How to use the camera module for object detection?

1. Open **terminal**. Make sure your Raspberry Pi is fully updated by entering the following command.

```
sudo apt-get update
```

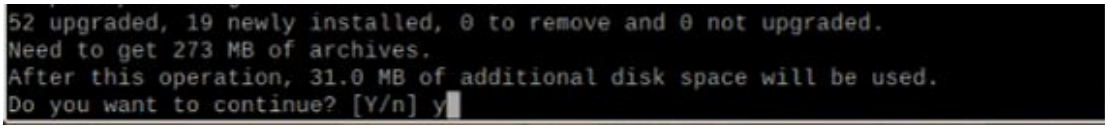
This command means download package information from all configured sources. `sudo` means "superuser do", it allows you to run administrative tasks.

After running this command, you should get the result similar to the following:

2. Install available upgrades of all packages currently installed on the system from the sources configured using the following command.

```
sudo apt-get dist-upgrade
```

Press "**Y**" to continue installation.



Noted that upgrade the system may end your TeamViewer session, just wait a few minutes before reconnect to the TeamViewer.

Reboot the system after upgrade finished.

```
sudo reboot
```

The system may end your TeamViewer session, just wait a few minutes before reconnect to the TeamViewer.

3. Install the packages, including tensorflow, libatlas-base-dev, pillow, lxml, jupyter, matplotlib, cython and python-tk. Press "**Y**" when the question "**Do you want to continue?**" is asked.

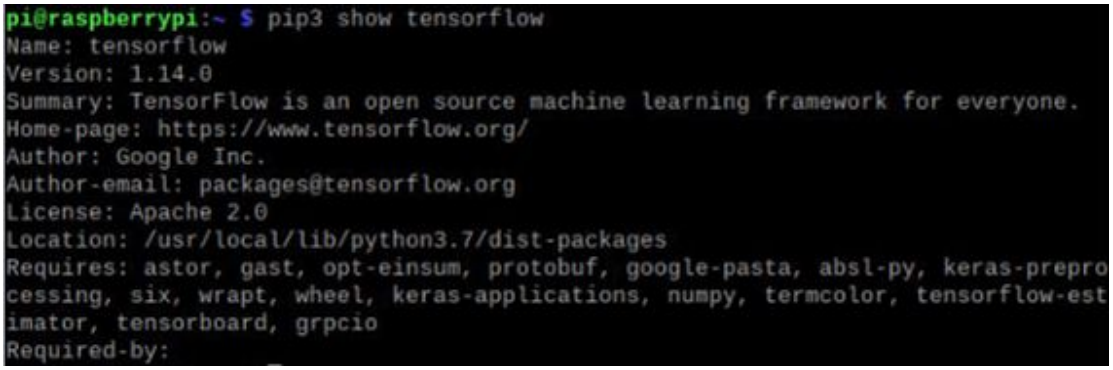
```
sudo pip3 install tensorflow

sudo apt-get install libatlas-base-dev

sudo pip3 install pillow lxml jupyter matplotlib cython

sudo apt-get install python-tk
```

You will notice we are using two ways to install a package, `pip3` and `apt-get`. `pip3` is used to download and install packages directly from Python Package index (PyPI). PyPI is hosted by Python Software Foundation. It is a specialized package manager that only deals with python packages. `apt-get` is used to download and install packages from Ubuntu repositories which are hosted by Canonical.

Check if tensorflow is installed.

```
pip3 show tensorflow
```

You should be able to see the following:



4. Install OpenCV and the packages needed.

```
sudo apt-get install libjpeg-dev libtiff5-dev
libjasper-dev libpng12-dev

sudo apt-get install libavcodec-dev libavformat-dev
libswscale-dev libv4l-dev

sudo apt-get install libxvidcore-dev libx264-dev

sudo apt-get install qt4-dev-tools libatlas-base-dev

pip3 install opencv-python==3.4.6.27
```

You will notice we are installing OpenCV version 3.4.6.27 specifically using command `pip3 install opencv-python==3.4.6.27`. It is because if you using command `pip3 install opencv-python`, it will install the latest version of OpenCV but the latest version of OpenCV does not support Raspberry Pi. Therefore, we need to specific the version we are installing.

