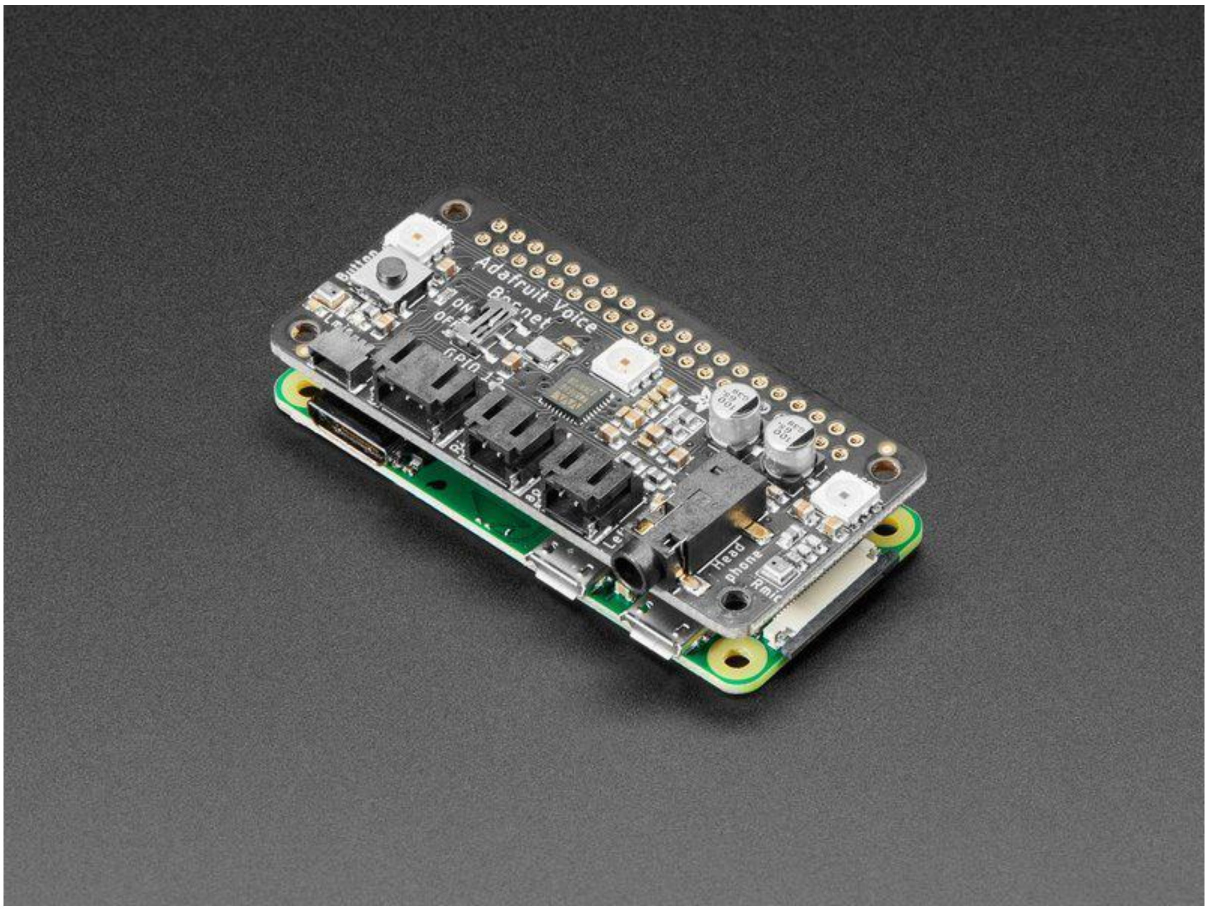
Adafruit Voice Bonnet

Created by Kattni Rembor

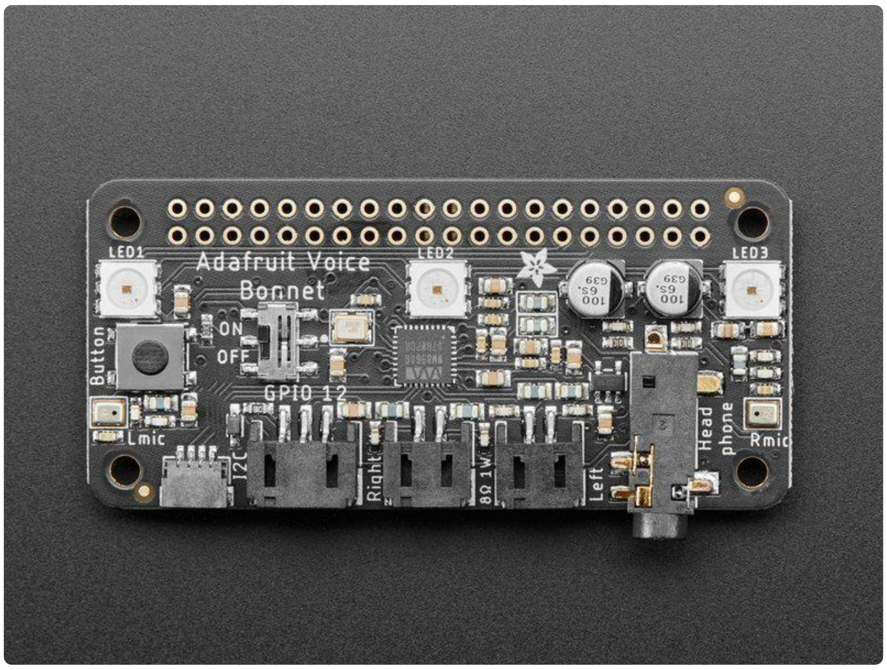


https://learn.adafruit.com/adafruit-voice-bonnet

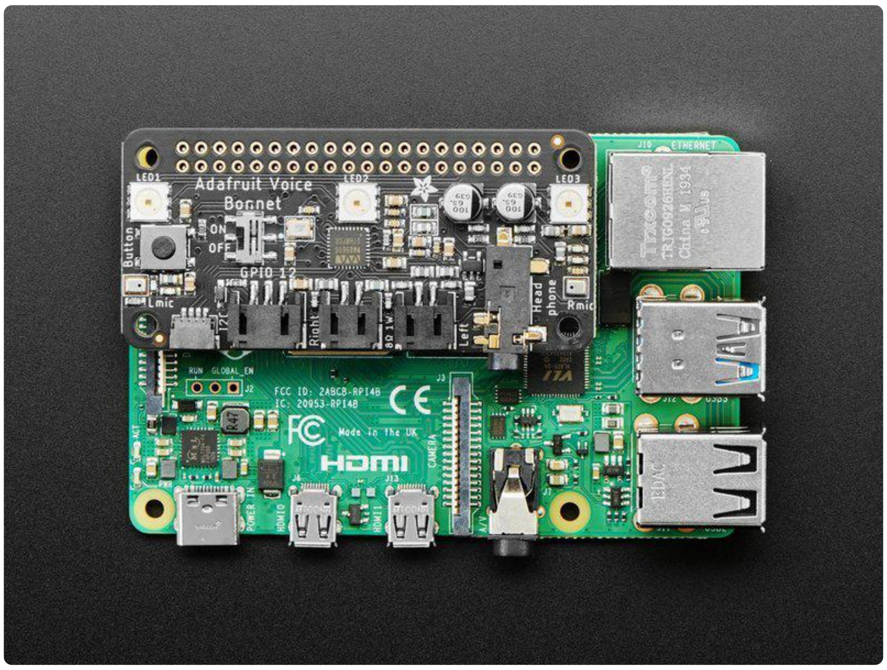
Last updated on 2023-04-14 01:59:38 PM EDT

Table of Contents

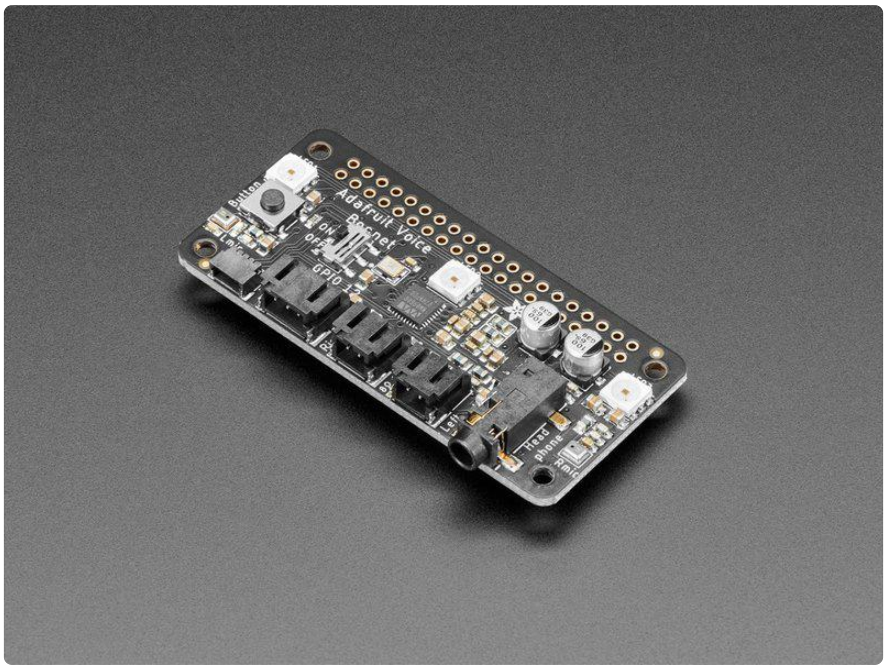
Overview



Your Raspberry Pi computer is like an electronic brain - and with the Adafruit Voice Bonnet you can give it a mouth and ears as well! Featuring two microphones and two 1 Watt speaker outputs using a high quality I2S codec, this Pi add-on will work with an y Raspberry Pi with a 2x20 connector - from the Pi Zero up to the Pi 4 and beyond (basically all but the very first ones made).

