



Machine Learning 101 with Lobe and BrainCraft

Created by Adi Azulay



<https://learn.adafruit.com/machine-learning-101-lobe-braincraft>

Last updated on 2023-08-29 04:31:03 PM EDT

Table of Contents

Overview	5
• Background Knowledge	
Required Parts	6
Train a Model with Lobe	6
• 1. Open Lobe and create a new project.	
• 2. In the Label tab, select Import (top right corner), and Camera from the drop down menu.	
• 3. In the bottom left corner, type a label for the first image.	
• 4. Take between 10 and 20 pictures of your objects using your computer's camera.	
• 5. Repeat steps 3 and 4 for the rest of your objects.	
• 6. Add a "Nothing" category.	
Test your Model	9
• 1. In Lobe, select the Use tab and choose Camera.	
• 2. Try different placements of the objects used for training and see how the model performs.	
• 3. Improve your model using the buttons in the bottom right.	
Export your Model	11
Set up Raspberry Pi and BrainCraft	12
Raspberry Pi Setup	13
• Step 1 - Burn SD Card	
• Step 2 - Configure log-in access	
• Step 3 - Log in & Enable Internet	
• Step 4 - Update/Upgrade	
Blinka Setup	15
Fan Service Setup	17
Fan Service Troubleshooting	20
Display Module Install	22
• Installing The 1.54" Kernel Module	
Kernel Module Troubleshooting	24
• Bullseye Desktop Version Breaking Changes	
• Static Issue	
• BrainCraft Audio Driver Reinstall	
• Unpinning the Kernel	
Camera Test	26
• Install the Pi Camera module	
Set up an FTP Connection	27
• Windows Instructions	
• Mac Instructions	

-
- 1. Connect your Pi to a power source and wait for it boot up.
 - 2. Open command prompt on a PC or terminal on Mac/Linux and connect to your Pi using SSH.
 - 3. Download the GitHub folder.
 - 4. Create a new folder called model in the home directory.
 - 5. Open the FTP connection from the previous step.
 - 6. Copy saved_model.tflite and signature.json from your exported Lobe model to the model directory on the Pi.
 - 7. In terminal on the Pi, run the following script to install Lobe and all it's dependences:
 - 8. In terminal on the Pi, run the Python program lobe-basic-prediction.py

Overview



Welcome to the wonderful world of Lobe, where you can train Machine Learning models without writing a single line of code!! Lobe is free and easy-to-use, all you have to do is take pictures and label them and Lobe does the rest.

In this guide we'll walk through training a Lobe model, exporting it, and running it on a Raspberry Pi 4.

This tutorial is part of a series which include the following tutorials:

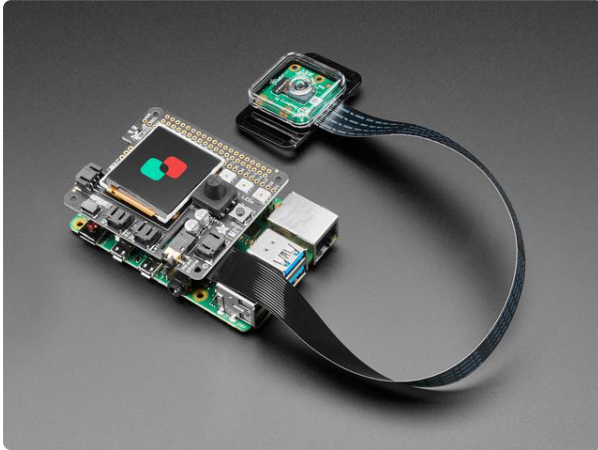
- [Tutorial 2 - Build an ML Rock Paper Scissors Game with Lobe \(\)](#)
- [Tutorial 3 - Build an ML Package Detector with Lobe \(\)](#)

Background Knowledge

To be successful with this project, you'll need some experience with the following:

1. Setting up and using the Raspberry Pi
2. Some familiarity with using the terminal window
3. Installing the Pi Camera

Required Parts



[Microsoft Machine Learning Kit for Lobe with Raspberry Pi 4 4GB](https://www.adafruit.com/product/4963)

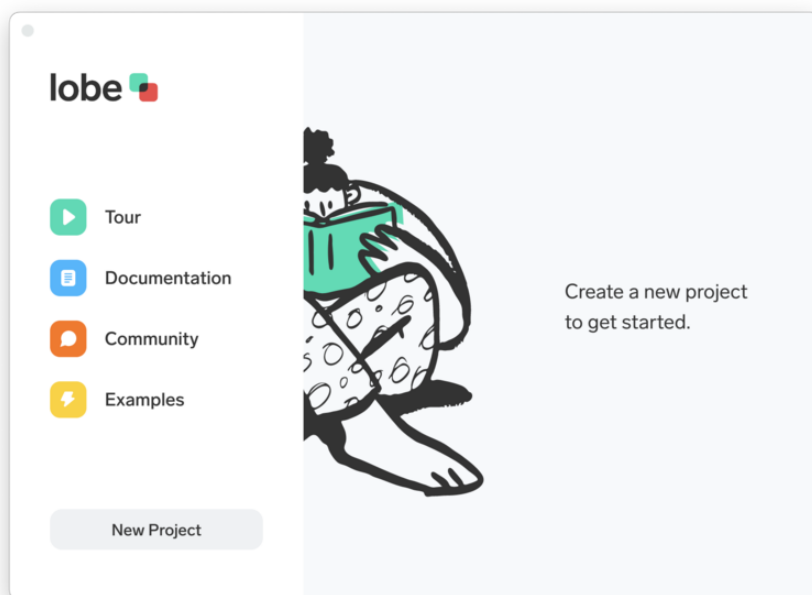
Machine learning is a transformative tool that's redefining how we build software—but up until now, it was only accessible to a small group of experts. At Adafruit, we... <https://www.adafruit.com/product/4963>

Train a Model with Lobe

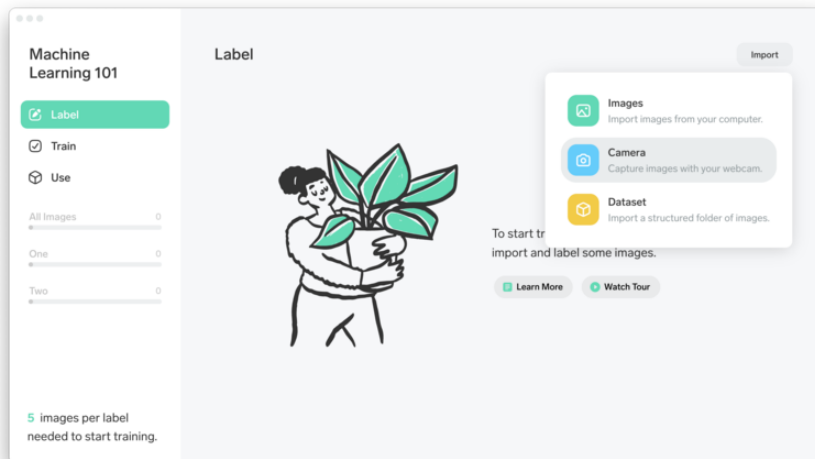
First, we'll train a custom machine learning model using objects on your desk. For my model I used a pen, Lobe sticker, and a succulent plant. Feel free to use any objects you'd like!

