STUDY GUIDE FOR MODULE NO. LAB 08

Facial Recognition Using RPi

MODULE OVERVIEW

Face recognition on Raspberry Pi (RPI) detects and identifies faces using the processor and camera modules. The setup comprises installing the Raspbian operating system, integrating OpenCV for image processing, and incorporating Python libraries for coding. Once the hardware and software are finished, OpenCV's algorithms can detect faces in images or live video feeds from the camera.

Recognition is the process of matching observed faces to a database, which is often accomplished using learning or classical algorithms trained on labeled datasets. It has numerous applications, including security and surveillance, access control, emotion analysis, and intelligent interfaces. Integration connects the system to databases or actuators to accomplish activities like access control. Optimization and ethical considerations, such as privacy and prejudice prevention, are crucial for the successful implementation and operation of facial recognition systems on Raspberry Pi.

This module will explore the use of RPi for facial recognition. The process of creating a facial recognition system using RPi will be tackled, as well as the steps involved in setting it up and showing users how to use it. Furthermore, this module will allow users to understand how facial recognition works on RPi.

MODULE LEARNING OUTCOMES

Upon completion of this module of Facial Recognition using Raspberry Pi, learners should be able to:

- ✓ Understand the basics of facial recognition technology and how it works.
- ✓ Learn how to install OpenCV, face_recognition, and imutils packages and dependencies on the Raspberry Pi.
- ✓ Collecting and managing a set of images to use as dataset for training.
- ✓ Training the Raspberry Pi to recognize faces using the images from the dataset.
- ✓ Witness how accurate the facial recognition from the live feed based on the given dataset.

LEARNING ACTIVITY

Name: Cerujano, Erman Ace M.________________ Due date: ___**March 25, 2024**_____

Summary on how to install Facial Recognition Using Rpi:

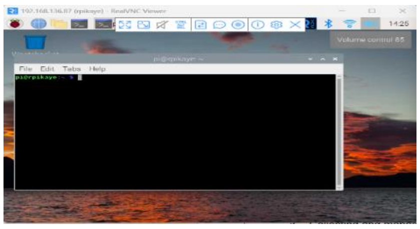
Prerequisites:

- Raspberry Pi 4 or better
- Raspberry Pi OS, preferably bullseye or bookworm.
- Internet Connection
- Remote Desktop (Real VNC viewer)

Steps:

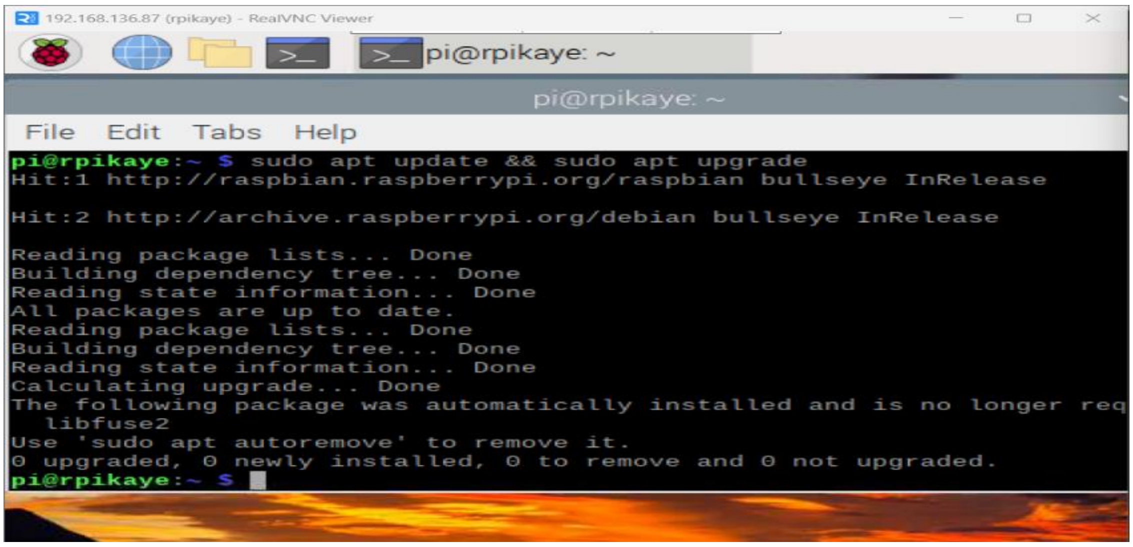
Step 1: Raspberry Pi and Remote desktop setup.

