` and process the output. If you've been programming in Python for a while, you may notice there is no main while loop. That's because the Voice2JSON is run inside of a subprocess and since it has a main loop, it isn't necessary to add one.

```
for output_line in shell_command(listen_command):
    process_output(output_line)
```