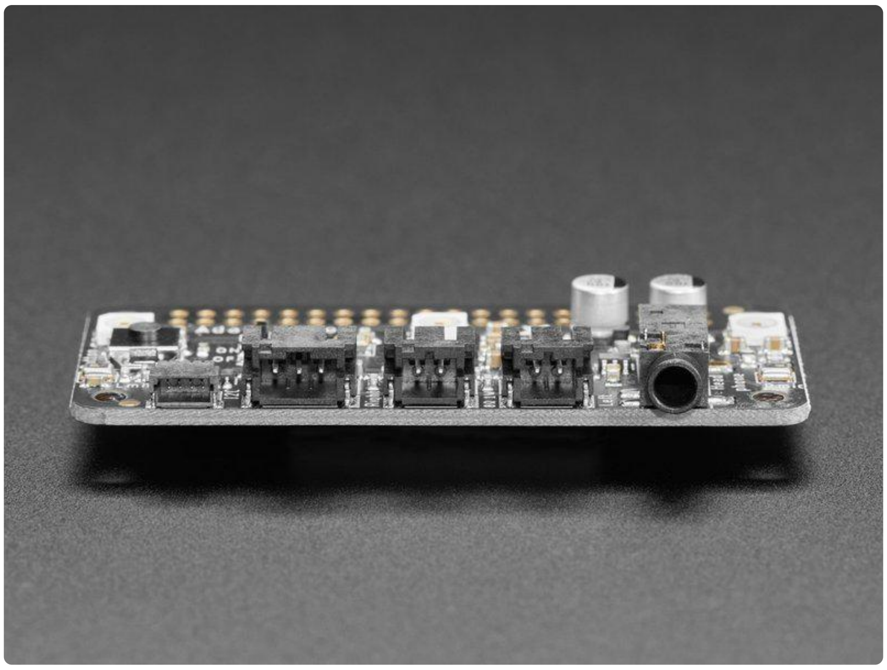


The on-board [WM8960 codec uses I2S digital audio for great quality recording and playback ()]() - so it sounds a lot better than the headphone jack on the Pi (or the no-headphone jack on a Pi Zero). We put ferrite beads and filter capacitors on every input and output to get the best possible performance, and all at a great price.



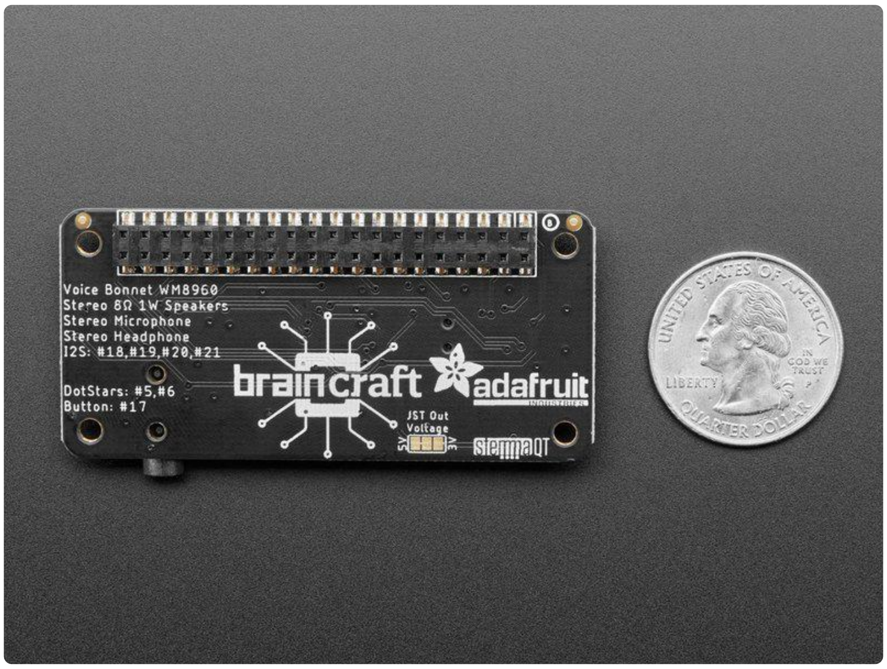
We specifically designed this bonnet for use with making machine learning projects such as DIY voice assistants - for example see this guide on creating a DIY Google Assistant (). But you could do various voice activated or voice recognition projects. With two microphones, basic voice position can be detected as well.

- WM8960 codec uses I2S digital audio for both input and output ()
- On/Off privacy switch to deactivate audio so you know it can't be recording.
- Two analog microphone inputs (left and right)
- Two 1W speaker outputs
- 3.5mm stereo headphone or line-out audio
- Plugs into any Raspberry Pi with 2x20 headers



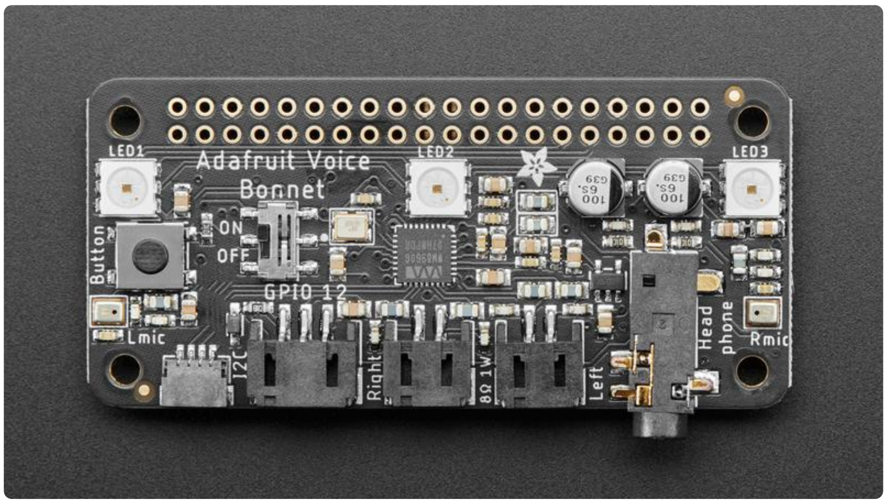
With the extra space on the PCB we also added some bonus extras!

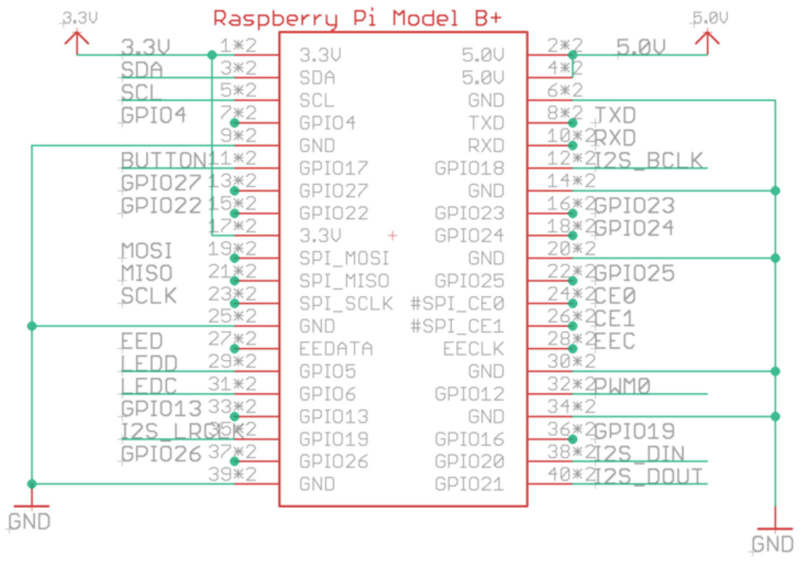
- Push button - Use to change modes, activate the voice assistant, anything you like!
- Three DotStar RGB LEDs - add LED feedback or make a rainbow light show
- STEMMA QT connector - plug in any of our I2C sensors, OLEDs, or accessories ().
- 3 Pin JST STEMMA connector - for larger accessories, like NeoPixels (), a relay () or servo ()



No assembly required! Simply pop it onto your Pi and install the microphone/speaker card using our installer script. Your Pi will then have stereo input and output for use in any software - they appear as any other speaker/mic would. Play music or record audio with ease. For audio input, the two microphones are built into the Bonnet. For audio output, you can either use the Line-out/Headphone 3.5mm stereo jack or you can plug in one or two of our enclosed speakers (). For DIY speakers, solder any 1W+ speaker to one of these JST 2-PH cables (). If you'd like to stack another HAT or bonnet on top, use a 2x20 stacking header () to feed through the 2x20 connector.

