Raspberry Pi Zero W RGB Image Processor

This document is explaining how to make simple Image Processor with a Raspberry Pi Zero W. The image processing is just to determine if an object is red, green or blue. It is good starting point to learn how images can be processed with a Raspberry Pi.

Raspberry Pi RGB Image Items

Following three items are the only parts necessary to make the RGB Image Processor:

1. **Raspberry Pi Zero W**

2. **Camera Raspberry Pi v2 8MP**

3. **RGB Items**

Raspberry Pi Zero Setup

Operating Systems

The first thing to do to setup and install the OS on the Raspberry Pi Zero W.
The easiest way is to use the Raspberry Pi imager if there was not a pre-installed OS on a microSD when the Raspberry Pi Zero W was purchased. The instruction on how to install the OS and to download the imager can be found on following link:
https://www.raspberrypi.org/software/
Note that the maximum recommended microSD Card size for Raspberry Pi models is 32GB.

Remote Access

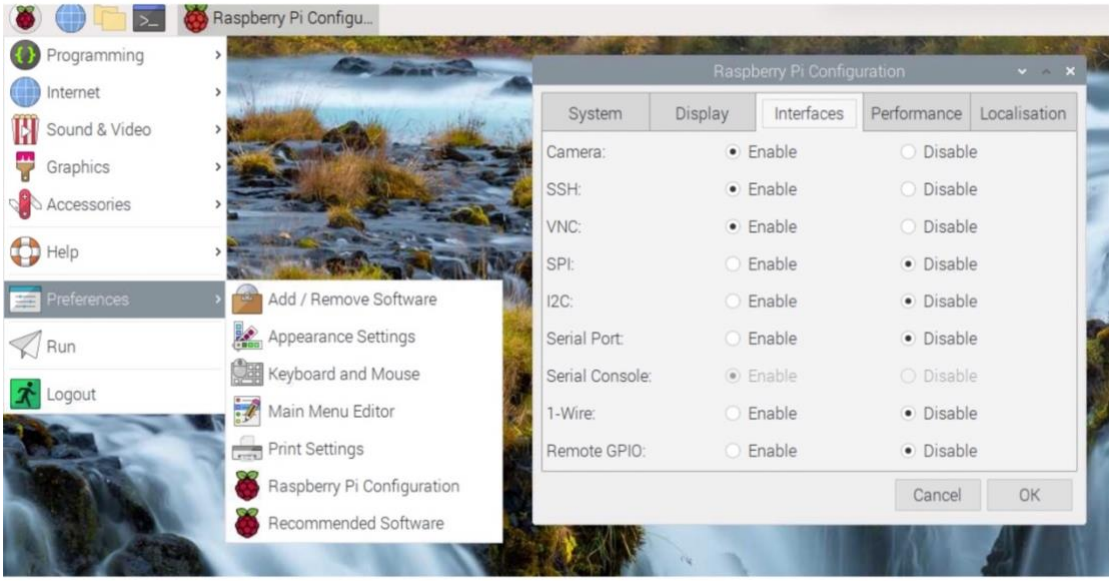
The second thing is to install the VNC on the your Raspberry Pi Zero W to get remote access.
Open the Terminal on your Raspberry Pi and type following commands

sudo apt-get update
sudo apt-get upgrade
sudo apt-get install realvnc-vnc-viewer

The third thing is to install the VNC on the your Windows/MacOS/Linux to be able to control your Raspberry Pi unit remotely.
In the following link you will find the VNC download for you OS system:
https://www.realvnc.com/en/connect/download/vnc/macos/
Following video is also showing on hoe to make remote access:
https://www.youtube.com/watch?v=JZ1pdVVTMrw