This setup operates on the 32-bit Pi OS using a USB webcam. However, it's important to mention that this procedure may also function on the 64-bit edition of the Pi OS once comprehensive support for camera modules is established. The Real VNC viewer is used to access the remote desktop.



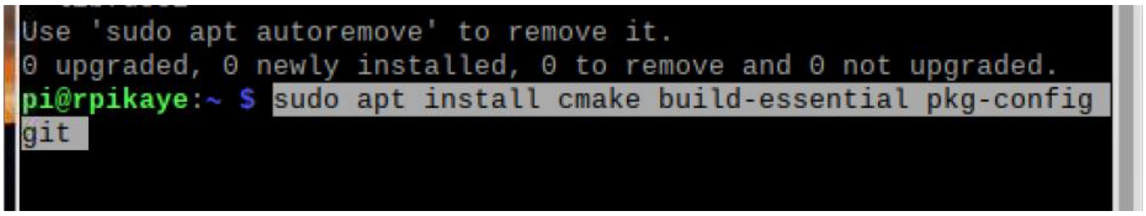
```
pi@rpikaye:~ $
```

Step 2: Update and upgrade the currently installed packages by running the following command.



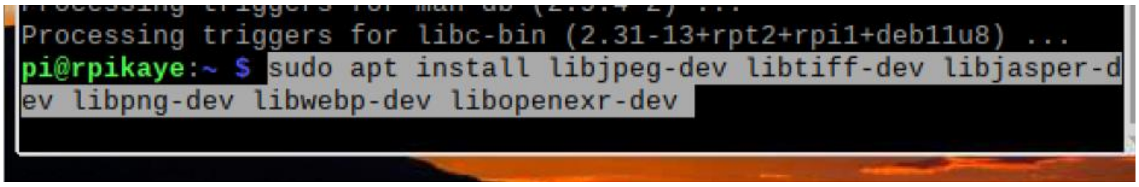
```
pi@rpikaye:~ $ sudo apt update && sudo apt upgrade
Hit:1 http://raspbian.raspberrypi.org/raspbian bullseye InRelease

Hit:2 http://archive.raspberrypi.org/debian bullseye InRelease

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following package was automatically installed and is no longer req
  libfuse2
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@rpikaye:~ $
```

Step 3: Install all the packages that needs to compile for OpenCV.

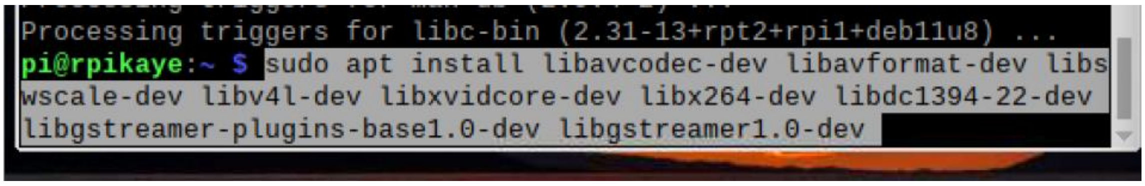


```
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@rpikaye:~ $ sudo apt install cmake build-essential pkg-config
git
```

Step 4: Install the packages that will add support for different image and video formats to OpenCV with the following commands.

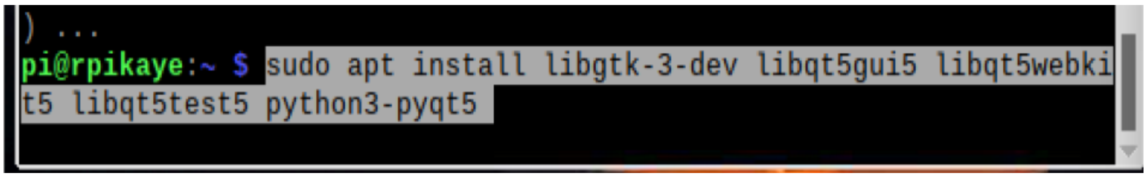


```
Processing triggers for libc-bin (2.31-13+rpt2+rpi1+deb11u8) ...
pi@rpikaye:~ $ sudo apt install libjpeg-dev libtiff-dev libjasper-d
ev libpng-dev libwebp-dev libopenexr-dev
```