Pinouts



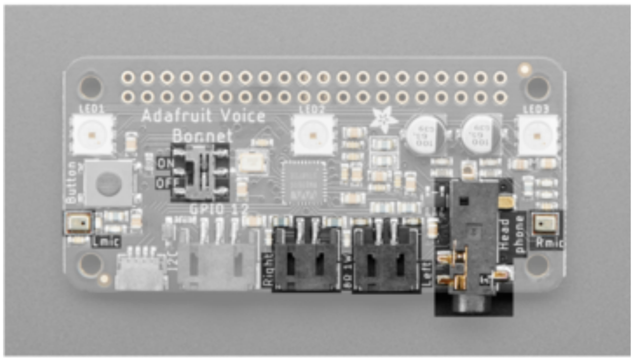


Power

- 5.0V - Connected to power on the audio amp
- 3.3V - Connected to the STEMMA QT / Qwiic connector

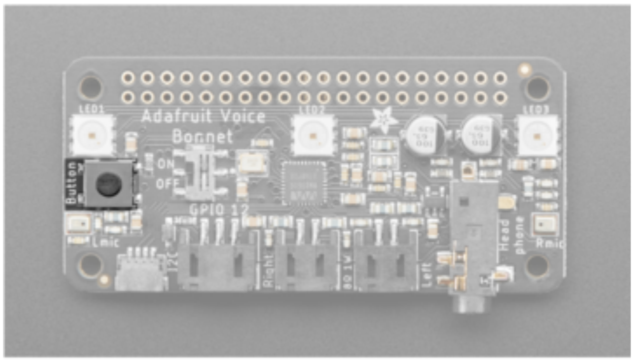
- GND - Ground for everything

Sound



GPIO #18, GPIO#19, GPIO #20, GPIO #21 - I2S Digital Audio.
Speakers - The two speaker connectors are located on the bottom in the middle, labeled Right and Left. Plug in an enclosed speaker () or use a JST 2PH cable to attach custom speakers ().
Microphones - The two mics are in the middle of each side, labeled L Mic and R Mic.
Headphones/Line Out - 3.5mm headphone jack located on the bottom right. Use any headphones or Line-Out to another audio system.
On/Off Switch - Located towards the left in the middle of the board, switches audio on and off.

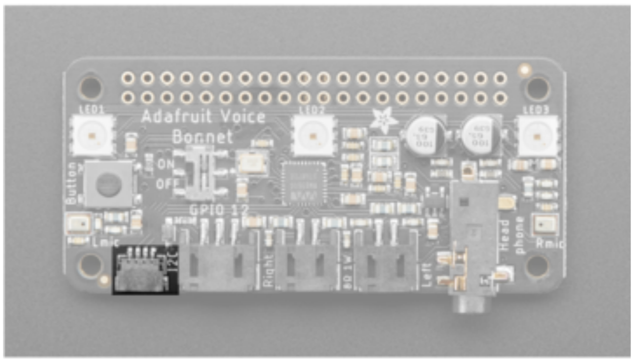
Button



This pin has a 10K pullup to 3.3V so when the button is pressed, you will read a LOW voltage on this pin.

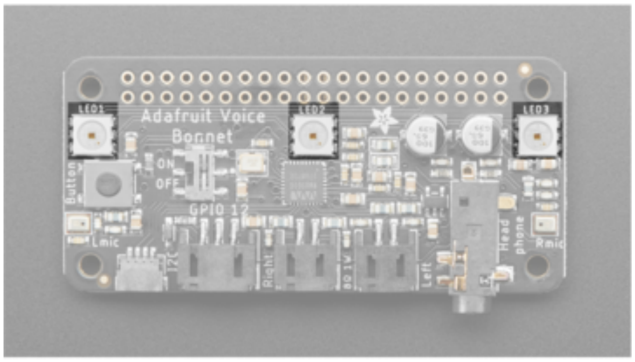
GPIO#17 - Button

STEMMA QT Connector



SCL, SDA - I2C data for the STEMMA QT / Qwiic connector. Can also use with Grove sensors with an adapter cable (). Great for quickly adding sensors or accessories with plug-and-play.

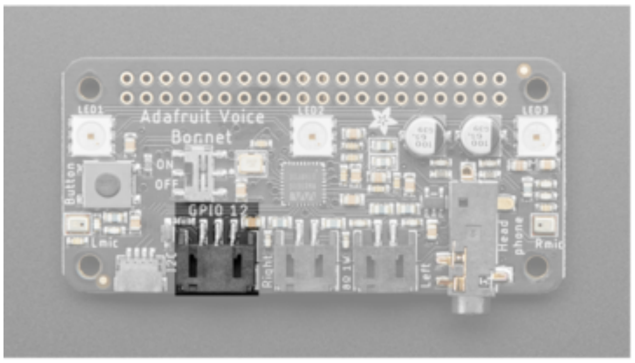
DotStar LEDs



Three fully color RGB addressable LEDs can provide feedback or a light show. Uses DotStar protocol (not NeoPixel) so any microcomputer can easily control the lights.

GPIO #5 - DotStar LED data pin.
GPIO #6 - Dotstar LED clock pin.

Digital/Analog Connector



GPIO#12 - 3-pin JST digital or analog connector. Easily connect things like NeoPixels () or servos () using this 2mm connector. You can also use a generic JST cable to connect to a breadboard. ()

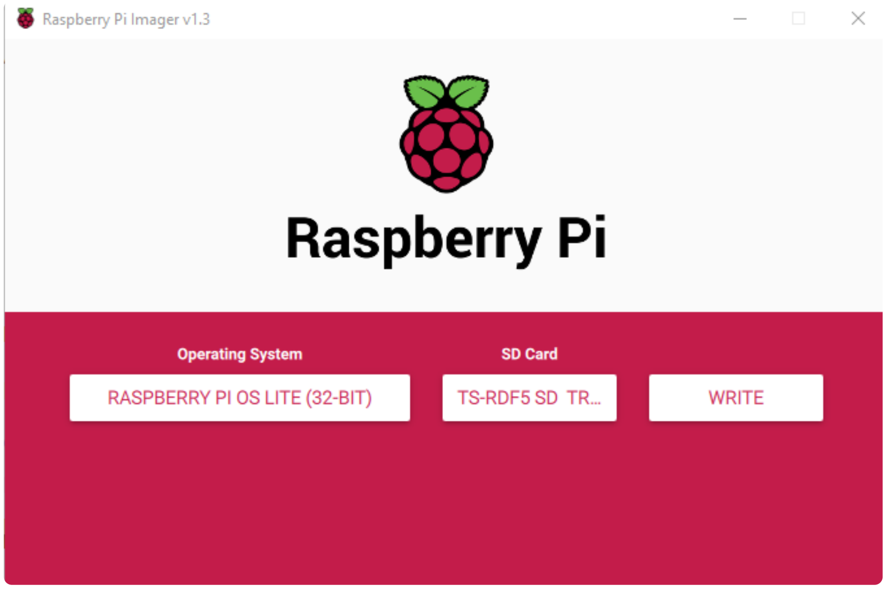
Raspberry Pi Setup

OK now you have all your parts in order, it's time to get your Raspberry Pi computer set up with the HAT or Bonnet.

Step 1 - Burn SD Card

Use Etcher or the Raspberry Pi Imager () to burn the latest Raspbian Lite to an SD card (you can use full but we won't be using the desktop software and it takes up a bunch of room.

If you are using the Raspberry Pi Imager, you can press Ctrl+Shift+x to get to advanced options.



If you enabled SSH and WiFi credentials in the Imager, you can skip steps 2 and 3

Step 2 - Configure log-in access

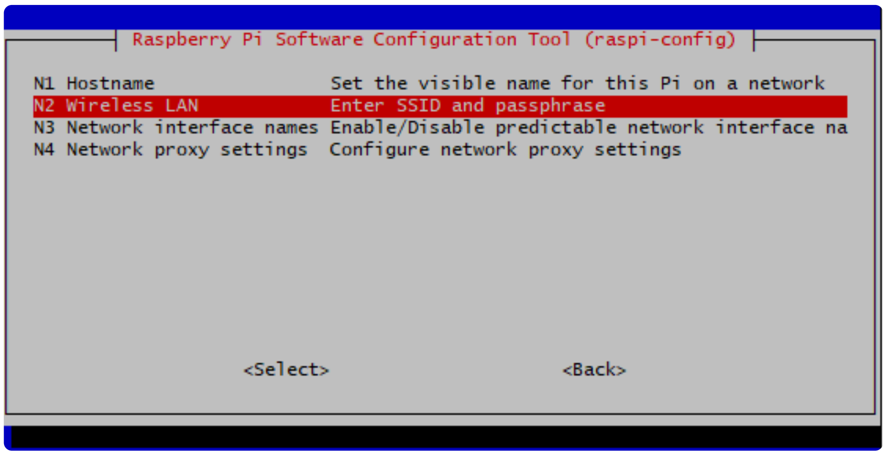
You'll need to be able to log into your Pi, either enable SSH access (and use and Ethernet cable) (), use a USB to serial cable, or connect a monitor and keyboard. Basically get it so you can log in.

We have a quickstart guide here () and here that you can follow (), or there's dozens of online guides. it is assumed by the next step you are able to log in and type commands in - ideally from a desktop computer, so you can copy and paste in some of the very long commands!

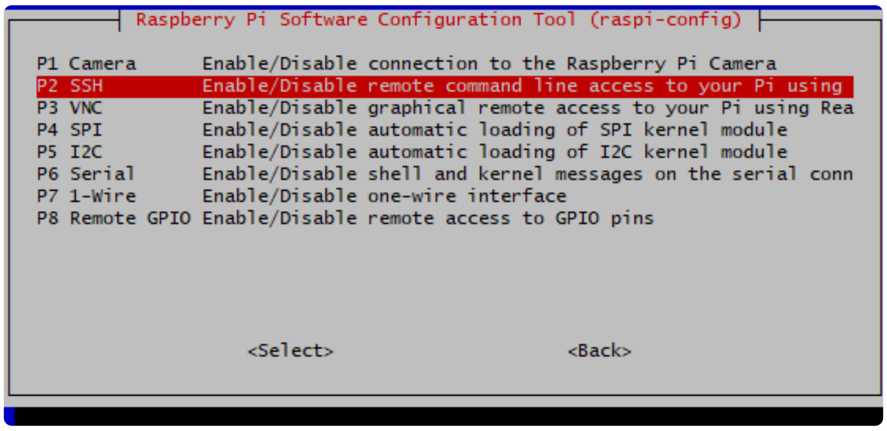
Step 3 - Log in & Enable Internet

Once you've logged in, enable WiFi (if you have built in WiFi) with sudo raspi-config () so you can ssh in.

