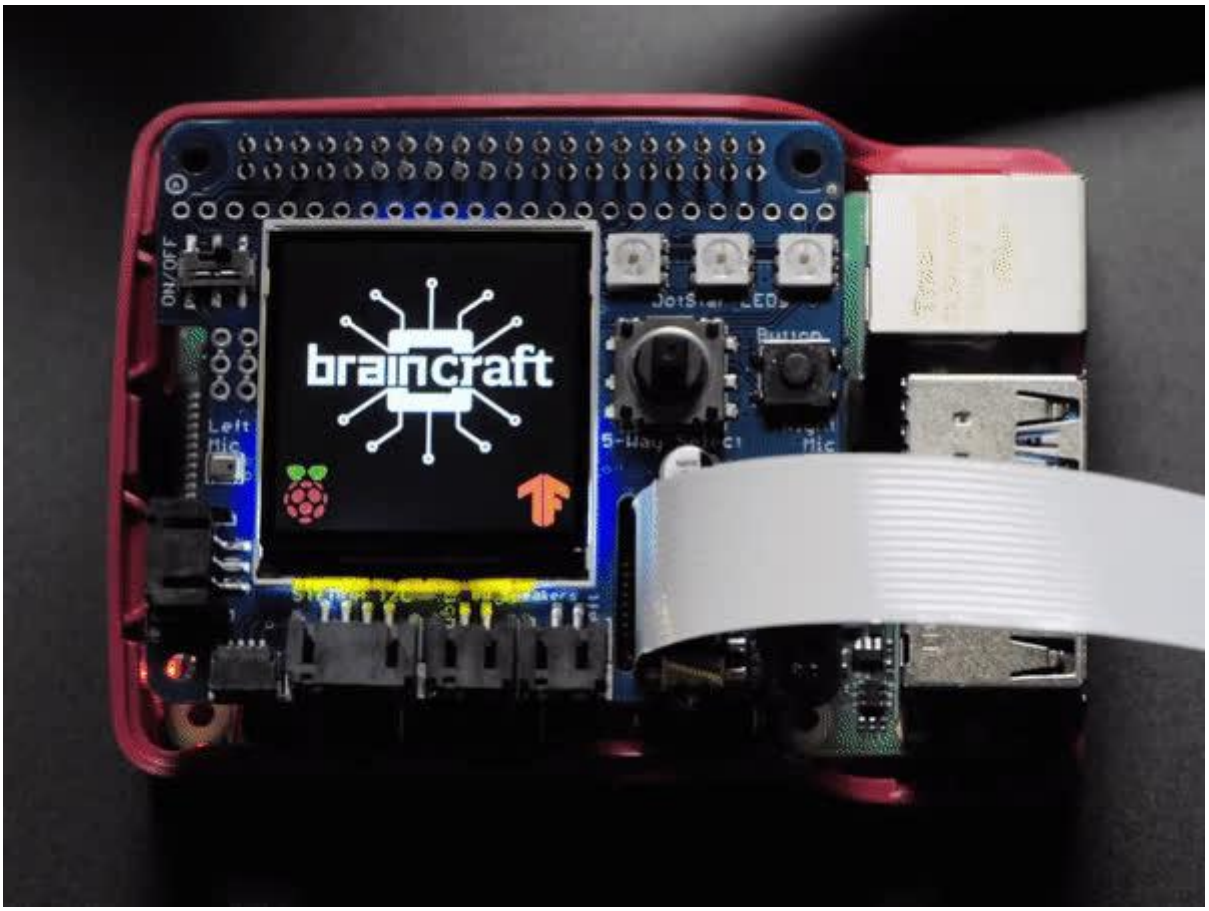
Adafruit BrainCraft HAT - Easy Machine Learning for Raspberry Pi

Created by lady ada

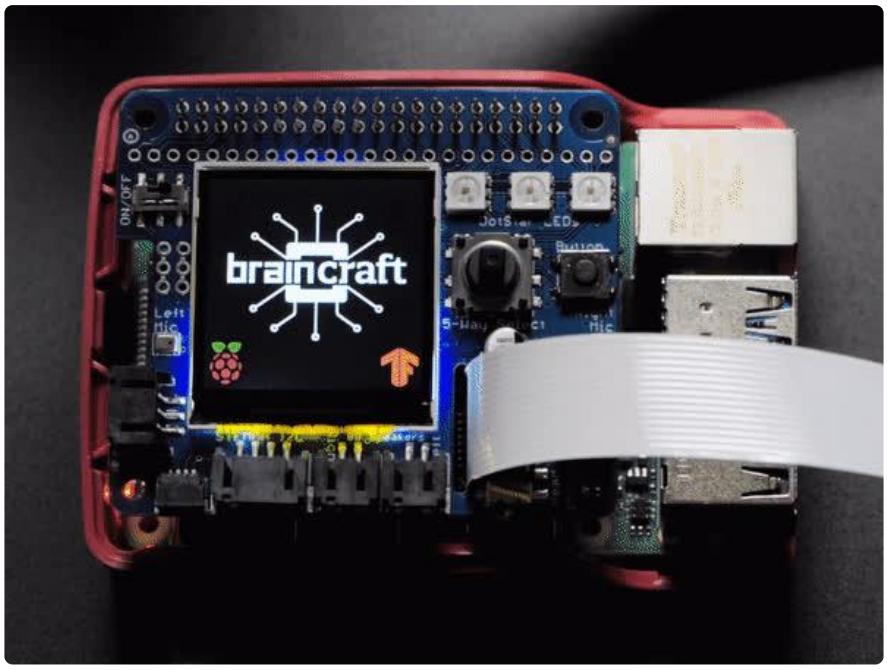


https://learn.adafruit.com/adafruit-braincraft-hat-easy-machine-learning-for-raspberry-pi

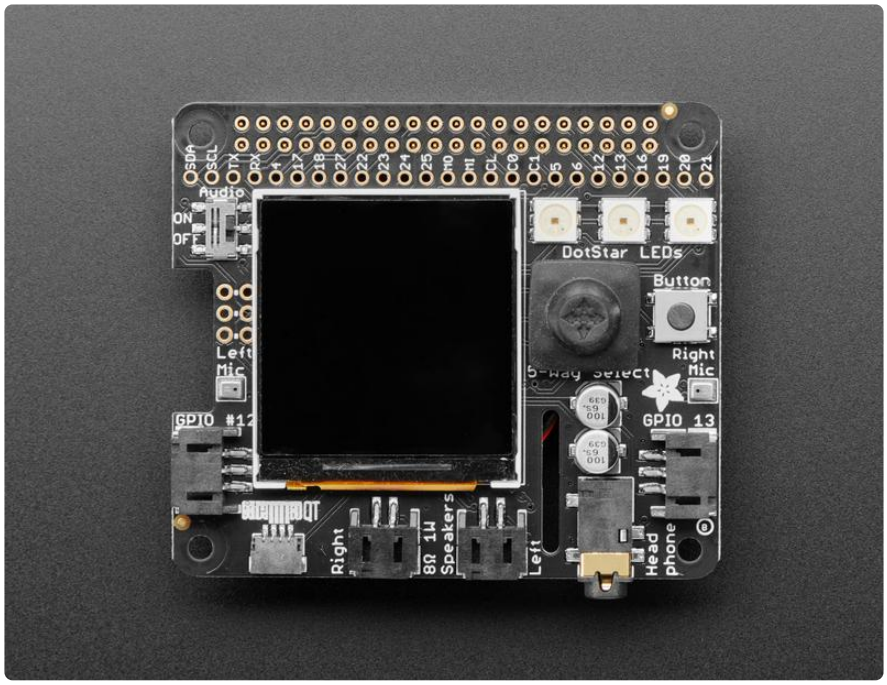
Last updated on 2023-01-05 12:57:07 PM EST

Table of Contents

Overview

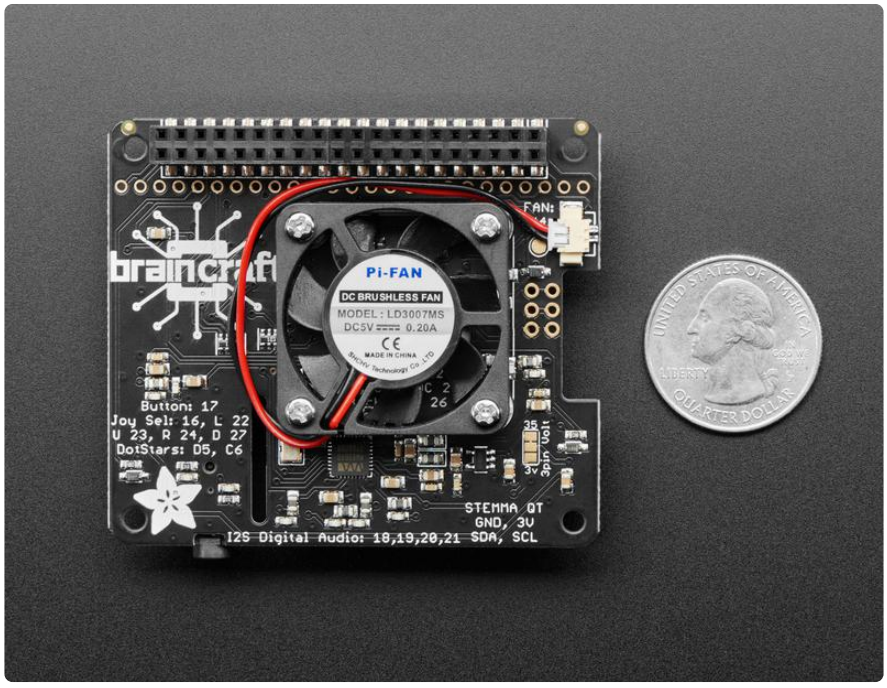


The idea behind the Adafruit BrainCraft HAT is that you'd be able to "craft brains" for Machine Learning on the EDGE, with Microcontrollers & Microcomputers. On ASK AN ENGINEER, our founder & engineer chatted with Pete Warden, the technical lead of the mobile, embedded TensorFlow Group on Google's Brain team about what would be ideal for a board like this.

