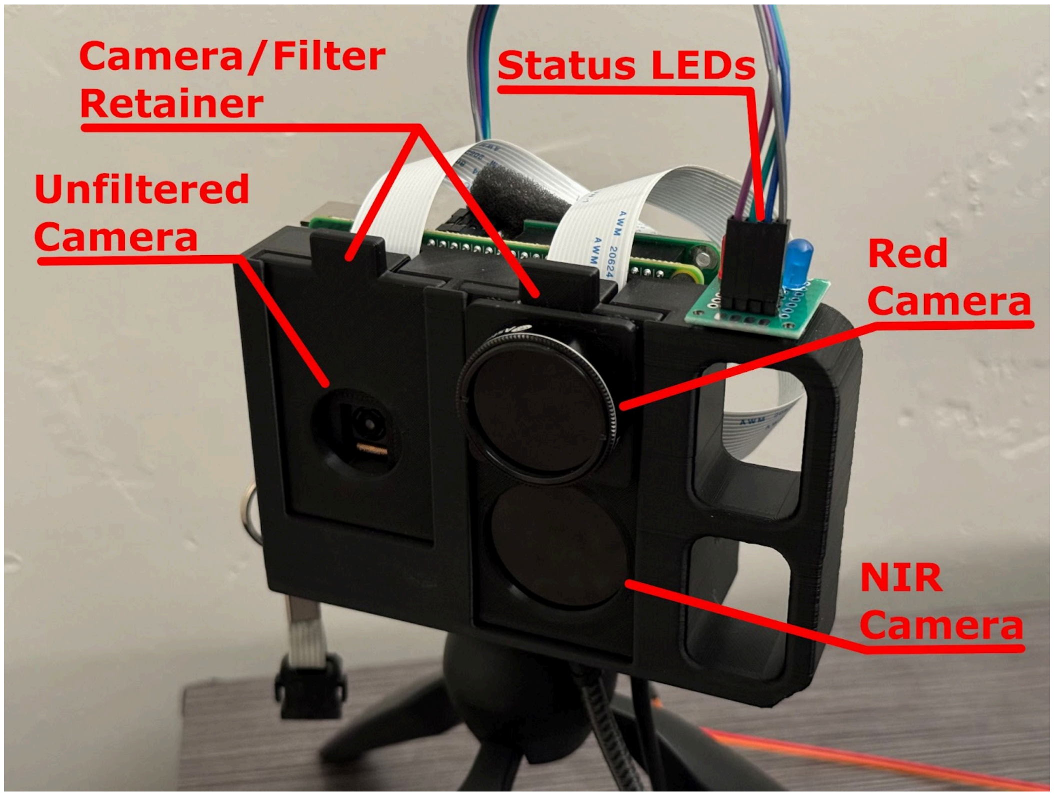
Raspberry Pi 4B Three-Camera System for Normalized Difference Vegetation Index (NDVI) Imaging and Machine Learning training.



All CAD files and code for this camera will be available in the project Github, found here: https://github.com/goviedo-ok/RGB-to-NDVI

If you haven't been briefed on the purpose of this project, basically it is to create accurate NDVI images with a single NoIR OSC camera. This is very difficult to do with purely mathematical approaches, so Machine Learning is used to simplify the process.

Setting up the Pi

Note: The code for this camera was prepared on a **Raspberry Pi 4B 4GB** with **Raspberry Pi OS Debian version 11 (Bullseye) 32 Bit.** I highly recommend using the same image. Download the image HERE, then use the imager to flash the OS to a microSD card (8GB minimum size, 32GB recommended).

Also, to simplify the next few steps, please name your user **picam**

Download and install the **libcamera library**. If you need information on how to do this, please reference the multi-camera adapter board documentation found HERE, also reference the libcamera documentation found HERE.

Now edit the config.txt file. Type **sudo nano /boot/config.txt** in the console.

Under line **[all]** type **dtoverlay=imx519**

Also, under line **# uncomment if hdmi display is not detected and composite is being output**, uncomment or paste **hdmi_force_hotpug=1**

This last part ensures the camera works when no display is connected.