Enable SSH as well if you haven't yet, also via sudo raspi-config

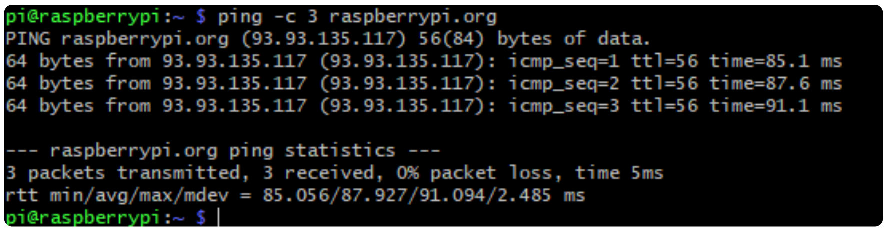


```
Raspberry Pi Software Configuration Tool (raspi-config)

N1 Hostname                 Set the visible name for this Pi on a network
N2 Wireless LAN             Enter SSID and passphrase
N3 Network interface names  Enable/Disable predictable network interface na
N4 Network proxy settings   Configure network proxy settings

                <Select>                  <Back>
```



```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using
P3 VNC         Enable/Disable graphical remote access to your Pi using Rea
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial conn
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins

                <Select>                  <Back>
```

After you're done, reboot, and verify you can log into your Pi and that it has internet access by running `ping -c 3 raspberrypi.org` and seeing successful responses.



```
pi@raspberrypi:~ $ ping -c 3 raspberrypi.org
PING raspberrypi.org (93.93.135.117) 56(84) bytes of data.
64 bytes from 93.93.135.117 (93.93.135.117): icmp_seq=1 ttl=56 time=85.1 ms
64 bytes from 93.93.135.117 (93.93.135.117): icmp_seq=2 ttl=56 time=87.6 ms
64 bytes from 93.93.135.117 (93.93.135.117): icmp_seq=3 ttl=56 time=91.1 ms

--- raspberrypi.org ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 5ms
rtt min/avg/max/mdev = 85.056/87.927/91.094/2.485 ms
pi@raspberrypi:~ $ |
```

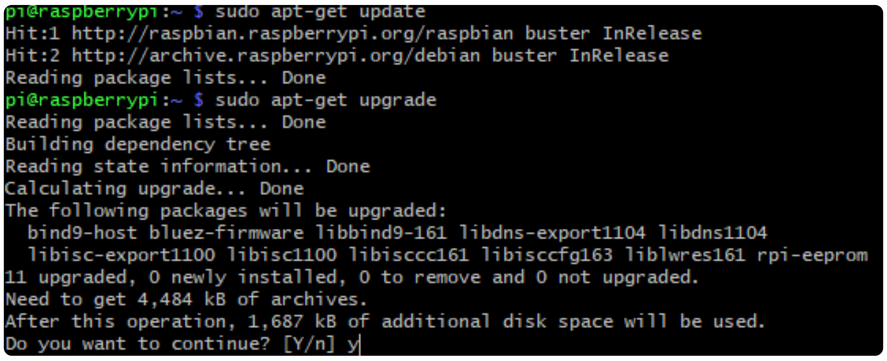
Step 4 - Update/Upgrade

Now that you are logged in, perform an update/update:

```
sudo apt update
sudo apt-get update
sudo apt-get -y upgrade
```

and

```
sudo apt-get install -y python3-pip
sudo pip3 install --upgrade setuptools
```



```
pi@raspberrypi:~ $ sudo apt-get update
Hit:1 http://raspbian.raspberrypi.org/raspbian buster InRelease
Hit:2 http://archive.raspberrypi.org/debian buster InRelease
Reading package lists... Done
pi@raspberrypi:~ $ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  bind9-host bluez-firmware libbind9-161 libdns-export1104 libdns1104
  libisc-export1100 libisc1100 libisccc161 libisccfg163 liblwres161 rpi-eeprom
11 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 4,484 kB of archives.
After this operation, 1,687 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

OK you've now got a nice, clean, connected, and up-to-date Pi!

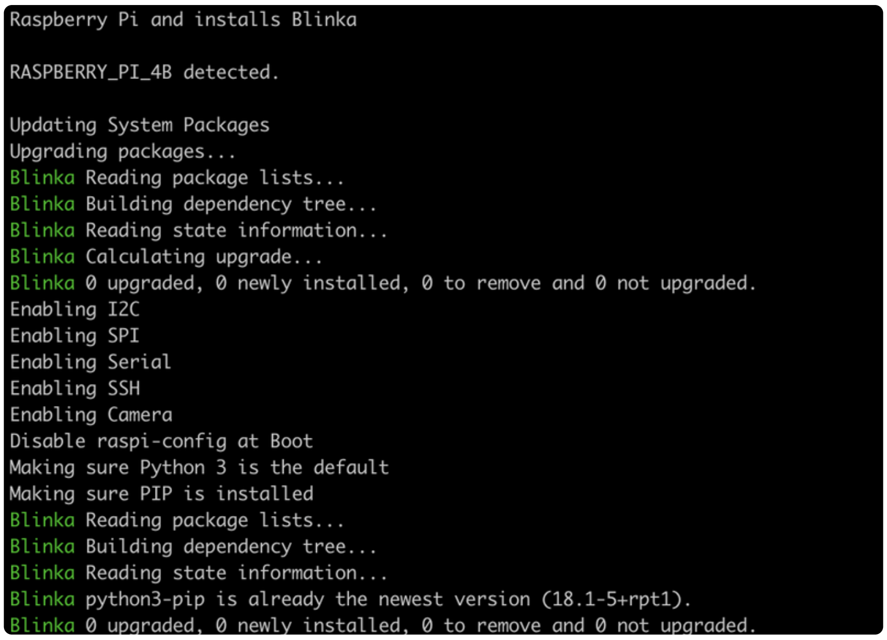
Blinka Setup

Blinka is our CircuitPython library compatibility layer. It allows many of the libraries that were written for CircuitPython to run on CPython for Linux. To learn more about Blinka, you can check out our CircuitPython Libraries on Linux and Raspberry Pi () guide.

We put together a script to easily make sure your Pi is correctly configured and install Blinka. It requires just a few commands to run. Most of it is installing the dependencies.

```
cd ~
sudo pip3 install --upgrade adafruit-python-shell
wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/
master/raspi-blinka.py
sudo python3 raspi-blinka.py
```

When it asks you if you want to reboot, choose yes.

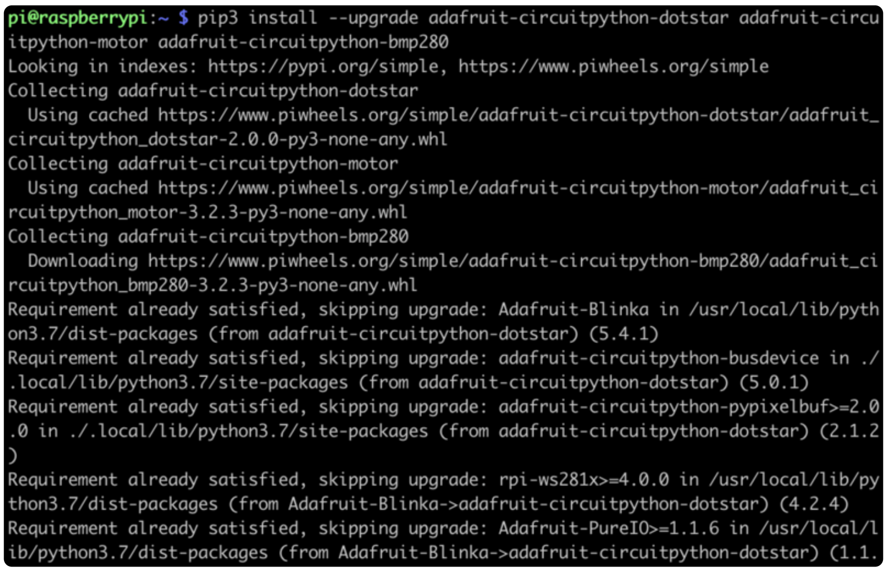


```
Raspberry Pi and installs Blinka

RASPBERRY_PI_4B detected.

Updating System Packages
Upgrading packages...
Blinka Reading package lists...
Blinka Building dependency tree...
Blinka Reading state information...
Blinka Calculating upgrade...
Blinka 0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Enabling I2C
Enabling SPI
Enabling Serial
Enabling SSH
Enabling Camera
Disable raspi-config at Boot
Making sure Python 3 is the default
Making sure PIP is installed
Blinka Reading package lists...
Blinka Building dependency tree...
Blinka Reading state information...
Blinka python3-pip is already the newest version (18.1-5+rpt1).
Blinka 0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Finally, once it reboots, there are just a couple CircuitPython libraries to install for the BrainCraft HAT or Voice Bonnet.

