

Setting up a Raspberry Pi Reef Cam

Part 1, The hardware

Raspberry Pi 3 B+ is recommended for HD streaming quality, however will likely work well with any of the more recent models.

<https://core-electronics.com.au/raspberry-pi-3-model-b-plus.html>

~\$54

Raspberry Pi Camera Board v2 - 8 Megapixels, this is a high quality camera and will produce excellent results with a reef tank, it can compensate for blue light etc... I have not tested any other model of raspberry pi camera so your results may vary if you choose another

<https://core-electronics.com.au/raspberry-pi-camera-board-v2-8-megapixels-38552.html>

~\$39

Optional hardware

Raspberry Pi POE Hat – If you have a POE compatible switch or a POE injector on your network you can power the latest Raspberry Pi 3 B+ over ethernet which can be very handy near your tank as it means you won't need to dedicate a valuable power socket to the Pi.

<https://core-electronics.com.au/raspberry-pi-poe-hat-official.html>

~\$39

Raspberry Pi compatible power supply. Any modern USB power supply capable of delivering at least 5 Volts 2.5amps will work well. If in doubt buy the official one. (Note this is not required if you use the POE hat).

<https://core-electronics.com.au/raspberry-pi-3-power-supply.html>

~\$17

While not essential, a cool case will make the whole setup look nicer and possibly offer some splash protection in a reef-tank adjacent position.

<https://core-electronics.com.au/pimoroni-pibow-3-rainbow-raspberry-pi-3-2-b.html>

~\$25

The camera mount case can help with finding the best place for the camera and giving it a discrete mounting.

<https://core-electronics.com.au/raspberry-pi-camera-mount-38585.html>

~\$7

The cable that comes with the Camera is quite short. You may want to consider getting a longer one to allow for more flexible mounting locations.

<https://core-electronics.com.au/flex-cable-for-raspberry-pi-camera-1-meter.html>

~\$7

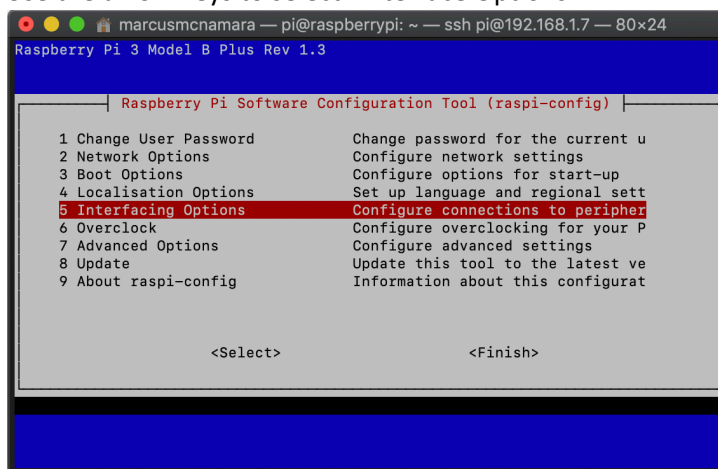
For networking I recommend hard-wiring the Pi with ethernet if possible. If you can't, I would suggest at least 'N' spec wifi, with 'AC' being recommended.

Part 2 – Setting up the Raspberry Pi, Camera and initial software installation

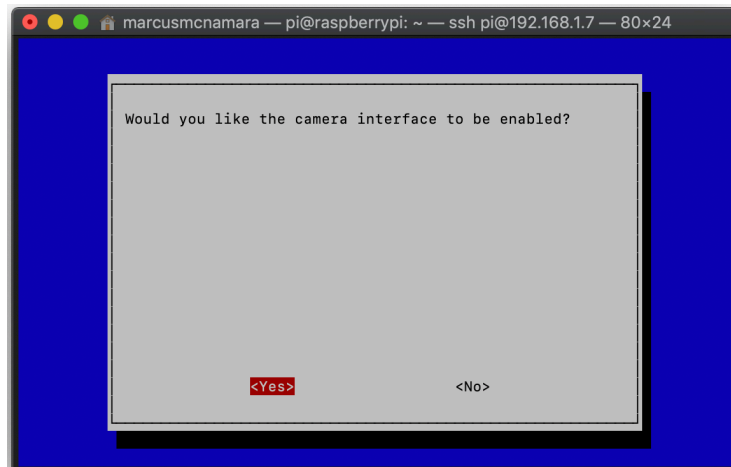
For this guide we are going to be using the latest version of Raspbian Stretch Lite as the operating system for the Pi. The Lite version does not include a graphical user interface, we are choosing this as it would just waste system resources and it not needed for our use case. However if you think you will want the GUI for other purposes with this Raspberry Pi, feel free to install the full version of Stretch instead. It won't change the instructions below.

