On PC side:

1. Create a venv environment so that when you clone github or install any requirements/packages, it does not save in your local pc. This can ensure that it does not conflict with your local pc packages.

Can follow this guide: <https://www.youtube.com/watch?v=ohlRbcasPAc&ab_channel=IDGTECHtalk>

1. Go to terminal and click on new terminal. You should see the venv or other names depends on what you put. This means that you now on the venv.



1. You can deactivate and it will become like this.



1. Can activate back to venv by typing “.\\venv\Scripts\activate” and run it. You should be back in venv.
2. Please make sure you are in the venv before continue.
3. Clone the yolov5 github. Run this in the terminal.

git clone <https://github.com/ultralytics/yolov5>

1. Run this in the terminal to get the latest pip: python -m pip install --upgrade pip
2. For requirements, please run cd yolov5 and then pip install -r requirements.txt
3. Installing the PyTorch and CUDA

<https://pytorch.org/get-started/locally/>

Chart, bar chart

Description automatically generated

The above settings you can choose based on what you need. So for my case is Pip install.

pip3 install torch==1.9.0+cu102 torchvision==0.10.0+cu102 torchaudio===0.9.0 -f <https://download.pytorch.org/whl/torch_stable.html>

1. Run in the terminal: pip3 install IPython
2. Needed step 10 before can run the codes below.

import torch

from IPython.display import Image, clear\_output  # to display images

clear\_output()

print(*f*"Setup complete. Using torch {torch.\_\_version\_\_} ({torch.cuda.get\_device\_properties(0).name if torch.cuda.is\_available() else 'CPU'})")

It should show your laptop GPU. For mine is:

Setup complete. Using torch 1.9.0+cu102 (NVIDIA GeForce GTX 1050 with Max-Q Design)

1. Or another way to check if you have any cuda gpu

import torch

from IPython.display import Image, clear\_output  # to display images

clear\_output()

print(f"Setup complete. Using torch {torch.\_\_version\_\_} ({torch.cuda.get\_device\_properties(0).name if torch.cuda.is\_available() else 'CPU'})")

print(torch.cuda.is\_available())

print(torch.cuda.device(0))

print(torch.cuda.device\_count())

print(torch.cuda.get\_device\_name(0))

1. In the YOLOv5 folder, find detect.py

Change the device='0' so now pytorch will use your cuda gpu instead of cpu

Text

Description automatically generated

**Establish connection and streaming video from rpi to pc using ImageZMQ**

For more info: <https://www.pyimagesearch.com/2019/04/15/live-video-streaming-over-network-with-opencv-and-imagezmq/>

Sender (Rpi) 🡪 Receiver (Local PC)

1. Ensure you on your venv before installing these packages. E.g.



Pip install all the packages below:

pip3 install opencv-contrib-python

pip3 install imagezmq

pip3 install imutils

1. For the testing purpose, I will be using this link <https://github.com/jeffbass/imagezmq#id5>

to demonstrate the connection between rpi to local pc.

1. Input these codes into your python file.

#run this program on the Mac to display image streams from multiple RPis

import cv2

import imagezmq

image\_hub = imagezmq.ImageHub()

while True:  # show streamed images until Ctrl-C

    rpi\_name, image = image\_hub.recv\_image()

    print('Test')

    cv2.imshow(rpi\_name, image) # 1 window for each RPi

    print('Test 1')

    cv2.waitKey(1)

    image\_hub.send\_reply(*b*'OK')

**On Rpi side:**

1. I will be using VNC viewer to access the Rpi UI or if you have external monitor for it also can use.
2. Take note, you need internet access for the installation of the packages
3. Install all the packages below:

pip3 install opencv-contrib-python

pip3 install imagezmq

pip3 install imutils

1. Run this command:

vcgencmd get\_camera

To check whether your camera is been detected and it should be 1. If is not 1, go secure it properly or is hardware issues.



1. Create a new python file and input these codes. Edit the code based on your local pc IP address when connected to the MDPGrp Wi-Fi. Can do ipconfig on your local pc terminal to find out your IP address. For example, for my case is: **(*connect\_to*='tcp://192.168.25.8:5555')**

# run this program on each RPi to send a labelled image stream

import socket

import time

from imutils.video import VideoStream

import imagezmq

sender = imagezmq.ImageSender(*connect\_to*='tcp://192.168.26.27:5555')

rpi\_name = socket.gethostname() # send RPi hostname with each image

picam = VideoStream(*usePiCamera*=True).start()

time.sleep(2.0)  # allow camera sensor to warm up

while True:  # send images as stream until Ctrl-C

    image = picam.read()

    sender.send\_image(rpi\_name, image)

**Testing the connection between Rpi and local pc**

1. Once done, run step 16 the python file and ensure you are in venv. For example:



1. Run step 16(computer) first then step 21(rpi)

If happens to have any errors below step 24-26, run the respective command for it:

1. [How to solve ImportError: libhdf5\_serial.so.103: cannot open shared object file: No such file or directory while Importing h5py](https://stackoverflow.com/questions/62433716/how-to-solve-importerror-libhdf5-serial-so-103-cannot-open-shared-object-file)

# Graphical user interface, text, application Description automatically generated

# Run this command:

# sudo apt-get install python3-h5py

# [ImportError: libcblas.so.3: cannot open shared object file: No such file or directory](https://stackoverflow.com/questions/53347759/importerror-libcblas-so-3-cannot-open-shared-object-file-no-such-file-or-dire)

Run this command:

pip3 install opencv-contrib-python; sudo apt-get install -y libatlas-base-dev libhdf5-dev libhdf5-serial-dev libatlas-base-dev libjasper-dev libqtgui4 libqt4-test

# ImportError numpy core multiarray failed to import

Run this command:

pip3 install -U numpy

1. Rerun the python files in rpi.
2. If there is nothing appears, please ensure that your IP address is correct. If not, a windows firewall setting will appear just accept it (Should be able to skip step 29)
3. (This step may or maynot be needed depending on step 28) Go enable firewall settings. Go to control panel 🡪 System and Security 🡪 Allow an app through Windows Firewall.

Text

Description automatically generated with low confidence

1. Ensure your python.exe is both checked on Private and Public. If is not, click on the python.exe and then click on the Change Settings as shown in red below. Checked on the private and public, once done just click on Ok which is at the bottom.

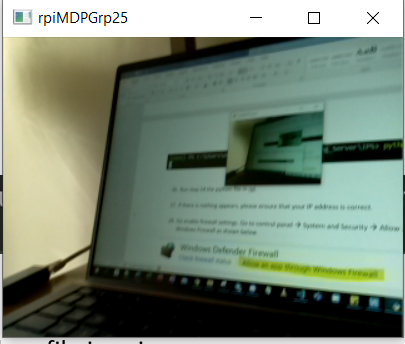
Graphical user interface, text, application, email

Description automatically generated

Issues of using ImageZMQ on windows: <https://github.com/jeffbass/imagezmq/issues/20>

1. To rerun the codes. Ctrl + C on the rpi side.
2. Reopen your visual studio code and ensure is in venv to run the python file. Wait for 4-5 seconds, it should be able to work.

For example:



1. To stop it from running. Ctrl + C on the rpi side and close the pop-up python UI. It will stop running on both rpi and local pc.

**Rpi Camera take images and send to local pc**

**On PC side:**

1. Create a folder and name it as raw\_images, ensure it is on the same folder as your python file.

Graphical user interface, application

Description automatically generated

1. Input these codes into your python file on PC.

import os

import sys

import numpy as np

from datetime import datetime

import imagezmq

import cv2

IMG\_ENCODING = '.png'

dir\_path = os.path.dirname(os.path.realpath(\_\_file\_\_)) + "/raw\_images"

SOURCE = dir\_path

image\_hub = imagezmq.ImageHub()

print('\nStarted Image Processing Server.\n')

while True:  # show streamed images until Ctrl-C

    print('Waiting for image from RPI...', flush=True)

    rpi\_name, image = image\_hub.recv\_image()

    print('Connected and received frame at time: %s' % str(datetime.now()), flush=True)

    # form image file path for saving

    raw\_image\_name = rpi\_name.replace(':', '') + ' ' + str(datetime.now().strftime('%d-%b\_%H-%M-%S')) + IMG\_ENCODING

    raw\_image\_path = os.path.join(SOURCE, raw\_image\_name)

    print(str(raw\_image\_path))

    save\_success = cv2.imwrite(raw\_image\_path, image)

    print(raw\_image\_path)

    print('Successfully saved')

    cv2.waitKey(1)

    image\_hub.send\_reply(b'OK')

**On Rpi side:**

1. Input these codes into your python file on Rpi. Remember to change your IP address for this ***connect\_to*='tcp://192.168.25.8:5555'.** You can change the time.sleep in the while loop to delay the speed of your rpi camera to take image.

#run this program on each RPi to send an image to PC

import socket

import time

from imutils.video import VideoStream

import imagezmq

from picamera import PiCamera

from picamera.array import PiRGBArray

sender = imagezmq.ImageSender(*connect\_to*='tcp://192.168.25.8:5555')

rpi\_name = socket.gethostname() # send RPi hostname with each image

camera = PiCamera(*resolution*=(2592, 1944)) #max resolution 2592,1944

rawCapture = PiRGBArray(camera)

time.sleep(1)  # allow camera sensor to warm up

while True:  # send images as stream until Ctrl-C

    time.sleep(2)

    camera.capture(rawCapture, *format*="bgr")

    image = rawCapture.array

    rawCapture.truncate(0)

    sender.send\_image(rpi\_name, image)

1. Once done, run step 2(pc) and then step 3(rpi). Wait for like 3-4 seconds. You can check your raw\_images folder on your pc, it should have the images took by rpi camera.

For example:

Graphical user interface, website

Description automatically generated