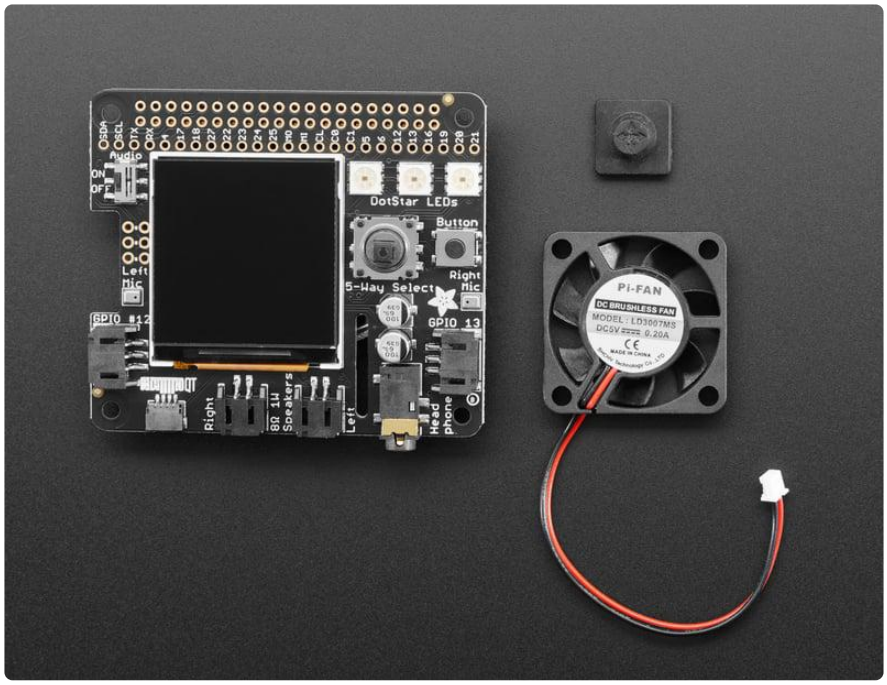


And here's what we designed! The BrainCraft HAT has a 240×240 TFT IPS display for inference output, slots for a camera connector cable for imaging projects, a 5 way

joystick and button for UI input, left and right microphones, stereo headphone out, a stereo 1 Watt speaker out, three RGB DotStar LEDs, two 3 pin STEMMA connectors on PWM pins so they can drive NeoPixels or servos, and Grove/STEMMA/Qwiic I2C port. This will let people build a wide range of audio/video AI projects while also allowing easy plug-in of sensors and robotics!

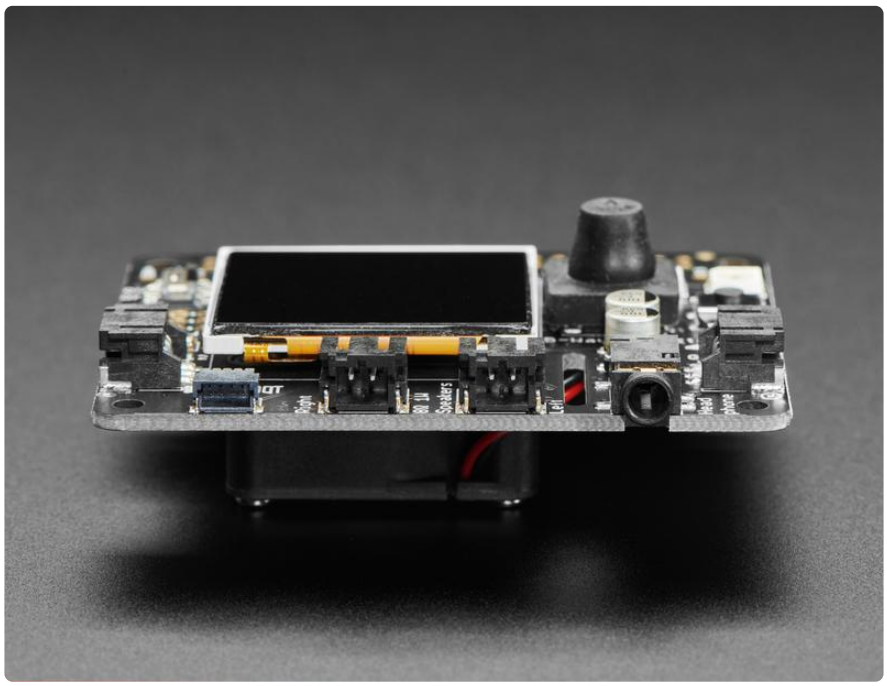


A controllable mini fan attaches to the bottom, and can be used to keep your Pi cool while doing intense AI inference calculations. Most importantly, there's an On/Off switch that will completely disable the audio codec, so that when it's off, there's no way it's listening to you.



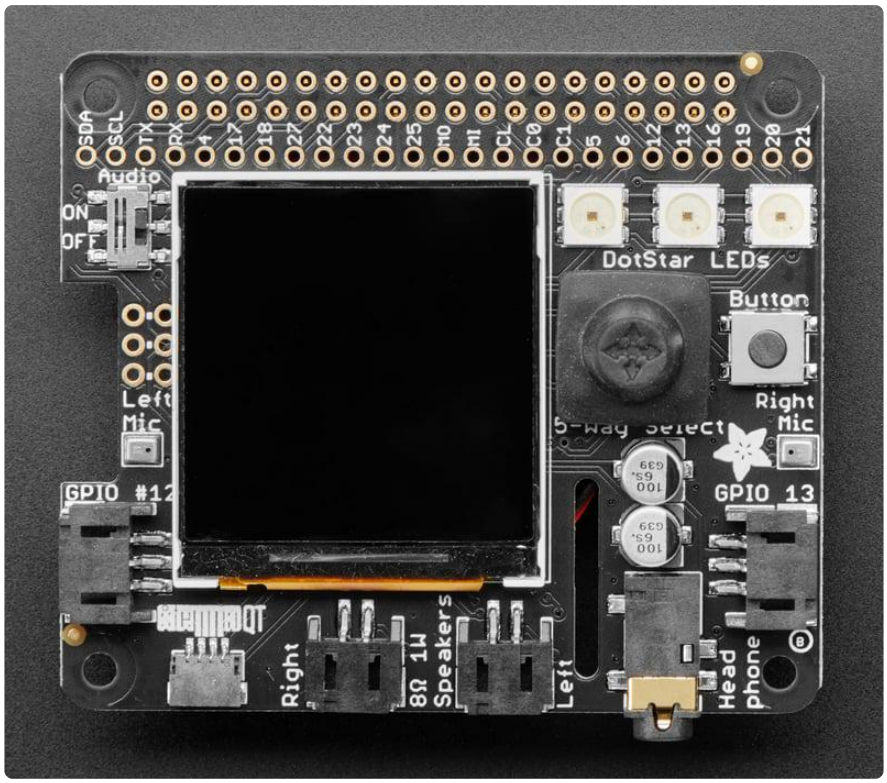
Features:

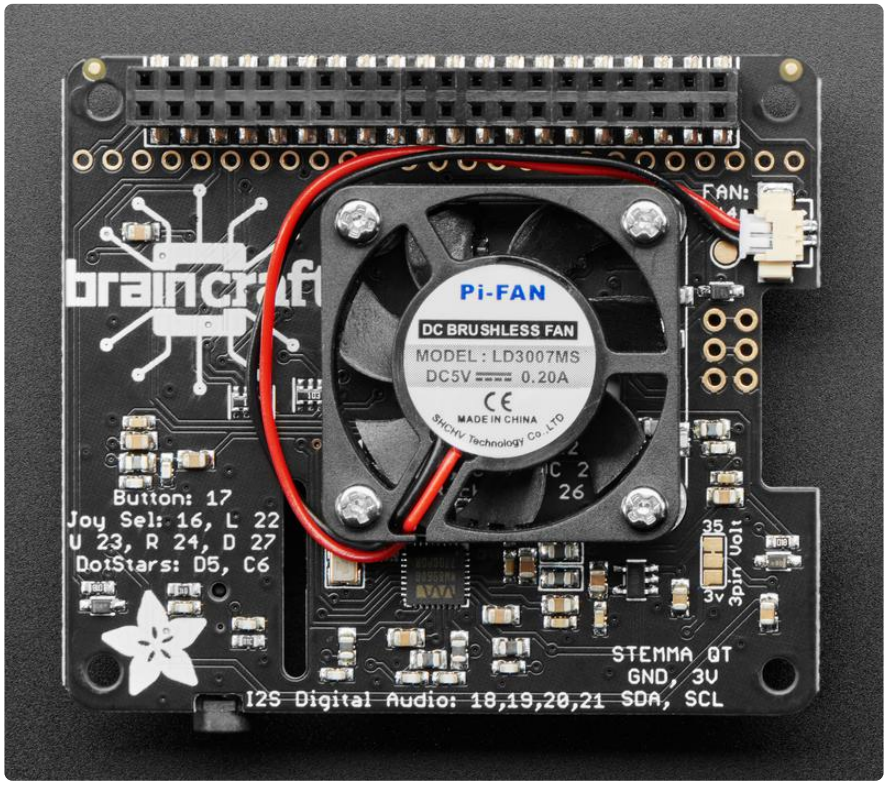
- 1.54" IPS TFT display with 240x240 resolution that can show text or video
- Stereo speaker ports for audio playback - either text-to-speech, alerts or for creating a voice assistant.
- Stereo headphone out for audio playback through a stereo system, headphones, or powered speakers.
- Stereo microphone input - perfect for making your very own smart home assistants
- Two 3-pin JST STEMMA connectors that can be used to connect more buttons (), a relay (), or even some NeoPixels! ()
- STEMMA QT plug-and-play I2C port (), can be used with any of our 50+ I2C STEMMA QT boards (), or can be used to connect to Grove I2C devices with an adapter cable ().
- 5-Way Joystick + Button for user interface and control.
- Three RGB DotStar LEDs for colorful LED feedback.