- Download the following onto your computer
 - Raspbian Stretch Lite - <https://www.raspberrypi.org/downloads/raspbian/>
 - Etcher - <https://www.balena.io/etcher/>
- Install Etcher (its Mac, Windows and Linux compatible)
- Connect an SD card reader with the SD card inside to your computer
- Open Etcher and select from your hard-drive folder the Raspbian .zip file you downloaded before
- Select the SD card
- Review your selections and click 'Flash!' to begin writing data to the SD card
- Once the process is completed, close Etcher, unplug the SD card, and plug it back in again
- A small partition of the SD card should be accessible on your computer, this is the boot partition
 - Place a file in this location called "ssh" with no file extension
 - To create this file on Mac
 - Open terminal
 - cd /Users
 - ls
 - Note the exact (case sensitive) name of your accounts users folder
 - touch /[username]/Desktop/ssh
 - Substitute [username] for the name of your Users folder you noted
 - A file blank file should appear on your desktop call 'ssh'
 - To create this file on Windows
 - Right click on your desktop
 - Go to 'New' and select 'Text Document'
 - Name the file 'ssh' and delete the the '.txt' extension while naming it
 - Say yes when asked if you want to change the file extension of the file
 - Copy the blank 'ssh' file into the boot partition of the SD card
- Now eject the SD card from your computer and insert it into the Raspberry Pi.
- Connect the camera module, pay attention to the orientation of the ribbon cable, the gold tabs on the cable need to be facing the side of the connector with the pins.
- Plug your Raspberry Pi into power and plug it into Ethernet, you may need to temporarily set it up in a location near your router/modem in order to do this if you plan on using WIFI later on.
- If you know how to get the IP of your Pi from your network from your router or some other method, please do so and note it down. If you do not, you may need to plug a monitor and keyboard into your Pi and type the command: hostname -l
 - This article may help if you are having trouble getting your Pi's IP local address <https://www.raspberrypi.org/documentation/remote-access/ip-address.md>
- From here on out we will be doing everything over SSH, sending commands remotely to the Raspberry Pi from our main computer/laptop
 - For Mac we use Terminal which is part of the operating system, substitute 192.168.1.7 with your Raspberry Pi's IP address that you found as above.

- Type: ssh [pi@192.168.1.7](https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md)
 - Then enter the password which is 'raspberrypi'
- For Windows you will need to download, install and open Putty - <https://www.putty.org/>
 - Enter the IP address where it says Hostname
 - Ensure connection type SSH is selected and the Port is 22
 - Click Open and then enter the username 'pi' and password 'raspberrypi'
- First thing we are going to do now that we are connected to our Raspberry Pi and able to send it commands is to update its firmware to make sure it has the latest version that is compatible with the camera module and the POE hat if you're using it.
 - sudo rpi-update
- Once that is complete, we need to make sure the camera module is enabled
 - sudo raspi-config
 - Use the arrow keys to select "Interface Options"