Now make the camera run on startup. To do this, type **sudo nano /lib/systemd/system/Camera.service** in the console

No file should exist, so allow it to create a new one. Now in the new file, paste the following:

```
[Unit]
Description-PiCounter
After=network.target

[Service]
ExecStart=/usr/bin/python3 /home/picam/Desktop/Camera.py
Restart=always
User=picam

[Install]
WantedBy=multi-user.target
```

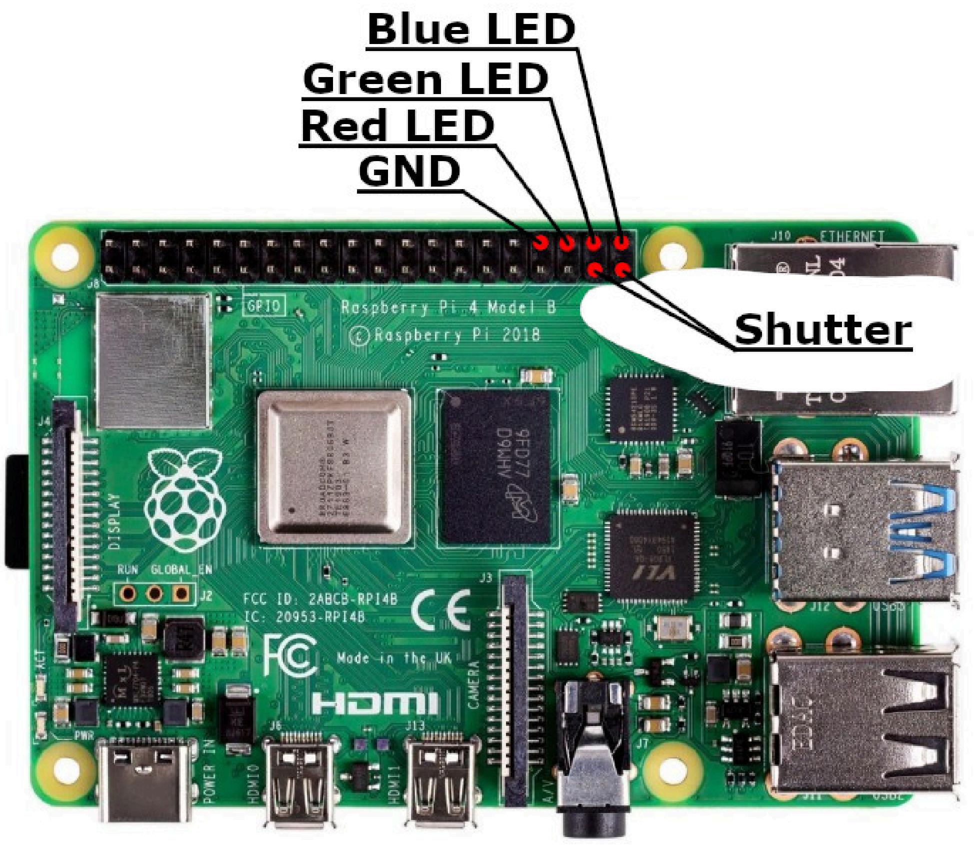
Now paste the following scripts from the Github to **/home/picam/desktop**

Reset.py
Camera.py
NDVIAutomatedAlign.py
FolderCombine.py

Due to the unreliability of the multi-camera adapter, the reset script is necessary to restart Camera.py when an exception occurs. I tried several methods, but this worked best.

Hardware setup

Plug the multi-camera adapter into the Pi according to the Arducam documentation. Also plug the RGB LED board and shutter release into the Pi according to the following diagram.



Now assemble the camera by snapping the Pi into the standoffs on the back of the body. Carefully press camera modules and filters into place and slide the filter drawer down to hold everything together. If done correctly there should be no play or loose parts. Also thread the additional filters in front of the Red camera. You may also need to add a neutral density filter in front of the unfiltered camera. I recommend a ND13 (~3 stop), which is what is used on the red camera. The STL for a thread adapter can be found in the Github. 3D print it and glue it on the filter drawer to allow the use of threaded 1.25” filters. Due to the nature of printing fine threads, multiple attempts may be necessary with varying horizontal expansion values in the slicer.

You may notice that the unfiltered camera ribbon cable will be rubbing against the adapter board. I recommend placing a small piece of foam between to prevent damage over time.

Now everything should be working. If the code doesn't run properly, see if you're missing a library. If you are, install it.

Note that for the camera, NDVI, or foldercombine scripts to work properly, a save location must be defined. I saved all images to a USB drive named **Samsung256GB**. If you use a drive with a different name, change all instances of **Samsung256GB** to your drive name. Additionally, in the root of your drive, create several folders and name them as shown below:

Buffer
Captures
Combined
NDVIInput
NDVIOutput
Red-NIR
Training

These are the save locations of the camera, NDVI, and folder combine scripts. Buffer only holds files when the camera script is running. Please inspect the code for more information.

I'd like to mention that the NDVI script can probably run a *lot* faster by using some clever Numpy matrix math (around lines 225-296). I didn't use this approach due to just learning Python and having very little time to make the camera work. But please feel free to improve the code as you see necessary.

Also, there are several stretch functions in the NDVIAutomatedAlign script. contrast_stretch() was found to be the most effective, however, I left the others in there in case you want to experiment.

Also also, the Pi may get hot while running the NDVIAutomatedAlign script. I recommend putting a small heat sink on the processor and placing a fan nearby to keep it cool.