TrailPi User Guide

Josue Aleman | Max Nedorezov | Brody Reeves | Matthew Quesada

Introduction

Background

The Quail Ridge Reserve is a 2,000 acre natural reserve located adjacent to Lake Berryessa, 45 minutes from the UC Davis campus. A unique and experimental wireless mesh and environmental sensor network was established in the reserve in 2004 covering about 45 sites, 40 of which run on batteries and solar power. These sites sync back to a central computer which has a single connection back to the UC Davis campus.

There used to be different types of trail cameras at different locations that took pictures of wildlife during the day and night. Many of these cameras are old and picture quality is not up to par with current camera technology. The best cameras cost around $700 but still require human intervention for camera collection.

The cameras collect the images onto SD cards and have to be manually recovered, often by students. After the images are collected they are manually transferred to a computer and false positives, images without wildlife due to false triggers, are identified and deleted. These images are then labeled with animal type, sorted, and saved for further analysis.

The collection of images from the cameras is wasteful of human resources and the wireless mesh network is currently under-utilized for image collection. Due to the limitations of the trail cameras, they have been retired and there is currently no solution for capturing images of wildlife at the reserve. This is where we come in.

Objectives

1. Design, build, and document a simple camera module that can be easily reproduced in order to collect images from each site over the network.
2. Build a database on a server that will manage the images from all of the sites.
3. Build a web application for easily accessing the database remotely to view, delete, and download the images locally.

Audience

This document contains information about assembling, maintaining, and updating the different components of TrailPi.

1. The hardware
2. The software running on the hardware
3. The client side website
4. The server
5. The database on the server
6. Machine learning algorithms assisting in image filtering

Contact Information

Max Nedorezov
- Developed hardware side including everything to do with the Camera Module
- maxned@yahoo.com

Matthew Quesada
- Developed the server and the client website
- quesadamatthew@yahoo.com

Brody Reeves
- Developed the database side
- brodyreeves@gmail.com

Josue Aleman
- Developed the machine learning image filtering
- josue.aleman87@gmail.com

Github Repository

All code has been under source control from the beginning and can be found in a public Github repository located here: https://github.com/maxned/TrailPi

Social & Legal Aspect

TrailPi is developed as an open source system and is released under the MIT License for anyone to use.

Glossary of Terms

Term	Definition
Trail camera	Weatherproof camera that is specifically designed to take pictures and/or video of wildlife during both the day and at night.
Raspberry Pi (RPi or Pi)	Small linux computer designed for any type of IoT projects. Used in this project for taking pictures and transferring them to AWS.
IR LED	Infrared LED for night illumination
PIR Sensor	Passive IR sensor for camera triggering based on heat
IoT	Internet of Things device
AWS	Amazon Web Services provides lots of services, of which image classification is one and computing power is another.
RDS	Amazon Relational Database Service - hosts the database instance
EC2	Amazon Elastic Compute Cloud - provides the security group used by RDS
DBMS	Database Management System
VM	Virtual Machine
S3	Object storage container provided by Amazon Web Services, used to store images
Flask	Python framework used to create a REST server for connecting our client, database, and various other resources
Elastic Beanstalk	Web server host used to deploy our Flask server to the public web
CNN	Convolutional Neural Network used in the image classification, to determine if the image has a false positive (image without an animal)

General Workflow of TrailPi

This is a description of the general workflow of the project.

1. The Camera Module is assembled, configured, and deployed in the field.
    a. It contains Raspberry Pis with one day camera and one night camera.
2. Once the Camera Module is deployed, the PIR sensor will detect an animal walking by.
3. The Raspberry Pi will take an image and save it to disk.
4. The Pi will then upload this image to the server running on AWS and will tag it as an animal or empty. This tag will be reflected on the web.
5. The server will accept this image, save the image into an S3 bucket, and insert an entry into an SQLite Database for this image, also running on AWS.
6. Concurrently, the Raspberry Pi will send “heartbeat” or “check-in” signals to the server to indicate that the Pi is still alive.
7. This process will repeat many times.
8. At some point in time, you might want to check on a site to view a live feed of what’s going on.
9. You will SSH into the Raspberry Pi at the site and turn on Live Camera Viewing.
10. You will view what is happening at the site and then turn on the TrailPi software again.
11. At another point in time, you will open the website to view what images have been uploaded to the server.
12. You will enter a date range and choose some sites and view the images at these sites.
13. You will be able to delete, download, or tag these images.
14. Once you download the images, in order to help you remove images without anything in them, you will run a machine learning algorithm on the images.
15. Now with the images in separate folders, you will be able to do whatever you wish with the collected images of animals.

Camera Module

The camera module consists of everything physical for the TrailPi project including the software that makes it run. All questions should be directed to Max Nedorezov.

Bill of Materials

These are the materials that must be bought for at least one full camera module.

Name of Part	Price	Quantity	Link to purchase	Total
SparkFun OpenPIR	$18	1	https://www.amazon.com/SparkFun-432843647 1-Sparkfun-13968-OpenPIR/dp/B0711JQH9 1/ref=sr_1_1?keywords=SparkFun+openpir&q id=1557891205&s=gateway&sr=8-1-spell	$18
IR Flash	$12	1	https://www.amazon.com/dp/B01D73XM24/ref =sxbs_sxwds-stvp_1?pf_rd_p=03dd84f0-0dd9 -4d32-b6cb-02a32e0ba336&pd_rd_wg=BWne e&pf_rd_r=9A6YE29T2C2G735FN8E9&pd_rd _i=B01D73XM24&pd_rd_w=RP7e8&pd_rd_r= 3f1fe8c9-6c28-405e-8d0b-2d60c7efc408&ie= UTF8&qid=1550258485&sr=1	$12
Jumper Wires (get both 4" and 8"; 4" are used more than 8")	$5	0.25	https://www.amazon.com/GenBasic-Solderles s-Dupont-Compatible-Breadboard-Prototyping /dp/B077N58HFK/ref=sr_1_4?keywords=fema le%2Bjumper%2Bwires&qid=1557891336&s= gateway&sr=8-4&th=1	$1
Power Supply from 12V	$4	1	https://www.amazon.com/Ship-Hobbywing-Sw itch-mode-UBEC-Lowest/dp/B008ZNWOYY/re f=sr_1_fkmr0_1?keywords=hobbywing+5v%2 F6v+3a+switch-mode+ultimate+bed&qid=155 7891133&s=gateway&sr=8-1-fkmr0	$4
RPi NoIR Camera (Night)	$32	1	https://www.amazon.com/Raspberry-Pi-Camer a-Module-1080P30/dp/B071WP53K7/ref=sr_1 _3?keywords=Raspberry+Pi+NoIR+Camera+ Board+v2&qid=1557891250&s=gateway&sr=8 -3	$32
RPi Camera (Day)	$30	1	https://www.amazon.com/Raspberry-Pi-Camer a-Module-1080P30/dp/B01ER2SKFS/ref=sr_1 _3?keywords=Raspberry%2BPi%2BNoIR%2B Camera%2BBoard%2Bv2&qid=1557891250& s=gateway&sr=8-3&th=1	$30
RPi Kit	$21	2	https://www.amazon.com/Raspberry-Zero-Arg on-Forty-Barebones/dp/B075FPYVTD/ref=sr_ 1_3?keywords=raspberry%2Bpi%2Bzero%2B w%2Bbarebones&qid=1551165953&s=gatew	$41

			ay&sr=8-3&th=1	
16GB Micro SD Card	$6	2	https://www.amazon.com/Sandisk-Ultra-Micro-UHS-I-Adapter/dp/B073K14CVB/ref=sr_1_3?keywords=16gb+micro+sd+card&qid=1557891633&s=electronics&sr=1-3	$12
1 Channel Relay	$10	0.2	https://www.amazon.com/WINGONEER-KY-019-Channel-Module-arduino/dp/B06XHJ2PBJ/ref=sr_1_4?keywords=1+channel+relay&qid=1557892057&s=gateway&sr=8-4	$2
1.75mm White PETG Filament	$21	0.25	https://www.amazon.com/eSUN-1-75mm-Filament-Printer-Opaque/dp/B00ZAUR6M0/ref=sr_1_4?keywords=1.75mm+petg+esun&qid=1558559214&s=gateway&sr=8-4	$5
			Total:	$158
One Time Purchase				
Male Headers	$6	1	https://www.amazon.com/OdiySurveil-2-54mm-Straight-Single-Header/dp/B00UVPT5RI/ref=sr_1_5?keywords=male+headers&qid=1557944916&s=gateway&sr=8-5	$6
M2 x 5mm Screws	$8	1	https://www.amazon.com/uxcell-M2x5mm-Thread-Button-Socket/dp/B01B1OD1D6/ref=sr_1_4?crid=4UJ8PT96Q6IN&keywords=m2+5mm+screw&qid=1558125230&s=gateway&sprefix=m2+5mm%2Caps%2C189&sr=8-4	$8
M3 x 5mm Screws	$8	1	https://www.amazon.com/uxcell-M3x5mm-Button-Socket-100pcs/dp/B01C3LHQ0K/ref=sr_1_6?keywords=m3+5mm&qid=1558130177&s=gateway&sr=8-6	$8

3D Printing the Camera Enclosure

The design of the camera enclosure is provided in the release of TrailPi on the Github Release page here. The files are provided as STL files which need to be imported into 3D printing software to be sliced and then printed.

The parts are provided with the proper face on the bottom, ready for printing. There are 3 separate STL files for each part and 1 STL file with all of the parts.

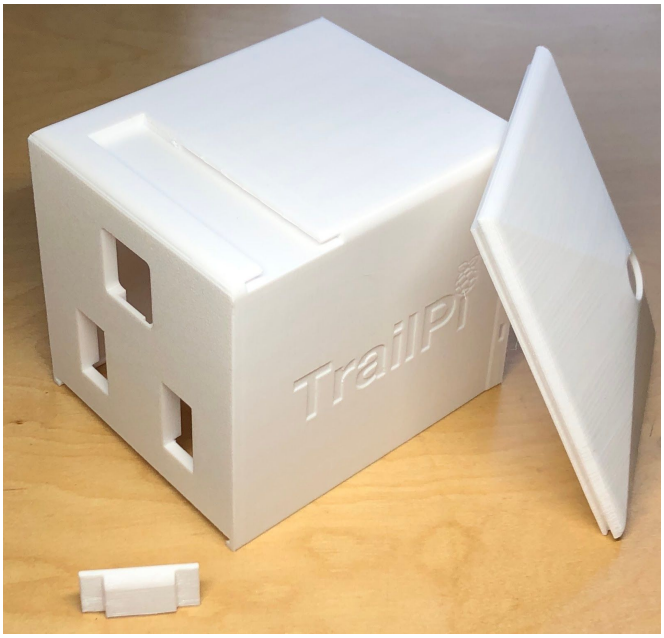
- The **Body** also called Enclosure throughout the guide is the biggest piece
- The **Bottom Cover** is the part that slides on to the bottom of the Body or Enclosure
- The **IR LED Endstop** is the little piece used to secure the IR LED

The enclosure is recommended to be 3D printed with PETG due to the ease of printing and the plastic's ability to withstand temperatures up to 80°C. It is easy to print due to the lack of lots of warping. Theoretically, the plastic is best 3D printed with ASA due to even better thermal properties and excellent UV withstanding properties. The cons of printing with ASA are the higher price and more difficult printing properties due to warping leading to cracks in the 3D printed part.

Since this part is meant to be waterproof, make sure the plastic is dry when printing. Wet plastic can lead to little holes being formed during printing.

Recommended Printing Settings

- Brim enabled to keep the corners from warping up
- For the body, support on build plate only
- No support for the bottom cover and IR LED endstop
- 0.3mm layer height with 20% infill
    - Lower layer heights can be used but they will take longer

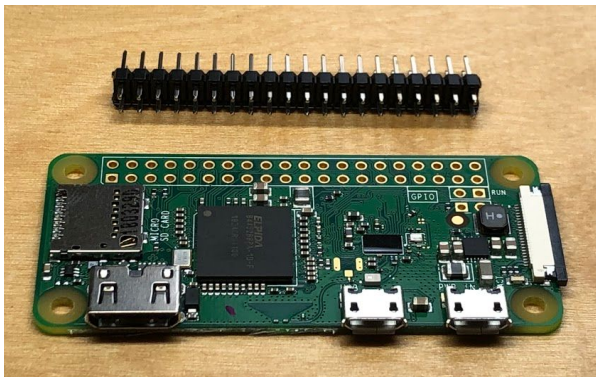


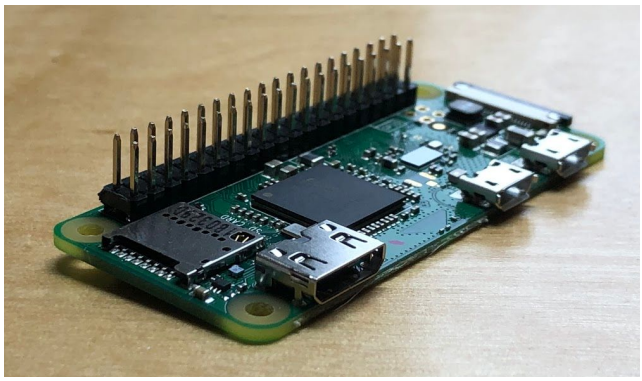
Preparing the Electrical Components for Assembly

Extra Tools Needed:

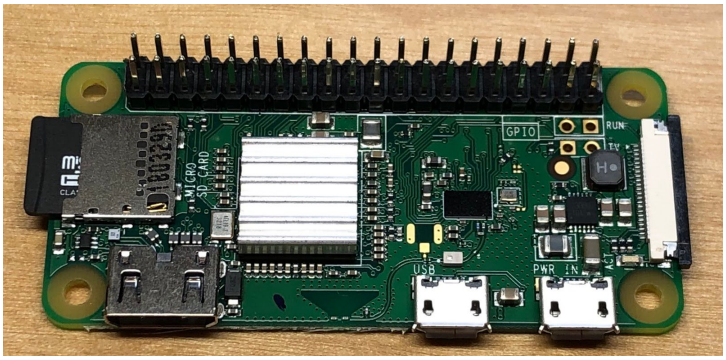
- Soldering Iron with the usual tools i.e. solder, wire crimpers, etc.