The DotStar library is for controlling the 3 on-board DotStar LEDs and the Motor library is for testing out the GPIO pins.

```
pip3 install --upgrade adafruit-circuitpython-dotstar adafruit-circuitpython-motor
adafruit-circuitpython-bmp280
```



```
pi@raspberrypi:~ $ pip3 install --upgrade adafruit-circuitpython-dotstar adafruit-circu
itpython-motor adafruit-circuitpython-bmp280
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting adafruit-circuitpython-dotstar
  Using cached https://www.piwheels.org/simple/adafruit-circuitpython-dotstar/adafruit_
circuitpython_dotstar-2.0.0-py3-none-any.whl
Collecting adafruit-circuitpython-motor
  Using cached https://www.piwheels.org/simple/adafruit-circuitpython-motor/adafruit_ci
rcuitpython_motor-3.2.3-py3-none-any.whl
Collecting adafruit-circuitpython-bmp280
  Downloading https://www.piwheels.org/simple/adafruit-circuitpython-bmp280/adafruit_ci
rcuitpython_bmp280-3.2.3-py3-none-any.whl
Requirement already satisfied, skipping upgrade: Adafruit-Blinka in /usr/local/lib/pyth
on3.7/dist-packages (from adafruit-circuitpython-dotstar) (5.4.1)
Requirement already satisfied, skipping upgrade: adafruit-circuitpython-busdevice in ./
.local/lib/python3.7/site-packages (from adafruit-circuitpython-dotstar) (5.0.1)
Requirement already satisfied, skipping upgrade: adafruit-circuitpython-pypixelbuf>=2.0
.0 in ./.local/lib/python3.7/site-packages (from adafruit-circuitpython-dotstar) (2.1.2
)
Requirement already satisfied, skipping upgrade: rpi-ws281x>=4.0.0 in /usr/local/lib/py
thon3.7/dist-packages (from Adafruit-Blinka->adafruit-circuitpython-dotstar) (4.2.4)
Requirement already satisfied, skipping upgrade: Adafruit-PureIO>=1.1.6 in /usr/local/l
ib/python3.7/dist-packages (from Adafruit-Blinka->adafruit-circuitpython-dotstar) (1.1.
```

That's it for Blinka and CircuitPython libraries.

Audio Setup

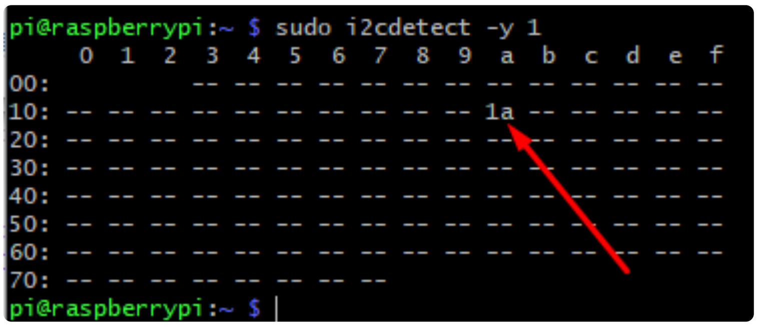
Start by making sure you've installed I2C support, see the previous page (Blinka Setup) on how to do it!

Install Voicecard software

Make sure you've got the BrainCraft HAT or Voice Bonnet installed, and I2C support installed as well!

When you run `sudo i2cdetect -y 1`

you should see an entry under 1A, indicating the hardware sees the audio card. The number may also appear as UU if you already installed software



```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- 1a -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

At the command line run:

```
cd ~
sudo apt-get install -y git
git clone https://github.com/HinTak/seeed-voicecard
cd seeed-voicecard
```

Depending on your kernel version, you may need to change your branch. You can check your kernel version by typing `uname -r`.

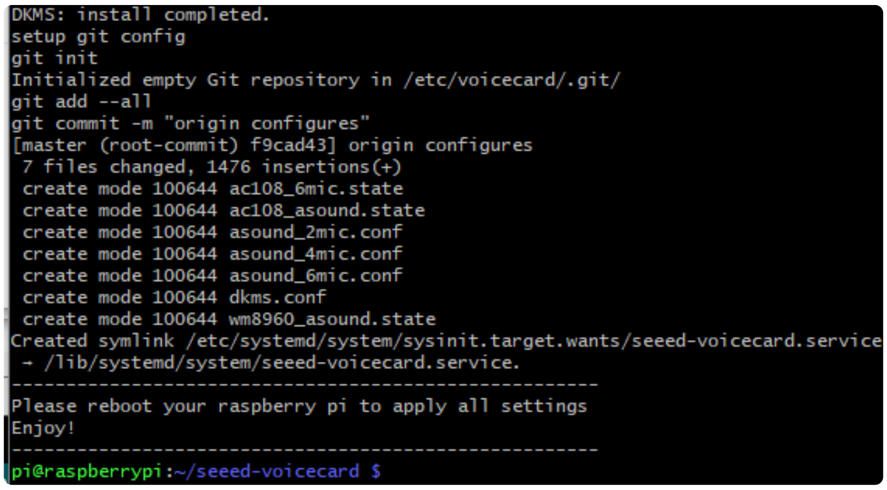
If your have a more recent kernel version of 5.10 or higher, use the following:

```
git checkout v5.9
sudo ./install.sh
```

If your running an older kernel version around 5.4, use the following:

```
git checkout v5.5
sudo ./install.sh
```

At the end you should see something like this:



```
DKMS: install completed.
setup git config
git init
Initialized empty Git repository in /etc/voicecard/.git/
git add --all
git commit -m "origin configures"
[master (root-commit) f9cad43] origin configures
 7 files changed, 1476 insertions(+)
 create mode 100644 ac108_6mic.state
 create mode 100644 ac108_asound.state
 create mode 100644 asound_2mic.conf
 create mode 100644 asound_4mic.conf
 create mode 100644 asound_6mic.conf
 create mode 100644 dkms.conf
 create mode 100644 wm8960_asound.state
Created symlink /etc/systemd/system/sysinit.target.wants/seeed-voicecard.service
 → /lib/systemd/system/seeed-voicecard.service.
------------------------------------------------------
Please reboot your raspberry pi to apply all settings
Enjoy!
------------------------------------------------------
pi@raspberrypi:~/seeed-voicecard $
```

Reboot with

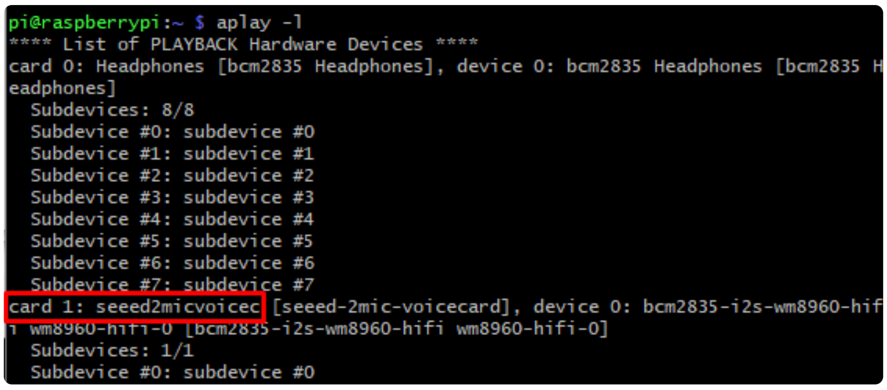
`sudo reboot`

and on reboot run

`sudo aplay -l`

To list all sound cards, you should see it at the bottom

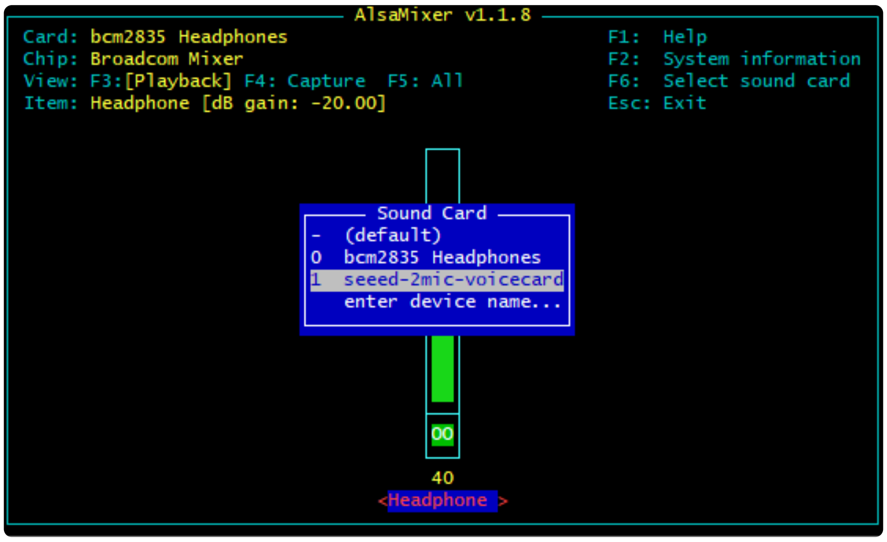
On a Raspberry Pi 4, your card number may be Card 2 instead of 1 because of the second HDMI port. You'll need to make some changes to some of the commands to reflect this further down.