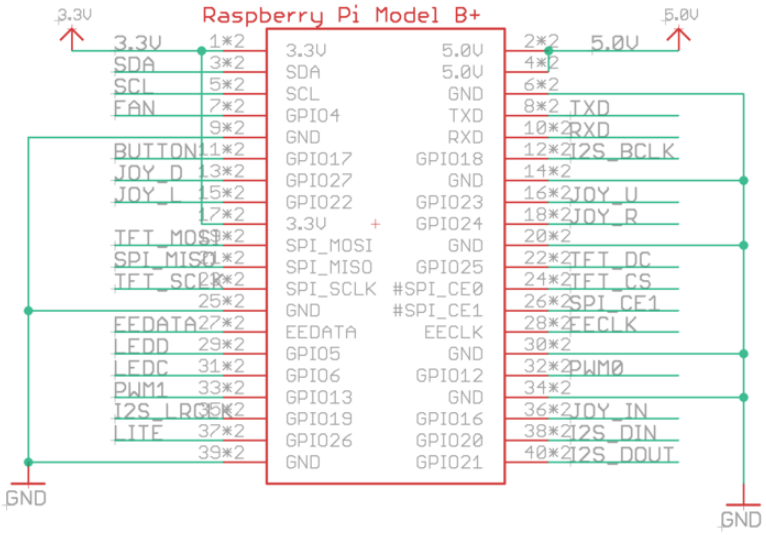


The STEMMA QT port means you can attach heat image sensors like the Panasonic Grid-EYE () or MLX90640 (). Heat-Sensitive cameras can be used as a person detector, even in the dark! An external accelerometer can be attached for gesture or vibration sensing such as machinery/industrial predictive maintenance projects

Pinouts


SDA
SCL
TX
RX
4
17
18
27
22
23
24
25
MO
MI
CL
C0
C1
5
6
12
13
16
19
20
21
Audio
ON
OFF
DotStar LEDs
Button
Left
Mic
Right
Mic
5-Way Select
GPIO #12
GPIO 13
STEMMA QT
Right
8Ω 1W
Speakers
Left
Head
Phone

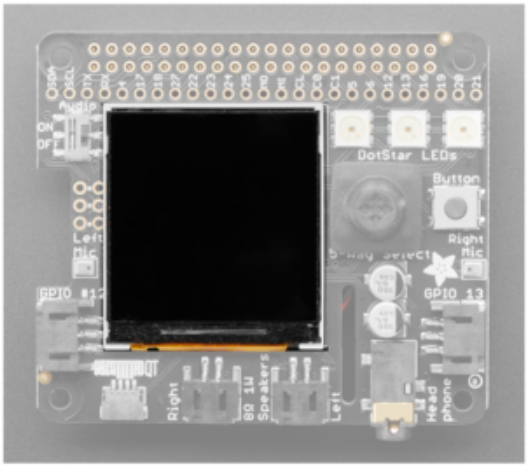

FAN:
Pi-FAN
DC BRUSHLESS FAN
MODEL : LD3007MS
DC5V 0.20A
MADE IN CHINA
Button: 17
Joy Sel: 16, L 22
U 23, R 24, D 27
DotStars: D5, C6
35
3pin Volt
3v
STEMMA QT
GND, 3V
I2S Digital Audio: 18,19,20,21 SDA, SCL



Power

- 5.0V - Connected to the display backlight
- 3.3V - Connected to the display power and also the STEMMA QT / Qwiic connector
- GND - Ground for everything

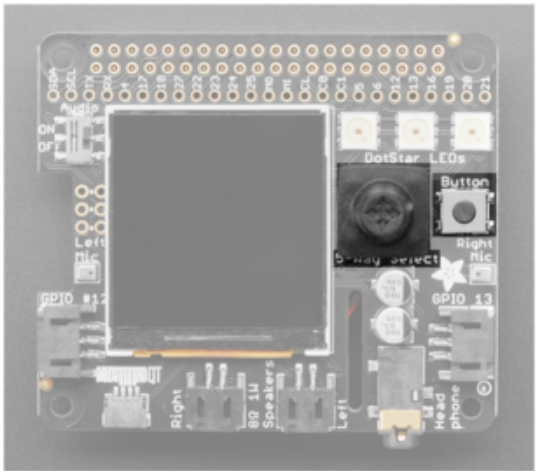
Display



The display is a 1.54" wide-angle TFT LCD.

MOSI, SCK, GPIO #25, CE0 - These are the display control pins. Note that MISO is not connected, even though it is an SPI pin, because you cannot read back from the display.
GPIO #26 - This is the display backlight pin. Used to turn on and off the backlight.

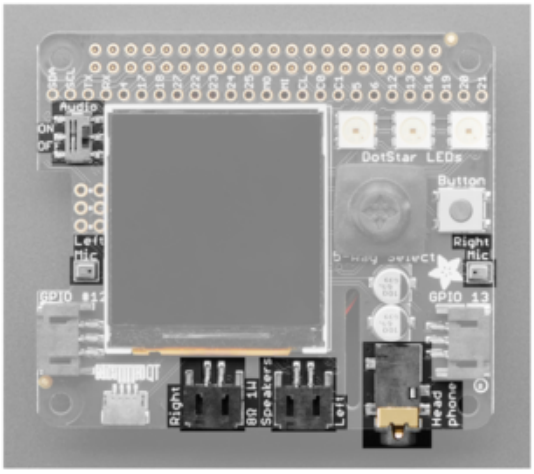
Buttons and Joystick



These pins have a 10K pullup to 3.3V so when the button is pressed or the joystick is moved/pressed, you will read a LOW voltage on these pins.

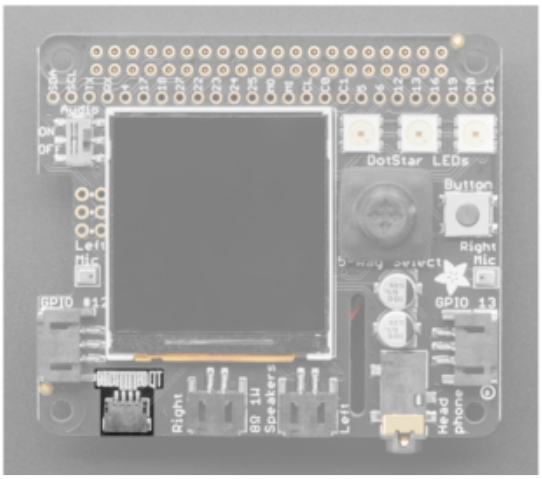
GPIO#17 - Button
GPIO #16 - Joystick select
GPIO #22 - Joystick left
GPIO #23 - Joystick up
GPIO #24 - Joystick right
GPIO #27 - Joystick down

Sound



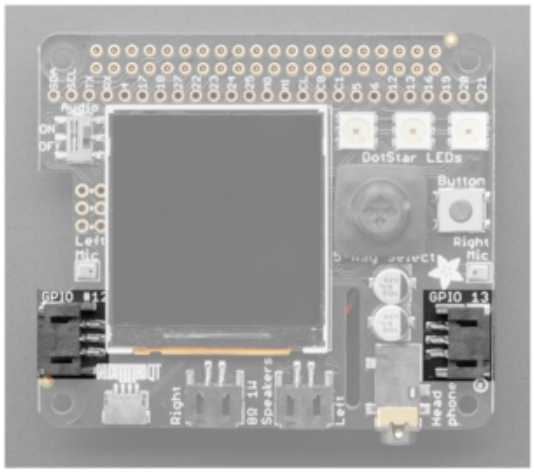
GPIO #18, GPIO#19, GPIO #20, GPIO #21 - I2S Digital Audio.
Speakers - The two speaker connectors are located on the bottom in the middle, labeled Right and Left. Plug in an enclosed speaker () or use a JST 2PH cable to attach custom speakers ().
Microphones - The two mics are in the middle of each side,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

STEMMA QT Connector



SCL, SDA - I2C data for the STEMMA QT / Qwiic connector. Not used by buttons or display. Can also use with Grove sensors with an adapter cable (). Great for quickly adding sensors or accessories with plug-and-play.

Digital/Analog Connectors



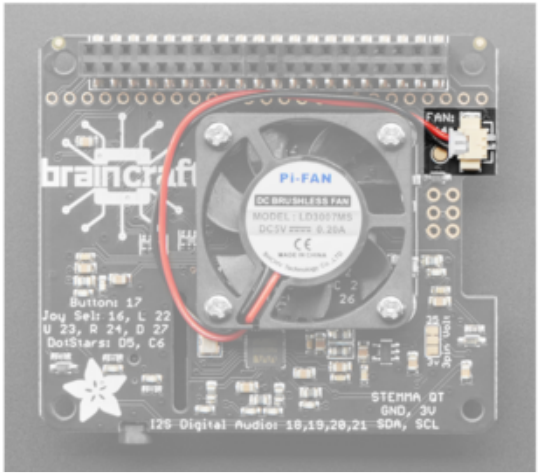
GPIO#12 and GPIO #13 - 3-pin JST digital or analog connectors. Easily connect things like NeoPixels () or servos () using this 2mm connector. You can also use a generic JST cable to connect to a breadboard. ()

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.

Fan



GPIO #4 - The fan is on this pin.