Prepare Raspberry Pi



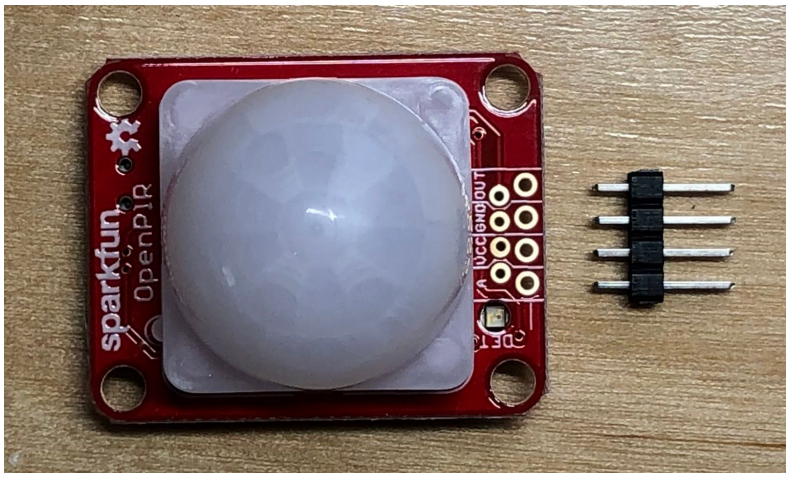


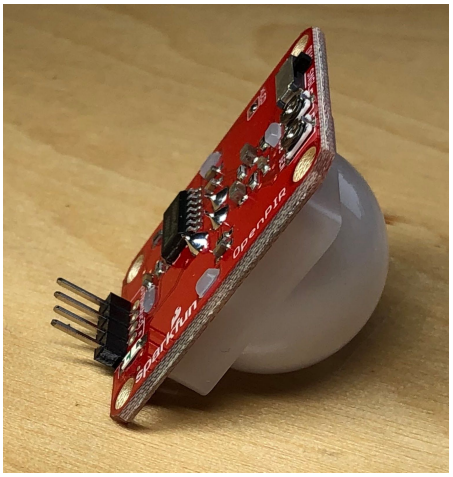
1. Solder the headers to the Pi with the long legs facing up towards the electronics on the board
    a. Be careful to not overheat the board and alternate which pins you solder to let the heat dissipate.



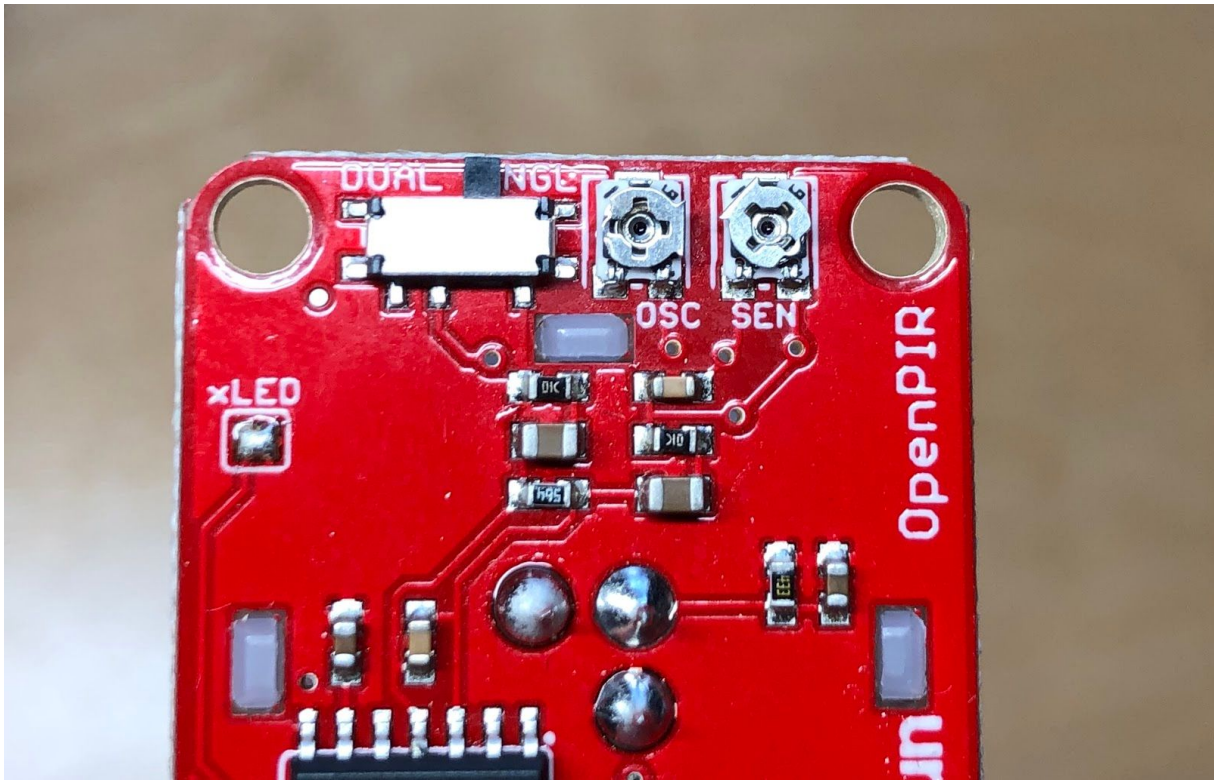
2. Attach the heat sink with adhesive on the back to the Pi, making sure it is centered on the chip.

Prepare PIR Sensor

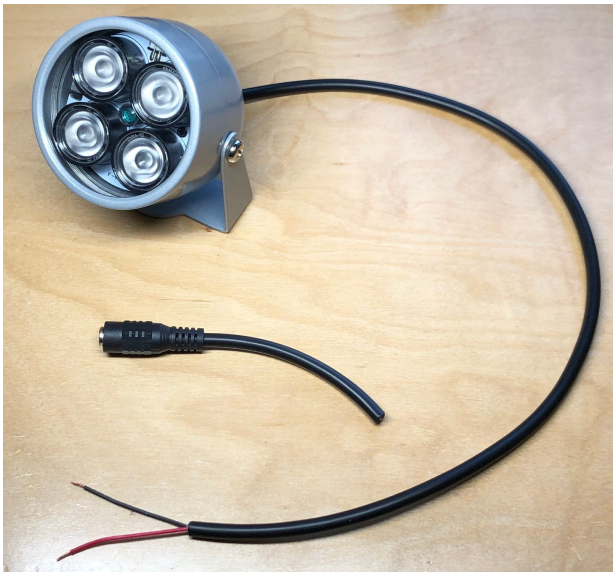




1. Solder a 4-pin or 3-pin male header to the PIR sensor with the long legs facing backwards (we will not be using the A pin, so you can either solder to it or not)
2. This is a good time to configure the sensor
    - Move the switch to specify SINGLE
    - Using a small phillips screwdriver, turn the OSC trimpot counter-clockwise so the flat piece was at 3 o'clock.
    - Turn the SEN trimpot all the way clockwise. Be extra careful to not turn it past the limit.
3. If you find the sensitivity too high, feel free to turn it down later. More information about this sensor can be found here. It may also be useful to change the DUAL mode to SINGLE.

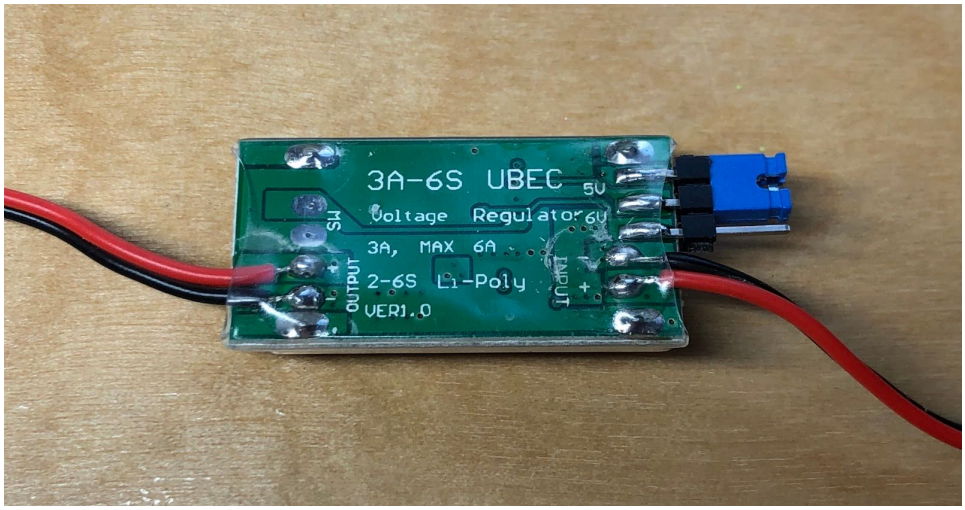


Prepare IR LED Light



1. Cut off the end of the wire a few centimeters from the end.
2. Take off about 5cm of thick shielding from the wire. A red and black wire should now be exposed.
3. Take off about 0.5cm of shielding off of the red and black wires.

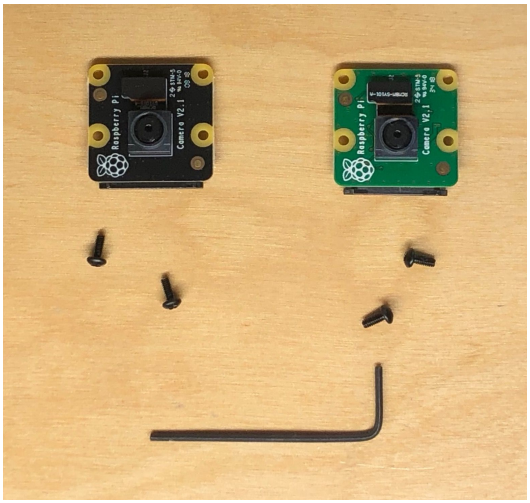
Prepare Power Supply



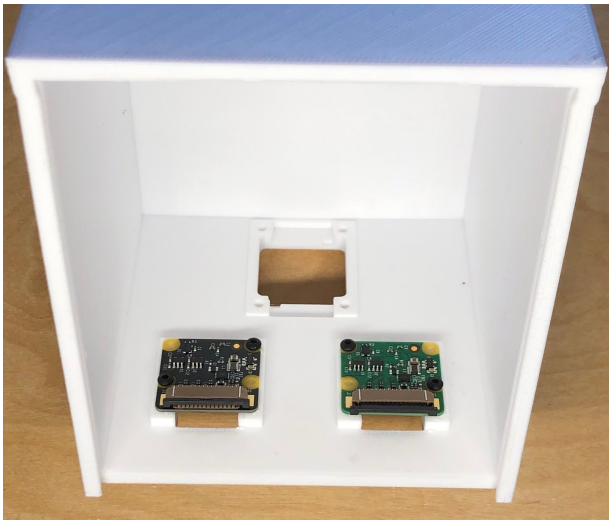
1. Verify that the blue jumper is set to 5V. – 6V WILL DESTROY THE RASPBERRY PI.

Assembling the TrailPi Camera Module

Install Cameras

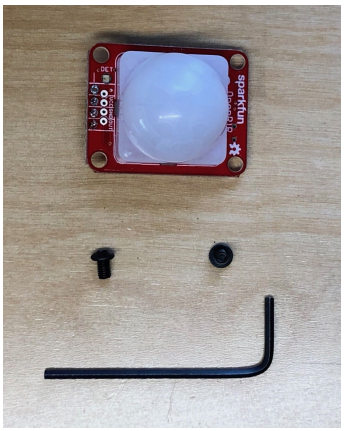


1. Gather both the day and night camera and take off any ribbon cable by carefully pulling back on the black plastic thing.
    a. The day camera is green
    b. The night camera is black
2. You will also need at least 4 M2 screws and a 1.5mm allen wrench.

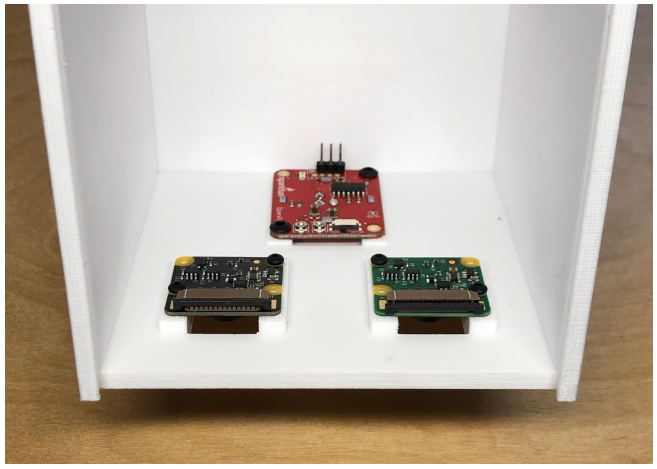


3. Install the cameras inside the 3D printed enclosure with the ribbon cable connector facing down.
    a. Make sure the black camera is on the left, and green camera is on the right.
    b. More than 2 screws for each camera can be used.