```
pi@raspberrypi:~ $ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: Headphones [bcm2835 Headphones], device 0: bcm2835 Headphones [bcm2835 H
eadphones]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 1: seeed2micvoicec [seeed-2mic-voicecard], device 0: bcm2835-i2s-wm8960-hif
i wm8960-hifi-0 [bcm2835-i2s-wm8960-hifi wm8960-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

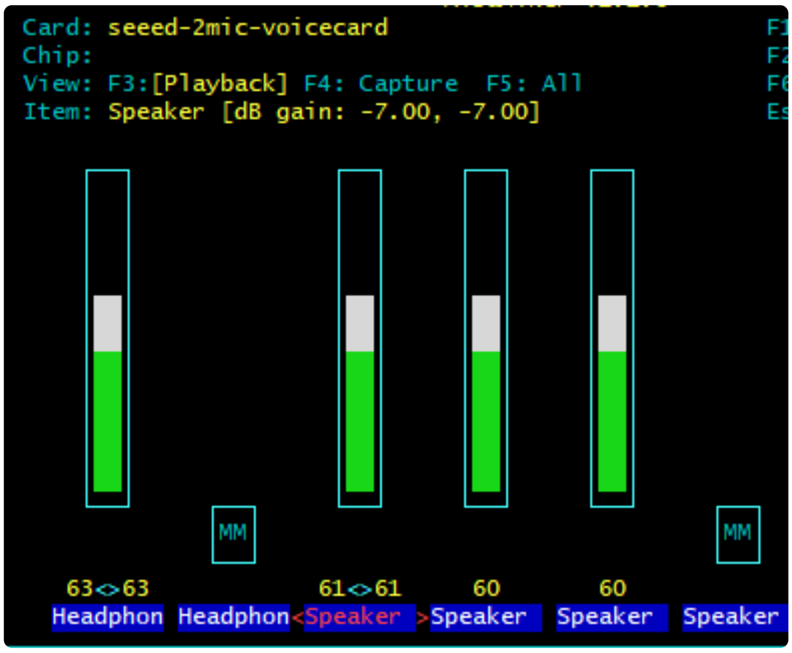
If your card number differs from the above image, take note of your number.

You can use `alsamixer` to adjust the volume, dont forget to select the card with F6



```
AlsaMixer v1.1.8
Card: bcm2835 Headphones                              F1:  Help
Chip: Broadcom Mixer                                  F2:  System information
View: F3:[Playback] F4: Capture  F5: All              F6:  Select sound card
Item: Headphone [dB gain: -20.00]                     Esc: Exit

                    Sound Card
                    -  (default)
                    0  bcm2835 Headphones
                    1  seeed-2mic-voicecard
                       enter device name...

                          00
                          40
                      <Headphone >
```

A gain of about 60% is plenty loud!



```
Card: seeed-2mic-voicecard
Chip:
View: F3:[Playback] F4: Capture  F5: All
Item: Speaker [dB gain: -7.00, -7.00]

                       MM                                  MM
  63<>63                 61<>61    60       60
 Headphon Headphon<Speaker >Speaker Speaker Speaker
```

Headphone/Speaker Test

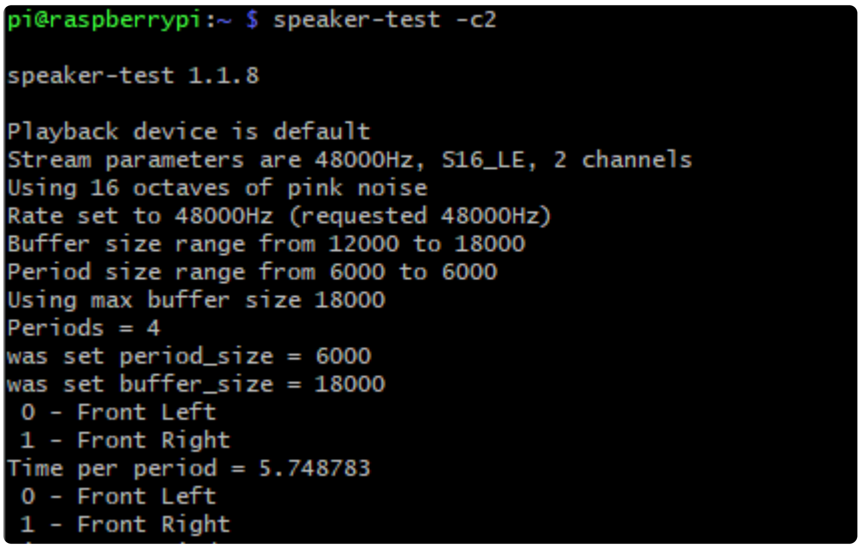
Make sure the Audio On/Off switch is set to ON!

With either headphones plugged into the headphone jack or a speaker attached to the speaker port, run

`speaker-test -c2`

you will hear white noise coming out of the speakers/headphones!

If you don't hear anything, make sure you have the audio on/off switch set!



```
pi@raspberrypi:~ $ speaker-test -c2

speaker-test 1.1.8

Playback device is default
Stream parameters are 48000Hz, S16_LE, 2 channels
Using 16 octaves of pink noise
Rate set to 48000Hz (requested 48000Hz)
Buffer size range from 12000 to 18000
Period size range from 6000 to 6000
Using max buffer size 18000
Periods = 4
was set period_size = 6000
was set buffer_size = 18000
 0 - Front Left
 1 - Front Right
Time per period = 5.748783
 0 - Front Left
 1 - Front Right
```

Microphone Test

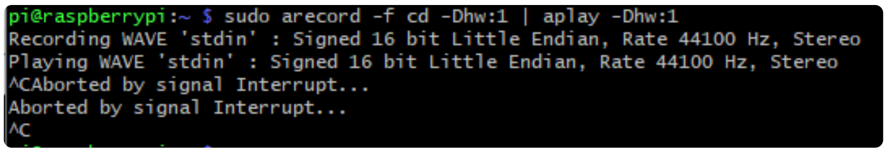
There are two microphones, and now we can test that they work. This test is best done with headphones, not using the speaker port, because it can cause a painful feedback effect if the speakers are next to the mics!

Run:

`sudo arecord -f cd -Dhw:2 | aplay -Dhw:2`

If your sound card ID is not #2, then replace the number in both of the `-Dhw:` parameters with your actual number.

Then either gently rub each microphone, or speak to hear yourself echoed!



```
pi@raspberrypi:~ $ sudo arecord -f cd -Dhw:1 | aplay -Dhw:1
Recording WAVE 'stdin' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Playing WAVE 'stdin' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
^CAborted by signal Interrupt...
Aborted by signal Interrupt...
^C
```

`Control-C` to quit when done

Your audio subsystem is now completely tested!

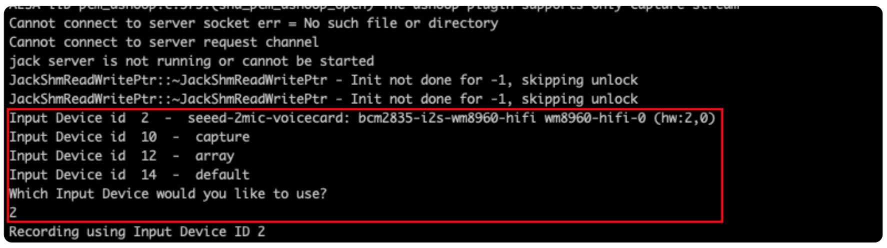
Python Libraries

The Microphone and Voice Card are installed as Linux level devices, so using them in is done as you would with any system level audio device. If you would like to make use of audio in Python, you can use the PyAudio library. To install pyaudio and its dependencies, run the following code:

```
sudo apt-get install libportaudio2 portaudio19-dev
sudo pip3 install pyaudio
```

Python Usage

Here is a basic test script to enumerate the devices and record for 10 seconds. When prompted, choose the device called seeed-2mic-voicecard.



```
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Input Device id  2  -  seeed-2mic-voicecard: bcm2835-i2s-wm8960-hifi wm8960-hifi-0 (hw:2,0)
Input Device id  10  -  capture
Input Device id  12  -  array
Input Device id  14  -  default
Which Input Device would you like to use?
2
Recording using Input Device ID 2
```

Copy and paste the following code into a file called audiotest.py.

```
import pyaudio
import wave

FORMAT = pyaudio.paInt16
CHANNELS = 1           # Number of channels
BITRATE = 44100        # Audio Bitrate
CHUNK_SIZE = 512       # Chunk size to
RECORDING_LENGTH = 10  # Recording Length in seconds
WAVE_OUTPUT_FILENAME = "myrecording.wav"
audio = pyaudio.PyAudio()

info = audio.get_host_api_info_by_index(0)
numdevices = info.get('deviceCount')
for i in range(0, numdevices):
    if (audio.get_device_info_by_host_api_device_index(0,
i).get('maxInputChannels')) &gt; 0:
        print("Input Device id ", i, " - ",
audio.get_device_info_by_host_api_device_index(0, i).get('name'))

print("Which Input Device would you like to use?")
device_id = int(input()) # Choose a device
print("Recording using Input Device ID "+str(device_id))

stream = audio.open(
    format=FORMAT,
    channels=CHANNELS,
    rate=BITRATE,
    input=True,
    input_device_index = device_id,
    frames_per_buffer=CHUNK_SIZE
)

recording_frames = []

for i in range(int(BITRATE / CHUNK_SIZE * RECORDING_LENGTH)):
    data = stream.read(CHUNK_SIZE)
    recording_frames.append(data)

stream.stop_stream()
stream.close()
audio.terminate()

waveFile = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
waveFile.setnchannels(CHANNELS)
waveFile.setsampwidth(audio.get_sample_size(FORMAT))
waveFile.setframerate(BITRATE)
waveFile.writeframes(b''.join(recording_frames))
waveFile.close()
```

Run the code with the following command:

`sudo python3 audiotest.py`

When finished, you can test playing back the file with the following command:

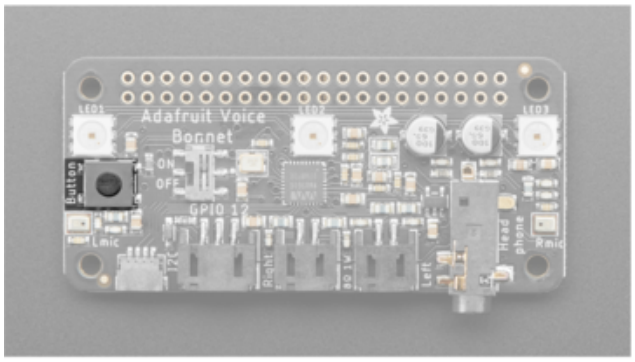
`aplay recordedFile.wav`

Python Usage

At this point, you should have just about everything already set up.