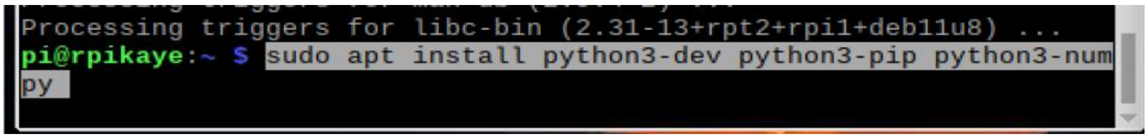
```
Processing triggers for libc-bin (2.31-13+rpt2+rpi1+deb11u8) ...
pi@rpikaye:~ $ sudo apt install libavcodec-dev libavformat-dev libs
wscale-dev libv4l-dev libxvidcore-dev libx264-dev libdc1394-22-dev
libgstreamer-plugins-base1.0-dev libgstreamer1.0-dev
```

Step 5: Install all the packages needed for OpenCV's interface.



```
) ...
pi@rpikaye:~ $ sudo apt install libgtk-3-dev libqt5gui5 libqt5webki
t5 libqt5test5 python3-pyqt5
```

Step 6: Install the final packages in order to compile OpenCV with support for Python on Raspberry Pi.

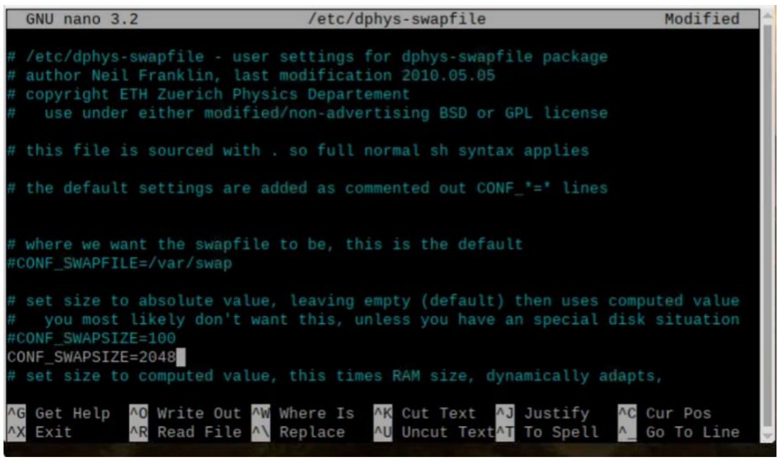


```
Processing triggers for libc-bin (2.31-13+rpt2+rpi1+deb11u8) ...
pi@rpikaye:~ $ sudo apt install python3-dev python3-pip python3-num
py
```

Step 7: Expand the swapfile temporarily. To begin expanding the swapfile, open dphys-swapfile for editing:

"Sudo nano /etc/dphys-swapfile"

Once the file is open, remove the line CONF_SWAPSIZE=100 and replace it with CONF_SWAPSIZE=2048. To save your modifications to dphys-swapfile, press Ctrl-X, Y, and then Enter. This is merely a temporary modification; we will reverse it after we have finished installing OpenCV.