First, download and install Lobe from the link below:

[Download Lobe](#)



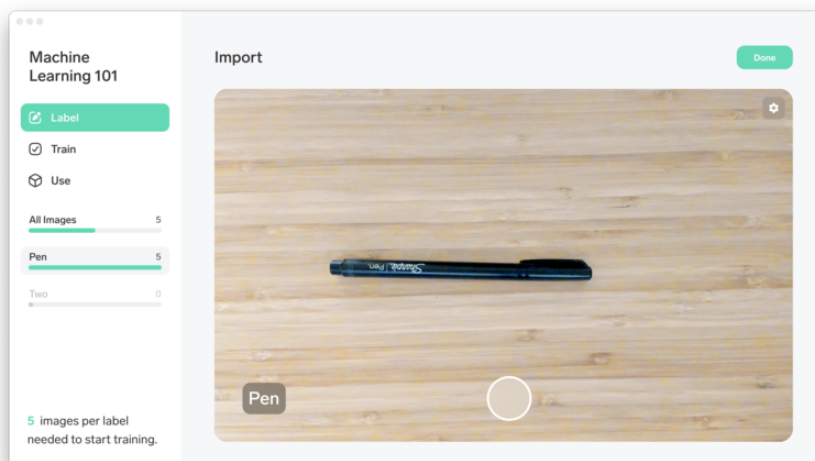
1. Open Lobe and create a new project.



2. In the Label tab, select Import (top right corner), and Camera from the drop down menu.

If this is your first time using Lobe you'll need to give it permission to use your camera.

If your computer doesn't have a camera you can take pictures using a cellphone or digital camera and import them using the Images option.

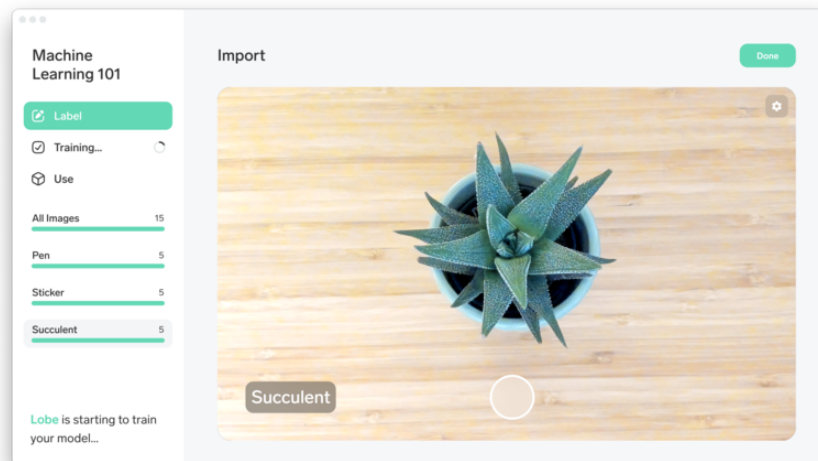
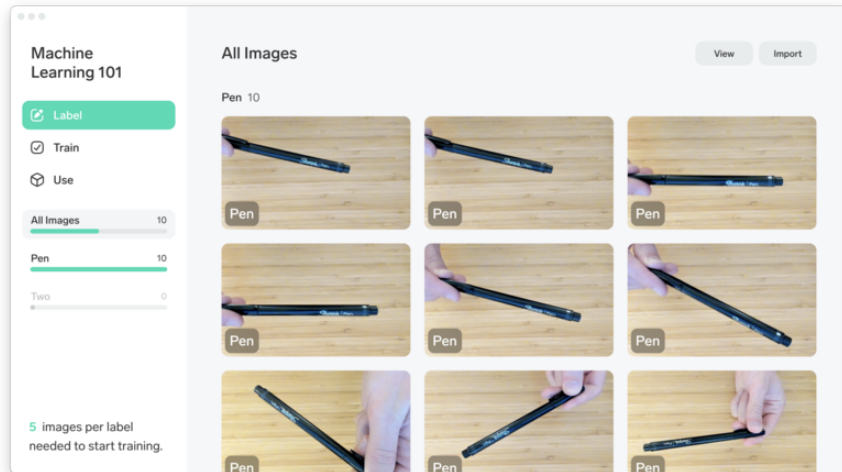


3. In the bottom left corner, type a label for the first image.

My first label is "Pen" because it's a picture of, well, a pen!

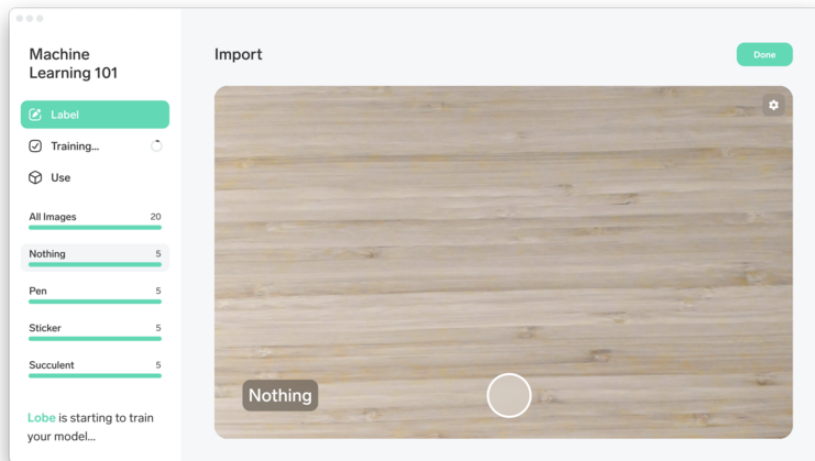
4. Take between 10 and 20 pictures of your objects using your computer's camera.

Take pictures from different angles, in different lighting conditions, and with different hand placements to improve model accuracy.



5. Repeat steps 3 and 4 for the rest of your objects.

Remember to add a new label for each object (e.g. "Sticker" and "Succulent").



6. Add a "Nothing" category.

This improves accuracy of the ML model. See below for more info.

Why is there always a prediction even when nothing is in the image?

Lobe will always predict one of your labels even if your image does not contain any related content. If you expect your model to see these types of images, create a 'None' label and add variations of these images as examples. You can use this 'None' label as a placeholder when waiting for relevant predictions.

Test your Model

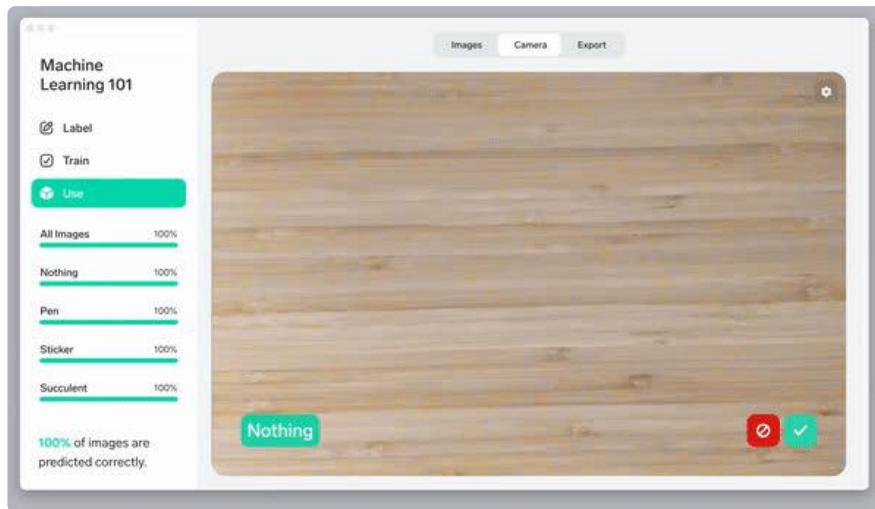
In Lobe, training happens automatically as soon as you add enough images (a minimum of 5 for each label).