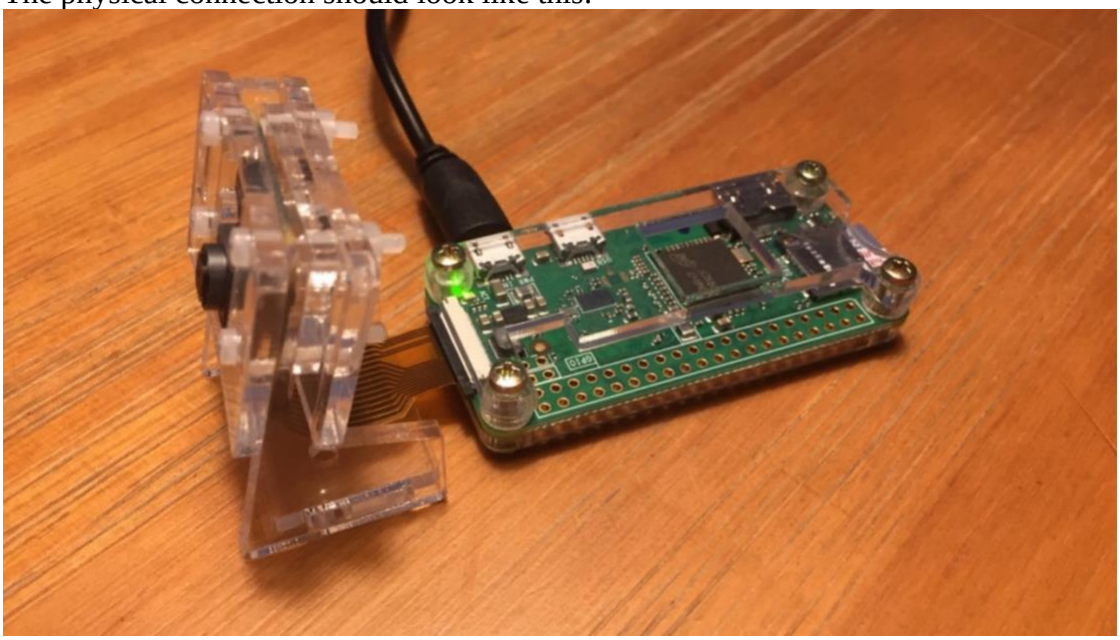
Raspberry Pi configuration

To be able to work remotely and to use the camera it is important that the Camera and VNC Interfaces in the Raspberry Pi configuration setting are enabled as follows:



Raspberry Pi Camera wiring and camera enabling

Following link explains how to connect the camera:
https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/2
The physical connection should look like this:



In this configuration a transparent case was added for the camera but this is not necessary for the RGB Image processor to work.

Raspberry Pi Camera Python Libraires

Before the python coding it is important that following python libraires are installed
Use the terminal of the Raspberry IP. Before installing the libraries do an update by following command: sudo apt update

Numpy (For working with image arrays)

sudo apt install python3-numpy

Matplotlib (For image plotting)

sudo apt install python3-matplotlib

Opencv (For image processing)

sudo apt install python3-opencv

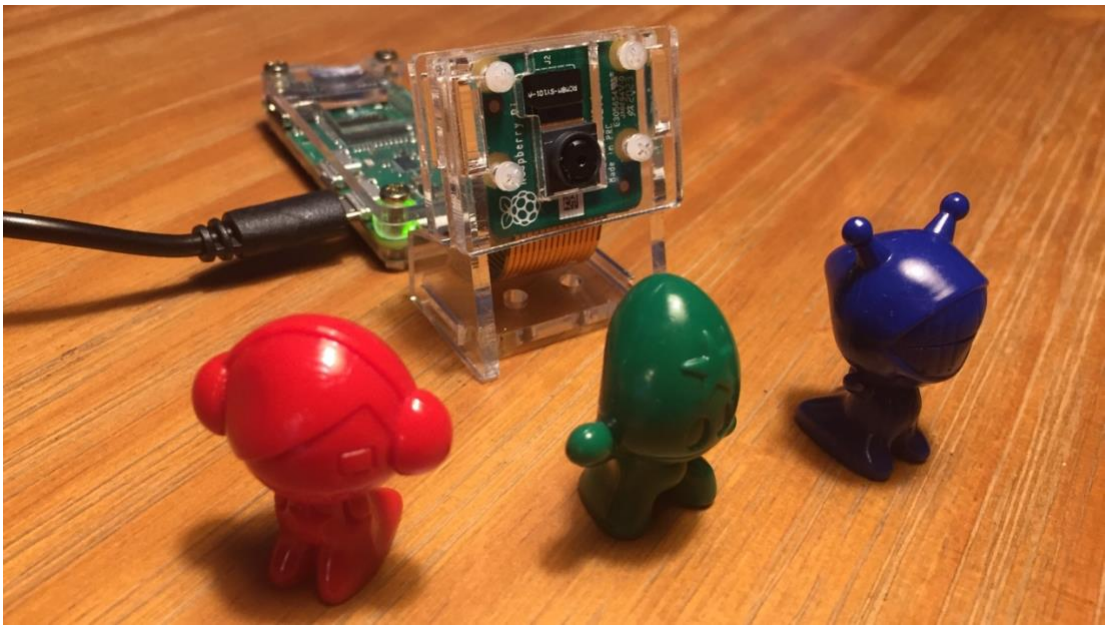
Imagemagick (For displaying images)