Required Parts

Before you start, you'll need some parts to make your BrainCraft HAT run! At a minimum you will need

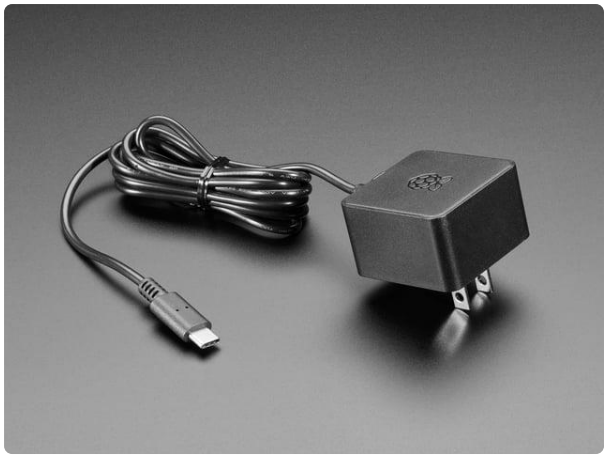
Raspberry Pi Computer

A working Raspberry Pi (Pi 4+ needed for vision projects, any Pi for audio-only projects)

Raspberry Pi 4 Model B - 2 GB RAM
NOTE: Due to stock limitations we may only be able to offer refunds or store credit for Pis that are defective, damaged or lost in...
https://www.adafruit.com/product/4292

Power supply

[5V 2.5A Switching Power Supply with 20AWG MicroUSB Cable]
Our all-in-one 5V 2.5 Amp + MicroUSB cable power adapter is the perfect choice for powering single-board computers like Raspberry Pi, BeagleBone, or anything else that's...
https://www.adafruit.com/product/1995



[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! Superfast with just the right amount of cable length to get your Pi 4...
https://www.adafruit.com/product/4298

And an SD card. It can be blank or come with software on it.

[SD/MicroSD Memory Card (8 GB SDHC)]
Add mega-storage in a jiffy using this 8 GB class 4 micro-SD card. It comes with a SD adapter so you can use it with any of our shields or adapters. Preformatted to FAT so it works out...
https://www.adafruit.com/product/1294

For Vision projects

you'll also want a Raspberry Pi camera, a long cable (optional) and camera case (optional)

[Raspberry Pi Camera Board v2 - 8 Megapixels]
Snap, snap! The Camera v2 is the new official camera board released by the Raspberry Pi Foundation!The Raspberry Pi Camera Board v2 is a high quality 8...
https://www.adafruit.com/product/3099

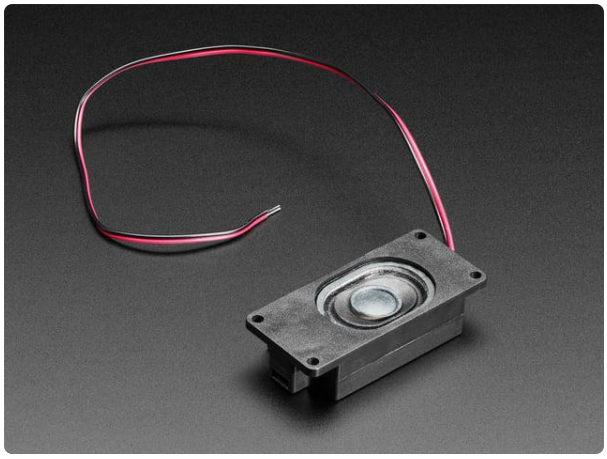
[Flex Cable for Raspberry Pi Camera or Display - 1 meter]
This cable will let you swap out the stock 150mm long flex cable from a Raspberry Pi Camera (either 'classic' or 'NoIR' type) or Raspberry Pi Display for a different...
https://www.adafruit.com/product/2143

[Adafruit Raspberry Pi Camera Board Case with 1/4" Tripod Mount]
This is a basic, classic Adafruit Raspberry Pi Camera Board Enclosure with a black base and a clear top. The case is as minimal as it gets, coming in just...
https://www.adafruit.com/product/3253

For Audio Projects

You'll want at least a set of headphones or plug-in speakers!

Cell-phone TRRS Headset - Earbud Headphones w/ Microphone
These earbud headphones are the perfect accessory for your FONA - they've been tested to work with our modules - but can be used with any iOS or Android device that uses a TRRS...
https://www.adafruit.com/product/1966



Mono Enclosed Speaker with Plain Wires - 3W 4 Ohm
Listen up! This single 2.8" x 1.2" speaker is the perfect addition to any audio project where you need 4 ohm impedance and 3W or less of power. We...
https://www.adafruit.com/product/4445

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

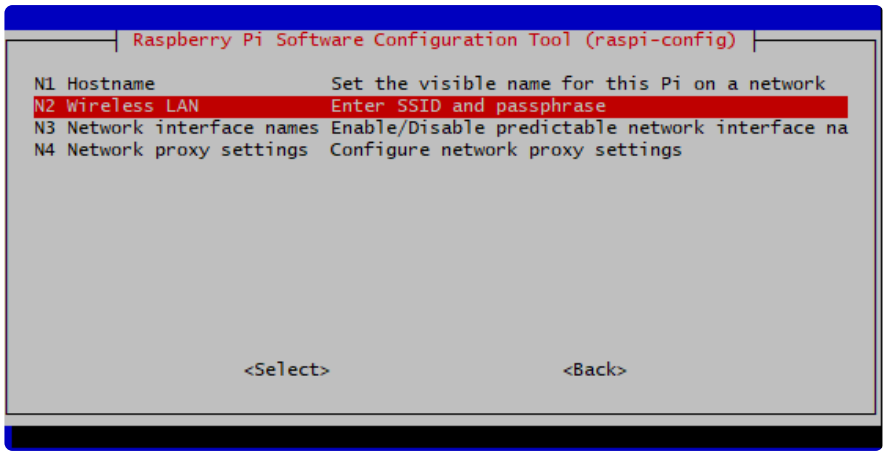
You'll need to be able to log into your Pi, either enable SSH access (and use and Ethernet cable) (), use a USB to serial cable, or connect a monitor and keyboard. Basically get it so you can log in.

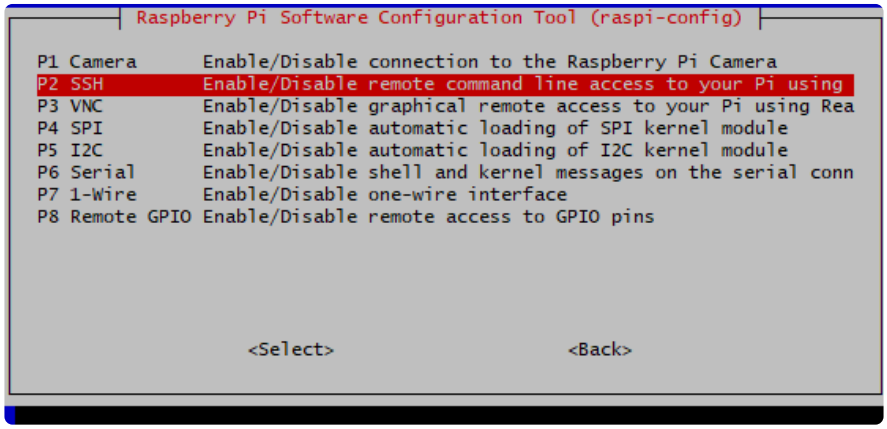
We have a quickstart guide here () and here that you can follow (), or there's dozens of online guides. it is assumed by the next step you are able to log in and type commands in - ideally from a desktop computer, so you can copy and paste in some of the very long commands!

Step 3 - Log in & Enable Internet

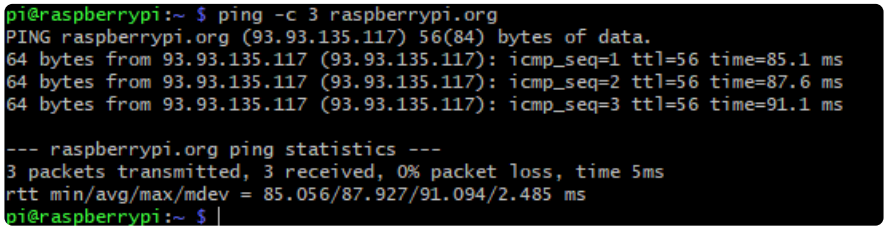
Once you've logged in, enable WiFi (if you have built in WiFi) with sudo raspi-config () so you can ssh in.

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





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.