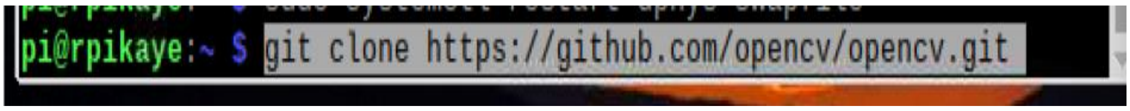


```
  GNU nano 3.2                 /etc/dphys-swapfile                   Modified

# /etc/dphys-swapfile - user settings for dphys-swapfile package
# author Neil Franklin, last modification 2010.05.05
# copyright ETH Zuerich Physics Departement
#   use under either modified/non-advertising BSD or GPL license

# this file is sourced with . so full normal sh syntax applies

# the default settings are added as commented out CONF_*=* lines


# where we want the swapfile to be, this is the default
#CONF_SWAPFILE=/var/swap

# set size to absolute value, leaving empty (default) then uses computed value
#   you most likely don't want this, unless you have an special disk situation
#CONF_SWAPSIZE=100
CONF_SWAPSIZE=2048
# set size to computed value, this times RAM size, dynamically adapts,

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

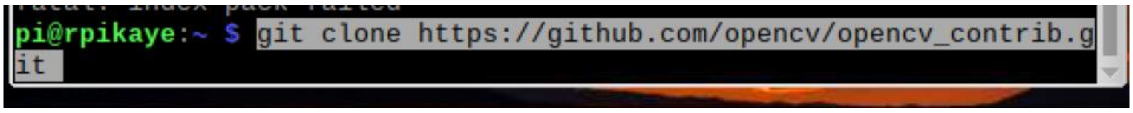
Afterwards, restart it's service by utilizing this command.

"sudo systemctl restart dphys-swapfile"

Step 8: Clone the OpenCV repositories needed for the Raspberry Pi.



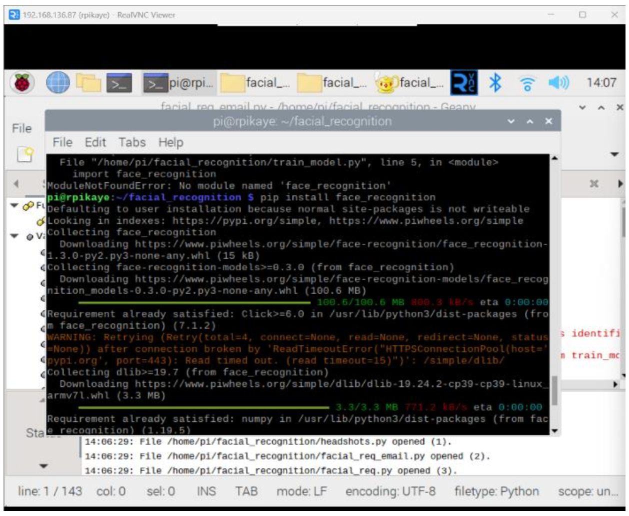
```
pi@rpikaye:~ $ git clone https://github.com/opencv/opencv.git
```



```
pi@rpikaye:~ $ git clone https://github.com/opencv/opencv_contrib.g
it
```

Step 9: Install **face_recognition** and **imutils** by the following commands :

"pip install facial_recognition" and ***"pip install imutils"***

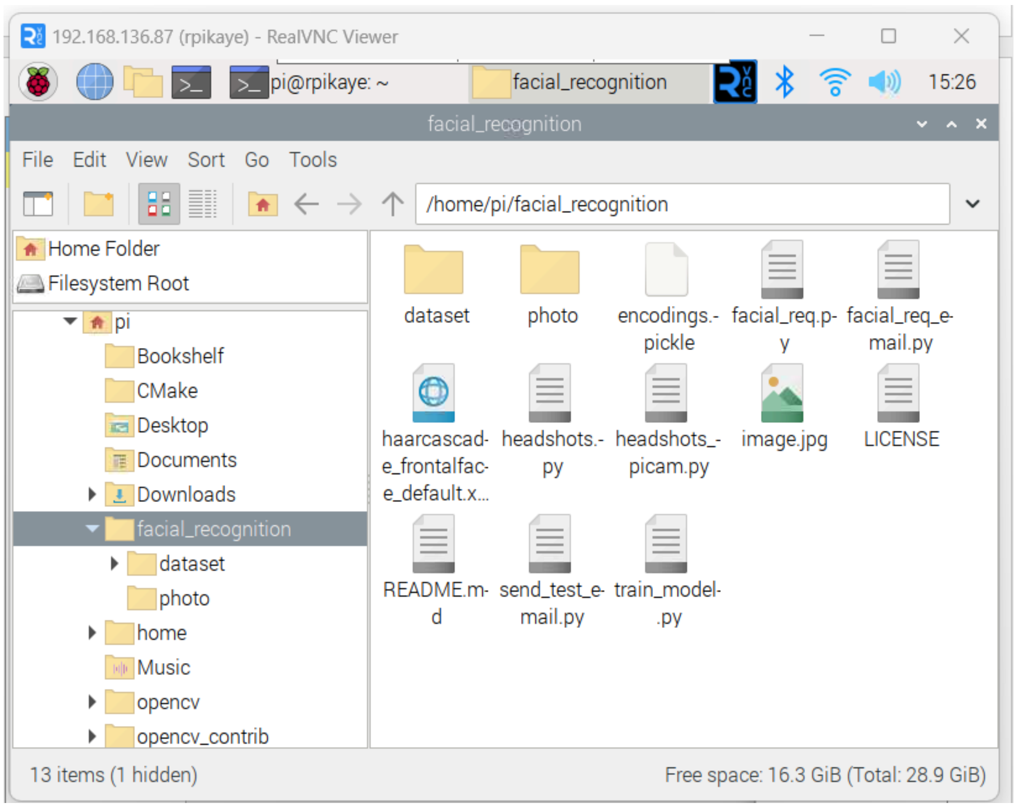