sudo apt install imagemagick

The Raspberry Pi Zero testbed for Image Processing

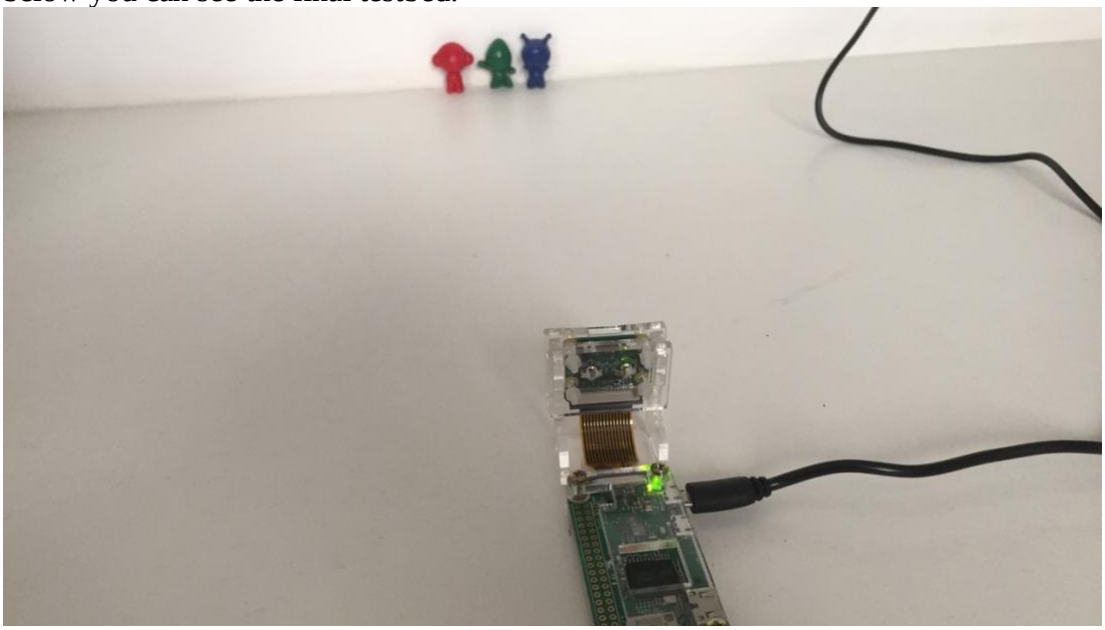
The RBG Test Objects

These are the RBG objects and the Raspberry Pi Zero W that was used in the test environment.



The testbed setup

The Raspberry Pi v2. Camera needs more or less 50cm to have a focus therefore in the figure below you can see the final testbed.



To get closer to a small object the “Camera.zoom” can then be used.

Raspberry Pi Camera Python Codes

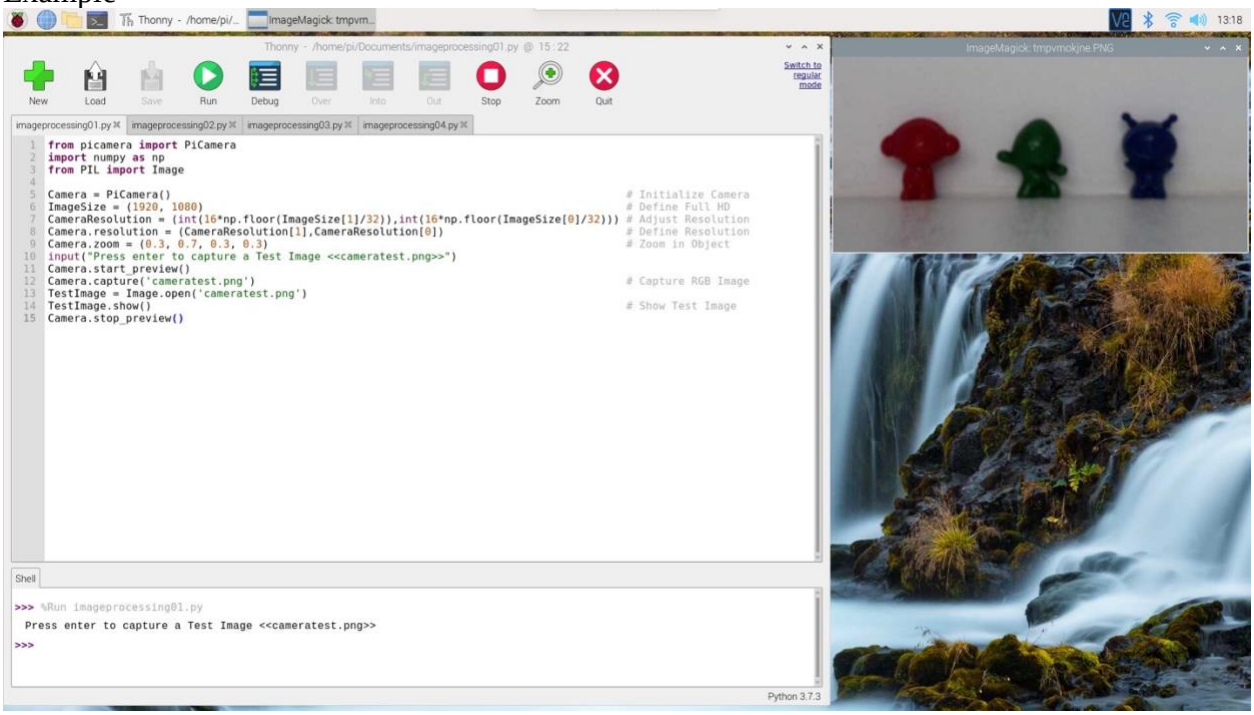
imageprocessing01.py

This python code is to test the camera.
It will take a picture and and save it as a png file and also show it on the screen.

```
from picamera import PiCamera
import numpy as np
from PIL import Image

Camera = PiCamera()                                                    # Initialize Camera
ImageSize = (1920, 1080)                                               # Define Full HD
CameraResolution = (int(16*np.floor(ImageSize[1]/32)),int(16*np.floor(ImageSize[0]/32)))   # Adjust Resolution
Camera.resolution = (CameraResolution[1],CameraResolution[0])          # Define Resolution
Camera.zoom = (0.3, 0.7, 0.3, 0.3)                                     # Zoom in Object
input("Press enter to capture a Test Image <<cameratest.png>>")
Camera.start_preview()
Camera.capture('cameratest.png')                                       # Capture RGB Image
TestImage = Image.open('cameratest.png')
TestImage.show()                                                       # Show Test Image
Camera.stop_preview()
```