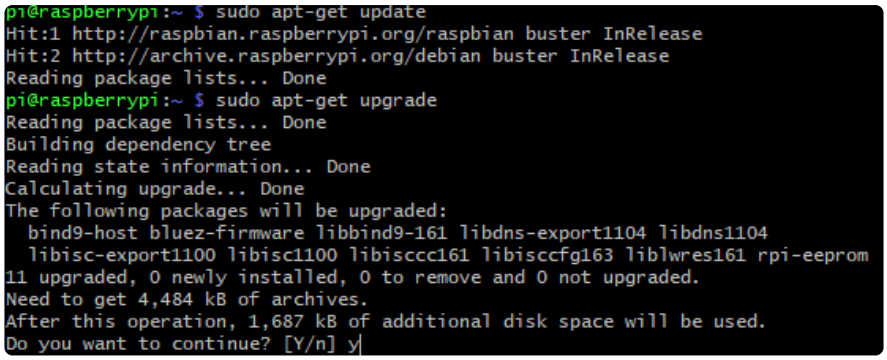
Step 4 - Update/Upgrade

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
```



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

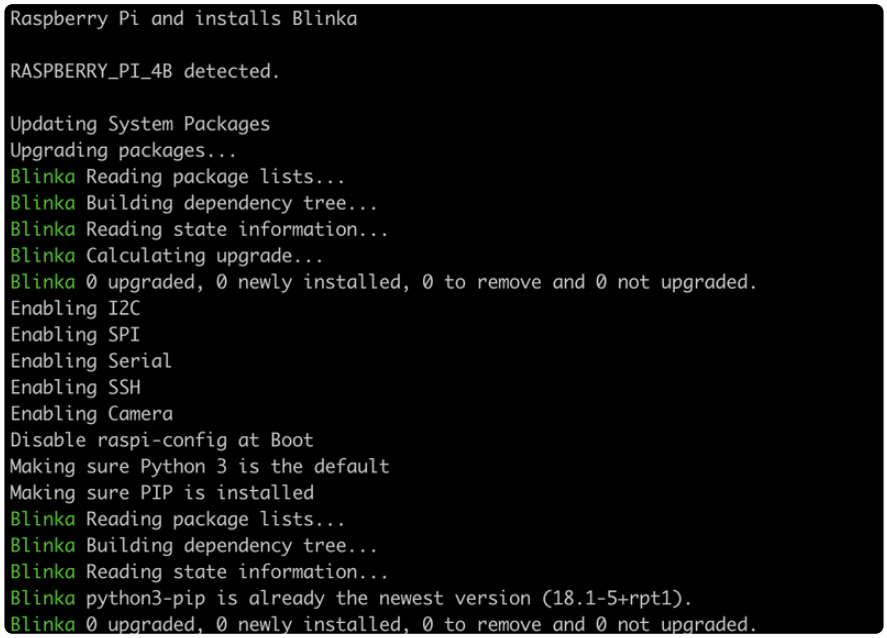
Blinka Setup

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 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.



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

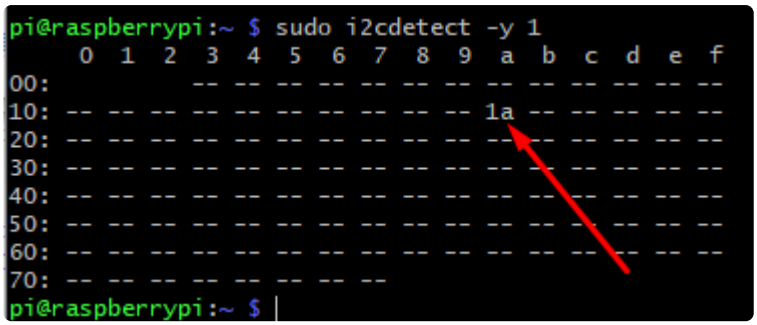
Start by making sure you've installed I2C support, see the previous page (Blinka Setup) on how to do it!

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



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`.

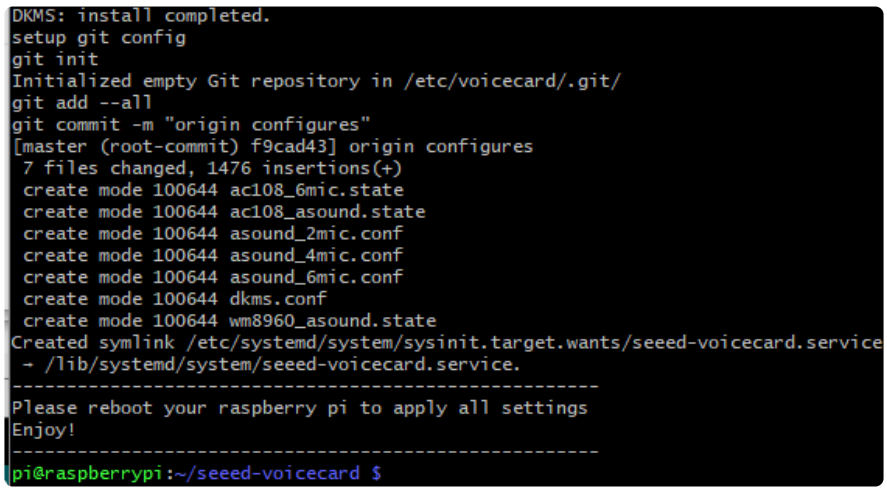
If your have a more recent kernel version of 5.10 or higher, use the following:

```
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:



Reboot with

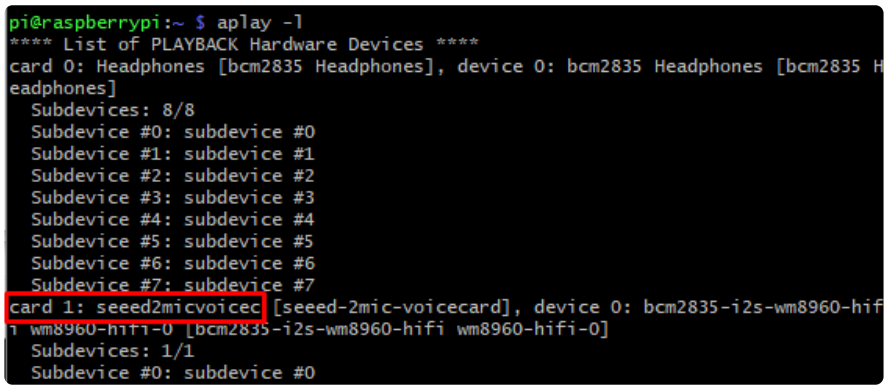
`sudo reboot`

and on reboot run

`sudo aplay -l`

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

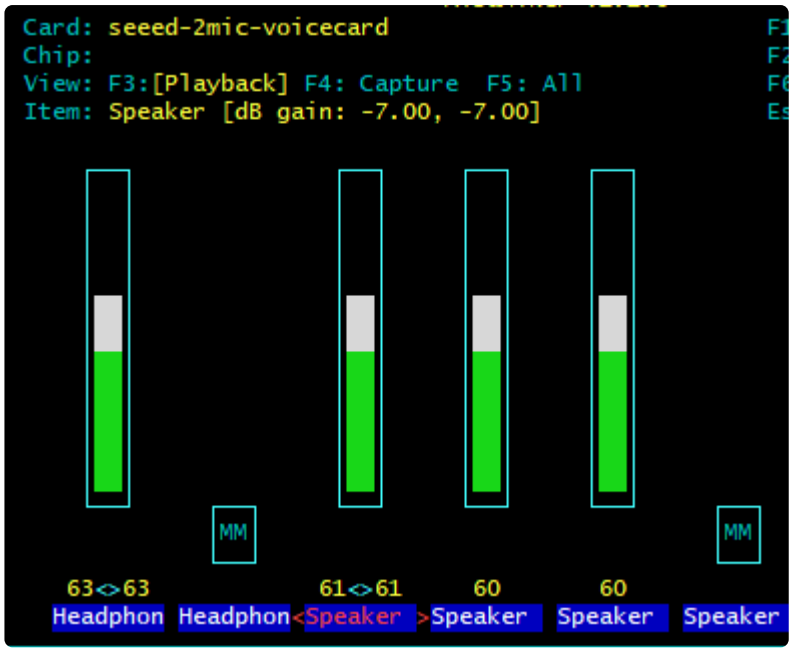
> On a Raspberry Pi 4, your card number may be Card 2 instead of 1 because of the second HDMI port. You'll need to make some changes to some of the commands to reflect this further down.



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

A gain of about 60% is plenty loud!



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

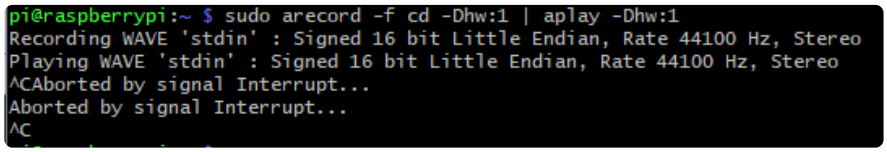
There are two microphones, and now we can test that they work. This test is best done with headphones, not using the speaker port, because it can cause a painful feedback effect if the speakers are next to the mics!

Run:

`sudo arecord -f cd -Dhw:1 | aplay -Dhw:1`

If your sound card ID is not #1, 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!



`Control-C` to quit when done

Your audio subsystem is now completely tested!

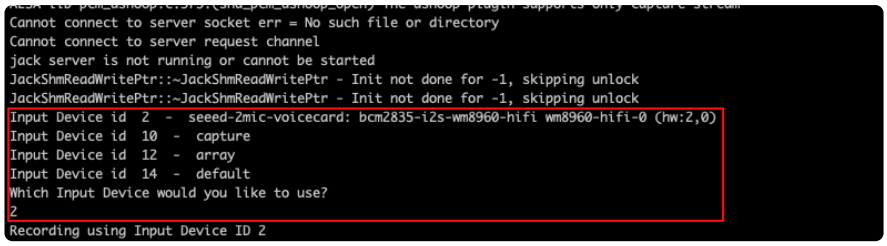
Python Libraries

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 -y 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.