When you have taken between 5 and 10 pictures of each object and training is complete, test out the model by following the steps below.

1. In Lobe, select the Use tab and choose Camera.



1a. This is the ML model prediction. The more full this bar is, the more confident the algorithm is with the prediction.



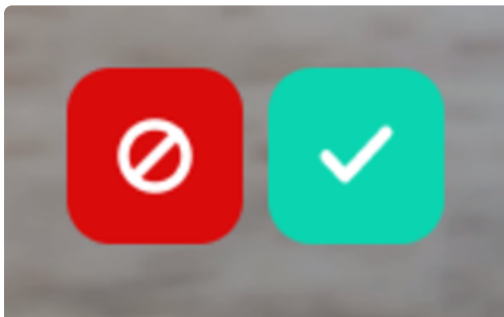
2. Try different placements of the objects used for training and see how the model performs.



Try other objects of the same type to check model accuracy. Also try out multiple objects at once. You will notice Lobe only makes one prediction at a time, even if there are multiple objects. See if you can trick your model into predicting the wrong thing and find ways to improve it.

3. Improve your model using the buttons in the bottom right.

My model did a great job at identifying sticky notes, but it mixed up the pen and highlighter.

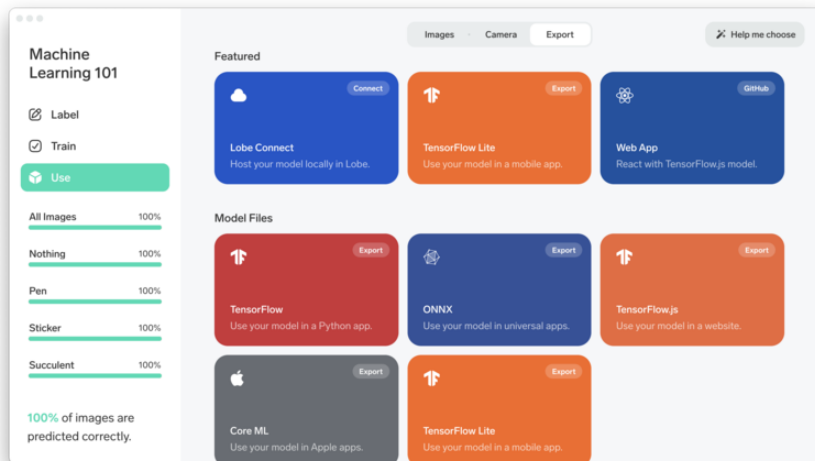


1b. Use these buttons to give Lobe feedback to improve your model.

Click the green button for images that the model predicts correctly. Select the red button for images the model predicts incorrectly.

Export your Model

Next, export your Lobe model to use on the Raspberry Pi. We'll use TensorFlow Lite which is a format that is optimized for mobile and edge devices, like the Pi.



In Lobe, navigate to the Use tab and click Export.

Select TensorFlow Lite and select a location to save the model. We'll transfer the model to our Raspberry Pi later in the tutorial.

Set up Raspberry Pi and BrainCraft

Now that the ML model is working on your computer, let's get your Raspberry Pi and BrainCraft HAT ready to make predictions on new images (also called inferencing). Your kit includes all the parts you need for this.

The next few steps walk you through setting up your BrainCraft HAT. We will skip audio setup since we aren't using it for this project, but if you can find the complete setup guide [here](#) ().

Please note that the following steps require installing a lot of packages on the Pi and it may take a few hours to download everything.

Logging into your Pi in a headless configuration via SSH? Use the "raspberrypi.local" IP (more info below)

As long as you're on the same WiFi as the Pi, you can use the IP address `raspberrypi.local`, as shown below:

```
ssh pi@raspberrypi.local
```

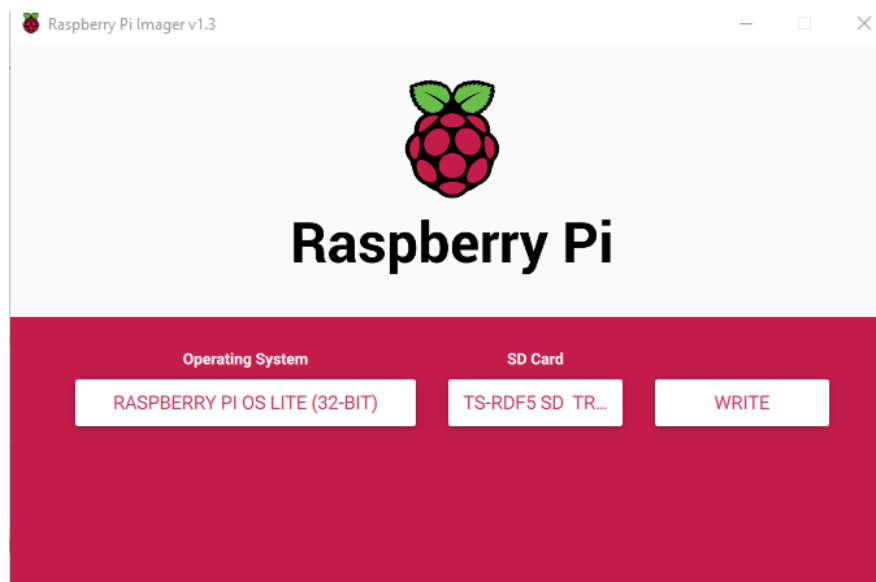
Raspberry Pi Setup

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.