Verify the OpenCV version.

```
pip3 show opencv-python
```

You should be able to see the following:

```
pi@raspberrypi:~ $ pip3 show opencv-python
Name: opencv-python
Version: 3.4.6.27
Summary: Wrapper package for OpenCV python bindings.
Home-page: https://github.com/skvark/opencv-python
Author: None
Author-email: None
License: MIT
Location: /home/pi/.local/lib/python3.7/site-packages
Requires: numpy
Required-by:
```

5. Install the Protocol Buffers compiler.

```
sudo apt-get install protobuf-compiler
```

Protocol Buffers (Protobuf) is a method of serializing structured data. It is useful in developing programs to communicate with each other over a wire or for storing data.

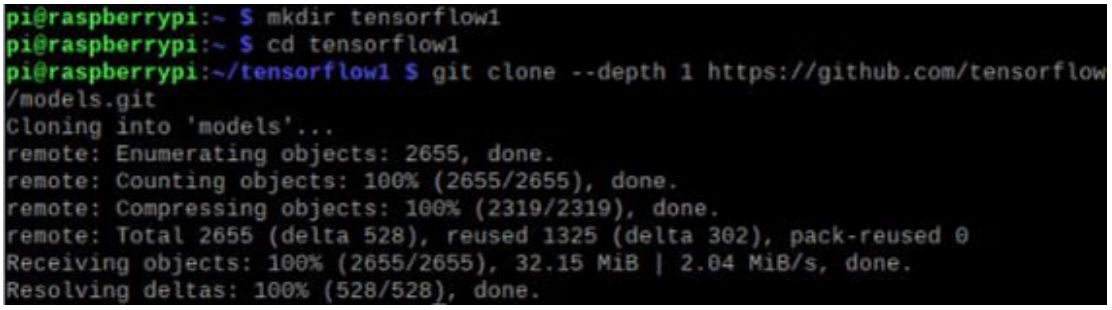
6. Verify the Installation of Protocol Buffers compiler by type in `protoc --version`. The result should be the following:

7. Set up **TensorFlow Directory** structure.

```
mkdir tensorflow1

cd tensorflow1

git clone --depth 1
https://github.com/tensorflow/models.git
```

`mkdir` is used to make a new directory. `git clone` is a Git command line utility which is used to target an existing repository and create a clone, or copy of the target repository.

After running this command, you should get the result similar to the following:

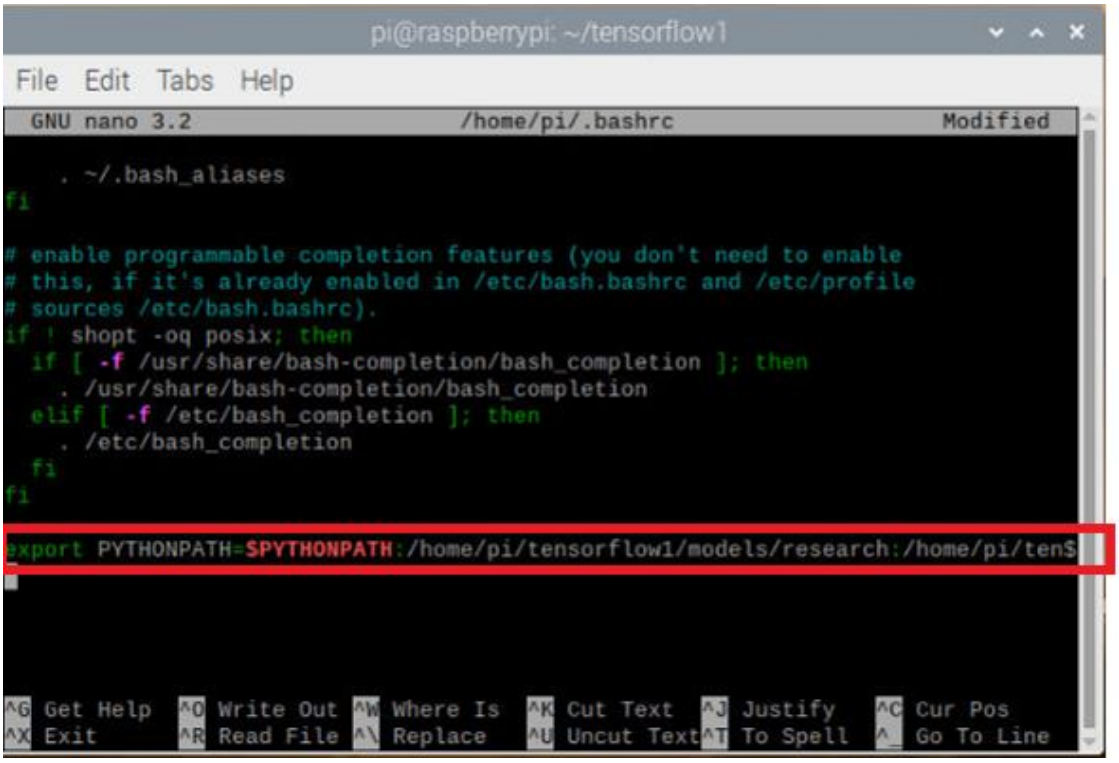


8. Edit the file `~/.bashrc`.

```
sudo nano ~/.bashrc
```

9. Add the following **at the end of the file**.

```
export
PYTHONPATH=$PYTHONPATH:/home/pi/tensorflow1/models/resea
rch:/home/pi/tensorflow1/models/research/slim
```



Noted that you cannot scroll down using mouse inside text editor, use **downwards arrow** to do so.

To exit the text editor and save the file, press **Ctrl + X → Y → Enter**.

10. Change the current directory into the following:

```
cd /home/pi/tensorflow1/models/research
```

11. Use the Protocol Buffers compiler to compile.

```
protoc object_detection/protos/*.proto --python_out=.
```

After running this command, you should get the result similar to the following:

```
pi@raspberrypi:~/tensorflow1/models/research $ protoc object_detection/protos/*.
proto --python_out=.
object_detection/protos/input_reader.proto: warning: Import object_detection/pro
tos/image_resizer.proto but not used.
```

12. Change the current directory to the following:

```
cd /home/pi/tensorflow1/models/research/object_detection
```

13. Download the SSDLite-MobileNet model.

```
wget
http://download.tensorflow.org/models/object_detection/s
sdlite_mobilenet_v2_coco_2018_05_09.tar.gz
```

`wget` is a command for downloading files from the Internet.

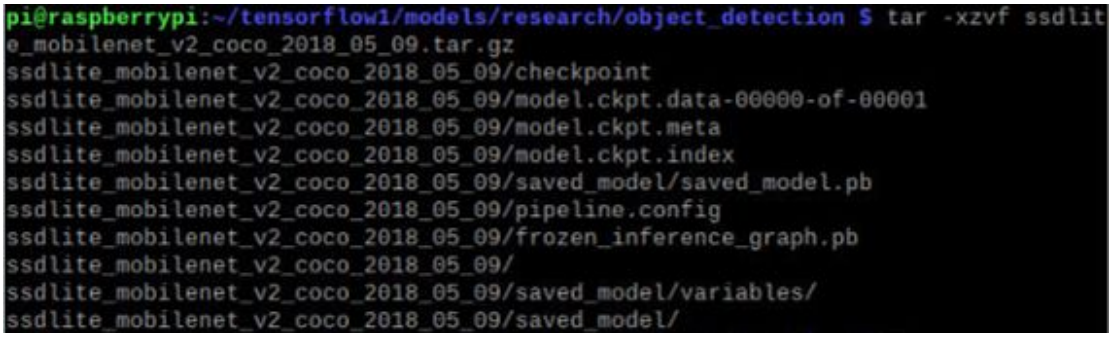
After running this command, you should get the result similar to the following:

14. Unpack the file you just downloaded.

```
tar -xzvf ssdlite_mobilenet_v2_coco_2018_05_09.tar.gz
```

`tar -xzvf` is a command that extract the contents of a .tar.gz file to the current directory.