Install PIR Sensor

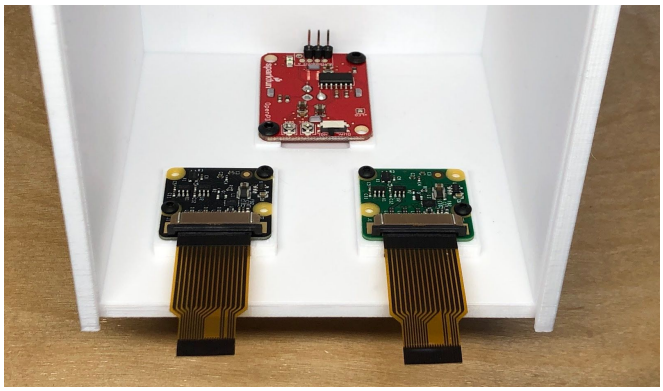


1. Gather the PIR sensor and 2 M3 screws with a 2mm allen wrench.



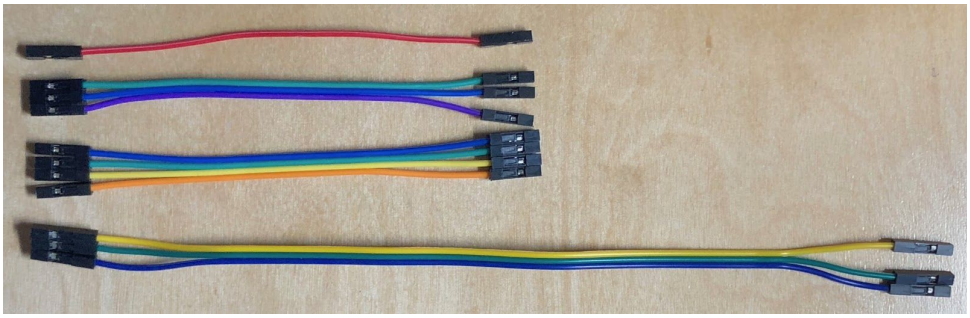
2. Attach PIR sensor with the pins against the top.
    a. The sensor can be installed in any orientation but this way provides easier access to the sensitivity trimpot.

Install Camera Ribbon Cable

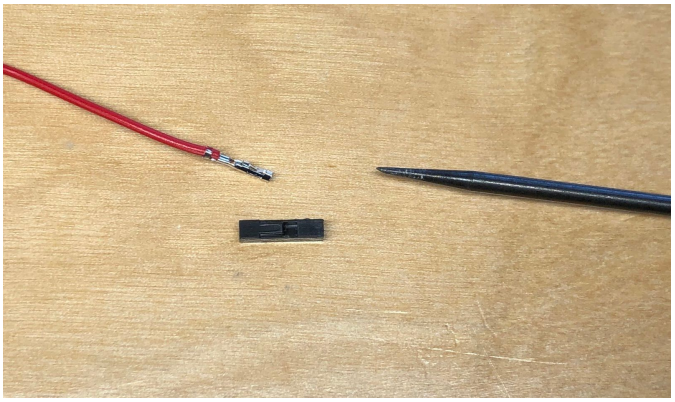


1. Move the black connector and install the ribbon cable as shown. Make sure to push the connector back into place.

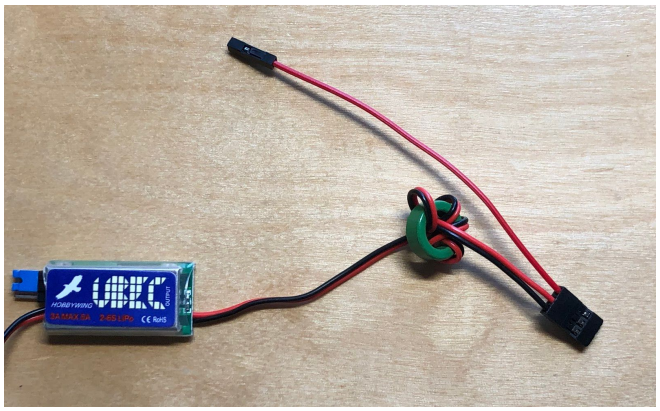
Wiring Everything Together



1. Gather the following wires, colors don't matter. Keeping them connected will make the wiring easier.
    a. 1 short wire
    b. 3 short wires
    c. 4 short wires
    d. 3 long wires

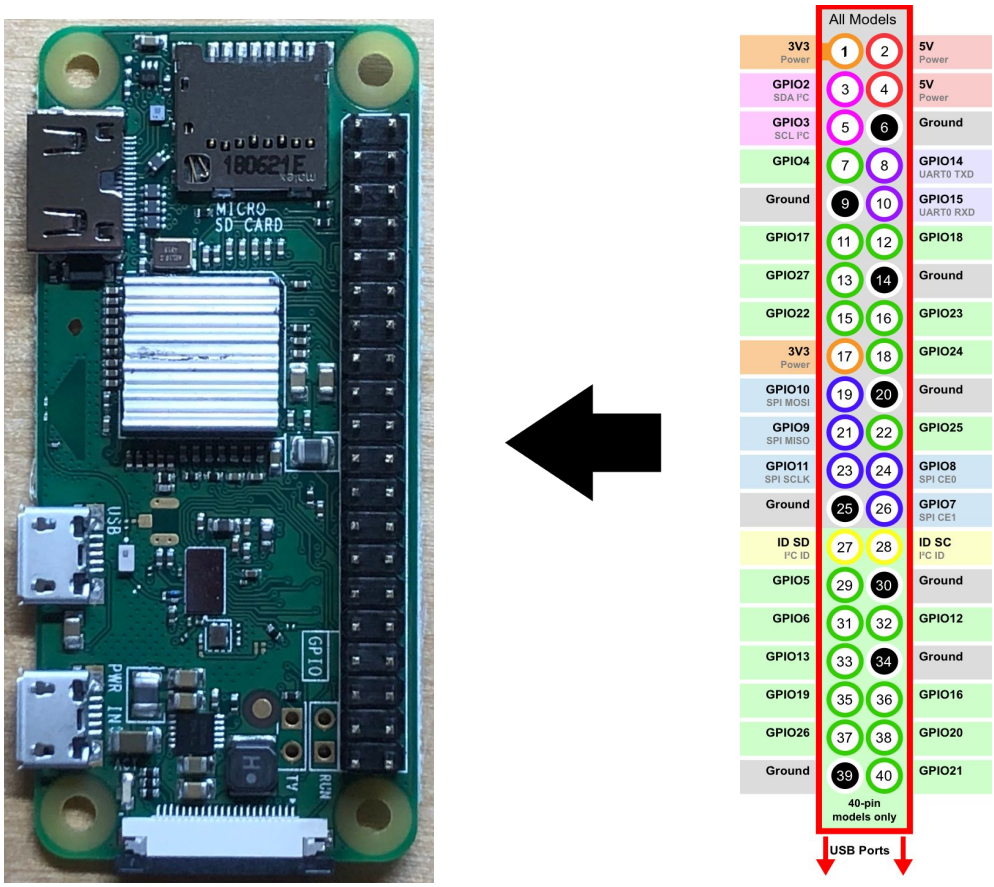


2. Using a sharp tool pick up the flap on the black connector of the single wire and take off the connector.



3. Take this connector and push it into the empty slot on the power supply converter. This will be used to share the power between both Pis.

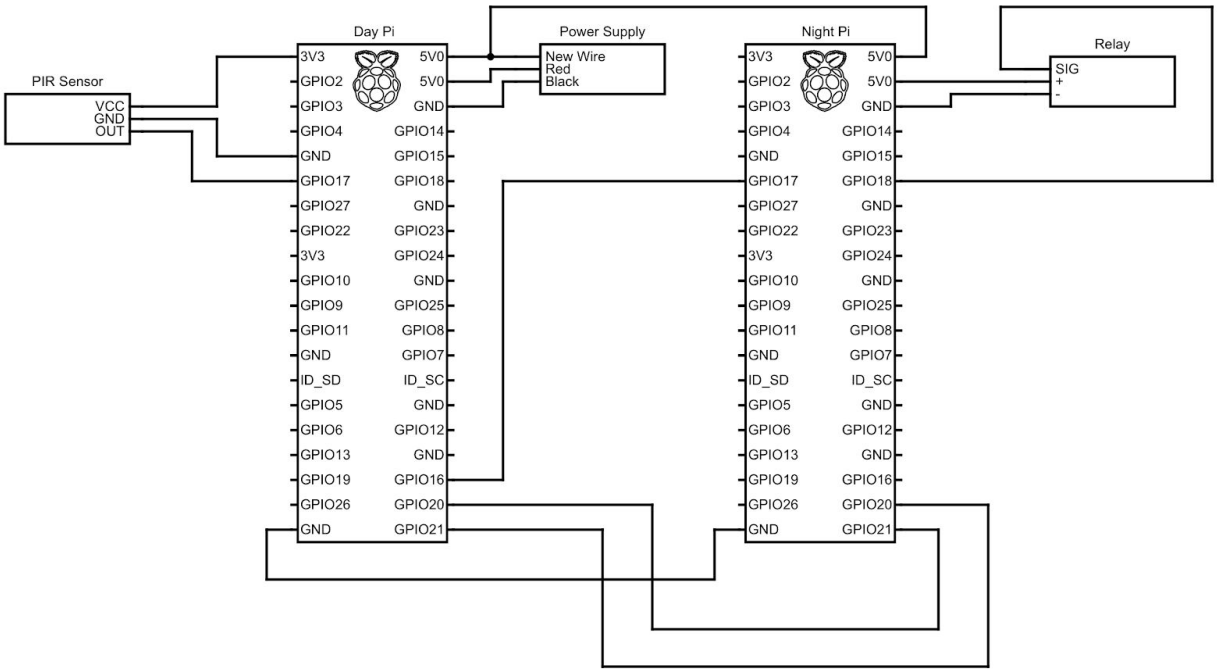
Raspberry Pi Wiring



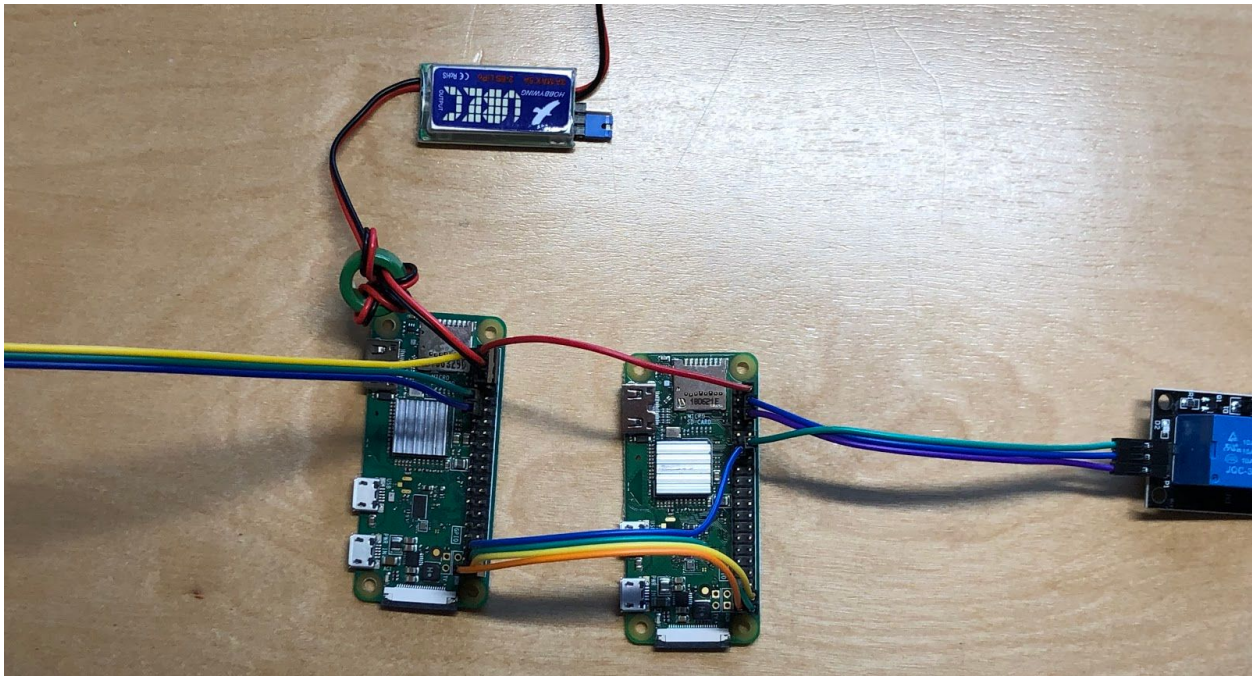
Use this diagram for referring to the pinout numbering on the Pi. Be extra careful when plugging in wires to make sure they are in the correct pins. Wrong pins can fry components or the Pi itself.

- In the configuration file the pins are numbered using the syntax **BOARD40** which would be the pin in the bottom right corner.

Follow the diagram in order to wire everything together correctly.



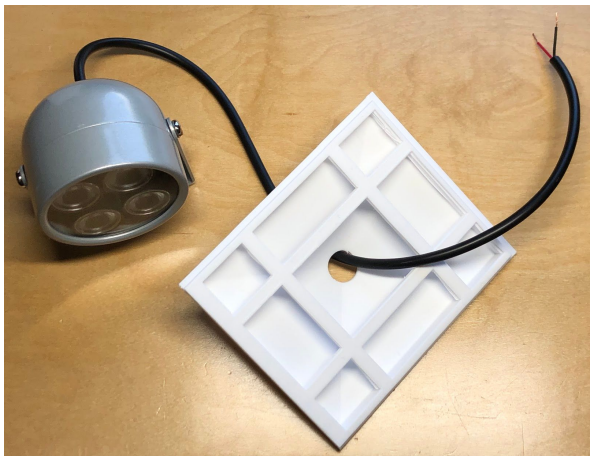
- Use the combined 4 short wires to wire the bottom of the Pi.
- Use the combined 3 short wires to wire the relay.
- Use the 3 long wires for wiring to the PIR sensor.