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
```

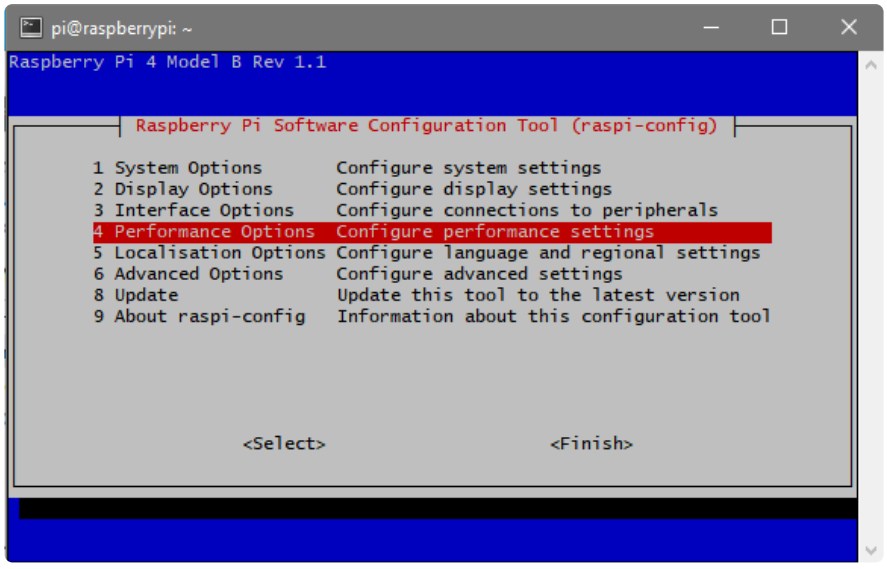
Fan Service Setup

We have a really simple fan service that will control the onboard fan. The reason we have it set up as a service instead of keeping the fan on all the time is so that it doesn't drain too much power from the Pi during the initial power on.

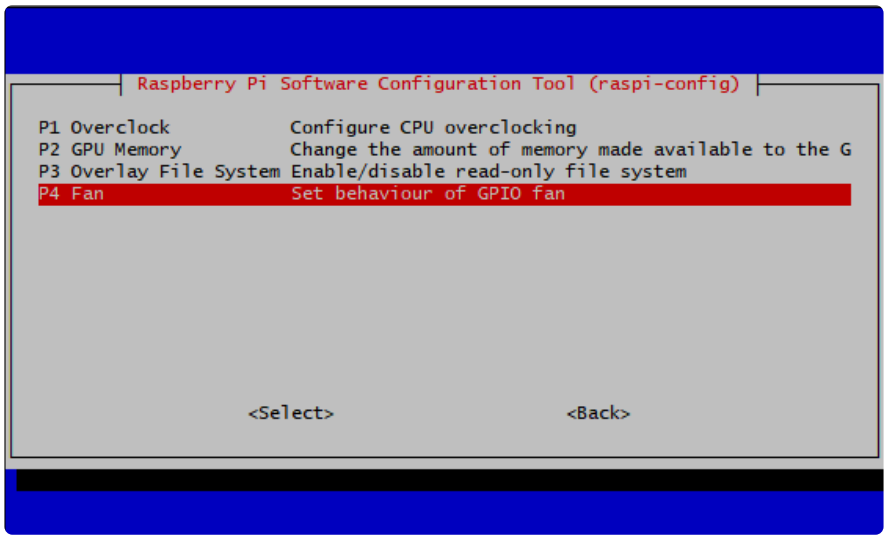
The fan service basically controls turning GPIO 4 on at startup, which is what the fan is connected to. Installing the fan service is really simple and we have a script for doing that.

To install, just type sudo raspi-config

Select Performance Options



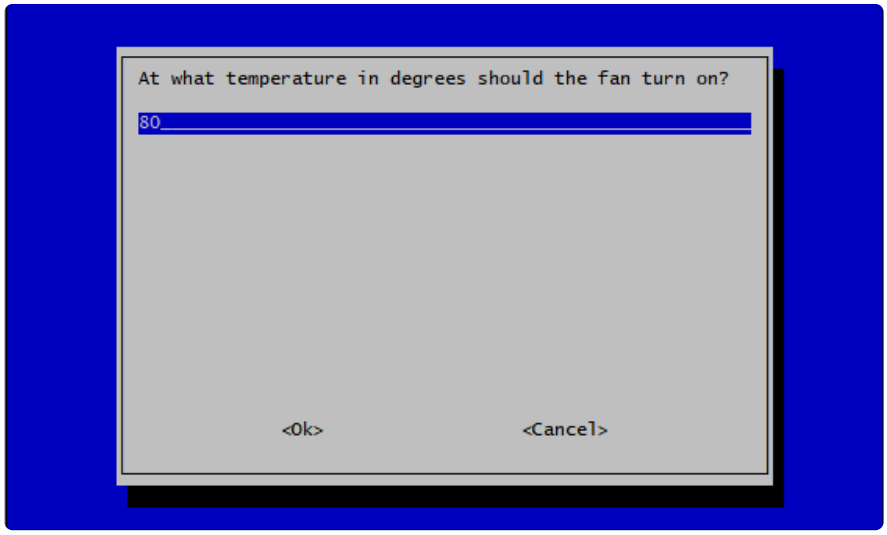
Select Fan



Select Yes

And make sure you put down GPIO pin 4 for the fan

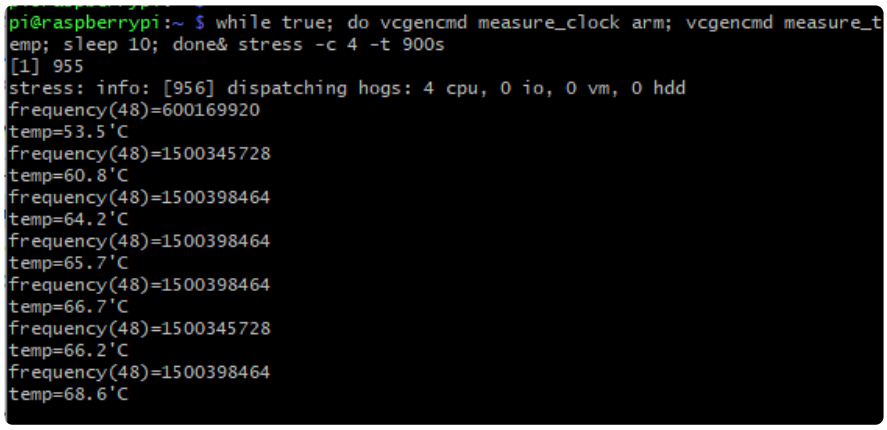
You can customize the fan temperature setting



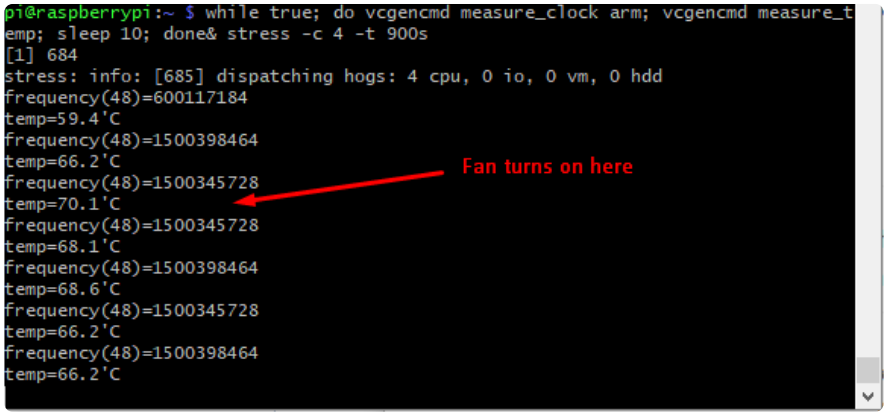
That's it!

You can then 'stress test' by running

- `sudo apt-get install stress`
- `while true; do vcgencmd measure_clock arm; vcgencmd measure_temp; sleep 10; done& stress -c 4 -t 900s`



When the temperature hits the limit you set earlier, the fan should turn on, and cool the pi back down (in this case I set it to 70 C):



Display Module Install

There's two ways you can use the 1.54" 240x240 display on the BrainCraft HAT. For machine learning purposes, the advanced method is the way to go, so that's what we'll be covering in this guide.

Be aware that you can only choose to do one way at a time. If you choose the advanced way, it will install the kernel driver, which will prevent you from doing it the easy way without uninstalling the driver first.

The easy way is to use 'pure Python 3' and Pillow library to draw to the display from within Python. This is great for showing text, stats, images etc that you design yourself. If you want to do that, the BrainCraft HAT has a pretty close layout to the Adafruit 1.3" Color TFT Bonnet () including the same type of display and a joystick, though the pinouts are slightly different. If you choose this option, You can skip this page and view the Python Setup page () for instruction for that display.

The advanced way is to install a kernel module to add support for the TFT display that will make the console appear on the display. This is cute because you can have any program print text or draw to the framebuffer (or, say, with pygame) and Linux will take care of displaying it for you. If you don't need the console or direct framebuffer access, please consider using the 'pure Python' technique instead as it is not as delicate.

If you plan on using the Pi Camera for vision projects, you will need to go with the advanced route!

Installing The 1.54" Kernel Module

We have tried to make this as easy as possible for you by providing a script that takes care of everything. There's only a couple of dependencies needed. To get everything setup, just run the following at the terminal:

```
cd ~
sudo pip3 install --upgrade adafruit-python-shell click
sudo apt-get install -y git
git clone https://github.com/adafruit/Raspberry-Pi-Installer-Scripts.git
cd Raspberry-Pi-Installer-Scripts
sudo python3 adafruit-pitft.py --display=st7789_240x240 --rotation=0 --install-
type=fbcp
```

If you want to use the BrainCraft HAT for vision projects, you will need to install the display driver as FBCP and not console.

```
pi@raspberrypi:~ $ cd ~
pi@raspberrypi:~ $ sudo pip3 install --upgrade adafruit-python-shell click==7.0
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting adafruit-python-shell
  Using cached https://www.piwheels.org/simple/adafruit-python-shell/adafruit_python_shell-1.2.0-py3-none-any.whl
Collecting click==7.0
  Using cached https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d7257104c434fe0b07e5a195a6847506c074527aa599ec/Click-7.0-py2.py3-none-any.whl
Requirement already satisfied, skipping upgrade: clint in /usr/local/lib/python3.7/dist-packages (from adafruit-python-shell) (0.5.1)
Requirement already satisfied, skipping upgrade: Adafruit-PlatformDetect in /usr/local/lib/python3.7/dist-packages (from adafruit-python-shell) (2.17.0)
Requirement already satisfied, skipping upgrade: args in /usr/local/lib/python3.7/dist-packages (from clint->adafruit-python-shell) (0.1.0)
Installing collected packages: adafruit-python-shell, click
Successfully installed adafruit-python-shell-1.2.0 click-7.0
pi@raspberrypi:~ $ sudo apt-get install -y git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.20.1-2+deb10u3).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~ $ git clone https://github.com/adafruit/Raspberry-Pi-Installer-Scripts.git
Cloning into 'Raspberry-Pi-Installer-Scripts'...
remote: Enumerating objects: 30, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 544 (delta 13), reused 18 (delta 7), pack-reused 514
Receiving objects: 100% (544/544), 181.33 KiB | 1.63 MiB/s, done.
```

When you get asked to reboot, reboot!

```
############## UPGRADING KERNEL ###############
Updating packages...
Upgrading packages...
Installing Kernel Headers...
PITFT make: Entering directory '/usr/src/linux-headers-5.4.51-v7l+'
PITFT Building modules, stage 2.
PITFT MODPOST 2 modules
PITFT LD [M]  /home/pi/Raspberry-Pi-Installer-Scripts/st7789_module/fb_st7789v.ko
PITFT LD [M]  /home/pi/Raspberry-Pi-Installer-Scripts/st7789_module/st7789v_ada.ko
PITFT make: Leaving directory '/usr/src/linux-headers-5.4.51-v7l+'
PITFT Success!

Settings take effect on next boot.