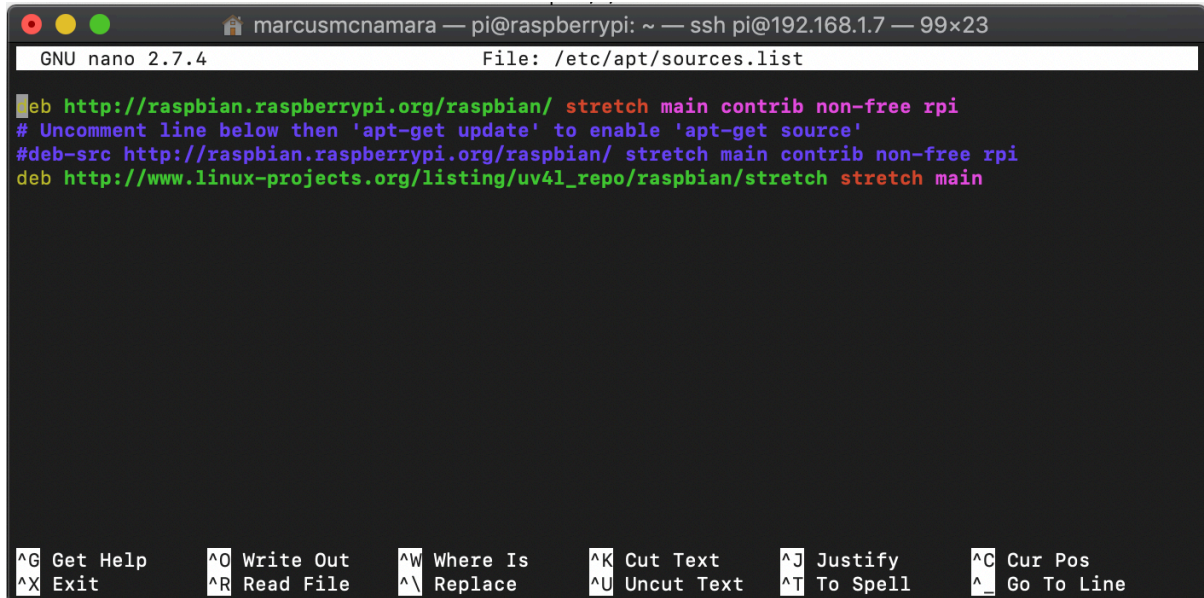
- Select the first option in the list, Camera and enable the Camera interface



- Select 'Finish' and reboot the Pi
- Sudo reboot
- You will likely need to re-connect to your Pi again via Terminal/Putty after it has rebooted
- If you want your Raspberry Pi connected to your WIFI instead of ethernet follow these instructions under 'Using raspi-config'
 - <https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md>
 - Note if switching your Raspberry Pi to WIFI its local IP address might change.

Part 3 – Installing UV4L and WebRTC support

- SSH into your Raspberry Pi
 - `curl http://www.linux-projects.org/listing/uv4l_repo/lpkey.asc | sudo apt-key add -`
 - `cd`
 - `sudo nano /etc/apt/sources.list`
 - Paste the following line into the file we just opened
 - `deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/stretch stretch main`
 - Save by pressing Ctrl + O then enter and exit with Ctrl + X



```
marcusmcnamara — pi@raspberrypi: ~ — ssh pi@192.168.1.7 — 99x23
GNU nano 2.7.4 File: /etc/apt/sources.list

deb http://raspbian.raspberrypi.org/raspbian/ stretch main contrib non-free rpi
# Uncomment line below then 'apt-get update' to enable 'apt-get source'
#deb-src http://raspbian.raspberrypi.org/raspbian/ stretch main contrib non-free rpi
deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/stretch stretch main
```

- Now we are ready to update the system and fetch and install the packages
 - `sudo apt-get update`
 - `sudo apt-get install uv4l uv4l-raspicam`
 - `sudo apt-get install uv4l-raspicam-extras uv4l-server uv4l-mjpegstream uv4l-demos uv4l-xmmp-bridge`
- We will be using WebRTC to stream the Camera
 - If using a Raspberry Pi 2 or 3 type
 - `sudo apt-get install uv4l-webrtc`
 - If using an Raspberry Pi 1 or Zero
 - `sudo apt-get install uv4l-webrtc-armv6`
- At any time you can restart the UV4L service by typing
 - `sudo service uv4l_raspicam restart`
- Do this now and **remember this command for the future**, we will be using it a lot moving forward; also anytime the camera is not working or throwing an error, this is the most common fix.

Part 4 – Test the camera and UV4L WebRTC Server

- So we now want to test that the UV4L service is running and the camera is working
- On your web-browser on your computer to <http://192.168.1.7:8080/> substituting 192.168.1.7 for your Raspberry Pi's IP address.
- You should see the UV4L streaming server interface load up