Besides the audio, the Voice Bonnet has quite a few other useful features on it that can be controlled through Python. We'll go through those and how to control them in Python.

Joystick and Button



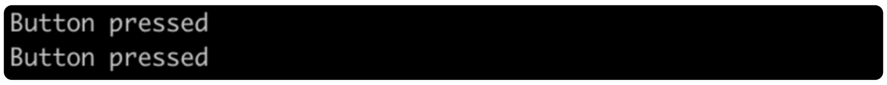
The button uses simple `digitalio`, so it's really simple to control. Here's a little script that will setup the GPIO, create an internal pull up, and then print out the value to the terminal.

```
import time
import board
from digitalio import DigitalInOut, Direction, Pull

button = DigitalInOut(board.D17)
button.direction = Direction.INPUT
button.pull = Pull.UP

while True:
  if not button.value:
    print("Button pressed")
  time.sleep(0.01)
```

Go ahead and save the above code onto your Pi as button_test.py and run it with the following command:

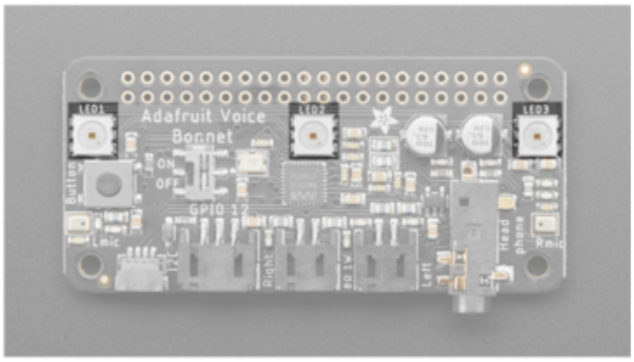
`python button_test.py`

Now try moving the joystick and press the button and you should see it print out what you're pressing.



```
Button pressed
Button pressed
```

DotStar LEDs



The 3 DotStar LEDS can be controlled with the DotStar CircuitPython Library. Here's a little script that will setup the DotStar LEDs and then color cycle them.

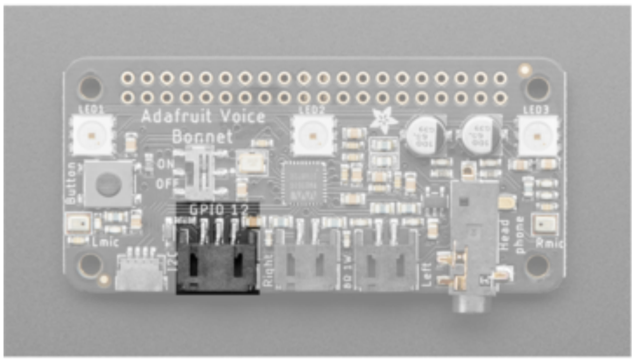
```
import time
import board
import adafruit_dotstar

DOTSTAR_DATA = board.D5
DOTSTAR_CLOCK = board.D6

dots = adafruit_dotstar.DotStar(DOTSTAR_CLOCK, DOTSTAR_DATA, 3, brightness=0.2)

def wheel(pos):
    # Input a value 0 to 255 to get a color value.
    # The colours are a transition r - g - b - back to r.
    if pos &lt; 0 or pos &gt; 255:
        return (0, 0, 0)
    if pos &lt; 85:
        return (255 - pos * 3, pos * 3, 0)
    if pos &lt; 170:
        pos -= 85
        return (0, 255 - pos * 3, pos * 3)
    pos -= 170
    return (pos * 3, 0, 255 - pos * 3)

while True:
    for j in range(255):
        for i in range(3):
            rc_index = (i * 256 // 3) + j * 5
            dots[i] = wheel(rc_index &amp; 255)
        dots.show()
        time.sleep(0.01)
```

Go ahead and save the above code onto your Pi as dotstar_test.py and run it with the following command:

```
python dotstar_test.py
```

The DotStar LEDs should start color-cycling in a rainbow.

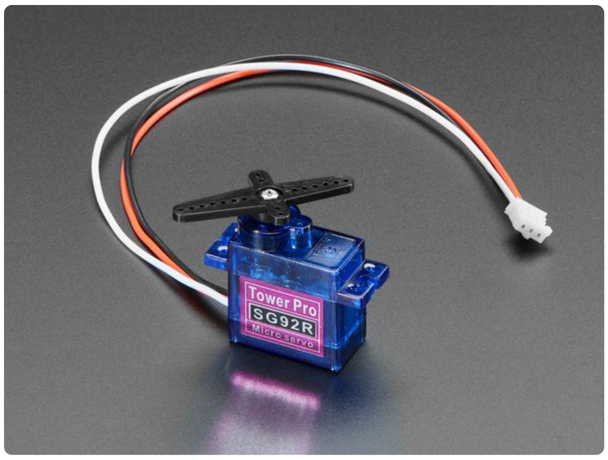
GPIO JST connector



GPIO 12 is accessible with the JST connector on the bottom edge of the Voice Bonnet.

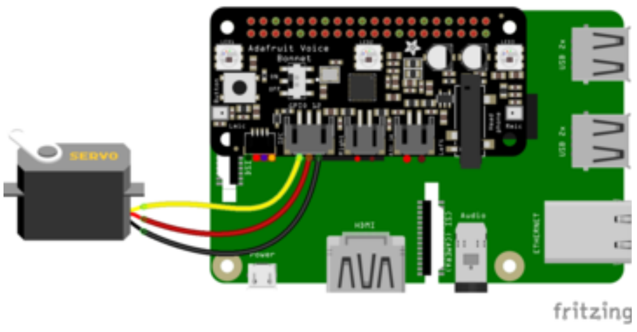
Parts

For this script, we'll just need one part that isn't included with the Voice Bonnet:



Micro Servo with 3-pin JST PH 2mm Cable - TowerPro SG92R
This tiny little servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds you're used to but smaller. You can use any...
https://www.adafruit.com/product/4326

Wiring



Connect the JST PH 3-pin plug into the GPIO #12 on the Voice Bonnet

Download the Fritzing diagram

Code

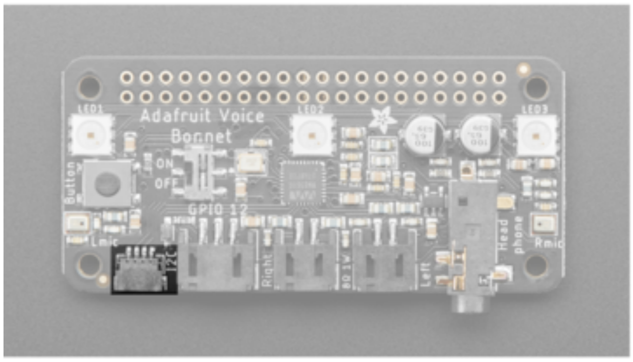
```
import time
import board
import pwmio
from adafruit_motor import servo

SERVO_PIN = board.D12
pwm = pwmio.PWMOut(SERVO_PIN, frequency=50)
servo = servo.Servo(pwm, min_pulse=750, max_pulse=2250)

while True:
    for angle in range(0, 180, 5):  # 0 - 180 degrees, 5 degrees at a time.
        servo.angle = angle
        time.sleep(0.05)
    for angle in range(180, 0, -5): # 180 - 0 degrees, 5 degrees at a time.
        servo.angle = angle
        time.sleep(0.05)
```

Go ahead and save the above code onto your Pi as servo_test.py and run it with the following command:

```
python servo_test.py
```

The servo should start sweeping back and forth in 5 degree increments.

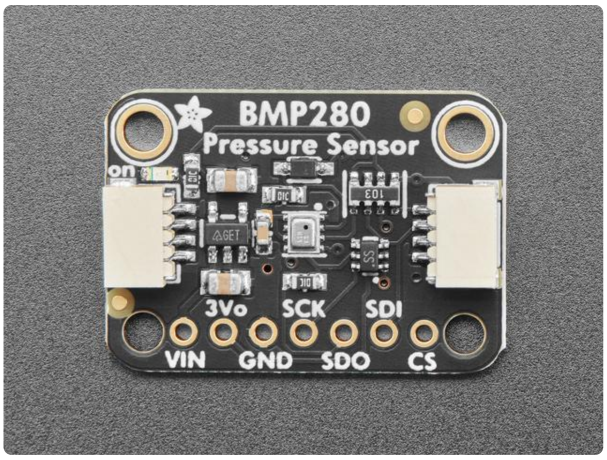
Stemma QT



For the Stemma QT port, you can use any of our 50+ sensors, but we're going to use a script that demonstrates using the BMP280 because it's so simple.

Parts

For this script, we'll just need a BMP280 and a Stemma QT cable:



Adafruit BMP280 I2C or SPI Barometric Pressure & Altitude Sensor

Bosch has stepped up their game with their new BMP280 sensor, an environmental sensor with temperature, barometric pressure that is the next generation upgrade to the...