```
  File "/home/pi/facial_recognition/train_model.py", line 5, in <module>
    import face_recognition
ModuleNotFoundError: No module named 'face_recognition'
pi@rpikaye:~/facial_recognition $ pip install face_recognition
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting face_recognition
  Downloading https://www.piwheels.org/simple/face-recognition/face_recognition-
1.3.0-py2.py3-none-any.whl (15 kB)
Collecting face-recognition-models>=0.3.0 (from face_recognition)
  Downloading https://www.piwheels.org/simple/face-recognition-models/face_recog
nition_models-0.3.0-py2.py3-none-any.whl (100.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100.6/100.6 MB 800.3 kB/s eta 0:00:00
Requirement already satisfied: Click>=6.0 in /usr/lib/python3/dist-packages (fro
m face_recognition) (7.1.2)
WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status
=None)) after connection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='
pypi.org', port=443): Read timed out. (read timeout=15)")': /simple/dlib/
Collecting dlib>=19.7 (from face_recognition)
  Downloading https://www.piwheels.org/simple/dlib/dlib-19.24.2-cp39-cp39-linux_
armv7l.whl (3.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.3/3.3 MB 771.2 kB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/lib/python3/dist-packages (from fac
e_recognition) (1.19.5)
```

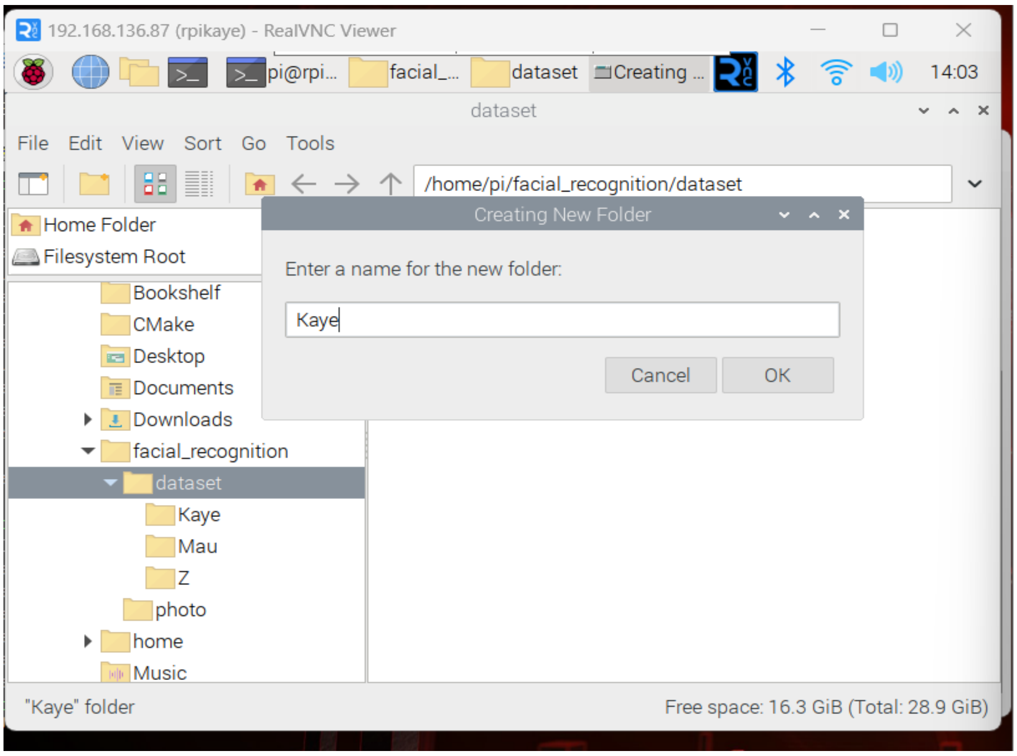
```
14:06:29: File /home/pi/facial_recognition/headshots.py opened (1).
14:06:29: File /home/pi/facial_recognition/facial_req_email.py opened (2).
14:06:29: File /home/pi/facial_recognition/facial_req.py opened (3).
```

192.168.136.87 (rpikaye) - RealVNC Viewer
pi@rpi...
facial_...
facial_...
facial_...
14:07
facial_req_email.py - /home/pi/facial_recognition - Geany
pi@rpikaye: ~/facial_recognition
File Edit Tabs Help
File
line: 1 / 143 col: 0 sel: 0 INS TAB mode: LF encoding: UTF-8 filetype: Python scope: un...

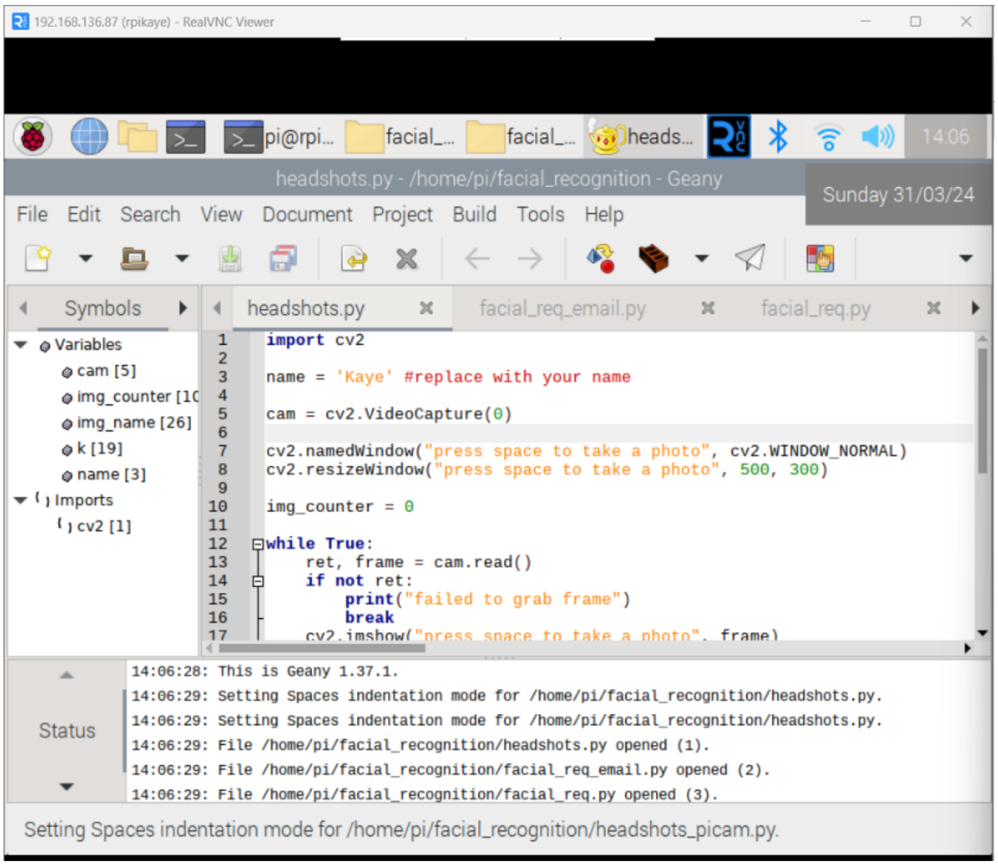
Step 10: Train the model by copying Python code files that are needed.
git clone *https://github.com/carolinedunn/facial_recognition*.



Step 11: Navigate to the facial_recognition folder and proceed to the dataset folder to create a New Folder. Within the newly created folder, input your name.



Step 12: Still in facial_recognition folder, now open headshots.py in Geany.

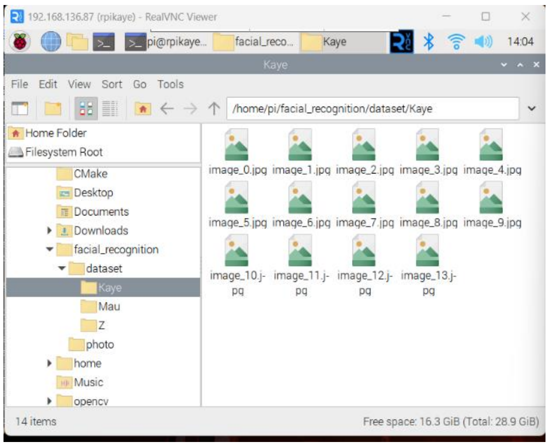