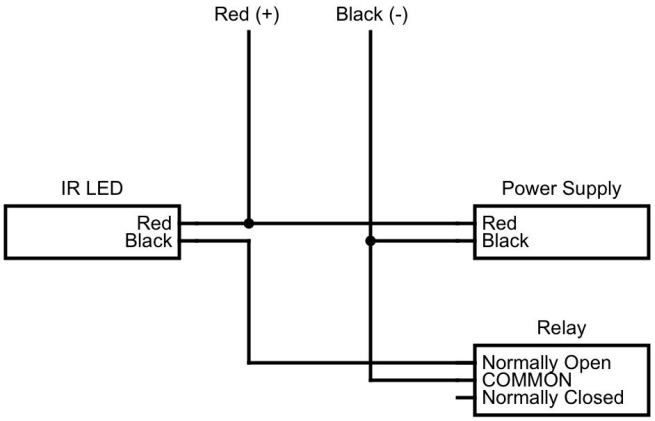
Here is an example of how the wiring should end up looking.

- At this point you can leave the wires to the PIR sensor unattached. It will make the next steps easier.

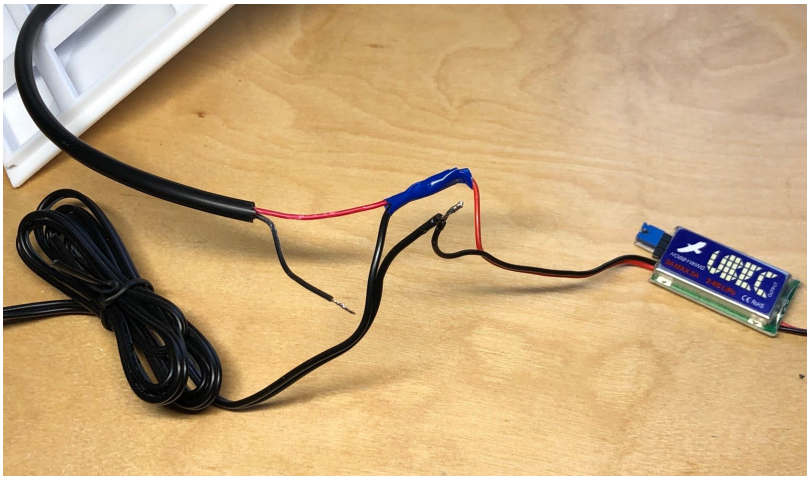
Wiring the IR LED and Power Supply Together



IMPORTANT: Pass the IR LED wire through the bottom cover because after these next few steps it will be impossible.

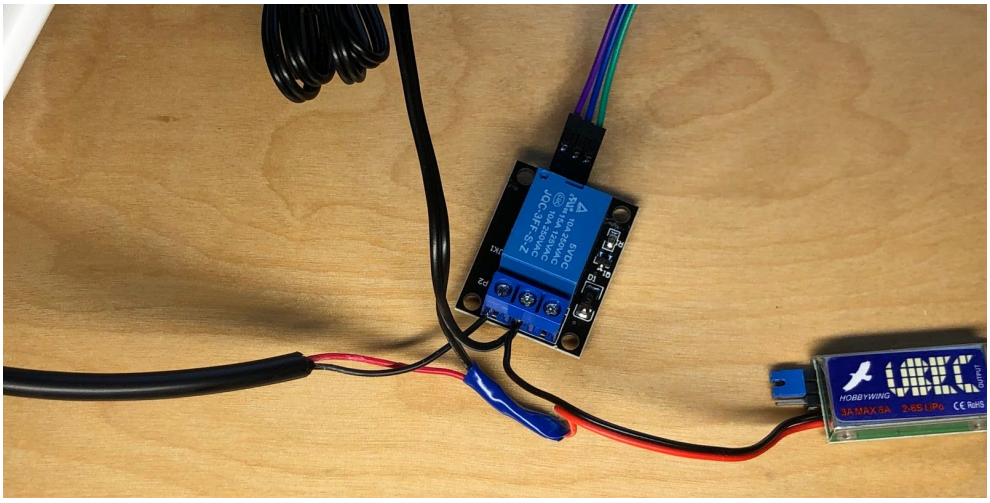


We will be following this diagram to wire the power so that there were only 2 wires leaving the TrailPi enclosure, which are the red and black wires at the top here.



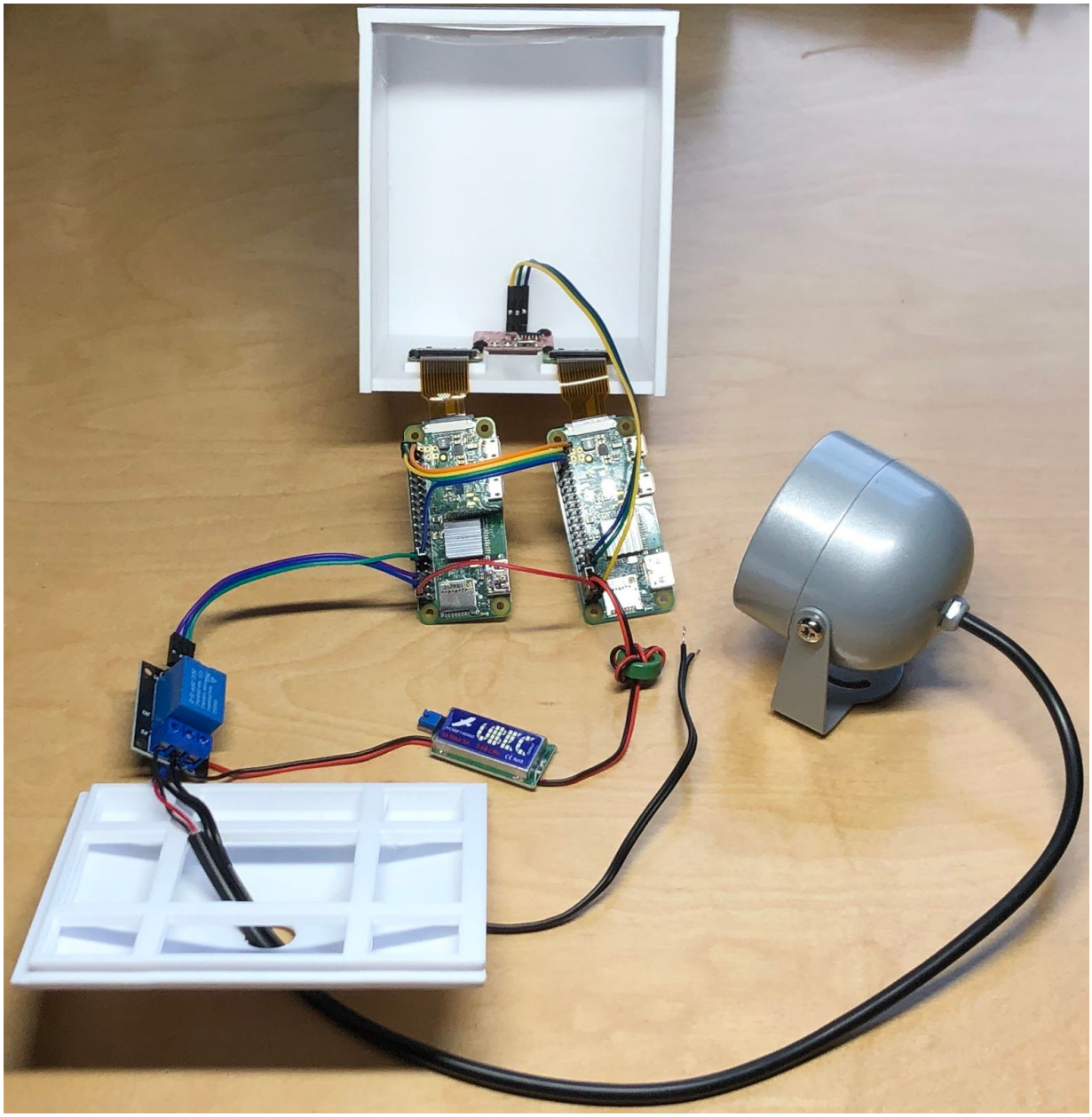
1. The bunched up black wire in the image is 2 wires which will be used as the red and black wires even though they are both black.

2. Solder the red wires facing each other and solder the extra red wire facing towards the IR LED wire so that it was easy for it to leave the bottom cover.
    a. Use electrical tape or shrink wrap to insulate the red wires.
3. Then solder the outside black wire and power supply black wire so that they were facing the same direction, resulting in one, single wire end.



4. Set the 2 connected wires into the middle opening of the relay.
5. Set the single black wire from the IR LED into the left hole.
    a. The connections on the relay from left to right are NO, CMN, NC just as in the diagram above.
6. Pass the new power wires out through the hole in the bottom cover.

Attaching the Pis to the Cameras and the PIR Sensor



1. Attach the Pis to the camera ribbon cable.
    a. The night Pi is on the left connected to the black camera and it has the relay connected to it.
    b. The day Pi is on the right connected to the green camera and it has the power supply connected to it.
2. Attach the long wires on the day Pi to the PIR sensor. Make sure you follow the wiring diagram above.

Install IR LED

1. Slide the IR LED into the slot at the top.
2. Then slide in the 3D printed IR LED Endstop to secure the IR LED.
3. Using some glue or later when sealing everything, secure the Endstop so that it did not come loose.
4. Angle the IR LED slightly down or whatever makes sense in the direction that the camera is facing.

Setup SD Cards

The time has come to setup the software of TrailPi!

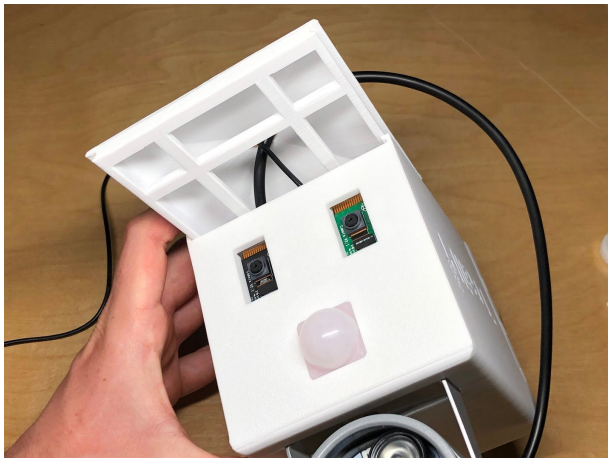
1. Follow the section **Typical TrailPi Setup, Configuration, and Use** to setup the SD cards for each Pi.
2. Then make sure to set the correct day or night config value along with the site number and insert it into the correct Pi.
    a. Day camera is green, night camera is black.
3. Test the Pis to make sure everything is working correctly.
4. Refer to the section **Powering TrailPi** for details on how to turn things on.

Attaching the Bottom Cover

This part is slightly tricky because it is not possible to simply slide the Bottom Cover into the Enclosure.

1. Push all of the components into the Enclosure and try to make the Pis be at the bottom.

2. Take the bottom cover and turn it around so that the flat edge would line up with the front face of the Enclosure when slid on all the way.
3. Insert one edge of the Bottom Cover into the Enclosure making sure that the wires are all inside. As shown in the picture.

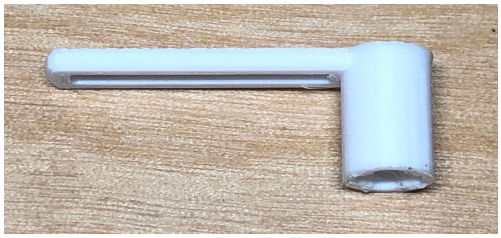


4. Now use some force with both hands to pull apart the Enclosure and the Bottom Cover should snap into place.
5. Slide it all the way down.
    a. It should also be secured with some silicone.
    b. Water should not easily get into the enclosure because the edge design prevents that.

Focus Cameras

The Raspberry Pi camera comes by default with focus set to infinity, meaning objects that are decently far away from the camera will be in focus. However, if you find that the focus is not what you want it to be when for example, animals walk by close to the camera, you will want to to adjust the focus.

Since there will be glass installed over the cameras, make sure to adjust the focus before gluing the glass. Otherwise, you will have to unscrew the camera and then set the focus. Before the glass is installed, we provide a camera focus tool shown below.



To make focusing possible, boot the camera and start the live camera view.

You might have to dig out some glue that holds the lens in place using a sharp scalpel or tweezers if you run into issues turning the camera with the focus tool.

Turning clockwise increases the focus distance and counter-clockwise brings the focus closer. Be mindful that small movements can change the focus quite a bit.

Install Glass Over Cameras

In order to protect the cameras from water, it is necessary to place a glass or plastic cover over them. Measure and cut out of glass or plastic a rectangle and glue it over the front of the enclosure.

Sealing Everything

In order to protect the whole enclosure from water, it is necessary to seal it with some silicone.

1. Seal the glass covering the cameras
2. Seal around the PIR Sensor
3. Seal/secure the IR LED Endstop at the top
4. Seal/secure the Bottom Cover to the Body
5. Plug the bottom hole of the Bottom Cover with some tape or putty so the enclosure did not become a habitat for some animal

Mounting the Enclosure

There are 3 slots at the back of the enclosure that are meant to be used as mounting points. Slip some zip ties through these slots and attach the enclosure where convenient.