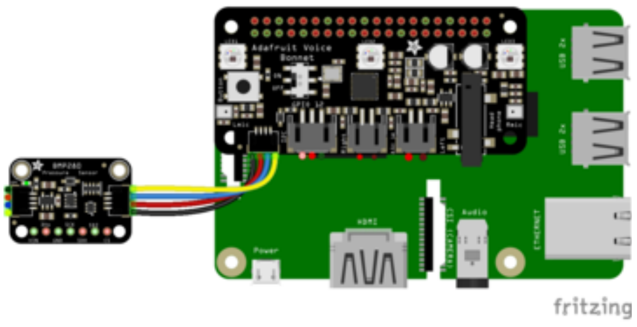
https://www.adafruit.com/product/2651

STEMMA QT / Qwiic JST SH 4-pin Cable - 100mm Long

This 4-wire cable is a little over 100mm / 4" long and fitted with JST-SH female 4-pin connectors on both ends. Compared with the chunkier JST-PH these are 1mm pitch instead of...

https://www.adafruit.com/product/4210

Wiring



Connect one side of the Stemma QT cable to either port on the BMP280

Connect the other side to the Stemma QT port on the Voice Bonnet

Download the Fritzing Diagram

Code

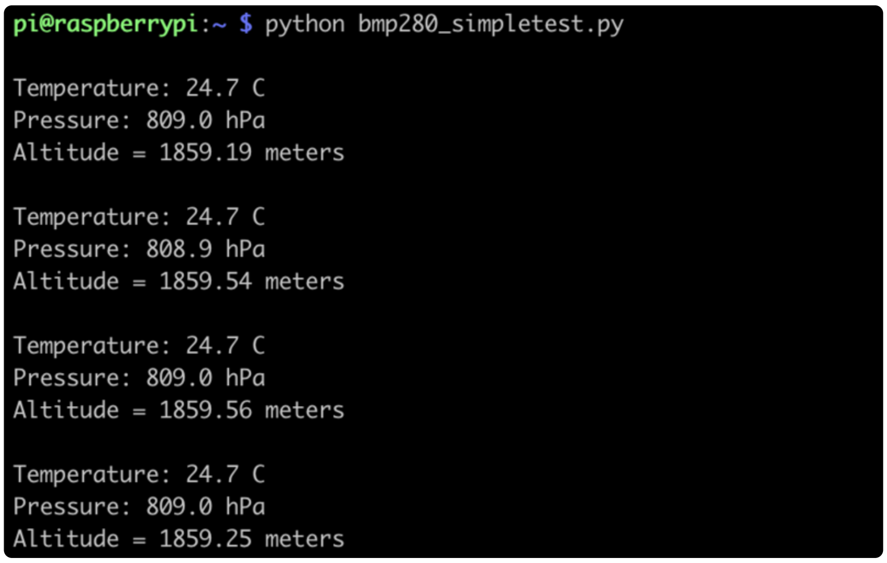
```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT
```

```
"""Simpletest Example that shows how to get temperature,
   pressure, and altitude readings from a BMP280"""
import time
import board

# import digitalio # For use with SPI
import adafruit_bmp280

# Create sensor object, communicating over the board's default I2C bus
i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c)

# OR Create sensor object, communicating over the board's default SPI bus
# spi = board.SPI()
# bmp_cs = digitalio.DigitalInOut(board.D10)
# bmp280 = adafruit_bmp280.Adafruit_BMP280_SPI(spi, bmp_cs)

# change this to match the location's pressure (hPa) at sea level
bmp280.sea_level_pressure = 1013.25

while True:
    print("\nTemperature: %0.1f C" % bmp280.temperature)
    print("Pressure: %0.1f hPa" % bmp280.pressure)
    print("Altitude = %0.2f meters" % bmp280.altitude)
    time.sleep(2)
```

Go ahead and save the above code onto your Pi as bmp280_simpletest.py and run it with the following command:

```
python bmp280_simpletest.py
```

The terminal should start printing out the detected measurements.



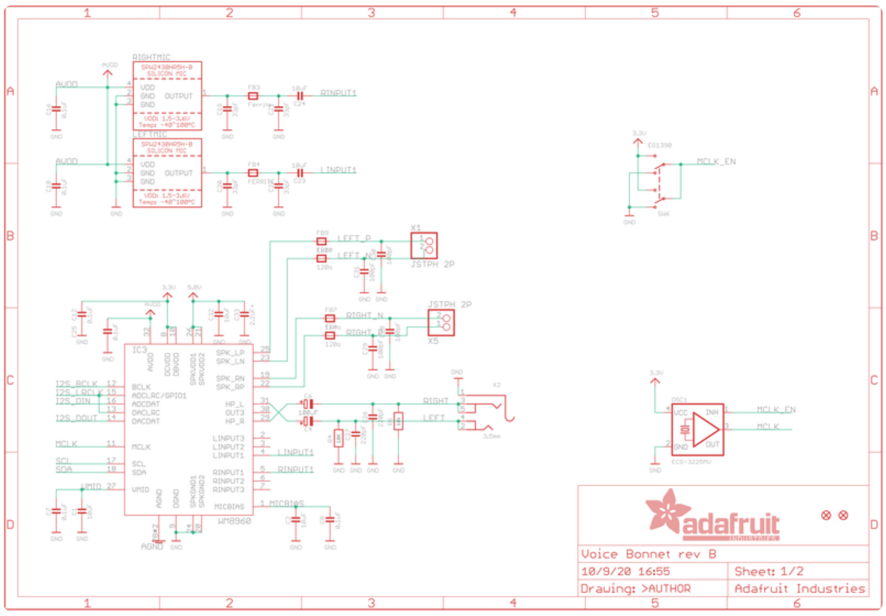
```
pi@raspberrypi:~ $ python bmp280_simpletest.py

Temperature: 24.7 C
Pressure: 809.0 hPa
Altitude = 1859.19 meters

Temperature: 24.7 C
Pressure: 808.9 hPa
Altitude = 1859.54 meters

Temperature: 24.7 C
Pressure: 809.0 hPa
Altitude = 1859.56 meters

Temperature: 24.7 C
Pressure: 809.0 hPa
Altitude = 1859.25 meters
```

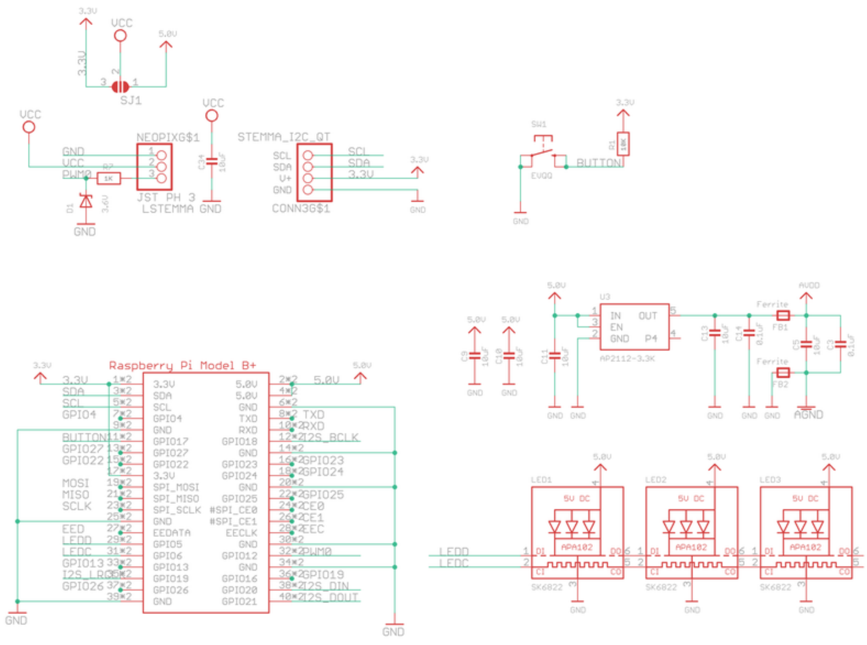
Downloads

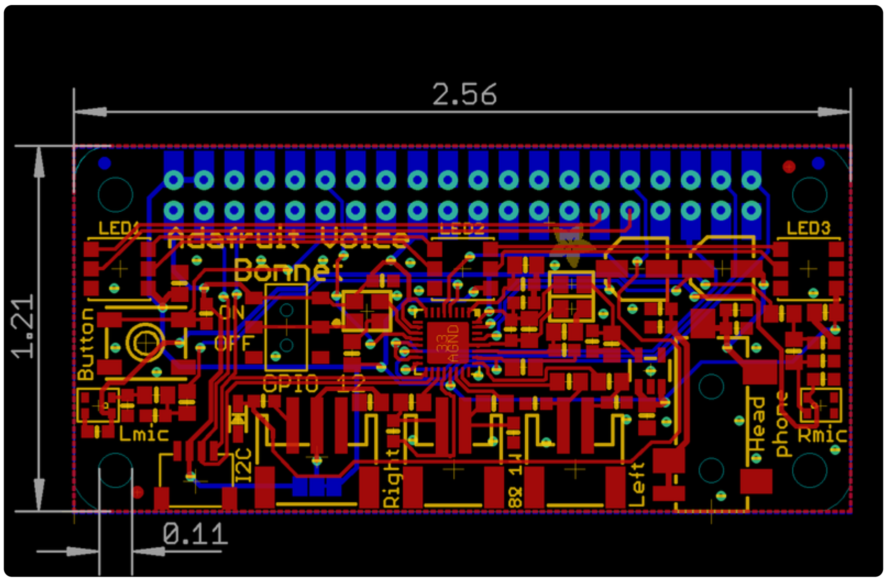
Files:

- WM8960 datasheet ()
- EagleCAD files on GitHub ()
- Fritzing object in the Adafruit Fritzing Library ()

Schematic and Fab Print






2.56
1.21
0.11
Adafruit Voice Bonnet
LED1
LED2
LED3
Button
ON
OFF
GPIO 12
Lmic
Rmic
I2C
Right
8Ω 1W
Left
Headphone
33
AGND