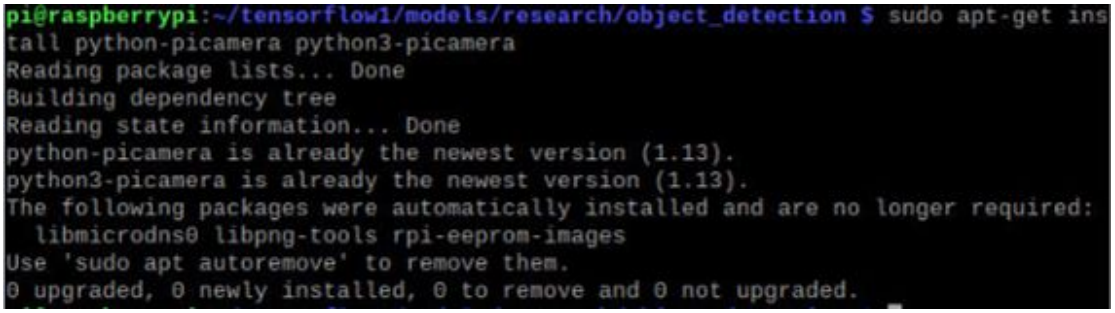
After running this command, you should get the result similar to the following:



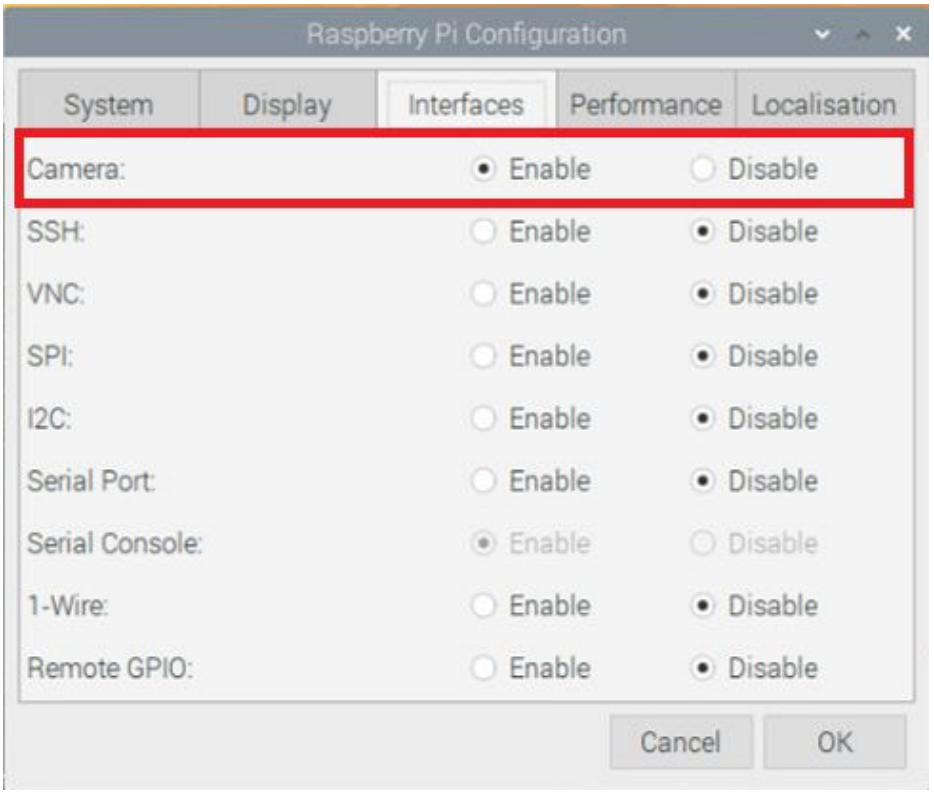
15. Install Pi camera.

```
sudo apt-get install python-picamera python3-picamera
```

After running this command, you should get the result similar to the following:



Make sure the camera is **enabled** in the "Raspberry Pi configuration" menu.