```
import cv2

name = 'Kaye' #replace with your name

cam = cv2.VideoCapture(0)

cv2.namedWindow("press space to take a photo", cv2.WINDOW_NORMAL)
cv2.resizeWindow("press space to take a photo", 500, 300)

img_counter = 0

while True:
    ret, frame = cam.read()
    if not ret:
        print("failed to grab frame")
        break
    cv2.imshow("press space to take a photo", frame)
```

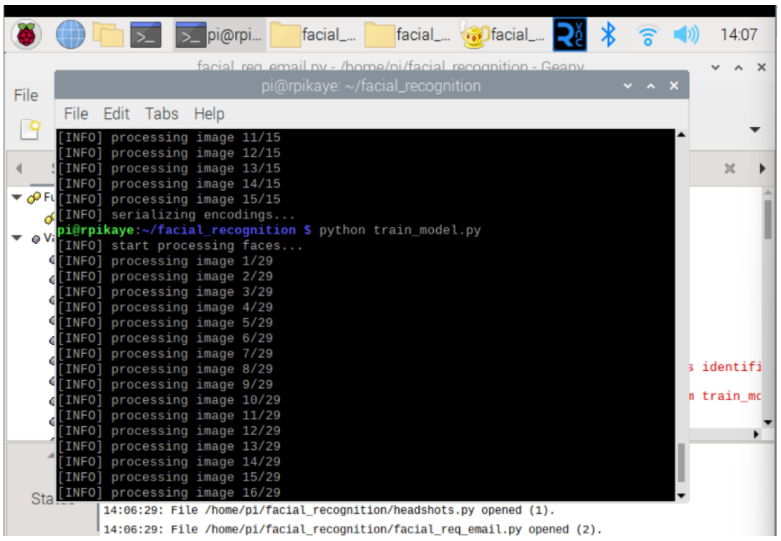
```
14:06:28: This is Geany 1.37.1.
14:06:29: Setting Spaces indentation mode for /home/pi/facial_recognition/headshots.py.
14:06:29: Setting Spaces indentation mode for /home/pi/facial_recognition/headshots.py.
14:06:29: File /home/pi/facial_recognition/headshots.py opened (1).
14:06:29: File /home/pi/facial_recognition/facial_req_email.py opened (2).
14:06:29: File /home/pi/facial_recognition/facial_req.py opened (3).
```

```
Setting Spaces indentation mode for /home/pi/facial_recognition/headshots_picam.py.
```

Step 13: To capture an image of yourself, direct the webcam towards you and press the spacebar. Each press of the spacebar triggers a new photo capture. We recommend taking approximately ten pictures of your face from different angles, ensuring to slightly turn your head in each shot. If you wear glasses, it's suggested to take a few pictures with and without them. Wearing hats in training shots is discouraged. These images will be used to train our model. Once you've finished taking pictures of yourself, press the Esc key.



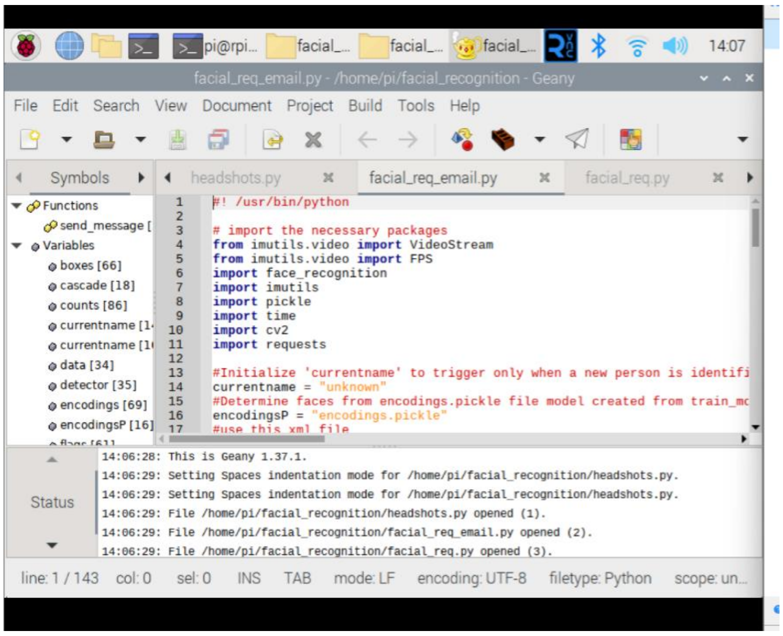
Step 14: In a new terminal, navigate to facial_recognition by typing ***"cd facial_recognition"*** and train the model by running the command "***python train_model.py".***