Example



imageprocessing02.py

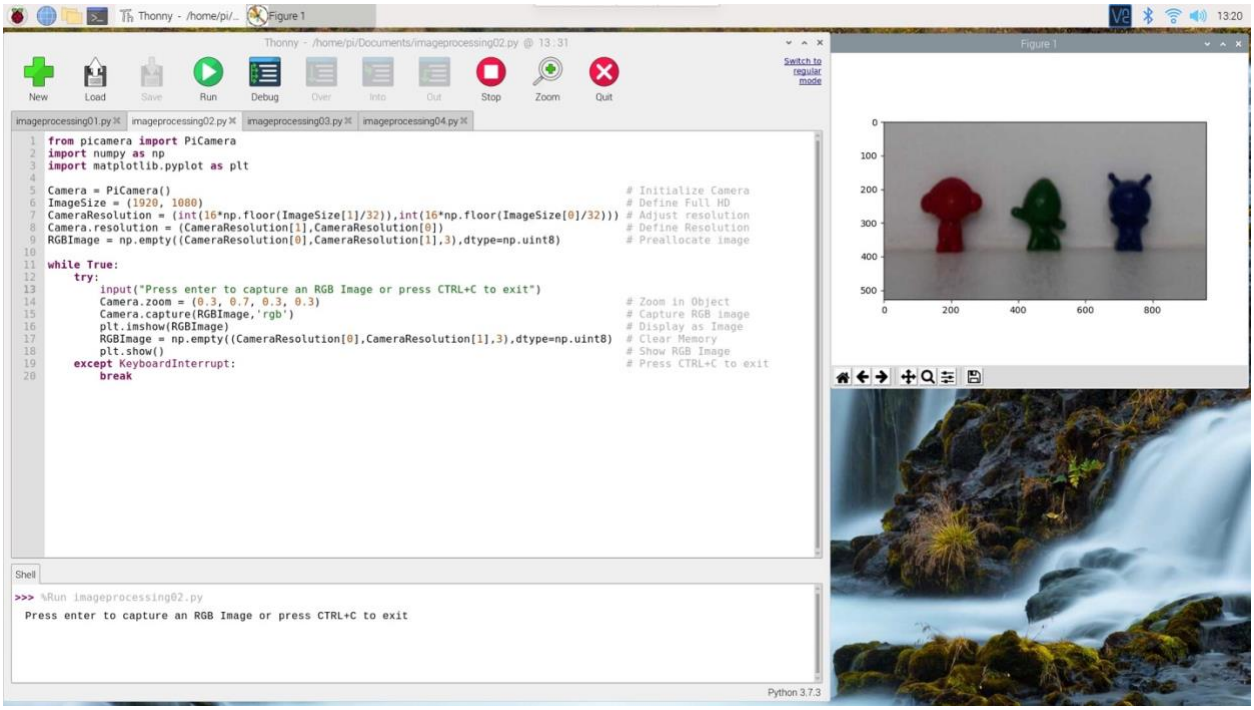
This pythoncode code show a simplest method for plotting the images is using matplotlib's 'imshow' function, which plots all three RGB colors in a traditional format seen by the human eye.

```
from picamera import PiCamera
import numpy as np
import matplotlib.pyplot as plt

Camera = PiCamera()                                                          # Initialize Camera
ImageSize = (1920, 1080)                                                     # Define Full HD
CameraResolution = (int(16*np.floor(ImageSize[1]/32)),int(16*np.floor(ImageSize[0]/32)))   # Adjust resolution
Camera.resolution = (CameraResolution[1],CameraResolution[0])                # Define Resolution
RGBImage = np.empty((CameraResolution[0],CameraResolution[1],3),dtype=np.uint8)  # Preallocate image

while True:
    try:
        input("Press enter to capture an RGB Image or press CTRL+C to exit")
        Camera.zoom = (0.3, 0.7, 0.3, 0.3)                                   # Zoom in Object
        Camera.capture(RGBImage,'rgb')                                       # Capture RGB image
        plt.imshow(RGBImage)                                                 # Display as Image
        RGBImage = np.empty((CameraResolution[0],CameraResolution[1],3),dtype=np.uint8)  # Clear Memory
        plt.show()                                                           # Show RGB Image
    except KeyboardInterrupt:                                                # Press CTRL+C to exit
        break
```

The plot should look something like the figure below, where the image's origin is the top left corner of the plot. We will be using this as the general layout for analyzing the images.