Powering TrailPi

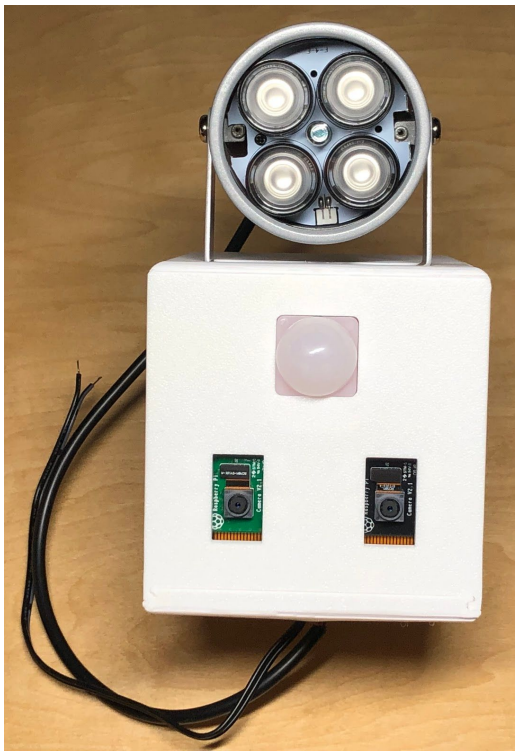
There are a couple ways to power everything.

1. The best way is to use the power supply wires coming out of the enclosure.
    a. Wire the (+) and (-) sides to 12V.
    b. The power supply will convert everything down to 5V for the Pis and the rest of the power will run the IR LED.
2. For testing purposes, you may attach a USB cable to either Pi and both Pis will turn on.

IMPORTANT: Make sure that when turning off the power you have sshed into both Pis and turned them off using the command **sudo halt** and waiting for about a minute.

All Done!

You have completed assembly of the Camera Enclosure! Congratulations! Now deploy it and hopefully it'll capture some good images.



Raspberry Pi Software Configuration

Advanced Image Setup

This section details how the original Raspbian OS that runs on the Raspberry Pi is configured from the default image. This should only be useful for advanced use if you want to see exactly what TrailPi needs installed to run from a stock Raspbian image.

Please follow the Normal Image Setup section for typical use.

OS Image Installation

We will now install the Raspbian Operating System onto an SD card.

Download the latest version of Raspbian Stretch Lite from here onto your computer.

Follow the instructions here to install the OS image onto the SD card.

You should now see the SD card named **boot** connected to your computer. If not, try reinserting the SD card and make sure the image burned properly.

Enabling Wifi

We need to connect the Pi to the current wifi network in order to configure it further. The wifi network details will have to be reconfigured once the Pi is ready to be deployed. Use a simple text editor for the next steps.

- Create a new, empty text file in the **boot** partition called **wpa_supplicant.conf**. This file will be deleted once the Pi starts up and if you will need to change the network details or to fix a mistake, you will need to create the file again.
- Paste the following into this file replacing **YOURSSID** and **YOURPASSWORD** with the wifi details. Make sure to not use a word processor but a simple text editor.

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US

network={
   ssid="YOURSSID"
   psk="YOURPASSWORD"
   scan_ssid=1
}
```

Enabling SSH

We also need to enable SSH to connect to the Pi after it boots up. In order to do this, create an empty file called **ssh** inside the same **boot** folder. This file will be deleted once the Pi starts up. You can use the command **touch ssh** to easily do this.

I took the relevant parts from this tutorial. Refer to it for more information.

Enabling the Camera and USB OTG

There are also some things that we need to enable while we are here such as the camera and interface to the other Pi.

Find the file named **config.txt** in the **boot** folder. Open it and add the following lines to the bottom.

 # Enable camera
start_x=1
gpu_mem=256

 # Enable USB communication
dtoverlay=dwc2

Enabling USB OTG

To make it easier to login to the Pi for testing if there is no wifi network, we want to enable a virtual network connection to the Pi over USB.

To do this, open the file named **cmdline.txt** in the **boot** folder and insert **modules-load=dwc2,g_ether** after **rootwait**. Make sure there is only one space before and after the text. Do not change anything else.

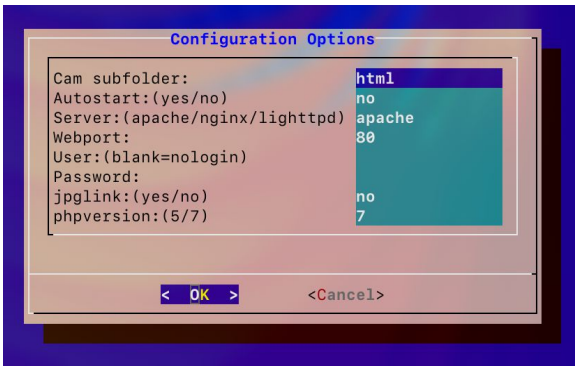
More information can be found here under the section called **Enabling OTG.**

Configure Raspberry Pi

These are the steps continuing to configure the Pi for setup that are not in the **Typical TrailPi Setup, Configuration, and Use** section.

1. Make sure you're on the same network or plugged into the USB port and type in **ssh pi@raspberrypi.local**
    a. The default password is **raspberry**
2. Type **passwd** and change the login password to **trailpi1234**
3. Set the timezone (run these commands)
    a. **sudo rm -f /etc/localtime**
    b. **sudo ln -sf /usr/share/zoneinfo/America/Los_Angeles /etc/localtime**
4. Update everything using **sudo apt-get update && sudo apt-get upgrade**
5. Install git **sudo apt-get install git**

6. Install PiCamera **sudo apt-get install python3-picamera**
7. Install GPIO Zero **sudo apt install python3-gpiozero**
8. Install pip **sudo apt install python3-dev python3-pip**
9. Install more stuff **sudo apt install libatlas-base-dev**
10. Install RPi-Cam-Web-Interface
    a. **git clone https://github.com/silvanmelchior/RPi_Cam_Web_Interface.git**
    b. **cd RPi_Cam_Web_Interface**
    c. **./install.sh**
        i. When prompted change Autostart to no
        ii. Hit tab to move to the OK then hit enter



        iii. After the install, do not automatically start RPi_Cam_Web_Interface when prompted
11. Change the default hostname to trailpi
    a. **sudo nano /etc/hostname** and change **raspberrypi** to **trailpi**
    b. **sudo nano /etc/hosts** and change **raspberrypi** to **trailpi**
    c. Then run **sudo reboot**
    d. All subsequent logins to the Pi can be **ssh pi@trailpi.local**

These are the steps for preparing the Raspberry Pi to run the image classifier neural network. Due to the limited memory on the Pi, we need to setup a swap file. These steps, especially the last one will take a very long time. Up to a day.

1. **sudo nano /etc/dphys-swapfile**
2. Add this to the bottom of the file **CONF_SWAPSIZE=1024**
3. **sudo dphys-swapfile setup**
4. **sudo dphys-swapfile swapon**
5. **sudo apt-get install libjpeg-dev zlib1g-dev**
6. **sudo apt install libtiff5**
7. **sudo apt-get install libjasper-dev**
8. **sudo apt-get install libilmbase-dev**
9. **sudo apt-get install libopenexr-dev**
10. **sudo apt-get install libgstreamer1.0-dev**
11. **sudo apt-get install libqtgui4**
12. **sudo apt-get install python3-pyqt5**
13. **sudo apt install libqt4-test**
14. **sudo apt-get install libhdf5-dev**
15. **sudo pip3 install Cython**

16. **sudo pip3 install -r requirements.txt** in order to install all the packages needed
17. **sudo dphys-swapfile swapon**

Installing TrailPi Software on the Pi

1. Download the latest version from the Github Release page here using **wget <link>** replacing **<link>** with the link from the Release page.
    a. Place it in the home directory so all of the files were in **~/TrailPi/**
    b. Make sure to first delete and existing TrailPi using **rm -rf ~/TrailPi**
2. To setup the script to run at launch, type **crontab -e** and add the following line to the bottom of it **@reboot /home/pi/TrailPi/trailpi_launcher.sh**
3. Copy the **trailpi_config.json** file to the **/boot** to make it easier to setup the default parameters.

This concludes the steps needed to setup the Pi for use with TrailPi from a default Raspbian Stretch Lite image. The following section will detail the actual setup of TrailPi for typical use.

Typical TrailPi Setup, Configuration, and Use

Installing the Image

We first need to install the TrailPi image onto a micro SD card of at least 4GB. In the BOM, they are 16GB.

Download the **trailpi_image.zip** from the Github Release page here.
Unzip the file and you should have the file named **trailpi.img**.

Follow the instructions here to install the OS image onto the SD card. After the image is burned onto the SD card. Eject it and mount it again on the computer.

Configuring TrailPi

TrailPi must be configured for each site. There are some configuration options that must be set and the network must be setup. Most of this can be done with access to the SD card, but getting the MAC address of the Pi will require booting it up.

Software Configuration

There is a single configuration file used for TrailPi that allows tweaking a lot of the parameters used to capture images and upload them to a server.

The configuration file is found in the **/boot/** folder in the file **trailpi_config.json**. If this file exists in the **/boot/** folder then it will be used by the software. If it is deleted, then the configuration file located in **~/TrailPi/** will be used instead. I suggest keeping the config file in the **/boot/** folder because it allows for easy setup while still having local access to the Pi SD card.

With the SD card still in the computer, open this file and edit it directly using a simple text editor. Be mindful of the quotes and commas and do not delete any of them or add extra ones. Keep the format as is, otherwise the file may not be able to be read by the software.

TrailPi Configuration File Fields

Specified is the field and the default value in parentheses.

site_number (1): This field specifies the site number. It should be any integer representing a site.

camera_type (“day”): This specifies the type of camera attached to the Pi. The only valid options are **“day”** or **“night”**. If only using one Pi or camera, default to specifying “day”.

resolution ([3280, 2464]): This is the resolution with which an image is taken. It should not exceed the maximum value of the sensor which may lead to issues. It is set by default to the

highest resolution of the 8MP cameras. Format is a tuple describing the width and height, respectively.

inter_capture_delay (0.5): This is the time in seconds between when images are taken. Effectively it governs how fast images are captured and how fast the camera responds to exposure changes. I would not recommend setting it lower than 0.25, but it can be set higher to split chunk captures apart a little.

quality (10): This is the quality with which a JPEG is saved of the captured image. In tests, this is virtually identical in appearance to full quality, but drastically reduces image size from about 4MB to 700KB. More information can be found here.

capture_chunk_size (3): This is the total number of images that will be saved during an image capture event i.e. an animal walks by. Say the capture_chunk_size is 5, then 2 images will be saved before an event happens, that is they are buffered, 1 image during the event, and 2 after the event.

ml_tagging (true): Turn on or off the ability to tag images when uploading them to the server using machine learning.

ml_animal_threshold (0.5): This is used by the Machine Learning algorithm on the Pi to automatically tag images before uploading them to the server. Theoretically, increasing this value will make it harder for an image to be classified as having an animal, and lowering it will lower the threshold for tagging an image as an animal. However, due to how the algorithm actually works, it predicts the image with values really close either 1 or 0.

day_to_night_analog_threshold (7.5): This specifies the analog gain threshold of the camera at which the day Pi will switch to the night Pi. Valid values are between 1.0 - 8.0. When it gets dark, the analog gain rapidly increases to 8.0 so keeping this value close to 8.0 is ideal. If pictures become too dark on the day camera, lower this value slightly. This threshold is also combined with the **day_to_night_digital_threshold**. If the gain goes above both of these, then the switch happens.

day_to_night_digital_threshold (1.1): This is the digital gain threshold which is combined with the analog gain threshold to know at what lighting conditions to switch to the night camera. Based on my testing, valid values are between 1.0 - 1.16. The digital gain starts to slightly increase after the analog gain is almost at the maximum.

night_to_day_analog_threshold (3): This is the same as the **day_to_night** but going from the night camera to the day camera. This value is much lower so the night camera switched when it is already a little bit brighter. Keeping these values further apart will decrease the chance of the cameras switching back and forth in the same lighting conditions.

night_to_day_digital_threshold (1.1): This is the same as the **day_to_night** but going in the opposite direction.

ir_led_always_on (true): This specifies if the IR LED is always on at night or it only turns on when the PIR sensor is triggered. The only other option is specifying **false**.

More information about pins in general can be found here. The format for the pins is followed as defined here.

pir_input_pin (“BOARD11”): This is the input pin on the Pi from the PIR sensor OUT on the day Pi. On the night Pi, this is connected to the day Pi’s **pir_output_pin**. In order to share the PIR sensor state, the day Pi mirrors it to the night Pi.

pir_output_pin (“BOARD36”): This output pin mirrors the state of the **pir_input_pin**.

pi2_input_pin (“BOARD38”): This is the input pin of the current Pi that should be connected to the other Pi’s **pi2_output_pin**. This is used to coordinate the day to night camera switch.

pi2_output_pin (“BOARD40”): This is the output pin of the current Pi that should be connected to the other Pi’s **pi2_input_pin**. This is used to coordinate the day to night camera switch.

relay_output_pin (“BOARD12”): This output pin controls the state of the relay.

debug (false): This enables some special debug functionality, namely manually triggering an image capture event using the SIGINT signal.

image_timestamp_format ("%m-%d-%y--%H-%M-%S-%f"): This is the format of the date that is saved as the filename of the image. This shouldn’t be changed unless also changed on the server. More information about the fields can be found here.

image_folder (“images”): This is the folder that images are saved into. Shouldn’t need to be changed.

image_upload_url ("http://flask-server.wqwtbemyjw.us-west-2.elasticbeanstalk.com/TrailPiServer/api/image_transfer"): This is the URL that accepts the properly formatted image to be accepted to the server.

image_upload_check_interval_sec (30): This is the interval in seconds between which the script tries to upload images to the server. It checks if there are any images that need uploading and uploads them, then deletes them.

image_upload_valid_image_delta (10): Since there are multiple scripts running at the same time, an image may be saving to disk at the precise moment that it may be tried to be uploaded to the server. In order to prevent a corrupt image from being uploaded, the script verifies that there have been at least this many seconds since the time that the image was taken before it can be uploaded to the server.

check_in_url ("http://flask-server.wqwtbemyjw.us-west-2.elasticbeanstalk.com/TrailPiServer/api/check_in"): This is the URL that accepts the check-in “heartbeat”. The properly formatted object can be found in the code.

check_in_interval_sec (1800): This is the interval in seconds between when the Pi checks in to the server. This is the “heartbeat” that tells the server that the Pi is still alive.

Per-Site Configuration

IMPORTANT: To properly set up the Pi at each site, it is important to set the **site_number** and **camera_type** fields. All other fields can be left as default without problem.

Network Setup

To connect the Pi to the network, there are 2 options.

This first method can be used with access to the SD card with the Pi off.
Use a simple text editor for the next steps.

- Create a new, empty text file in the **boot** partition called **wpa_supplicant.conf**. This file will be deleted once the Pi starts up and if you will need to change the network details or to fix a mistake, you will need to create the file again.
- If the network is SECURED, paste the following into this file replacing **YOURSSID** and **YOURPASSWORD** with the WiFi details. Make sure to not use a word processor but a simple text editor.

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US

network={
   ssid="YOURSSID"
   psk="YOURPASSWORD"
   scan_ssid=1
}
```

- If the network is NOT SECURED, paste the following into this file replacing **YOURSSID** with the WiFi name. Make sure to not use a word processor but a simple text editor.