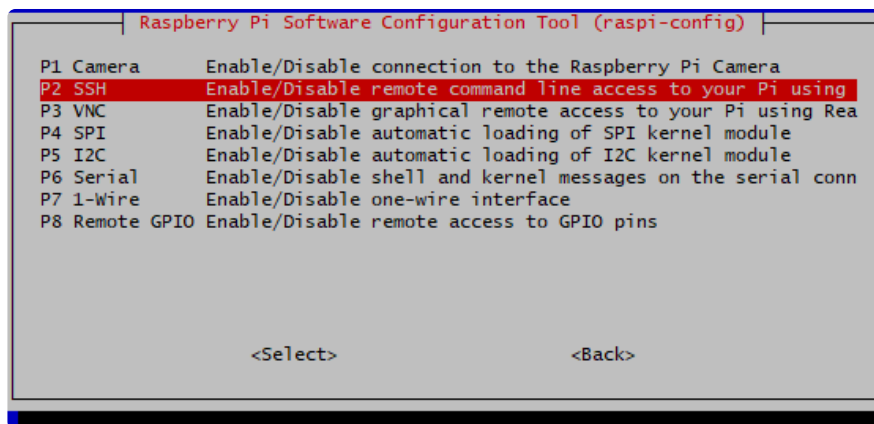
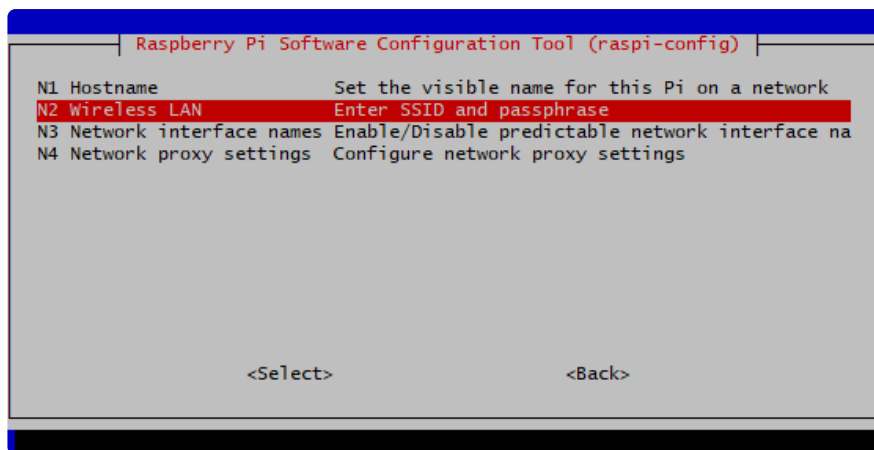
[We have a quickstart guide here](#) () and [here that you can follow](#) (), or there's dozens of online guides. it is assumed by the next step you are able to log in and type

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.

```
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:~$
```

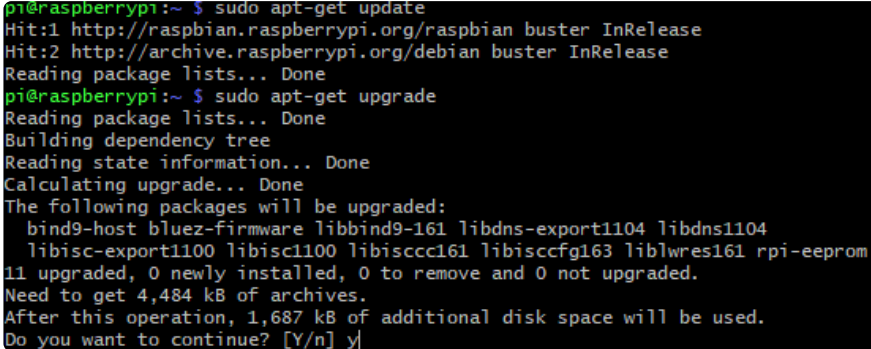
Step 4 - Update/Upgrade

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 libiscfg163 liblwres161 rpi-eeeprom
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.

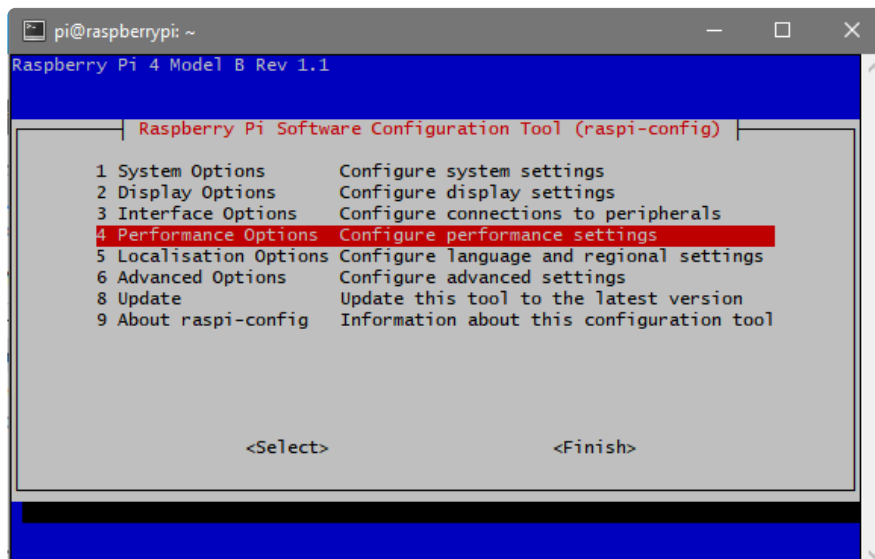
Fan Service Setup

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.

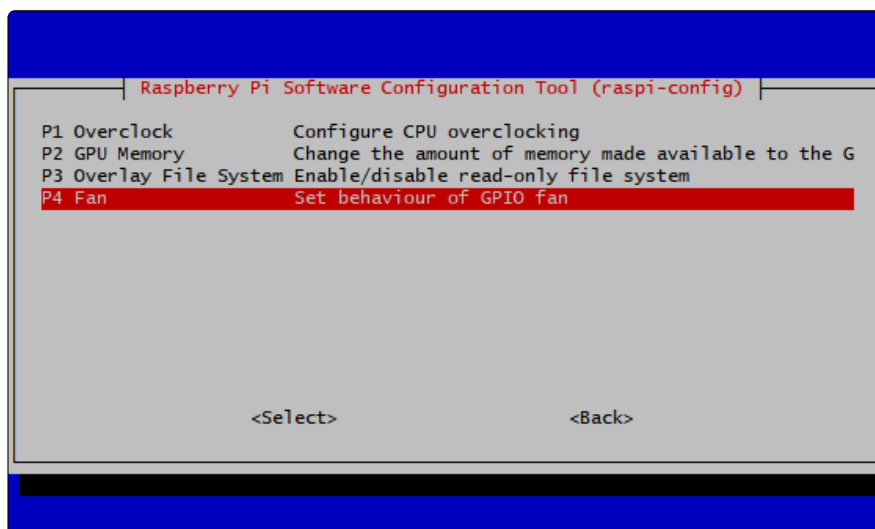
The fan service basically controls turning GPIO 4 on at startup, which is what the fan is connected to. Installing the fan service is really simple and we have a script for doing that.