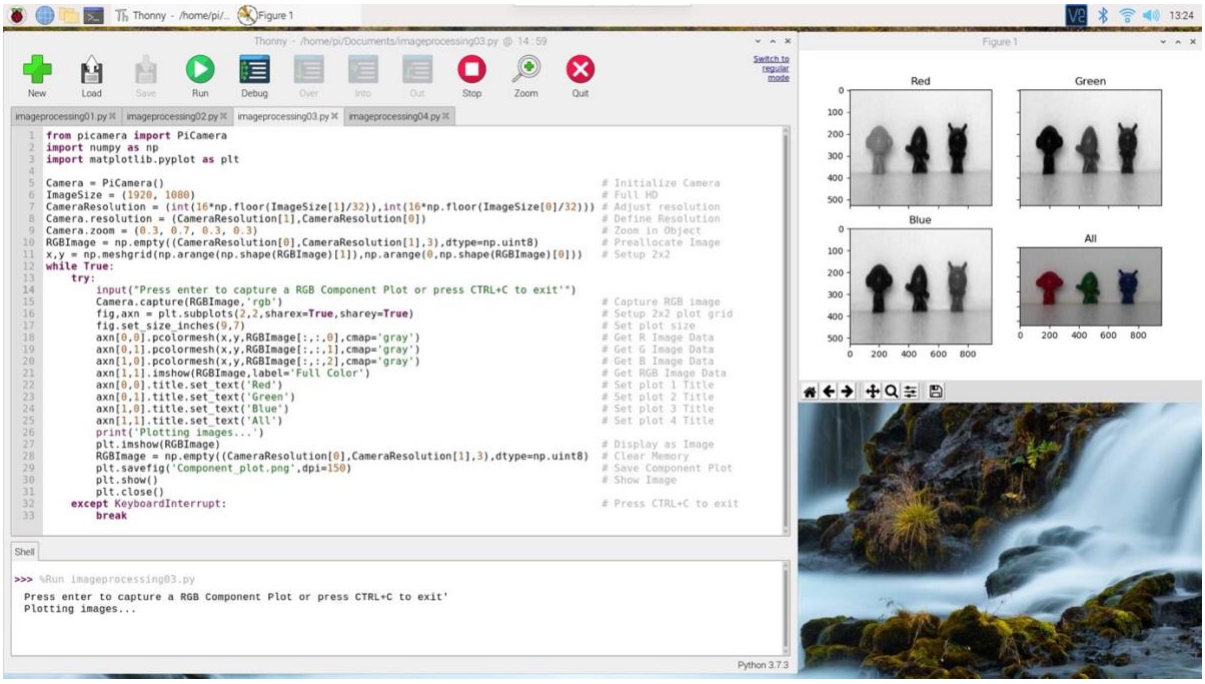


imageprocessing03.py

This code decompose the image into its three color components: red, green, and blue.

```
from picamera import PiCamera
import numpy as np
import matplotlib.pyplot as plt

Camera = PiCamera()                                                 # Initialize Camera
ImageSize = (1920, 1080)                                            # Full HD
CameraResolution = (int(16*np.floor(ImageSize[1]/32)),int(16*np.floor(ImageSize[0]/32)))  # Adjust resolution
Camera.resolution = (CameraResolution[1],CameraResolution[0])       # Define Resolution
Camera.zoom = (0.3, 0.7, 0.3, 0.3)                                  # Zoom in Object
RGBImage = np.empty((CameraResolution[0],CameraResolution[1],3),dtype=np.uint8)  # Preallocate Image
x,y = np.meshgrid(np.arange(np.shape(RGBImage)[1]),np.arange(0,np.shape(RGBImage)[0]))  # Setup 2x2
while True:
    try:
    input("Press enter to capture a RGB Component Plot or press CTRL+C to exit'")
    Camera.capture(RGBImage,'rgb')                                  # Capture RGB image
    fig,axn = plt.subplots(2,2,sharex=True,sharey='True)            # Setup 2x2 plot grid
    fig.set_size_inches(9,7)                                        # Set plot size
    axn[0,0].pcolormesh(x,y,RGBImage[:,:,0],cmap='gray')            # Get R Image Data
    axn[0,1].pcolormesh(x,y,RGBImage[:,:,1],cmap='gray')            # Get G Image Data
    axn[1,0].pcolormesh(x,y,RGBImage[:,:,2],cmap='gray')            # Get B Image Data
    axn[1,1].imshow(RGBImage,label='Full Color')                    # Get RGB Image Data
    axn[0,0].title.set_text('Red')                                  # Set plot 1 Title
    axn[0,1].title.set_text('Green')                                # Set plot 2 Title
    axn[1,0].title.set_text('Blue')                                 # Set plot 3 Title
    axn[1,1].title.set_text('All')                                  # Set plot 4 Title
    print('Plotting images...')
    plt.imshow(RGBImage)                                            # Display as Image
    RGBImage = np.empty((CameraResolution[0],CameraResolution[1],3),dtype=np.uint8)  # Clear Memory
    plt.savefig('Component_plot.png',dpi=150)                       # Save Component Plot
    plt.show()                                                      # Show Image
    plt.close()
  except KeyboardInterrupt:                                         # Press CTRL+C to exit
    break
```