16. Skip this step if your camera already **enabled.**

Reboot the system after **enabled** the camera. The system may end your TeamViewer session, just wait a few minutes before reconnect to the TeamViewer.

Then, **change directory** to the following:

```
cd /home/pi/tensorflow1/models/research/object_detection
```

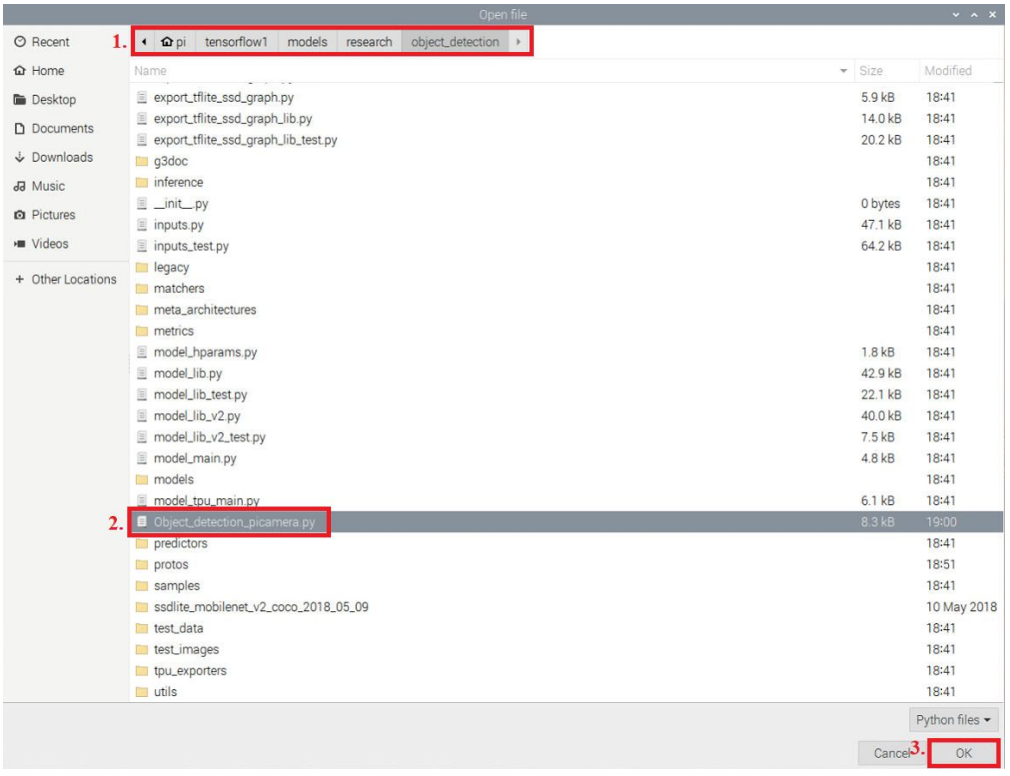
17. Download the `Object_detection_picamera.py` file into the object_detection directory by issuing:

```
wget
https://raw.githubusercontent.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi/master/Object_detection_picamera.py
```

After running this command, you should get the result similar to the following:

```
pi@raspberrypi:~/tensorflow1/models/research/object_detection $ wget https://raw
.githubusercontent.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspbe
rry-Pi/master/Object_detection_picamera.py
--2020-06-05 19:00:44--  https://raw.githubusercontent.com/EdjeElectronics/Tenso
rFlow-Object-Detection-on-the-Raspberry-Pi/master/Object_detection_picamera.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.76.13
3
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.76.1
33|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8317 (8.1K) [text/plain]
Saving to: 'Object_detection_picamera.py'

Object_detection_pi 100%[===================>]   8.12K  --.-KB/s    in 0.001s

2020-06-05 19:00:50 (13.1 MB/s) - 'Object_detection_picamera.py' saved [8317/831
7]
```

18. Open Thonny Python IDE. Click "**Load**" on the top navigation bar. Navigate to the following path: **/home/pi/tensorflow1/models/research/object_detection**. Select the file `Object_detection_picamera.py`. Click "**OK**".