To install, just type `sudo raspi-config`

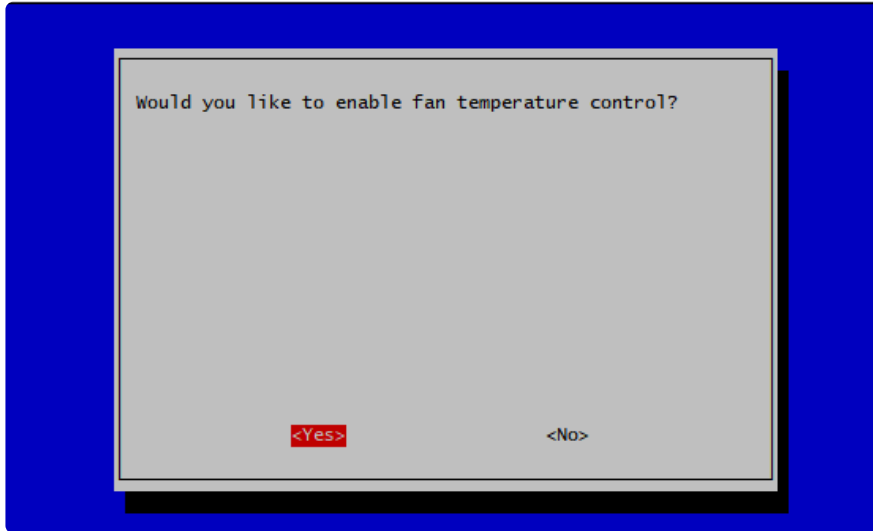
Select Performance Options



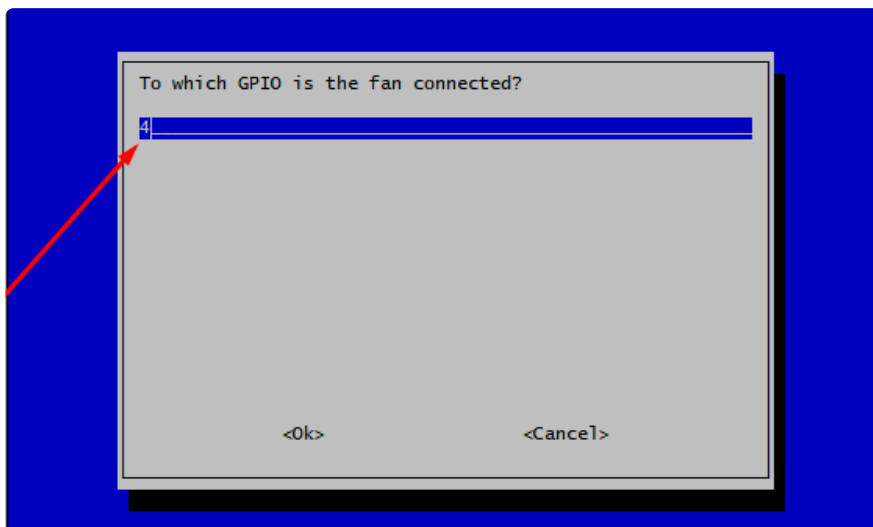
Select Fan



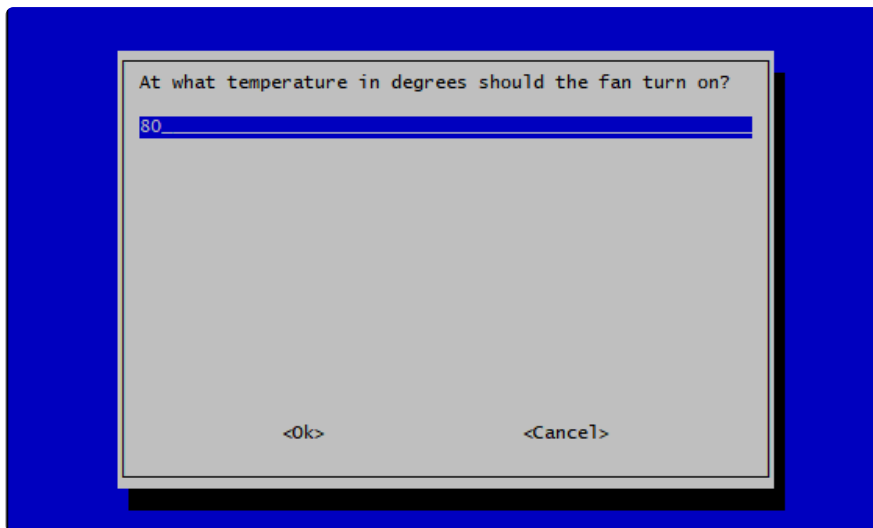
Select Yes



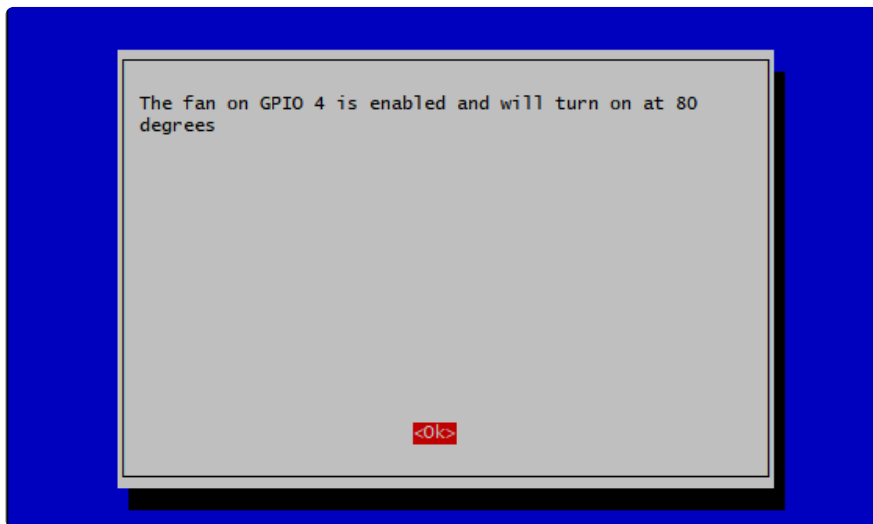
And make sure you put down GPIO pin 4 for the fan



You can customize the fan temperature setting



That's it!



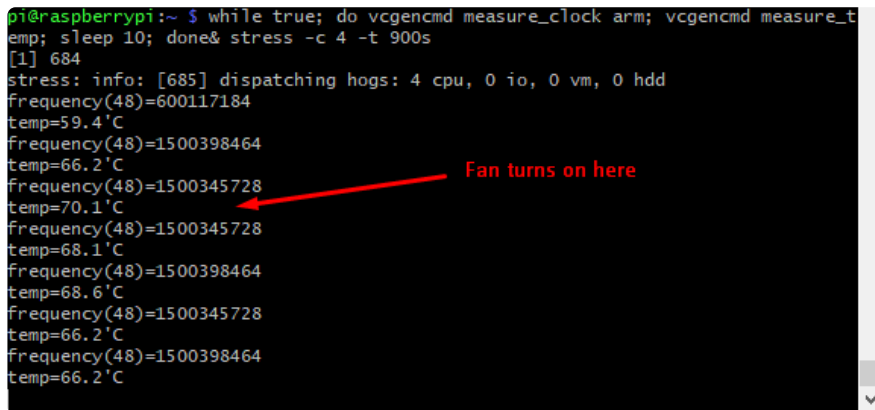
You can then 'stress test' by running

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

```
pi@raspberrypi:~ $ while true; do vcgencmd measure_clock arm; vcgencmd measure_t
emp; sleep 10; done & stress -c 4 -t 900s
[1] 955
stress: info: [956] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
frequency(48)=600169920
temp=53.5'C
frequency(48)=1500345728
temp=60.8'C
frequency(48)=1500398464
temp=64.2'C
frequency(48)=1500398464
temp=65.7'C
frequency(48)=1500398464
temp=66.7'C
frequency(48)=1500345728
temp=66.2'C
frequency(48)=1500398464
temp=68.6'C
```

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):

```
pi@raspberrypi:~$ while true; do vcgencmd measure_clock arm; vcgencmd measure_t
emp; sleep 10; done& stress -c 4 -t 900s
[1] 684
stress: info: [685] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
frequency(48)=600117184
temp=59.4'C
frequency(48)=1500398464
temp=66.2'C
frequency(48)=1500345728
temp=70.1'C
frequency(48)=1500345728
temp=68.1'C
frequency(48)=1500398464
temp=68.6'C
frequency(48)=1500345728
temp=66.2'C
frequency(48)=1500398464
temp=66.2'C
```