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US

network={
   ssid="YOURSSID"
   key_mgmt=NONE
   scan_ssid=1
```

}

The second method is boot and log in to the Pi and edit the file **sudo nano /etc/wpa_supplicant/wpa_supplicant.conf** directly and paste the above. Then make sure to run **sudo reboot**

Static IP Address Setup

If you are using Linux, then you can setup the static IP address right now with the SD card plugged into the computer. If not, then you will need to boot the Pi and ssh into it to setup the static IP address. You can follow the instructions in **Booting and Logging in to the Pi** to do that.

If using Linux, then you can navigate to the folder **cd /media/<USERNAME>/rootfs/** to get to where the volume **rootfs** has been mounted.

We need to edit the file **/etc/dhcpcd.conf** which can be done while ssh'd into the Pi or from Linux by editing it directly on the SD card. This has to be done with root privileges **sudo nano /etc/dhcpcd.conf**

At the bottom of the file add the following lines.

interface wlan0
static ip_address=192.168.1.53/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1

The **ip_address** field is the IP address that you want plus the subnet mask size in CIDR format.
The **routers** field is the IP address of the gateway router.
The **domain_name_servers** field is the IP address of the DNS server that is being used, which could be the IP address of the gateway router depending on how you have things set up. There could be multiple addresses that should be separated with a single space.

Getting the MAC Address

To obtain the MAC address, it is necessary to boot the Pi. Follow the **Booting and Logging in to the Pi** section to login. Once you have sshed into the Pi, return back here and run the following command to get the MAC address.

Get the MAC address using the command **ip link show wlan0 | awk '/ether/ { print $2 }'**

Booting and Logging in to the Pi

To gain access into the Pi we will use ssh. This will be easiest on a Mac or Linux machine. This is also possible

If the Pi is local, then plug a micro usb cable into the USB port. We will use a USB OTG connection which virtualizes a network over USB allowing us to connect to the Pi without a

nearby wifi network. After about 30 seconds after plugging in the Pi, you should be able to run the following command on your machine, **ssh pi@trailpi.local** and gain access to the Pi.

The password is !**TrailPi@1234!** for the user **pi**.

If you want to change the password for ssh, run the command **passwd** on the Pi and follow the prompts. This will be the new password when sshing into the Pi.

If the Pi is connected to another power source but is on the same network then you are fine and can run **ssh pi@trailpi.local** just as above.

If the Pi is on another network, or there are more than one Pi's connected to the same network, you will not be able to use the **trailpi.local** hostname, but will need the IP address of the Pi. One you have the IP address, **ssh pi@192.168.1.53** for example.

Extra Commands

exit - exit an ssh session
sudo halt - turn off the Pi
sudo reboot - reboot the Pi

Precautions for the Pi

- Do not unplug the power. If possible, SSH into the Pi and **sudo halt**. Unplugging the Pi may corrupt the SD card, resulting in you needing to reflash it or replace the SD card in a worst case scenario.
- Do not connect or disconnect the camera when the Pi is powered on.
- Do not under any circumstance disconnect the SD card while the Pi is on.

Live Camera Viewing (Bonus)

TrailPi provides the ability to get a live view into a camera. In order to do this there are a few steps that must be followed.

Turning on Live Camera View

1. SSH into the Pi and **cd ~/TrailPi/** (we need to be in the TrailPi directory)
2. Run **./start_live_view.sh**

Live camera viewing is now enabled. Now, in a web browser on your computer or phone, navigate to the IP address of the Pi. There you should see the live view.

Live camera viewing can be individually stopped without turning on TrailPi by passing the **-s** option to the script as such **./start_live_view.sh -s**

IMPORTANT: Once finished viewing the live view, you need to start TrailPi again. TrailPi and the live view cannot run concurrently.

Restarting TrailPi

1. Once again SSH into the Pi if you closed the connection and **cd ~/TrailPi/** (we need to be in the TrailPi directory)
2. Run **./start_trailpi.sh**

TrailPi can be stopped by running the script with the **-s** argument as such **./start_trailpi.sh -s**

You are now done. TrailPi automatically launches on a reboot so you don't have to worry about launching it manually every time.

Dealing with Problems on the Pis

While TrailPi is as robust and tested as possible, there is still a possibility of something going wrong. Debugging different issues may be difficult without good knowledge of Linux. In order to help with this, TrailPi logs what it does and any errors that it may run into.

TrailPi keeps logs of normal operation in a folder called **logs** inside the **TrailPi** directory. There is a separate log per each file. These may be helpful in a debugging event.

There are also crash logs that are saved, which are a **stderr** output for each file in the case of a crash or other errors that the normal logs do not save. These should also be viewed in the case of any errors.

In the case of wireless connection problems, it may be necessary to access the Pi's locally.

If an SSH session can be established but there are issues with taking pictures or switching cameras, the Pi is alive and well, but there may be an issue with the wiring.

The best way to figure out what went wrong is looking at the logs and crash logs.

Tip: TrailPi can always be relaunched by running **./start_trailpi.sh** but I doubt this would help with anything.

If the Pi seems to not want to connect to a camera, try reseating the ribbon cable. If it still does not want to connect, the camera might have died and needs to be replaced.

The Pi can fail hardware wise. If it still boots up but freezes or kernel panics, it may mean that the hardware failed and needs to be replaced.

TrailPi Setup Checklist

1. All 3 parts of enclosure are printed
2. All wires and components are connected properly
3. Double check all connections
4. Enclosure is assembled completely
5. Enclosure is sealed and waterproofed where necessary
6. Each Pi is configured with **site_number** and **camera_type**
7. Pi is connected to network
8. All done!

Future Work

These are some potential features that may be implemented into the Raspberry Pi TrailPi software.

1. Make the check-in heartbeat mean more by only sending it if all scripts look healthy and are not crashing.
2. When starting the live camera view, also start the IR LED to see at night.
3. Add a smart option to the image capture to stop saving images if too many are taken within a short period of time.
4. Run a web server on the Pi that allows easy switching between TrailPi software image collection and live camera view.
5. Make the TrailPi customization options available on a website running on the Pi.
6. Make the TrailPi live camera view available through the main website.
7. Make each Pi camera be customizable from the main website.

Machine Learning

Image recognition using a 5-layer CNN

Implementing an image classifier as either a stand alone application or integrated into the camera module gives the user the advantage of sorting images that are tagged as empty, reducing the amount of data that is in the server or on local storage. The current script has the ability to sort a folder with images that are not tagged and create two folders that contain animals or humans in one and nothing in the other one.

When the neural network is integrated into the Raspberry Pi zero, it has a predictor that will return the probability that the image the Raspberry Pi took contains an animal or human in it. Depending on the threshold that the configuration file in the module has, it will either tag the image as a good picture (one with an animal or human in it) or as a bad picture.

Requirements

To install packages that will work with the CNN model type into the terminal:

```
$ pip3 install -r requirements.txt
```

Running the Test Script

The script used to run:

```
$ python3 test.py
```

The script will load in the model and then will ask the user if they want to apply the model on a folder or a single image :

```
$ Folder(F) or Single File(S)?: _
```

Choosing Folder(F)

1. If user selects folder, then the script will check locally if `**animals_and_humans**` and `**nothing**` folders already exists, if they don't then it will create them. Then the user is asked to enter the folder name (local to the script) of the images they want to classify:

```
$ Folder Name: folder_name_here
```

2. Hit `**Return**` and the script will begin sorting the images into their respective folders (`**animals_and_humans**` or `**nothing**`).

3. Once complete, the prompt will ask if the user would like to continue and sort other files or folders with the following prompt:

$ Continue (Y/N): _

4. If user selects **`'N'`** then the script will terminate, and if the user selects **`'Y'`** then the script will repeat the steps above asking if the user wants a folder or single file.

Choosing Single File(S)

1. If the user selects single file, then the script will prompt the user to give the name of the image (must be local to the script) as such:

$ ImageName: my_image.jpg

2. The script will then print out if the image contains an animal/human or if there is nothing in the image.
3. Once complete, the prompt will ask if the user would like to continue and sort other files or folders with the following prompt:

$ Continue (Y/N): _

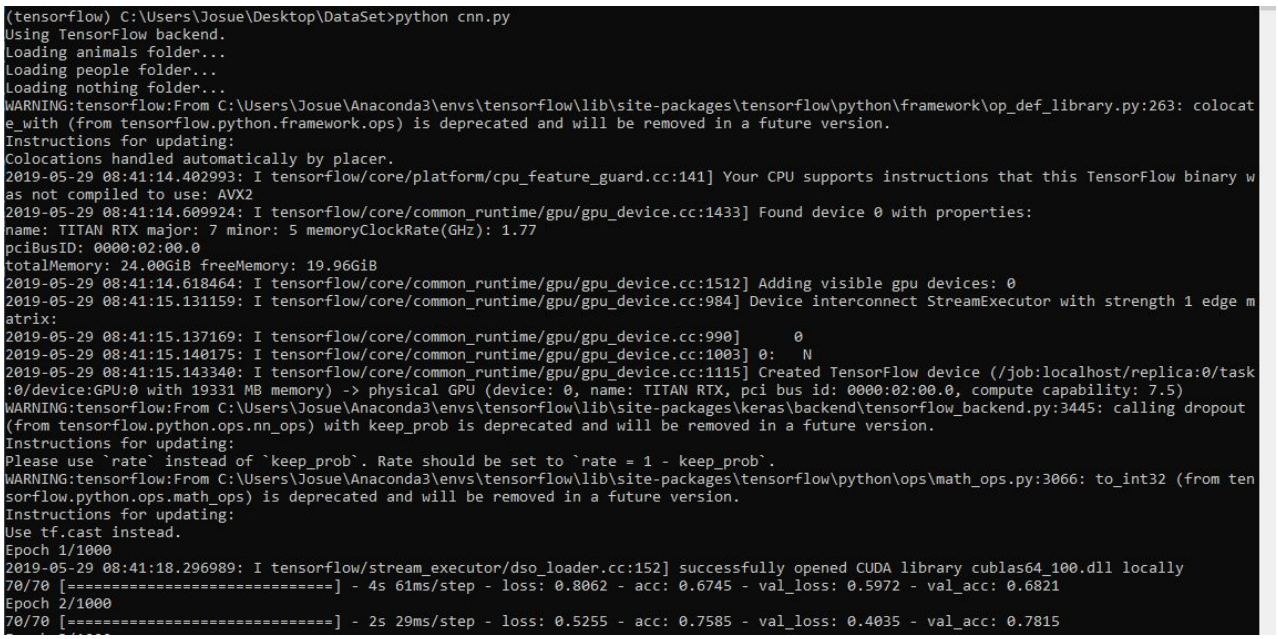
4. If user selects **`'N'`** then the script will terminate, and if the user selects **`'Y'`** then the script will repeat the steps above asking if the user wants a folder or single file.

Retraining the Image Classifier

1. Ensure that the **requirements** are installed before running the model.
2. Ensure that the script `cnn.py` has `animals`, `nothing` and `people` folder local to it. The reason the `animals` and `people` folder are kept separate is in case the user may want to modify the classes to determine if its a person or animal instead.
3. In a terminal window run:

$ python cnn.py

4. The script is now creating a new model with the provided images. A prompt like the following will show:



5. A **`labels.npy`** and **`trail_pics.pny`** file will be created, these are the numpy representation of the images, very useful if later the user decides to use the same images, the user would uncomment the following lines in the **`cnn.py`** file:

```
trail_pics = np.load("trail_pics.npy")
labels = np.load("labels.npy")
```

6. Once the neural network is done with it's training and validation, two files will be created locally named **`model.h5`** (containing the weights for the model) and **`model.json`** (containing the actual model).
7. Replace the **`model.h5`** and **`model.json`** that are local to the **`test.py`** and **`classifier.py`** folder with the new ones generated.
8. The new model will now be used by the **`test.py`** to make predictions on new images.

Note: This script can use both **tensorflow (v1.12.0)** and **tensorflow-gpu(**. For a faster training, it is recommended to use tensorflow-gpu. Details on requirements and installation for tensorflow-gpu can be found on https://www.tensorflow.org/install/gpu

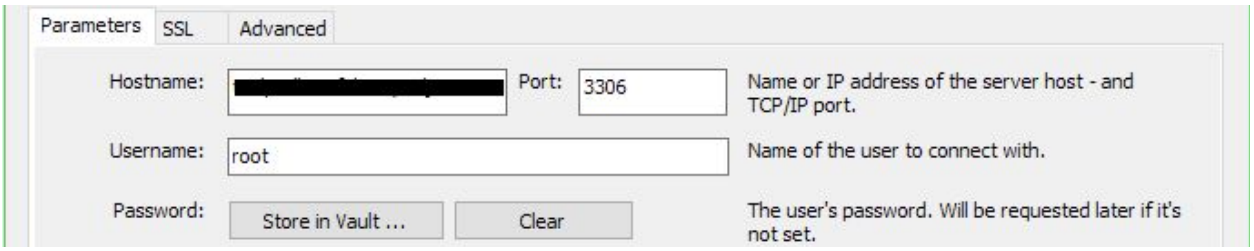
Database

The database is used to store and organize the data for each picture, such as the date taken, site number and site name. It also doubles as the backend for authentication of users and storing the user's credentials. It is worth noting that retrieving the data from the database has a cheaper cost and is also more efficient than simply retrieving the data from the S3 bucket every time we want to check what pictures are stored.

Accessing the Database

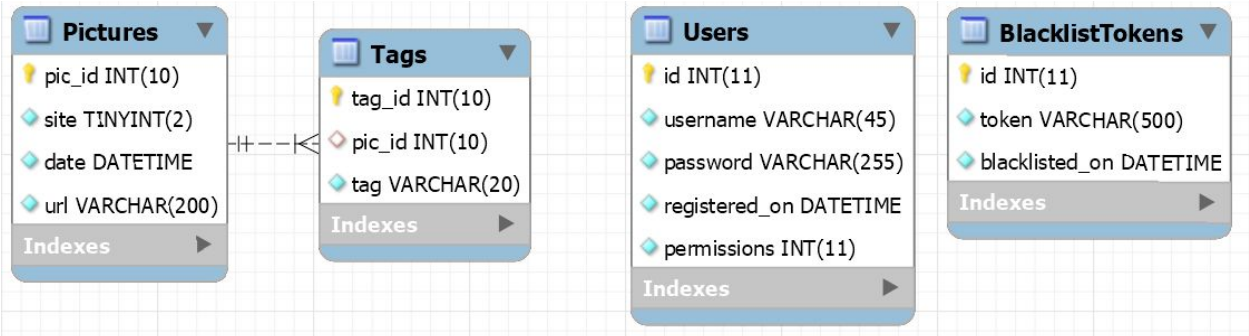
In case you wish to log in to the database and view the tables, or manage them in some form, follow the steps below to setup a connection that allows execution of SQL commands on the database.

1. Log into AWS management console at https://console.aws.amazon.com using the Amazon account login info
2. Go to 'Services' > 'RDS' > 'DB Instances' > Click on the database for TrailPi
3. Looking at 'Connectivity & Security', note the **endpoint** and **port** used by the database
4. Click on the 'VPC security group', then on the 'Inbound' Tab click 'Edit' and 'Add Rule'
5. Fill in the rule as follows: Type = **MYSQL/Aurora**, Source = **My IP** on the dropdown, add a description if you want, then save it
6. Launch your database-accessing program, such as MySQL Workbench (https://www.mysql.com/products/workbench/), and set up the connection.



   a. Hostname = DB endpoint noted earlier
   b. Port = DB port noted earlier
   c. Username = DB username set during creation (**root**)
   d. Store in Vault or get prompted for DB password set during creation (**jglgoMtkp2kfm-5ab**)
7. Connect to the database and execute SQL commands as desired

Database Schema



Server Routes

Registering a New User

Outline of how to add a new user to the authentication system, such as when setting up a camera for a new site.

Expected form:

Headers={Content-Type='application/json'}
Data={username='uname', password='pw', permissions=1}

1. Select a username, password, and permission level for the user. In the case of a new site camera, the username will probably want to follow a naming convention. Permissions will determine what the user can do (0 = upload images only, 1 = view and download, 2 = deletions)
2. Download Postman to formulate requests (https://www.getpostman.com/downloads/)
3. Open Postman and start a new request
    a. Set to a **POST** request
    b. Enter the URL next to the request type ('**URL HERE/auth/register**')
    c. Params
        i. Key 1 = **username** | value = **your_username**
        ii. Key 2 = **password** | value = **your_password**
        iii. Key 3 = **permissions** | value = **your_permission_level**
    d. Headers
        i. Key 1 = **Content-Type** | value = **application/json**

Other Routes and Expected Forms

Route	Expected Form	Response Form	Purpose
POST /auth/login	Data={username, password}	Status Message Auth_token	Logs a user in, returns a valid auth token

GET /auth/status	Headers={Authorization}	Status Data={user_id, username, permissions, registered_on}	Provides info on the user with the auth token
POST /auth/logout	Headers={Authorization}	Status Message	Forcefully invalidates a token
POST /TrailPiServer/api/check_in	Data={site}	Status	Checks a camera site in for heartbeat
POST /TrailPiServer/api/image_transfer	Files={file, data = {site}}	Status	Uploads a new image to the server/database
GET /TrailPiServer/api/files	N/A	Filenames	List files in the S3 bucket
GET /TrailPiServer/api/download/<list:filenames>	Handled through route	Triggers download through: Header={Content-Disposition}	Retrieves file from S3 and starts download through browser
GET /TrailPiServer/api/images/<startDate>/<endDate>/<list:requested_sites>	Handled through route	Images, list of objects with {id, site, timestamp, url} fields	Return images between date interval from selected sites
POST /TrailPiServer/api/delete/<list:image_ids>	Headers={Authorization} Rest handled through route	Status Message	Delete images from S3 and database entries

Web Client

React Client File Structure

The layout for the client side is detailed below. The root directory for the React project is located in the **/trailpi-client** folder of the repository.

- **Build** - contains the compiled, packaged web-app that is deployed to the internet
- **Node_modules** - contains the dependencies for the project
- **Public** - application entrypoint (should not really need much modification)
- **Src**
    - **Components**
        - **Common** - constants for the map, route, and sidebar configuration
        - **Map** - leaflet initialization
        - **Pages** - react components for the application's main pages
        - **Pictures** - picture component
        - **Sidebar** - sidebar component for the map page
    - **Utils** - contains helper functions such as HTTP request handlers

Deploying React Client Changes

Should you wish to deploy new features to the web application (the graphical user interface initially hosted at http://trailpi-client.s3-website-us-west-2.amazonaws.com/), the following steps must be taken. If you simply wish to deploy changes to the existing website, perform steps 7 and 8 in the Package.json configuration section below.

Identity Access Management (IAM) Configuration

This section must only be completed once in order to instantiate the host container for the website.

1. Login to the AWS console at https://aws.amazon.com/console/.
2. Navigate to IAM by typing **IAM** into the search bar at the top of the service page.
3. Click **Users** in the left column and click **Add User** on the resulting tab.
4. Give the new user the name of your choosing, and click **Next: Permissions**.
5. Switch to the **Attach existing policies directly** tab and select **AdministratorAccess**.
6. Click **Next: Review** and then click **Create user.**
7. Click **Download .csv** and store the .csv somewhere safe on your local machine for later use. It is very important to maintain your credentials in a safe location, as you will only be able to access them once without a credentials reset.

S3 Configuration

1. Return to the main AWS console page and search for **S3** to navigate to the S3 service.
2. On the top toolbar, click **Create bucket**. Enter a unique bucket-name (our original name was trailpi-client, but this name will be unavailable since AWS uses this bucket name in the DNS).
3. Click **Next** twice.
4. On the **Set permissions** tab, select **Grant public read access to this bucket** and click next. Hit **Create bucket**.
5. Click on your newly created bucket from the S3 console. Click the **Properties** tab. Click the **Static website hosting** box.
6. Click **Use this bucket to host a website** and enter **index.html** for the **Index document**. Click save.
7. Click the **Permissions** tab and select **Bucket Policy**. In the editor window that appears, copy the following JSON snippet and then hit save.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPublicReadAccess",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::example-bucket/*"
      ]
    }
  ]
}
```

Package.json Configuration

1. On your local machine, open your terminal application and install the AWS CLI application (the instructions for various operating systems are located here: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html). For our application, we used macOS, and the command to install the CLI was done via homebrew with **brew install awscli**.
2. Once the CLI has been installed, type **aws configure**.
3. Follow through the prompts, entering your AWS credentials from the previous section where necessary. For our case, the **Default region name** is **us-west-2**. The **Default output format** can be skipped by hitting the enter key.

4. If you have not done so already, install Node.js and the Node Package Manager (Npm) with **brew install node** (other operating systems here: https://nodejs.org/en/).
5. Navigate to the **trailpi-client** directory from the project root folder.
6. Inside the **scripts** section of your **package.json**, add two new lines: **"build": "react-scripts build"** and **"deploy": "aws s3 sync build/ s3://<your-bucket-name>"**.
7. In your terminal in the **trailpi-client** directory, type **npm run build && npm run deploy**.
8. In your browser of choice, navigate to http://<your-bucket-name>.s3-website-us-west-2.amazonaws.com/ to see the deployed website. Done!

For a more detailed guide with more screenshots, visit https://medium.com/@serverlessguru/deploy-reactjs-app-with-s3-static-hosting-f640cb49d7e6

Deploying Server Changes

This section details adding new features to the Flask server, which is written in Python and uses pip (the python package manager) to manage dependencies and handle migration from your local machine to Elastic Beanstalk service on Amazon Web Services. This tutorial assumes you are on a Unix/Linux based machine and have Python 3.6+ (https://www.python.org/downloads/) installed on your machine.

If you simply need to update an existing virtual environment (re-deploy the server), skip to the **Updating an Elastic Beanstalk Environment** section below.

Configuring a Local Virtual Environment

1. On your local machine, run the command **pip install virtualenv**.
2. Run the command **virtualenv <environment-name>** using the environment name of your choosing.
3. Run the command **source <environment-name>/bin/activate** to activate the virtual environment on your machine.
4. Navigate to the **/server** directory from the root of the project repository.
5. Run the command **pip install -r requirements.txt** to install packages from the **requirements.txt** file we have provided.

Adding Additional Dependencies

1. Ensure that you have activated your virtual environment using **source <environment-name>/bin/activate**.
2. If you have added additional packages with the **pip install** command, make sure to use **pip freeze > requirements.txt** while inside the **/server** directory to track dependencies. You will need to do this to ensure that Elastic Beanstalk has the necessary packages when your server is hosted.

Initializing an Elastic Beanstalk Environment

1. To initialize an environment to be uploaded to the Elastic Beanstalk service, type **eb init -p python-3.6 <application-name> --region us-west-2** using the application name of your choice.
2. Create the environment using the command **eb create <environment-name>** using the environment name of your choice.

Updating an Elastic Beanstalk Environment

1. Launch your virtual environment in your terminal using the command **source <environment-name>/bin/activate**. If you have not yet created a virtual environment, refer to the **Configuring a Local Virtual Environment** section above.
2. In the **/server** directory, run the command **pip freeze > requirements.txt**.
3. To deploy the server to Elastic Beanstalk, run the command **eb deploy**. If you have not yet initialized an Elastic Beanstalk environment, refer to the **Initializing an Elastic Beanstalk Environment** section above.

Making Configuration Changes to your Live Server

1. Login to the AWS console at https://aws.amazon.com/console/.
2. Type **Elastic Beanstalk** into the management console search bar and open the Elastic Beanstalk service.
3. If you have deployed your server correctly with the above steps, you will see your application and environment correctly shown in the control panel.
4. Click the Configuration tab on the left side of the screen to see attributes about your server.
5. In particular, you may be interested in viewing/modifying environment variables for the server. On the software box, click **modify**.
6. Navigate to the bottom of the screen to the **Environment properties** section to see the environment variables and make changes.

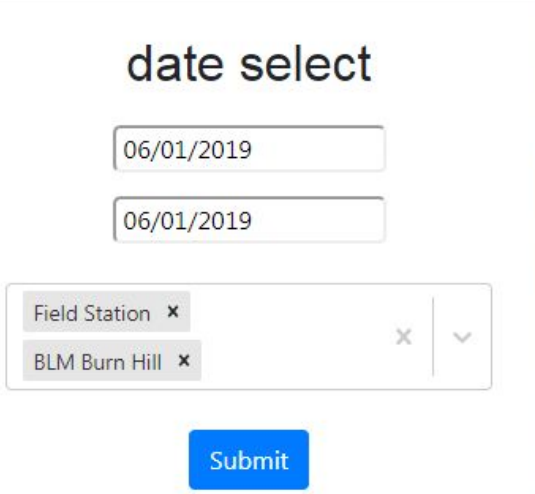
Accessing Images on Amazon S3

1. Login to the AWS console at https://aws.amazon.com/console/.
2. Type **S3** into the management console search bar and open the S3 service.
3. Click the name of your bucket to view your images. For this project, our bucket name was **trailpi-images** (showed below).

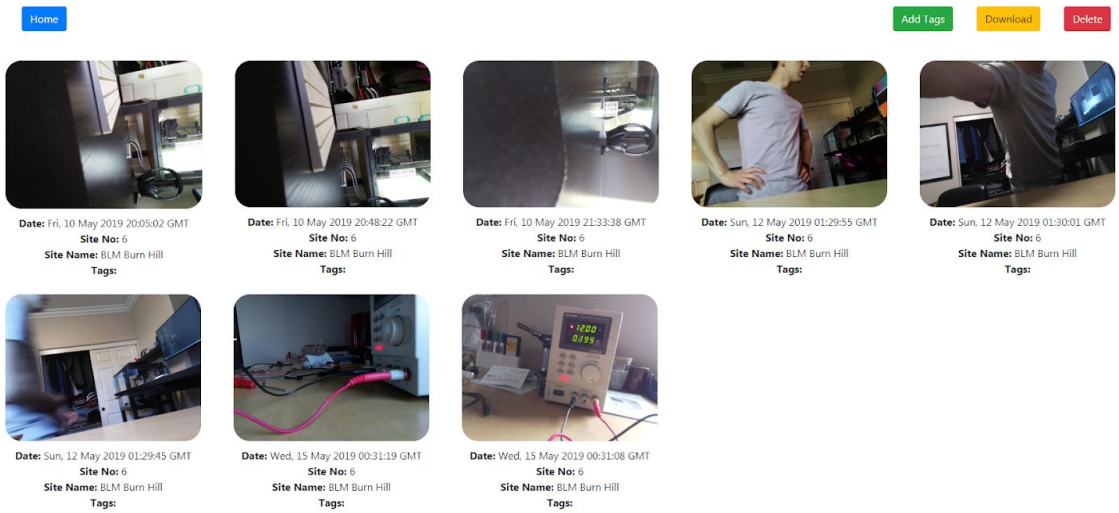
Using the Frontend GUI (React Client)

The web application allows the user to interact with the various sites remotely, giving the user an look at the state of each site. The web application also allows the user to tag images in order to help researchers study animal habits and location.