19. Find the following section of the program.

```
27 import tensorflow as tf
28 import argparse
29 import sys
30
31 # Set up camera constants
32 IM_WIDTH = 1280
33 IM_HEIGHT = 720
34 #IM_WIDTH = 640   # Use smaller resolution for
35 #IM_HEIGHT = 480   #slightly faster framerate
36
37 # Select camera type (if user enters --usbcam when calling this script,
38 # a USB webcam will be used)
39 camera_type = 'picamera'
40 parser = argparse.ArgumentParser()
41 parser.add_argument('--usbcam', help='Use a USB webcam instead of picamera',
42                     action='store_true')
43 args = parser.parse_args()
44 if args.usbcam:
45     camera_type = 'usb'
```

Change the resolution from 1280x720 to 640x480 as shown below:

```
27 import tensorflow as tf
28 import argparse
29 import sys
30
31 # Set up camera constants
32 # IM_WIDTH = 1280
33 # IM_HEIGHT = 720
34 IM_WIDTH = 640   # Use smaller resolution for
35 IM_HEIGHT = 480   #slightly faster framerate
36
37 # Select camera type (if user enters --usbcam when calling this script,
38 # a USB webcam will be used)
39 camera_type = 'picamera'
40 parser = argparse.ArgumentParser()
41 parser.add_argument('--usbcam', help='Use a USB webcam instead of picamera',
42                     action='store_true')
43 args = parser.parse_args()
44 if args.usbcam:
45     camera_type = 'usb'
```

20. Find the following section of the program.

Object_detection_picamera.py

```
120
121 ### Picamera ###
122 if camera_type == 'picamera':
123     # Initialize Picamera and grab reference to the raw capture
124     camera = PiCamera()
125     camera.resolution = (IM_WIDTH,IM_HEIGHT)
126     camera.framerate = 10
127     rawCapture = PiRGBArray(camera, size=(IM_WIDTH,IM_HEIGHT))
128     rawCapture.truncate(0)
129
130     for frame1 in camera.capture_continuous(rawCapture, format="bgr",use_video_port=True):
131
132         t1 = cv2.getTickCount()
133
134         # Acquire frame and expand frame dimensions to have shape: [1, None, None, 3]
135         # i.e. a single-column array, where each item in the column has the pixel RGB value
136         frame = np.copy(frame1.array)
137         frame.setflags(write=1)
```

Copy the following code.