Here you can see in the first three pictures the image decomposed into its three color components: red, green, and blue

imageprocessing04.py

This is the final code is a simple introduction to image processing.
It will analyze the color content in an image. In this case it is used normally distributed mean and standard deviation.

```
import time
from picamera import PiCamera
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv

Camera = PiCamera()                                                                    # Initialize Camera
ImageSize = (1920, 1080)                                                               # Full HD
CameraResolution = (int(16*np.floor(ImageSize[1]/32)),int(16*np.floor(ImageSize[0]/32)))  # Adjust resolution
Camera.resolution = (CameraResolution[1],CameraResolution[0])                          # Define Resolution
Camera.zoom = (0.35, 0.75, 0.2, 0.2)                                                   # Object Zoom
RGBImageRaw = np.empty((CameraResolution[0],CameraResolution[1],3),dtype=np.uint8)     # Preallocate Image
RGBImage = np.empty((CameraResolution[0],CameraResolution[1],3),dtype=np.uint8)        # Preallocate Image
Camera.framerate = 30                                                  # Fix Framerate
time.sleep(2)                                                          # Settle Camera
Camera.iso = 200                                                       # Fix ISO value
Camera.shutter_speed = Camera.exposure_speed                           # Fix shutter speed
Camera.exposure_mode = 'off'                                           # Fix White Balance
gain_set = Camera.awb_gains                                            # AWB Query
Camera.awb_mode = 'off'                                                # Fix exposure gains
Camera.awb_gains = gain_set                                            # Set AWB
Noise = np.empty((CameraResolution[0],CameraResolution[1],3),dtype=np.uint8)           # Preallocate Image
x,y = np.meshgrid(np.arange(np.shape(RGBImage)[1]),np.arange(0,np.shape(RGBImage)[0]))  # Create Rectangular grid
rgb_text = ['Red','Green','Blue']                                      # Array for naming color
input("Press enter to capture background noise image (Remove Background Noise)")Camera.capture(Noise,'rgb')          # Capture Background Noise
RGBImage = cv.subtract(RGBImageRaw, Noise)                                             # Substract Noise
while True:
    try:
        print('==========================')
        input("Press enter to capture an RGB Image or press CTRL+C to exit")
        Camera.capture(RGBImage,'rgb')                                 # Capture RGB image
        mean = []                                                              # Define Mean Vector
        std = []                                                               # Define STDev Vector
        for c in range(0,3):                                           # Calculate Mean and STDev
          mean.append(np.mean(RGBImage[:,:,c]-np.mean(RGBImage)))      # Append Mean value
          std.append(np.std(RGBImage[:,:,c]-np.mean(RGBImage)))        # Append STDev valie
          print(rgb_text[c]+'---mean: {0:2.2f}, stdev: {1:2.2f}'.format(mean[c],std[c]))
        print('The Object is: {}'.format(rgb_text[np.argmax(mean)]))
        plt.imshow(RGBImage)                                           # Plot image
        RGBImage = np.empty((CameraResolution[0],CameraResolution[1],3),dtype=np.uint8)  # Clear Memory
        plt.show()                                                             # Show the image
    except KeyboardInterrupt:                                          # Press CTRL+C to exit
        break
```

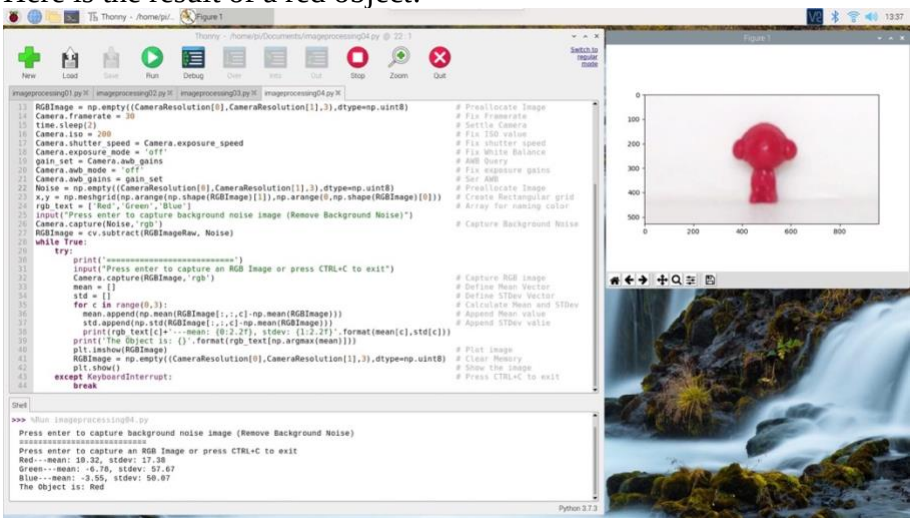
As you can see in code the we have fixed the shutter speed and the white balance, This is done for not letting the camera change these parameters after each taken image. We need also to subtract the background before we do any RGB test. This to avoid any background color to interfere.

Raspberry Pi Camera RGB Image Processing Test

To see if the RGB Image Processing is working do following:
Run the the imageprocessing04.py without any object to until the program reach done the “Press enter to capture background noise image (Remove Background Noise" and reached the “Press enter to capture an RGB Image or press CTRL+C to exit”. Before pressing enter, place an object with red, green or blue color. The result should match the color of the object.