- Click on the MJPEG/Still stream option and a high res still from the camera should load, don't be alarmed if its slow to load, this is not the video stream we will be using; just a basic test for camera quality and function
- Go back and click on the first option Web RTC – Two Way Audio/Video. We will be using this tool temporarily to test the WebRTC function, steaming quality, and browser compatibility.
 - At the very bottom click the green call button
 - The stream should load and be playing in near real time – if it doesn't, try with another web-browser, I have had best results with Chrome and Firefox.
 - Click Hang-Up
 - Note: you need to click Hang-Up every time after you have pressed Call, even if the stream failed to load. Otherwise you may need to restart UV4L using the command we noted down at the end of the last section.
 - There is an option to 'force the use of hardware codec' click this on
 - Select a resolution and call again – if it doesn't, try with another web-browser, I have had best results with Chrome and Firefox.
 - The quality should be better and the stream smoother with the hardware codec enabled and the resolution higher.
 - Hang up again
 - Test various resolutions with hardware codec enabled and find the best option for you, factors that will make it better or worse include the model of Raspberry Pi, your local network speed and which web-browser your using and the computer you are watching on.
 - Note what you think is a good balance between image quality and stream smoothness, we will be adding this to the config of the UV4L server later.

Part 5 – Configuring the UV4L server and website

- Open a new SSH session with the Raspberry Pi or use the old one if its still open
- We are going to edit the default configuration file of the UV4L server. After making changes, ctrl-o and enter to save, then ctrl-x and enter to exit. Then restart the UV4L server using the command we noted down.
 - `cd`
 - `sudo nano /etc/uv4l/uv4l-raspicam.conf`
 - Using the arrow keys to navigate the file we will be un-commenting some lines by deleting the # in front of them, and modifying the values
- If you image is upside down or you want the camera orientation to change the values of these two settings to yes or no accordingly
 - `Hflip = no`
 - `Vflip = no`
- Find the sections below (some are right near the bottom of the file) and make these changes and uncomment by removing the #

WebRTC options:

```
server-option = --enable-webrtc=yes
server-option = --enable-webrtc-datachannels=yes
server-option = --webrtc-datachannel-label=uv4l
server-option = --webrtc-datachannel-socket=/tmp/uv4l.socket
server-option = --enable-webrtc-video=yes
server-option = --enable-webrtc-audio=no
```

...

```
server-option = --webrtc-max-plaintext-delay=34
```

...

These options are specific to the HTTP/HTTPS Server

serving custom Web pages only:

```
server-option = --enable-www-server=yes
server-option = --www-root-path=/usr/share/uv4l/demos/reefcam/
server-option = --www-index-file=index.html
server-option = --www-port=8888
```

...

```
server-option = --www-max-queued-connections=8
server-option = --www-max-threads=4
server-option = --www-thread-idle-time=10
```

...

```
server-option = --www-webrtc-signaling-path=/webrtc
```

- Save and exit the config file once you have made those changes (ctrl – o and ctrl – x) and then restart the UV4L service.
 - `sudo service uv4l_raspicam restart`

- Go to this path
 - `cd`
 - `cd /usr/share/uv4l/demos`
 - `sudo mkdir reefcam`
 - `cd reefcam`
 - `sudo wget https://github.com/MulletBoy/Raspberry-Pi-FishCam-Demo-Site/archive/master.zip`
 - `sudo unzip -j master.zip`
-
- In your web-browser now to 192.168.1.7:8888 with your IP Raspberry Pi's address and noting the 8888 port location instead of the 8080 we went to before
- You should see the ReefCam website.
- Test it by starting the steaming.
- Then click stop streaming.
- As before, test a few different browsers with and without hardware acceleration (h264 codec) to get a feel what works for you and your devices.
- For now you just want to know what works and what doesn't so we can move forward. We can focus on getting full compatibility with all browsers and all configurations later.

Part 6 – Tailoring the website to your preferences

- Back in the terminal or SSH session
 - `cd`
 - `cd /usr/share/uv4l/demos/reefcam`
 - `sudo nano index.html`
- In this file you can edit the title and some of the styling of the website. Exactly how to do this is out of scope of the tutorial, however it should be fairly straight forward for you to at least change the title of the webpage, look for the `<h1>Reef Nerd FishCam</h1>` code. Ctrl-o and enter to save, then ctrl-x and enter to exit.
- Back in the terminal or SSH session
 - `sudo nano signalling.js`
- Look for the block of code comments which lists all the values for camera resolution and framerates. Below that you will see I have set the `vformat:` to 60 by default. Pick another resolution if you want to change it higher or lower based on your testing. Restart the raspicam service afterwards.
 - `sudo service uv4l_raspicam restart`