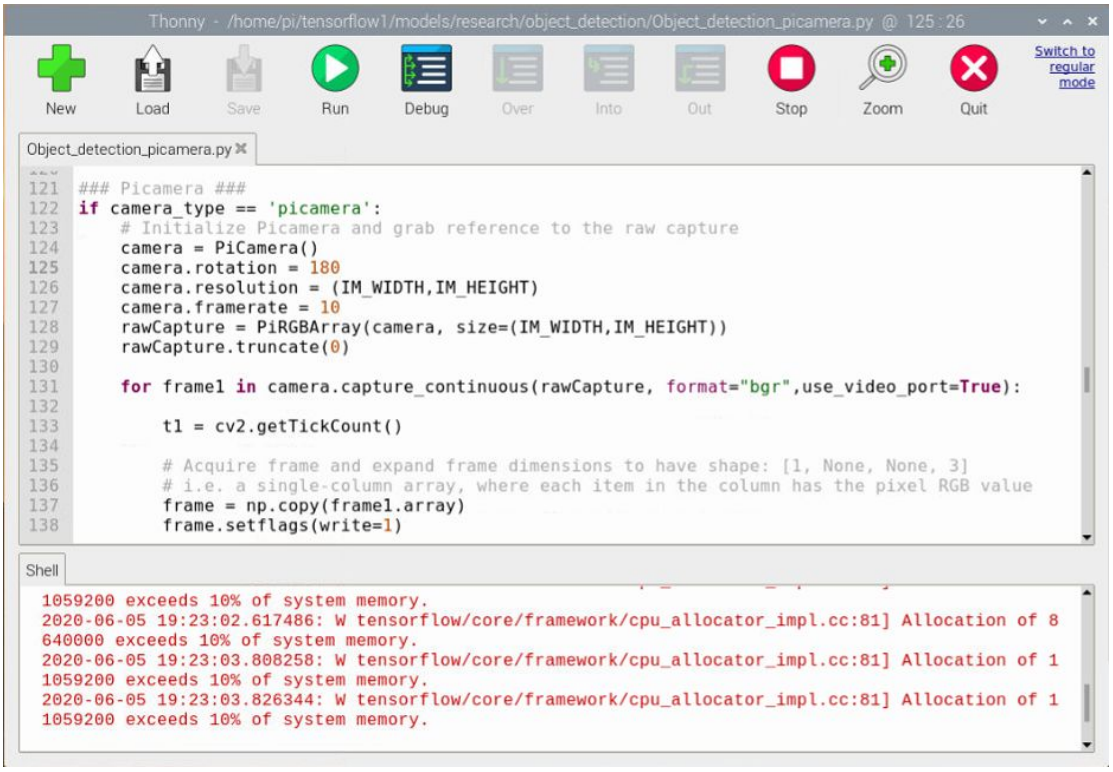
```
camera.rotation = 180
```

Add it into the position shown below to turn the live stream right side up.

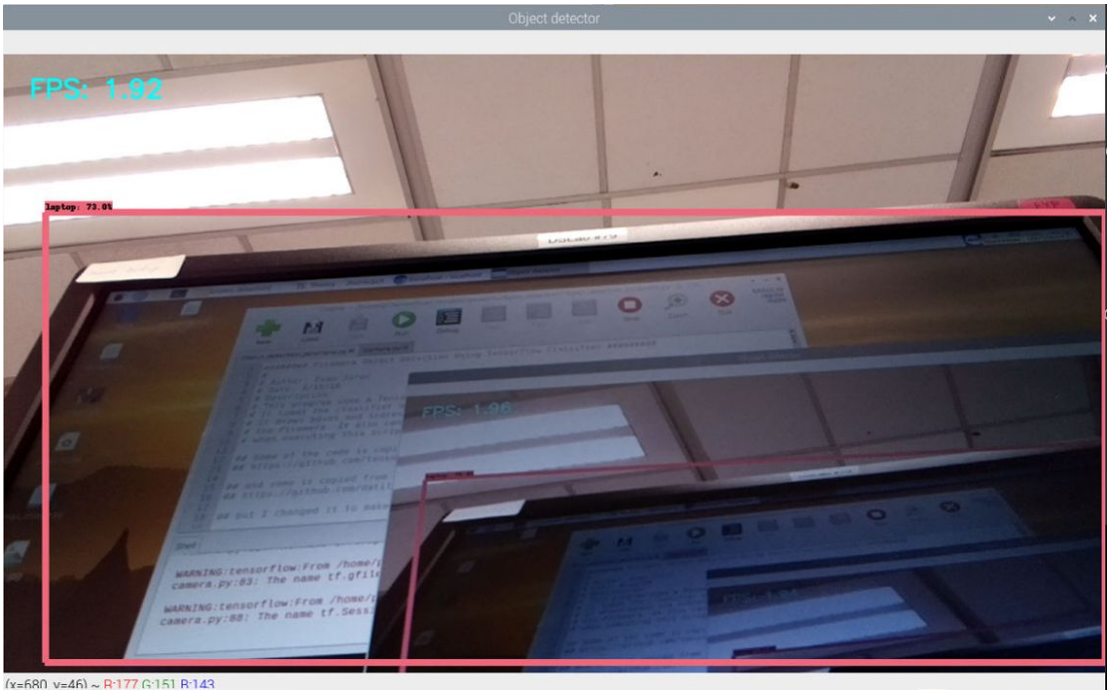
Object_detection_picamera.py

```
120
121 ### Picamera ###
122 if camera_type == 'picamera':
123     # Initialize Picamera and grab reference to the raw capture
124     camera = PiCamera()
125     camera.rotation = 180
126     camera.resolution = (IM_WIDTH,IM_HEIGHT)
127     camera.framerate = 10
128     rawCapture = PiRGBArray(camera, size=(IM_WIDTH,IM_HEIGHT))
129     rawCapture.truncate(0)
130
131     for frame1 in camera.capture_continuous(rawCapture, format="bgr",use_video_port=True):
132
133         t1 = cv2.getTickCount()
134
135         # Acquire frame and expand frame dimensions to have shape: [1, None, None, 3]
136         # i.e. a single-column array, where each item in the column has the pixel RGB value
```