REBOOT NOW? [Y/n] y
```

That's it! You will now have the BrainCraft HAT with a console display on it

---

Display Module Troubleshooting

The latest Raspberry Pi Bullseye release is new and may have issues with the PiTFT. In that case, you can try the previous buster release.

Bullseye Desktop Version Breaking Changes

Raspberry Pi recently release a new major version of Raspberry Pi OS called Bullseye. In our testing the desktop version, which is the default installation with the Raspberry Pi imager, it may not work.

The last known for-sure tested-and-working version is May 28, 2021 (https://downloads.raspberrypi.org/raspios_armhf/images/raspios_armhf-2021-05-28/ ()) from https://downloads.raspberrypi.org/raspios_armhf/images/ ().

We have applied a fix, but it hasn't been thoroughly tested. Please let us know if you are having issues and you can use the previous release in the meantime.

Static Issue

The Raspberry Pi Kernel sometimes updates firmware, which can which can break the Frame Buffer Copy mechanism. In this particular case, it only seems to affect the Raspberry Pi 4. The issue appears as a garbled screen that looks like static.

To check your kernel version, run the following command:

```
dpkg -l raspberrypi-kernel
```

You should see output similar to the following. If the kernel version is later than 1:1.20210527, then the following fix should work.

```
pi@raspberrypi:~ $ dpkg -l raspberrypi-kernel
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name               Version        Architecture Description
+++-==================-==============-============-=================================
ii  raspberrypi-kernel 1:1.20210831-1 armhf        Raspberry Pi bootloader
```

We have a script that is able to set the kernel version to the kernel version prior to it breaking. To "pin" the kernel version to an older version prior to it breaking, you'll need to run a few commands. You can either SSH into the Pi or hook up an HDMI cable, though the display may appear small.

Once you'd at a command prompt, run the following commands. Note that the `1:` prefix in the version number is on purpose because of the way that pinning was recently changed.

```
cd ~
sudo pip3 install --upgrade adafruit-python-shell
wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/main/rpi_pin_kernel_firmware.py
sudo python3 rpi_pin_kernel_firmware.py 1:1.20210527-1
```

After it finishes, reboot the Pi.

Once the Pi is back up, the display may appear inverted. To fix this, just run the Adafruit PiTFT script again and reboot a second time.

You can check the new kernel version by running the `dpkg` command again:

```
dpkg -l raspberrypi-kernel
```

This time, your version should be 1:1.20210527-1.

```
pi@raspberrypi:~ $ dpkg -l raspberrypi-kernel
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name             Version        Architecture Description
+++-================-==============-============-=================================
ii  raspberrypi-kernel 1:1.20210527-1 armhf        Raspberry Pi bootloader
```

BrainCraft Audio Driver Reinstall

If your display is a BrainCraft HAT and you have pinned your kernel, you should be running a kernel version of around 5.10. You can check this by typing `uname -r`.

```
pi@raspberrypi:~/Raspberry-Pi-Installer-Scripts $ uname -r
5.10.17-v7l+
```

If you pinned to an older version that uses a kernel of 5.4, you may need to reinstall the audio drivers at this point to get sound working. Be sure to follow the BrainCraft HAT Audio Setup instructions () for a kernel version around 5.4 when reinstalling.

Unpinning the Kernel

To unpin the kernel, just delete the file `/etc/apt/preferences.d/99-adafruit-pin-kernel` and update the Pi with the following commands:

```
sudo apt update
sudo apt upgrade
```

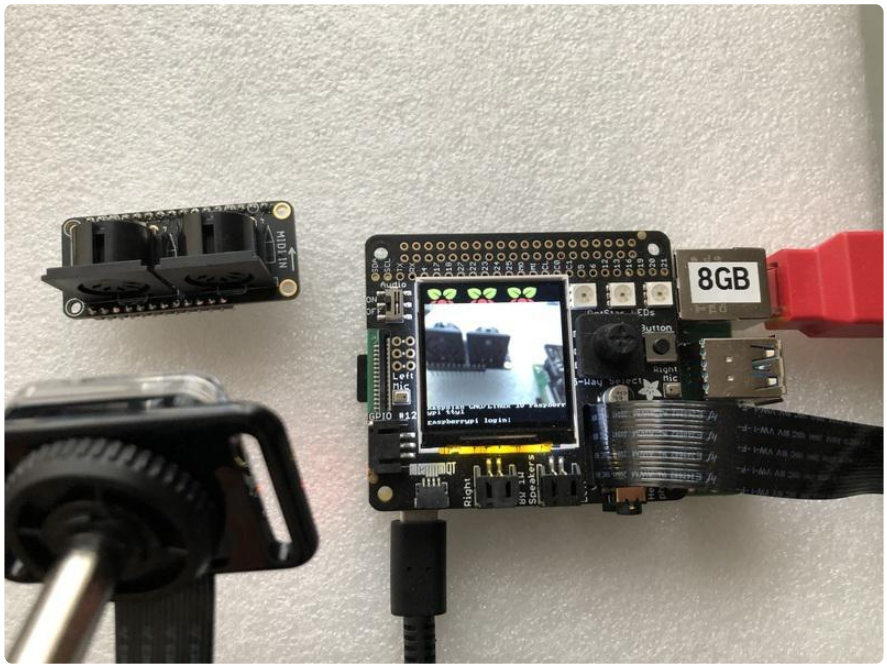
Camera Test

Now that you have everything set up, it's time to do an initial test with the camera. This should display what the camera sees on the display.

`sudo pip3 install picamera`

`raspistill -t 0`

raspistill is no long a part of the new Bullseye release of Raspberry Pi OS.



Python Usage

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

Besides the display, audio, and fan, this board 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 5-way Joystick and button just use simple `digitalio` and each uses a separate GPIO, so they're really simple to control. Here's a little script that will setup the GPIOs, Create Internal Pullups, and then print out the value to the terminal.

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

BUTTON_PIN = board.D17
JOYDOWN_PIN = board.D27
JOYLEFT_PIN = board.D22
JOYUP_PIN = board.D23
JOYRIGHT_PIN = board.D24
JOYSELECT_PIN = board.D16

buttons = [BUTTON_PIN, JOYUP_PIN, JOYDOWN_PIN,
           JOYLEFT_PIN, JOYRIGHT_PIN, JOYSELECT_PIN]
for i,pin in enumerate(buttons):
  buttons[i] = DigitalInOut(pin)
  buttons[i].direction = Direction.INPUT
  buttons[i].pull = Pull.UP
button, joyup, joydown, joyleft, joyright, joyselect = buttons

while True:
  if not button.value:
    print("Button pressed")
  if not joyup.value:
    print("Joystick up")
  if not joydown.value:
    print("Joystick down")
  if not joyleft.value:
    print("Joystick left")
  if not joyright.value:
    print("Joystick right")
  if not joyselect.value:
    print("Joystick select")

  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.

```
Joystick left
Joystick left
Joystick left
Joystick left
Joystick left
Joystick left
Joystick left
Joystick left
Joystick up
Joystick up
Joystick up
Joystick up
Joystick up
Joystick up
Joystick up
Joystick up
Joystick up
Joystick up
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.

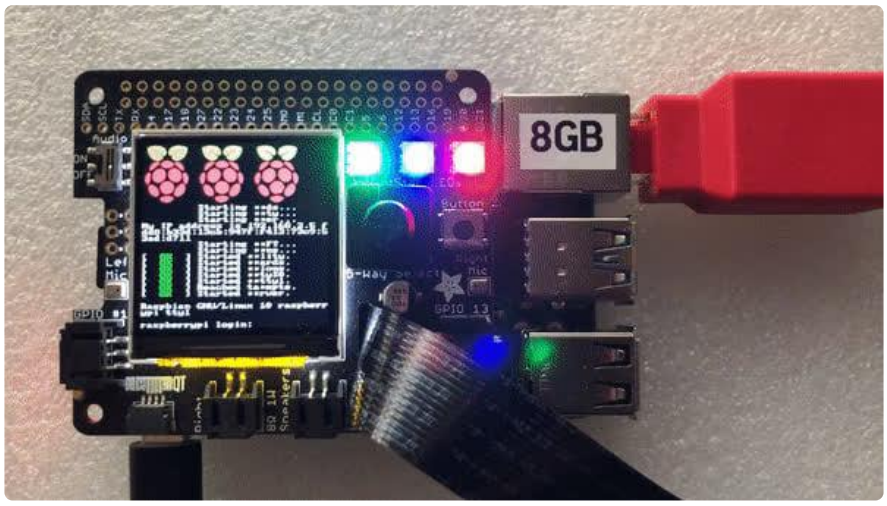
```
import time
import board
import adafruit_dotstar

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;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.

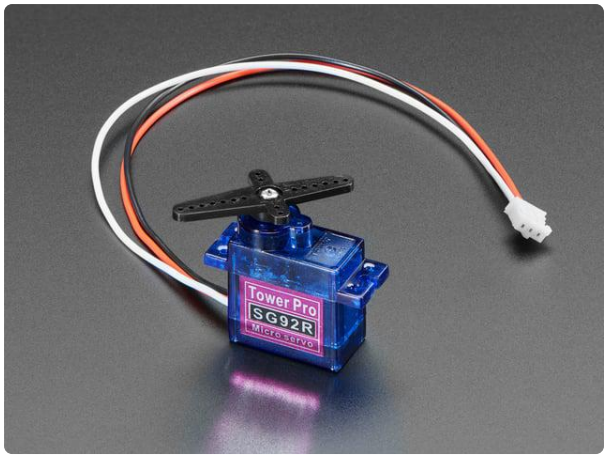


GPIO JST connectors

GPIOs 12 and 13 are accessible with the JST connectors on either side of the BrainCraft HAT.

Parts

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



Micro Servo with 3-pin JST PH 2mm Cable -

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 side of the BrainCraftHAT

Download Fritzing Diagram

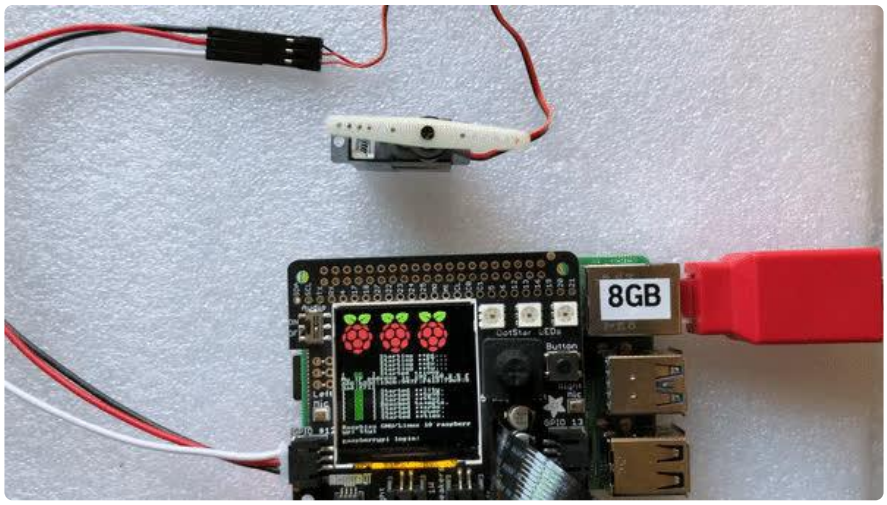
Code

```
import time
import board
import pulseio
from adafruit_motor import servo

SERVO_PIN = board.D12
pwm = pulseio.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.