```
[INFO] processing image 11/15
[INFO] processing image 12/15
[INFO] processing image 13/15
[INFO] processing image 14/15
[INFO] processing image 15/15
[INFO] serializing encodings...
pi@rpikaye:~/facial_recognition $ python train_model.py
[INFO] start processing faces...
[INFO] processing image 1/29
[INFO] processing image 2/29
[INFO] processing image 3/29
[INFO] processing image 4/29
[INFO] processing image 5/29
[INFO] processing image 6/29
[INFO] processing image 7/29
[INFO] processing image 8/29
[INFO] processing image 9/29
[INFO] processing image 10/29
[INFO] processing image 11/29
[INFO] processing image 12/29
[INFO] processing image 13/29
[INFO] processing image 14/29
[INFO] processing image 15/29
[INFO] processing image 16/29
```

```
14:06:29: File /home/pi/facial_recognition/headshots.py opened (1).
14:06:29: File /home/pi/facial_recognition/facial_req_email.py opened (2).
```

pi@rpi...
facial_...
facial_...
facial_...
14:07
facial_req_email.py - /home/pi/facial_recognition - Geany
pi@rpikaye: ~/facial_recognition
File Edit Tabs Help



```
#! /usr/bin/python

# import the necessary packages
from imutils.video import VideoStream
from imutils.video import FPS
import face_recognition
import imutils
import pickle
import time
import cv2
import requests

#Initialize 'currentname' to trigger only when a new person is identifi
currentname = "unknown"
#Determine faces from encodings.pickle file model created from train_mc
encodingsP = "encodings.pickle"
#use this xml file
```

```
14:06:28: This is Geany 1.37.1.
14:06:29: Setting Spaces indentation mode for /home/pi/facial_recognition/headshots.py.
14:06:29: Setting Spaces indentation mode for /home/pi/facial_recognition/headshots.py.
14:06:29: File /home/pi/facial_recognition/headshots.py opened (1).
14:06:29: File /home/pi/facial_recognition/facial_req_email.py opened (2).
14:06:29: File /home/pi/facial_recognition/facial_req.py opened (3).
```

SUMMARY / CONCLUSION

In conclusion, with the use of the technology known as face recognition, a computer can recognize or confirm an individual's identity by examining patterns in their facial features. A tiny, inexpensive computer called the Raspberry Pi can be used to create a variety of tasks, such as facial recognition software.

Detailed instructions for teaching your Raspberry Pi to identify faces are included in this module. This module also describes how to train a Raspberry Pi using the OpenCV, face_recognition, and imutils packages using a collection of photos that we offer as our dataset. To assign names to faces in the file encodings.pickle, run python train_model.py, which will examine the photos in our dataset. To recognize and identify faces, run facial_req.py after the Raspberry Pi has been trained.

REFERENCES

[15] Dunn, C. (2022, September 17). How to Train your Raspberry Pi for Facial Recognition. *Tom's Hardware*. https://www.tomshardware.com/how-to/raspberry-pi-facial-recognition

[16] Geerlingguy. (n.d.). *GitHub - geerlingguy/facial_recognition: Live facial recognition example for Raspberry Pi 4 based on OpenCV*. GitHub. https://github.com/geerlingguy/facial_recognition?fbclid=IwAR2uvchxynd2QGnGLUoeo6dW_v0zgN9oTKMhOQwQDhKngrdsKf7by3jlEn8