21. Click “**Run**”. Once the script initializes (which can take up to **30 seconds**), you will see a window showing a live stream from your camera.



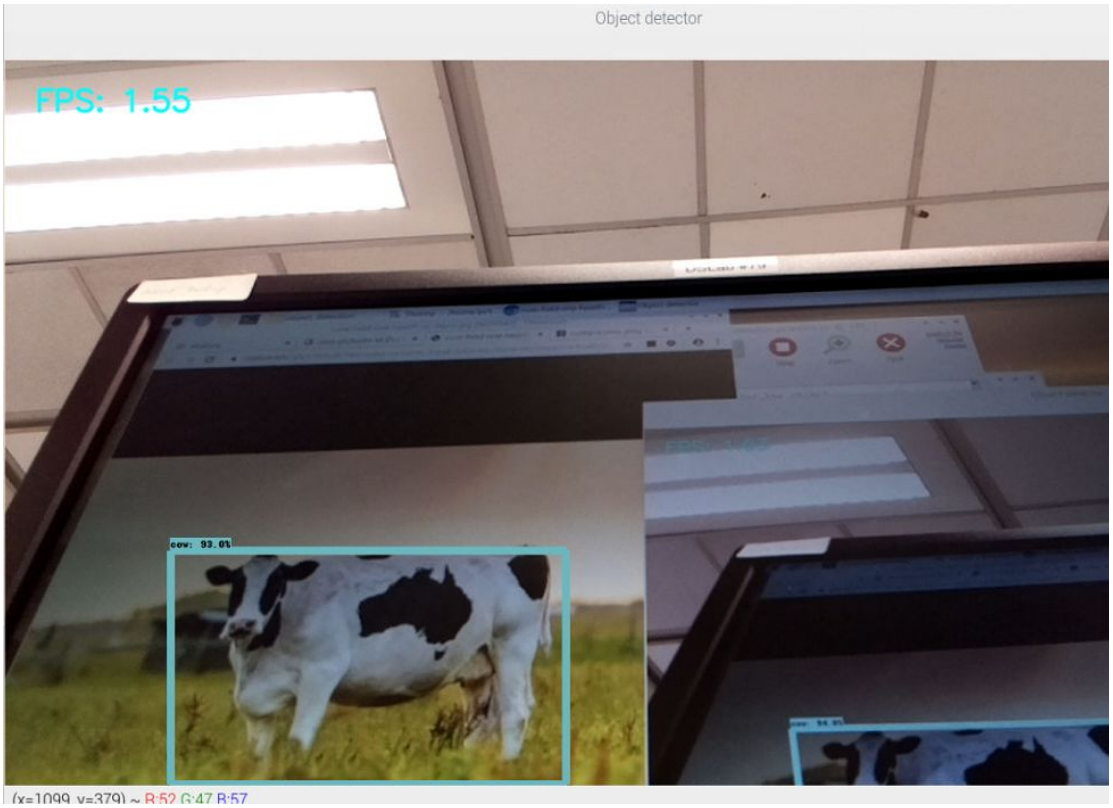
22. You should be able to see the following:



If your camera cannot see the monitor properly, please request assistance from the technical staffs to adjust the camera position.

23. Click the **web browser icon** at the top menu bar.

24. Find an image that your object detection program is able to recognize and label correctly. The following is an example:



HINT: You can try to find some picture of foods, animals and household appliances.

25. After finish this step, demonstrate the result to the technical staffs.

26. Click "**Stop**" on the top navigation bar of the Thonny Python IDE if you want to end the live stream.