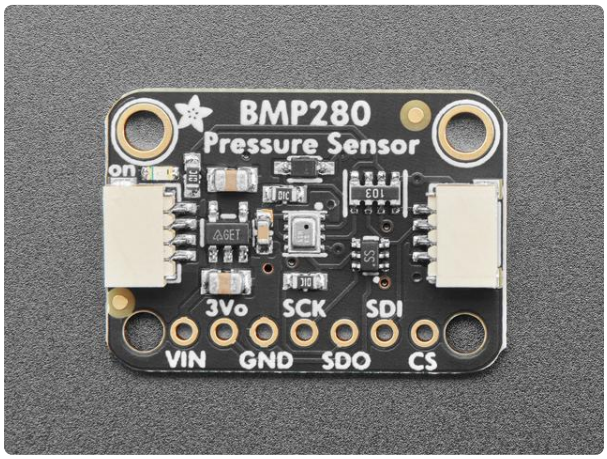


Stemma QT

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 BrainCraft HAT

Download 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.

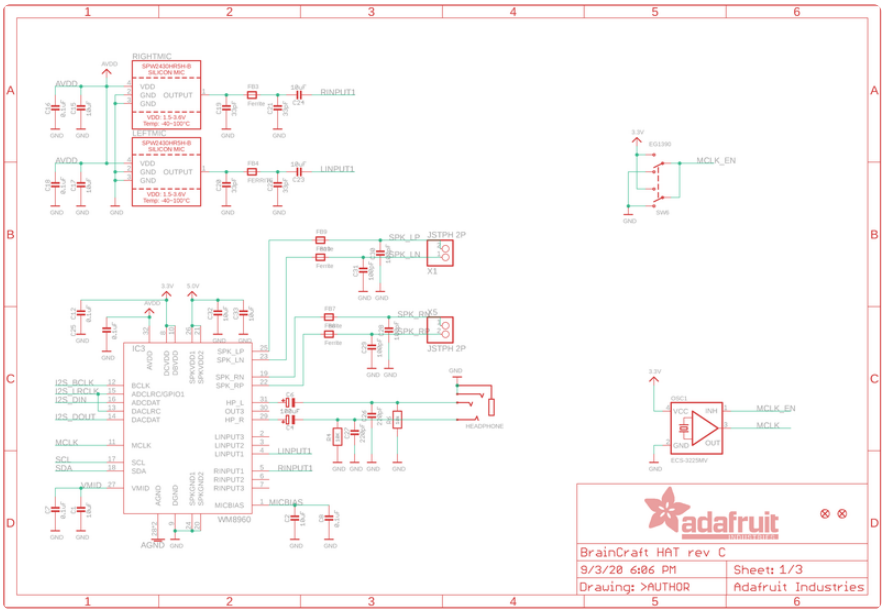
```
pi@raspberrypi:~ $ python bmp280_simpletest.py

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
```

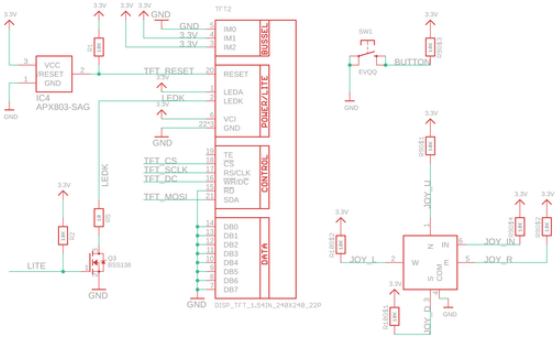
Downloads

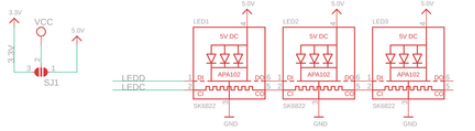
Files

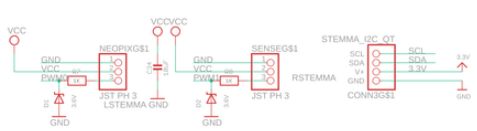
- ST7789 datasheet ()
- EagleCAD PCB files on GitHub ()
- 3D Model on GitHub ()
- Fritzing object in Adafruit Fritzing Library ()

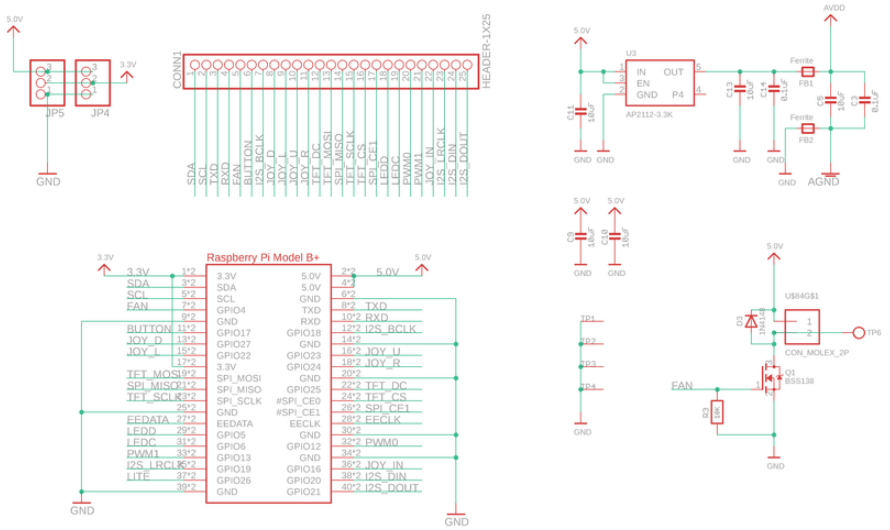
Schematic


BrainCraft HAT rev C
9/3/20 6:06 PM
Sheet: 1/3
Drawing: >AUTHOR
Adafruit Industries

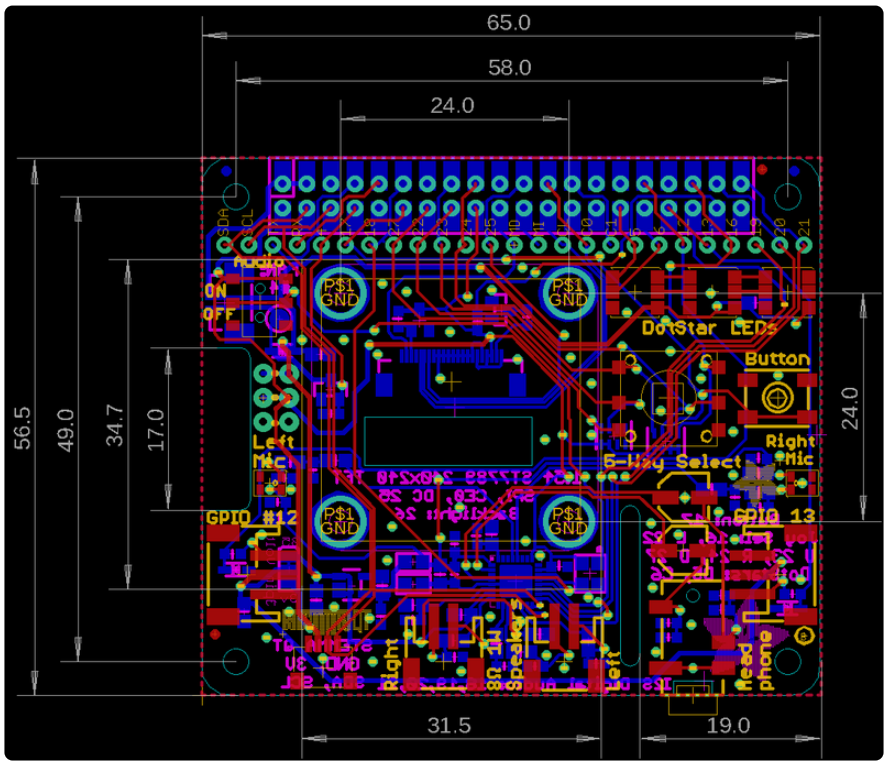







JP5
JP4
GND
CONN1
HEADER-1X25
Raspberry Pi Model B+
3.3V
5.0V
GND
AP2112-3.3K
AGND
U884GS1
BSS138
CON_MOLEX_2P
FAN

Fab Print


65.0
58.0
24.0
56.5
49.0
34.7
17.0
24.0
31.5
19.0
DotStar LEDs
Button
5-Way Select
Left Mic
Right Mic
GPIO #12
GPIO 13
Audio
ON
OFF
P$1
GND
Left
Right
8Ω 1W
Speaker
Head phone