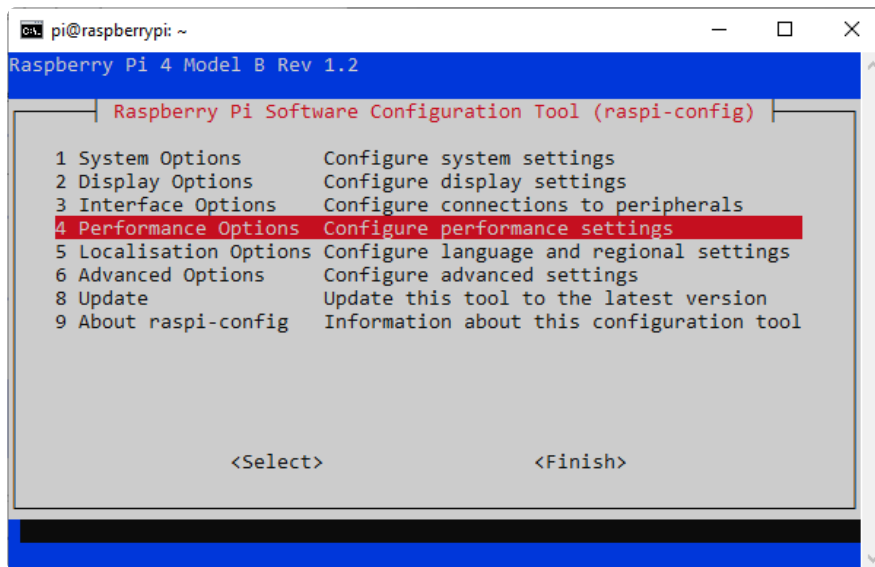


Fan Service Troubleshooting

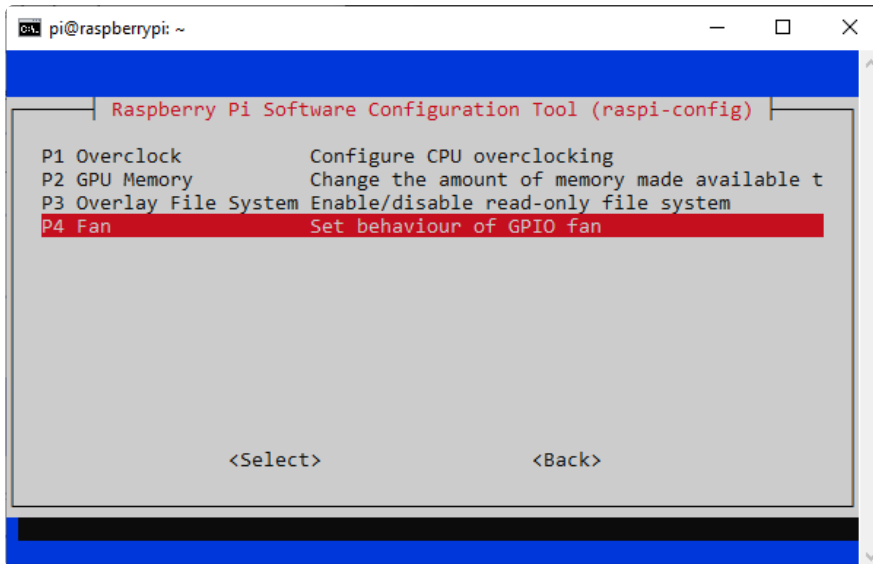
On some newer versions of Raspberry Pi OS the Fan service fails to start. Luckily these newer version of the OS have a built in fan control you can turn on.

In command prompt or terminal connect to the pi using SSH and run the command:

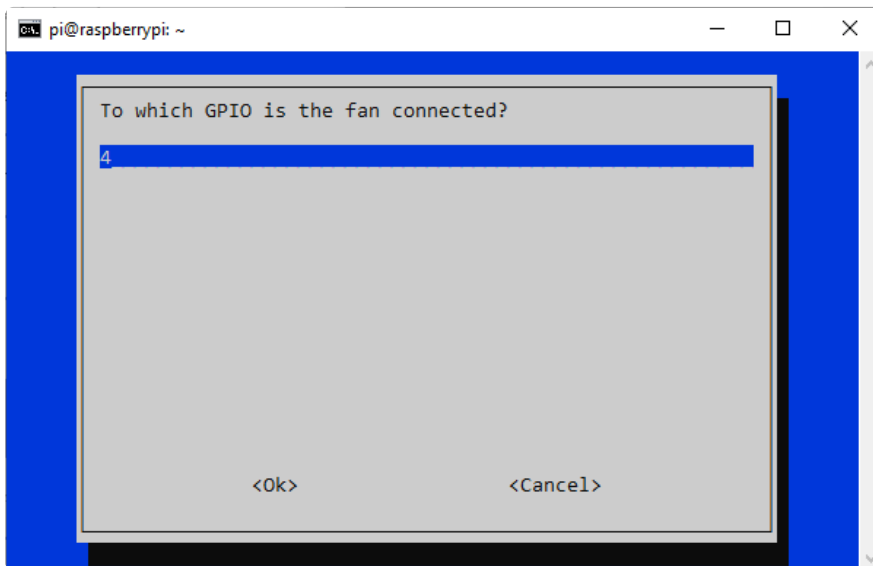
```
sudo raspi-config
```



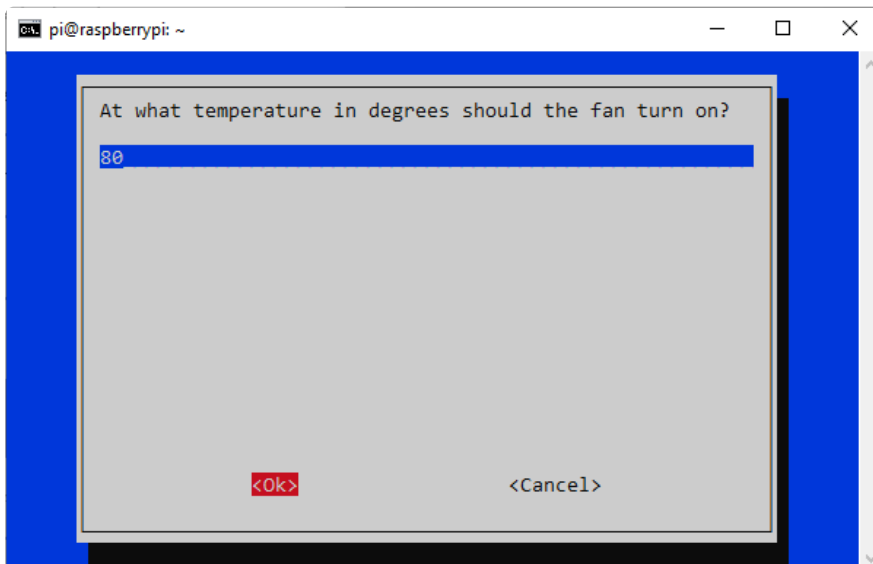
Select Performance Options



Select Fan



The default GPIO pin is 14, but the BrainCraft Hat has the fan connected to pin 4.