Red Objects

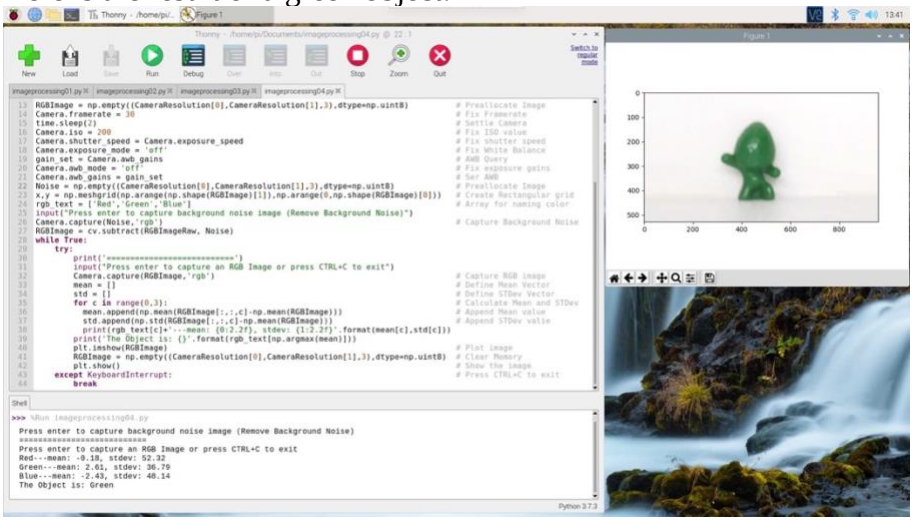
Here is the result of a red object:



The Red has a highest mean (10,32) and therefore the object was interpreted as red.

Green Objects

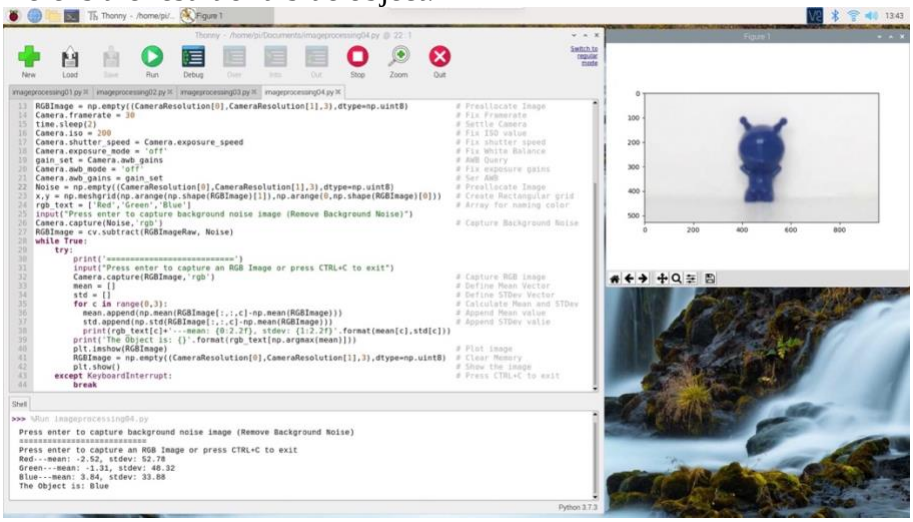
Here is the result of a green object:



The Green has a highest mean (2,61) and therefore the object was interpreted as green.

Blue Objects

Here is the result of a blue object:



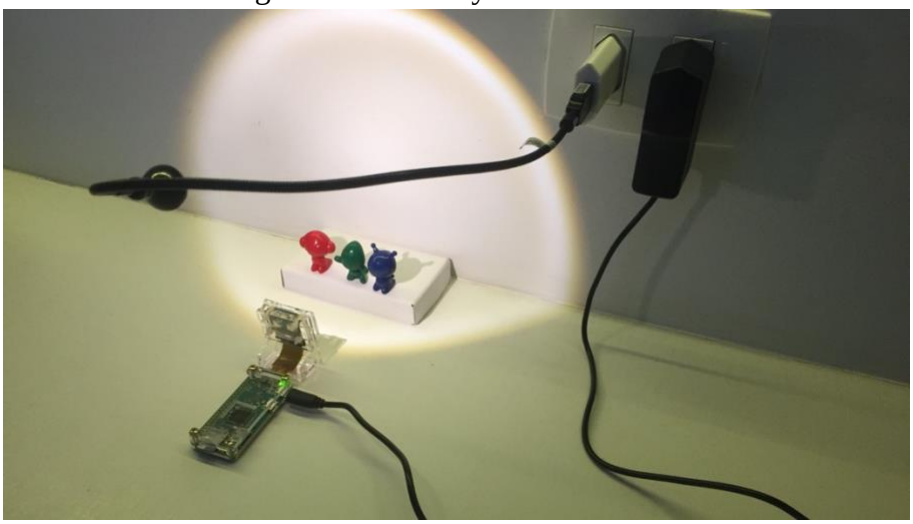
The Blue has a highest mean (3,04) and therefore the object was interpreted as blue.