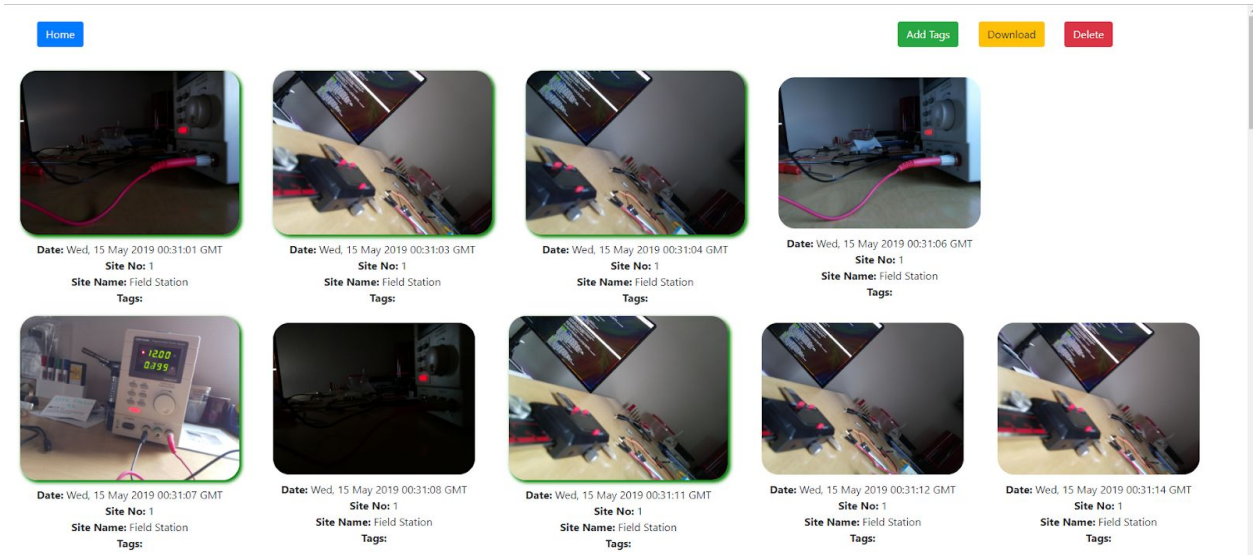
1. First, go to http://trailpi-client.s3-website-us-west-2.amazonaws.com/
2. On the top left section of the website, use the date selection tool, first date is the start date and second date is the end date, along with the site dropdown tool to pull up images between the selected dates for the given site (note: you can select as many sites as necessary) hit **Submit** once dates and sites are selected :



3. The image selection screen will appear with the queried images, each image will contain the **Date, Site No., Site Name** and **Tags** that the user has saved:



4. Images can be selected simply by clicking on them. An enlarged image with a green border indicates that the image is selected as shown below:



5. To add tags, select the **Add Tags** button. Ensure that you are only selecting one image at a time to perform this operation(Note: this operation requires a valid username and password).
6. The **Download** button will initiate a zip file download onto your local machine.
7. The **Delete** button will permanently remove the image from the S3 Bucket, the MySQL database, and the current view.
8. The **Home** button will return user to the start page in order to select other dates and sites.

Appendix

This section details some extra information that gives some insight into how this project was developed.

Technology Survey

This section details why certain technologies were chosen over other ones.

Camera Module Technology

For the creation of the camera module, there are a few different options with regard to price and image quality. The camera module encompasses everything that is needed to install at a site which will take a picture when needed and send it over the network to the central server. For the first 2 options, one or two PIR sensors are needed in order to trigger image capture during the night and an IR flashlight must be used to illuminate the dark landscape. A custom 3D-printed enclosure will be designed for the camera module because it is the lowest-cost option and most fit for the need. The full enclosure need not be completely 3D printed, a simple box may be used as the enclosure with 3D printed mounts for the inside.

Option 1: 2 RPi Zero W's with separate Day & Night cameras

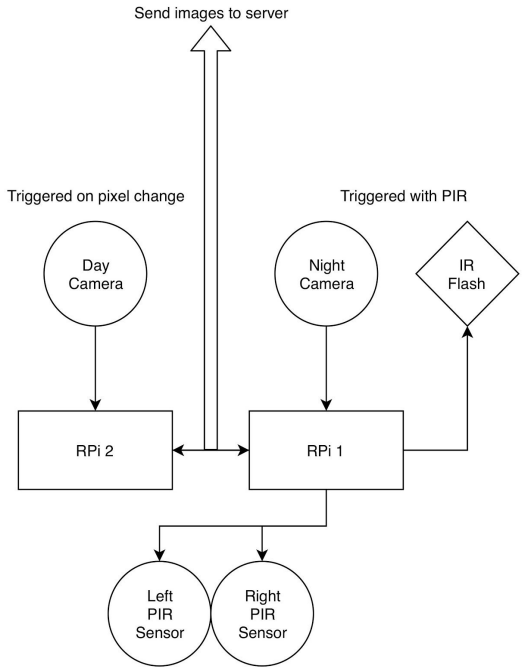
Pros

- A RPi Zero W is inexpensive (only $10) and we are not doing too much at each site
- Official RPi cameras are of high quality (8 MP)
- Using a separate camera for day and night allows the strengths of each camera to be used
- Using a special night camera may lead to better pictures at night with a custom IR flash

Cons

- 2 RPi's and 2 cameras may seem wasteful when other 2-in-1 solutions exist
- Duplicate hardware increases the chance of failure
- RPi Zero is much weaker than a RPi Model 3 B+
- Custom interface required for both devices to talk to each other

Architecture



Option 2: Single RPi Model 3 B+ or Zero W with integrated Day/Night camera

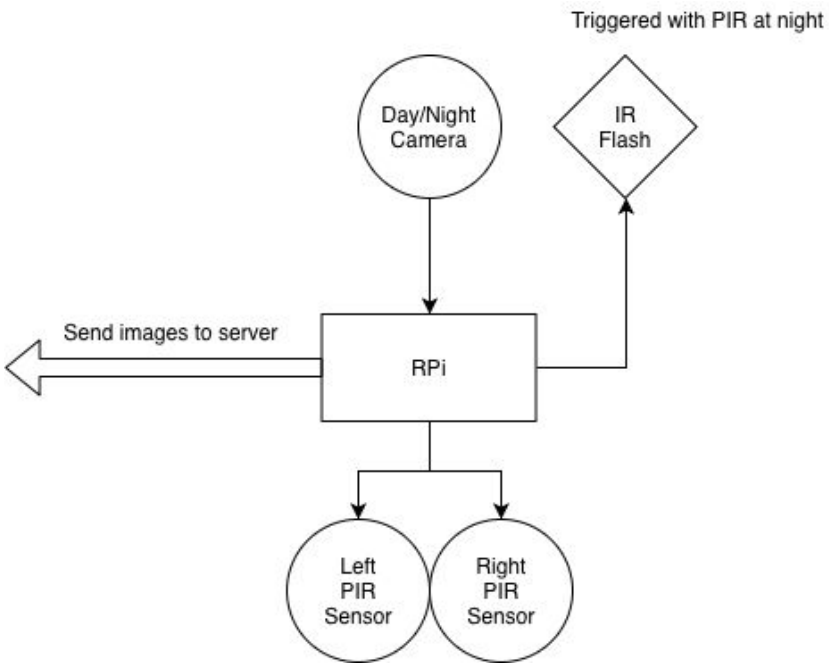
Pros

- Less components are in play so less chance of failure
- Saved money on one camera and one RPi

Cons

- Camera quality is much lower for both day and night (non-official RPi cameras exist but quality is 5MP compared to 8MP)
- Integrated day/night camera have always glowing IR leds which may distract or attract animals

Architecture



Option 3: RPi Model 3 B+ or Zero W with Toshiba Air SD card in current trail cameras

Pros

- Least components in use
- Simplest system to implement
- Cheapest solution
- Uses existing trail cameras

Cons

- Current trail cameras are old and have bad image quality
- Toshiba Air SD card is known to be unreliable
- Still requires a RPi for network communication

Camera Module Results

Out of all 3 options, we believe the first option provides the best results because of the best image quality. Image quality is extremely important when the goal is to take pictures of wildlife. The complexity of the first option is not that high because the number of components being used is still minimal.

Using a Raspberry Pi Zero W is much more cost effective than using a Model 3 B+ because a lot of computational power is not needed. The RPi's responsibility is taking a picture, controlling a PIR sensor or two, turning on an IR floodlight, and sending the picture over the network.

Client-Side Technology

React

Pros

- Supports modern HTML/CSS/Javascript features with rigid component-based architecture
- Gives us access to rich library of features available via javascript frameworks in the Node Package Manager
- Allows us to have a persistent live web application which is easily shareable to other researchers and people with an interest in the study

Cons

- Requires hosting via either our own server or a third-party service

Server-Side Technology

Flask: Server

Pros

- Light-weight and built for Python
- Extendable for more features
- Simple to use

Cons

- Limited in features and documentation

MySQL: Database

Pros

- Multi-platform support
- Efficient SELECT queries
- A simple DBMS to set up
- Easy to test on an individual's development computer

Cons

- Unable to retrieve more than 4GB of data in one query

S3: Storage

Pros

- Well-documented and easy to integrate with other AWS services
- Stable and durable
- Not too expensive

Cons

- Requires setup of a 'bucket' for file storage
- Complex pricing system

Machine Learning Technology

Keras/TensorFlow: Image processing

Pros

- Enable detection of false positive images
- Reduce the amount of images being sent over the network

Cons

- Requires a lot of labeled training data for higher accuracy in animal detection
- Requires a powerful computer to run the CNN

Real-World Constraints

Camera Module

- Price: goal is around $100 per module
- Reproducibility: goal is to cover 40 sites with this module
- Weatherproof: camera must work in the rain and sun
- Reliability: must work with 100% uptime to capture every animal
- Efficient: must use little electricity due to running on solar power and batteries

Database

- Storage space available
- Bandwidth and ability to transfer files from the reserve sites to the database

Server/Storage

- Scaling cost with usage
- Requires sufficient bandwidth to transfer images from camera to server

Image Processing

- Model created should be able to mark false positive images quickly to avoid bottleneck
- Should be easy to improve accuracy if more data is provided

Web Application

- Flexibility: application should be able to handle slow and dropped connections to the server
- Reliability: must be able to avoid crashing as much possible or have an easy recovery mechanism

User Stories and Requirements

1. As a wildlife researcher, I want to be able to capture high quality pictures of animals during the day/night to be able to better understand their behavior.
2. As a wildlife researcher, I want to capture images of wildlife both during the day & night without downtime so that I did not miss any animals walking by.
3. As a busy user, I want to be able to collect the images from the cameras using the available network so that I did not have to waste time collecting images manually.
4. As a user, I want to store all of the images from each site in a central location so that I had easy access to them wherever I am.
5. As the camera module assemblyman, I want an easy way to setup the Raspberry Pi with the correct configuration for each site so that I did not have to debug any of the code running on it.
6. As the client, I want the whole system to be well-documented so that after the project I could manage it on my own.
7. As the camera module assemblyman, I want an easy to use guide to put together the camera module so that I could quickly assembly one.
8. As an application user, I want to be able to select a date range I am interested in downloading from the central server so that I can view a small subset of all the data available.
9. As an application user, I want to be able to select one, several or all cameras to view their pictures so that I can find trends in wildlife activity.
10. As a researcher, I want to be able to be able to easily share this client-side interface with other researchers and colleagues so they may participate and gain interest in our study.
11. As an application user, I want to have the ability to download the images I selected as a zip file so that I can upload those images to a website such as zooniverse.com.
12. As a wildlife researcher, I want to use existing technology to filter out false positive images that are taken so that I did not have to waste time dealing with many pictures without animals.
13. As a wildlife researcher, I want to use crowd-sourcing options for labeling the animals in the pictures taken at each site so that I can better track these animals and their count.
14. As an assistant for the wildlife researcher, I want an easy to use guide when tasked out to service, or inspect the cameras set-up so that I spend more time tracking, observing and getting valuable data for the researcher.
15. As an application user, I want to be able to see an overview of various attributes about the camera sites (how many pictures taken in the last 24 hours, etc.)

Acceptance Test

In order to consider the TrailPi project acceptable, the following conditions must be satisfied.

- Design and implementation of the camera module including:
    - Bill of Materials
    - Assembly/user guide
    - Working prototype
- Camera module must support:
    - Taking pictures during the day & night based on input from a PIR sensor
    - Saving the pictures to disk
    - Uploading the pictures to the web server
    - Turning on/off the IR flash during the night to take a picture
- Camera module must have an enclosure design that safely houses all of the components
- Image processing model deployed that filters out false positives with an accuracy of > 70%
- Server must be able to receive and store pictures, along with maintaining a database of pictures and metadata such as sites and tags associated with each picture
- Server data should be stable and durable, to avoid loss of data
- Web application should enable client to safely login to delete images
- Client is able to select multiple sites in order to extract the images taken
- Web application can download/edit/delete images in bulk