You can set the temperature to anything between 60-120 degrees Celsius. 80 is a good midway point in that range.

You'll need to reboot after changing the settings.

Now your fan will come on when ever the board is over 80 degrees. To check the current temperature of your board you use this command:

```
/opt/vc/bin/vcgencmd measure_temp
```

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=fbc
```

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.
Resolving deltas: 100% (205/205), done.
```

When you get asked to reboot, reboot!

```
removing old section...
##### 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

Kernel 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

Install the Pi Camera module

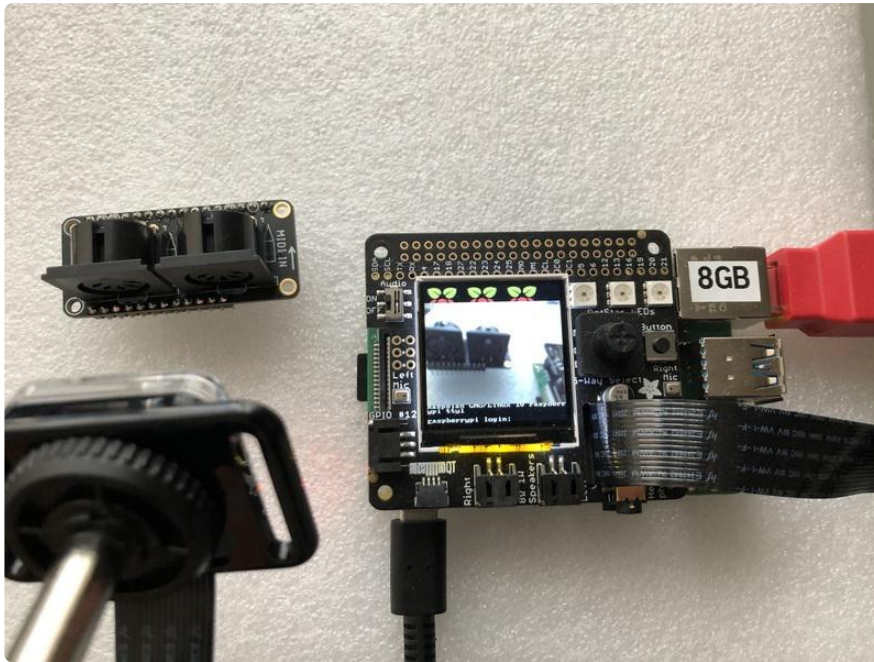
Make sure you have the Pi camera module by running the following command:

```
pip3 install picamera
```

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.

```
raspistill -t 0
```

Exit the camera test by pressing `CTRL + C`

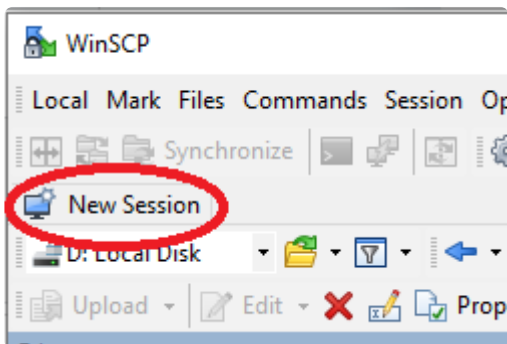


Set up an FTP Connection

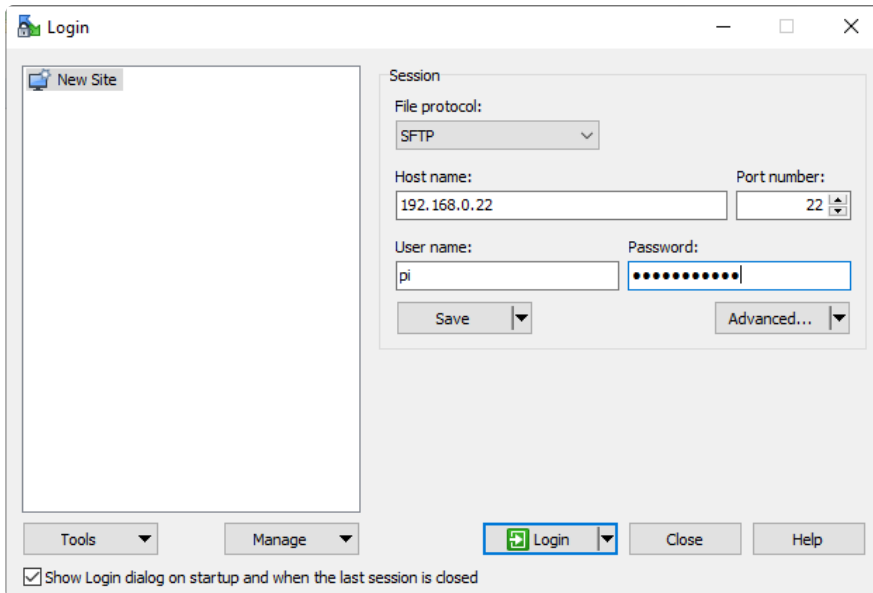
Since we're using the Pi in a headless configuration, we'll use an FTP connection to transfer files between our computer and the Pi.

Windows Instructions

Download and install [WinSCP \(\)](#)



Open WinSCP and start a New Session

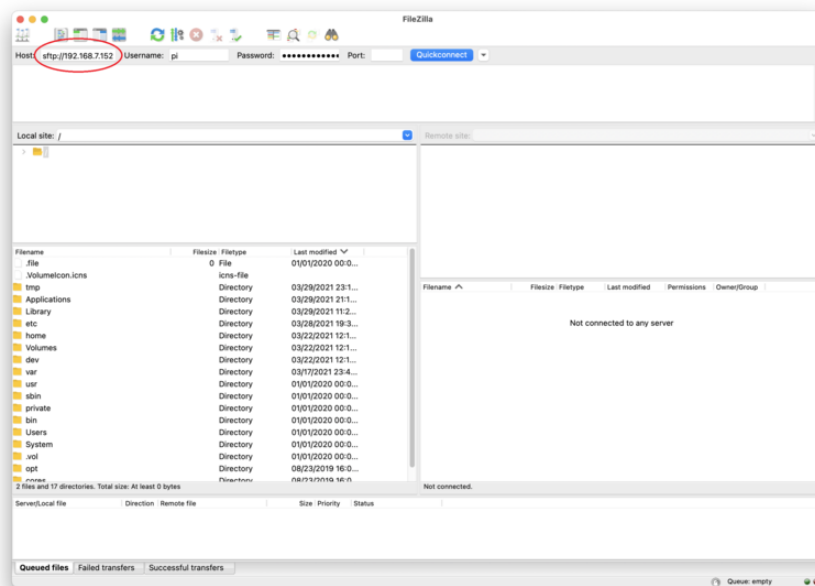


Select an SFTP connection, fill in the IP address of your Pi, set the username to Pi, and put in your password.

Your Pi's IP address is on the screen of the BrainCraft. You can also use the hostname address, e.g. "raspberrypi.local" (pi@raspberrypi.local).

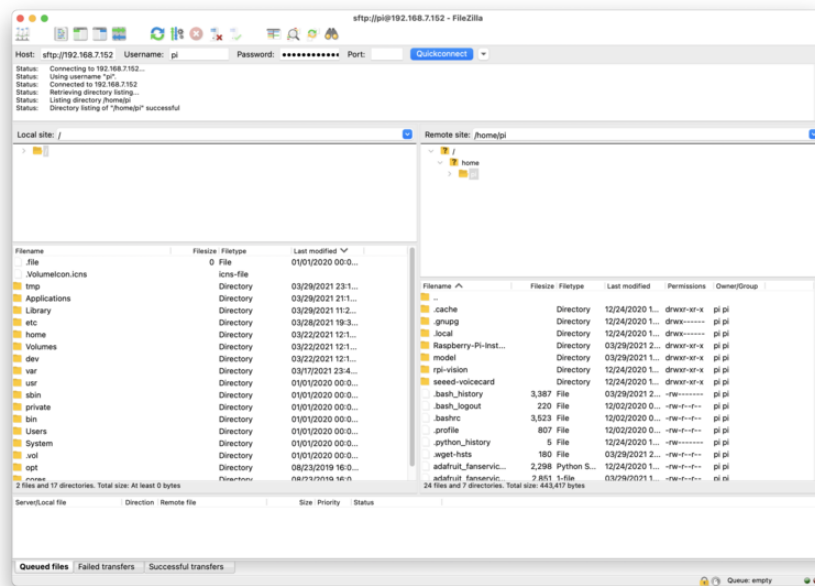
Mac Instructions

Download and install [FileZilla](#) (). When it's done installing, open the program.



Type `sftp://` followed by the IP address of your Pi. Set the username to `pi` and put in your password.

Your Pi's IP address is on the screen of the BrainCraft. You can also use the hostname address, e.g. "raspberrypi.local" (pi@raspberrypi.local).



Get Predictions on the Pi

1. Connect your Pi to a power source and wait for it boot up.

You should see a solid red light and an intermittently flashing green light.

2. Open command prompt on a PC or terminal on Mac/Linux and connect to your Pi using SSH.

Type the following command but replace the bolded number below with IP address of your Pi:

```
ssh pi@192.168.0.22
```

Your Pi's IP address is on the screen of the BrainCraft. You can also use the hostname address, e.g. "raspberrypi.local" (pi@raspberrypi.local)

3. Download the GitHub folder.

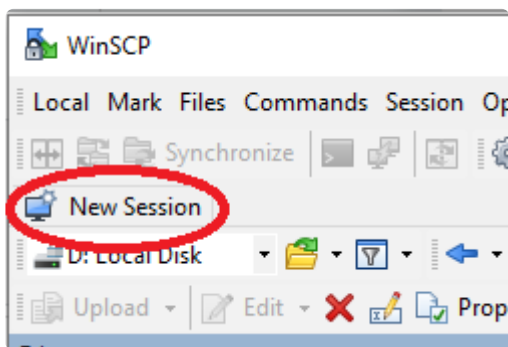
Run the following commands to download the sample code from GitHub:

```
cd ~
git clone https://github.com/lobe/lobe-adafruit-kit.git
```

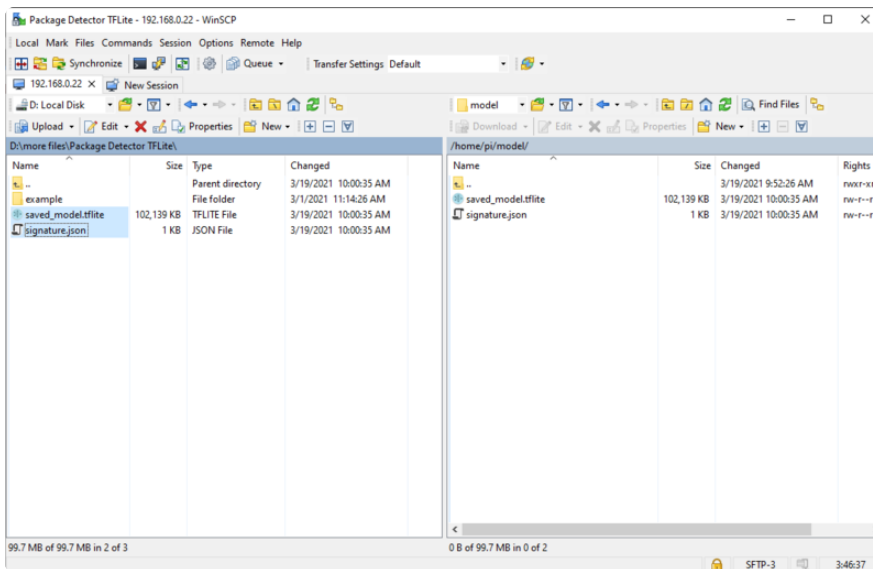
4. Create a new folder called **model** in the home directory.

Type the following commands:

```
cd ~
mkdir model
```



5. Open the FTP connection from the previous step.



6. Copy `saved_model.tflite` and `signature.json` from your exported Lobe model to the `model` directory on the Pi.

7. In terminal on the Pi, run the following script to install Lobe and all its dependencies:

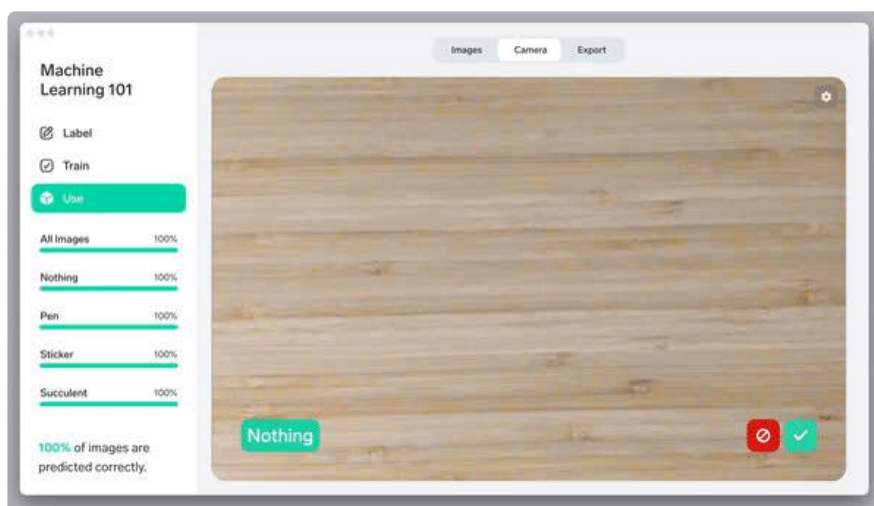
```
cd ~
wget https://raw.githubusercontent.com/lobe/lobe-python/master/scripts/lobe-rpi-install.sh
sudo bash lobe-rpi-install.sh
```

8. In terminal on the Pi, run the Python program `lobe-basic-prediction.py`

Type the following commands:

```
cd ~
cd lobe-adafruit-kit
python3 lobe-basic-prediction.py
```

Going Further



Keep testing the model on the Pi and see how it works. If you find that the prediction is consistently wrong, you can add more images to the model to improve its performance.

You can train an ML model to recognize all sorts of objects and then use the BrainCraft to trigger actions in the physical world!

To learn how to do this, check out these more advanced projects:

- [Tutorial 2: Build a rock paper scissor game \(\)](#)
- [Tutorial 3: Build a package detector \(\)](#)