Final Words and caution

If you have followed all the steps and setups the RGB Image Processing should work just fine.
Try with different object to see if it is work.
One thing important is that with this simple configuration there must be white light.
If you use yellow light you probable will get following result:

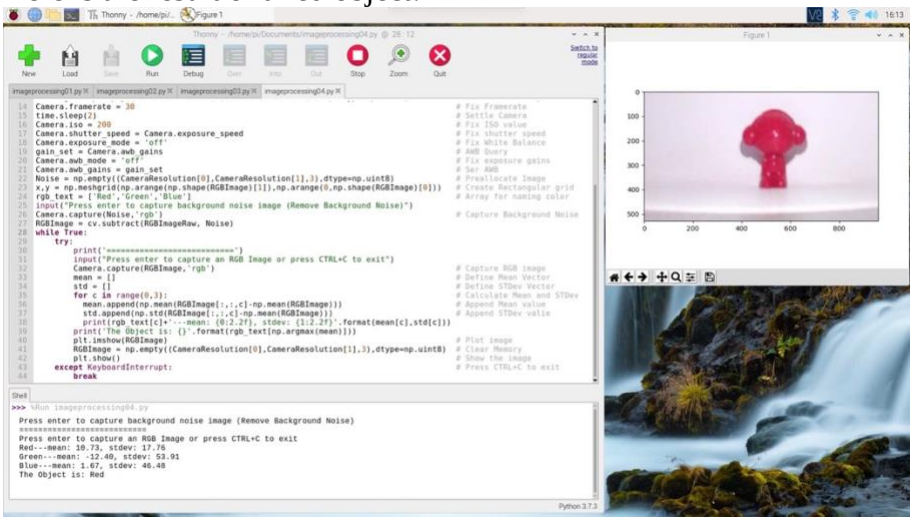
The testbed setup with yellow light

In this testbed the light came from a yellow led.



Red Objects

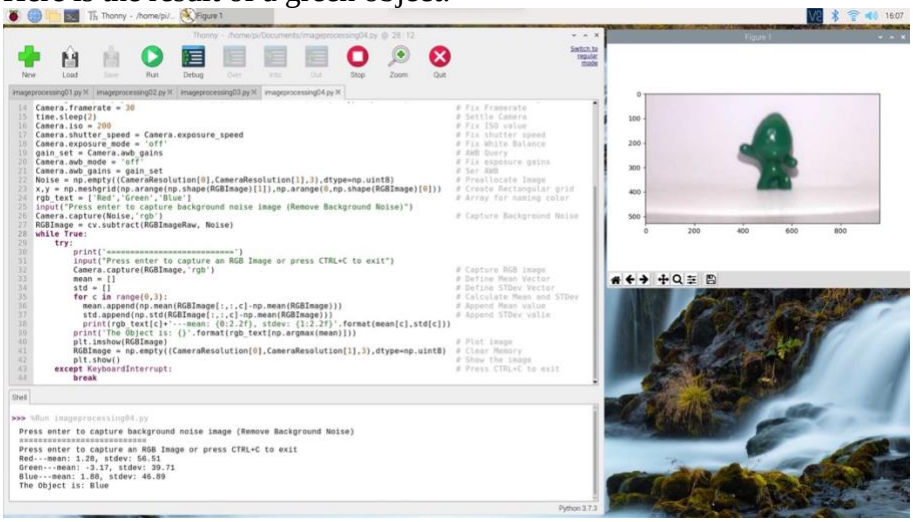
Here is the result of a red object:



The red has a highest mean (10,73) and therefore the object was interpreted correctly as red.

Green Objects

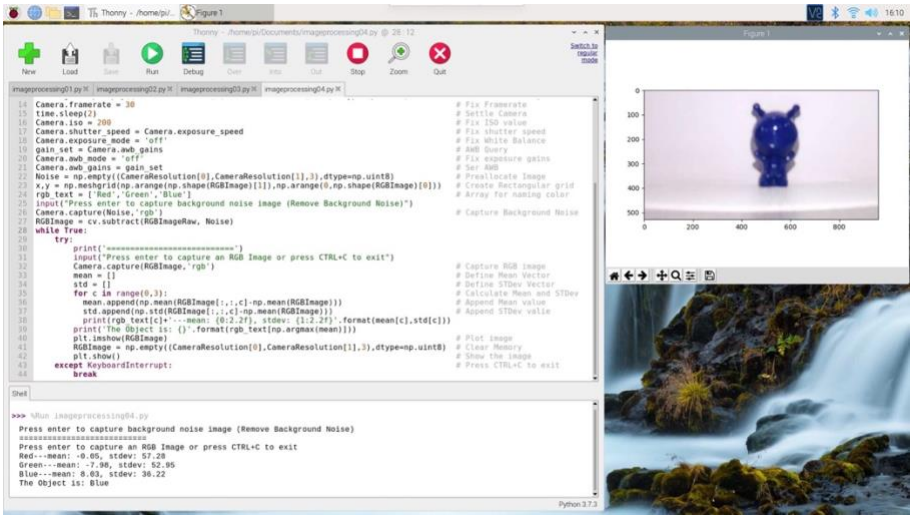
Here is the result of a green object:



The Blue has a highest mean (1,88) and therefore the object was interpreted as blue.
This is wrong and this is because of the yellow light. Therefore to make it work you need to use white light.

Blue Objects

Here is the result of a blue object:



The blue has a highest mean (8,03) and therefore the object was interpreted as blue. Which is correct.