Part 7 – Make it accessible externally

- So now we have a fit for purpose website that we can use to see our camera, however it is currently only accessible locally. i.e. on our home network.
- To make it accessible externally, say from our phone or on public/roaming internet anywhere in the world there are a few options. Some are out of scope of this guide such as:
 - Setup a VPN to be able to browse your local network externally as if you were at home (plenty of guides online how to do this and many ways to achieve it)
 - Buy your own domain, setup a web-server and have that push the raspberry pi hosted web-page through a reverse proxy to your self hosted and secure site. This is what I have done personally as I run a home server.
- For now what I am going to show you how to setup is how to setup a Dynamic DNS listing through a free domain provider and have that point to your Raspberry Pi with port forwarding on your router.
- Type 'whats my ip' into google and note down the result
- Go to www.duckdns.org and login using whatever account you prefer from the menu at the top (I used my google account)
- Now create a free subdomain which will be *something*.duckdns.org. It will have to be unique so if your first choice is already taken, try again
- Enter your IP into the current IP field and click the update button
- You have just manually told DuckDNS what your IP is, however it will change from time to time. If only we had an always-on low power computer connected to the internet that could tell DuckDNS every time our IP address changes... we will use our Raspberry Pi for this!
- I would give you the instructions for this but the DuckDNS website does it really well
 - Click install at the top of the site and select 'pi' from operating systems
 - Follow those instructions, choosing your domain and copying the commands into a new SSH session on your Raspberry Pi
 - You'll see there are many different ways to install a DuckDNS update service on many different devices and operating systems. If you have a fancy router, chances are it will have a DuckDNS compatible Dynamic DNS feature built into it, so you could use that instead, all you would have to do is add your domain name and the token DuckDNS gives you into its configuration.
- Assuming that went well, your IP address recorded with DuckDNS will update automatically now. Which means when you hit *yourdomain*.duckdns.org it will know how to find your internet router.
- The next step is to setup port forwarding on your router to point to your Raspberry Pi on port 8888 which is our Fish Cam website!
- Now unfortunately I can't give precise instructions here as everyone's router is different but the general method is as follows
 - Login to your router. Usually at 192.168.1.1 or 192.168.0.1 or something like that. Check for a sticker on the back/bottom of your router if in doubt. If you have a modem/router all-in-one from your ISP it may have come with instructions to get into its config page.
 - Often the default login is admin/admin or admin/password but again, check for a sticker with the details or google search the modem model number and you should find it
 - Navigate to your routers port forwarding section, also frequently called virtual server.
 - Create the port forward entry in your router. Usually the following settings would be required

- Name: FishCam
- Port: 8888
- IP Address: 192.168.1.7 (address of your Raspberry Pi)
- Forward Port: 8888
- Protocol: Both or TCP/UDP
- You may need to save and reboot your router. Then to test go to google and type 'whats my ip' copy the number into your address bar and add ':8888' to the end
 - For example <http://115.11.222.33:8888>
- If that works and brings up the FishCam site, happy days!
- Now test <http://yourdomain.duckdns.org:8888>
 - If that does not work don't stress, domain propagation is a likely cause, you may have to wait up to 48 hours for it to propagate fully
 - Check here <https://dnschecker.org/#A/yourdomain.duckdns.org> substituting [yourdomain] for the one you setup
- If everything has gone well, you should have a fully functional externally accessible on-demand Fish Cam!

Some further notes

- Your web-browser might complain about non-secure web-rtc streams. i.e. your website isn't hosted on HTTPS. There is usually a setting in the web-browser to turn this security feature off. Google should help you out if you encounter it.
- The configuration file *uv4l-raspicam.conf* has many many things you can tinker with, you can set up password login requirement to see the camera, and manipulate all kinds of other settings. Read the UV4L documentation to find out more
 - <https://www.linux-projects.org/documentation/uv4l-server/>
- The next step with this project would be to buy your own custom domain with a free dydns service (namecheap.com offers this), setup a self hosted web server such as NGINX and use it as a reverse proxy to the UV4L server. You could also use the LetEncrypt service to generate a HTTPS signed certificate. At that point your camera would be accessible on <https://yourdomain.com> and would be much more secure.
 - I achieved this using an Unraid server running the LinuxServer.io Docker container for LetEncrypt which includes NGINX within it.
 - <https://blog.linuxserver.io/2019/04/25/letsencrypt-nginx-starter-guide/>
 - Its what I have done with my system but the setup of something like this goes beyond my ability to write a guides/tutorials. It is however a really good learning experience and do-able for moderately tech savvy people.