**Real time Face Recognition based Attendance on Raspi**

**FR code in python using Face Recognition and DLIB for Face Detection,Open CV in Raspberry Pi or any other PC.**

**Setup & Install Dependencies:**

Before we begin: Grab your Raspberry Pi 4B and flash BusterOS to your microSD

Let’s review the hardware requirements for this Raspibian 4B

**Raspberry Pi:** This tutorial assumes you are using a Raspberry Pi 4B 1GB, 2GB or 4GB,8GB hardware.

**Operating system**: These instructions only apply to Raspbian Buster.

**64 GB microSD:** We recommend the high-quality SanDisk 64GB 120 Mb/s cards. Here’s an example on Amazon (however you can purchase them on your

favorite online distributor).

**microSD adapter:** You’ll need to purchase a microSD to USB adapter(Card Reader) so you can flash the memory card from your laptop.



Once you have the hardware ready, you’ll need to flash a fresh copy of the Raspbian Buster operating system to the microSD card.

1.Head on over to the official raspbian website, and start your download. I recommend the “Raspbian Buster with Desktop and recommended software”.

Setp the Raspberry Pi 4B Operating System

Raspberry Pi OS with desktop and recommended software

Release date: January 11th 2021

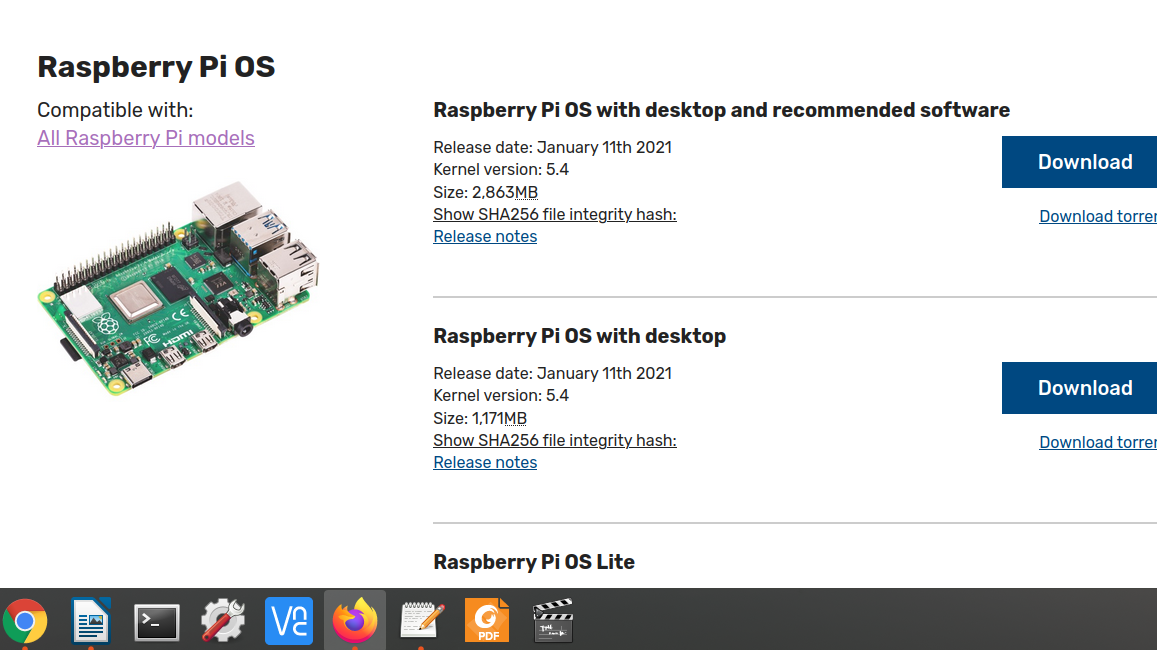
Kernel version: 5.4

Size: 2,863MB

Link:

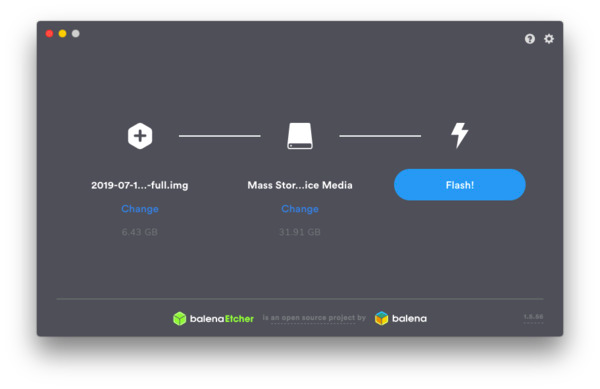
[https://www.raspberrypi.org/software/operating-systems/#raspberry-pi-os-32-bit](https://www.raspberrypi.org/software/operating-systems/" \l "raspberry-pi-os-32-bit).

Folow below figure 1.



**Figure:1**.Download Raspbian Buster for your Raspberry Pi and OpenCV 4

2.Download [Balena Etcher](https://www.balena.io/etcher/) — software for flashing memory cards. It works on every major OS. Use Etcher to flash BusterOS to your memory card. Folow below figure 2.

**Figure 2**:Flash Raspbian Buster with Etcher.

From there you can move on to the rest of the OpenCV Packages on Raspiberry pi

### Step #1: Expand filesystem and reclaim space

For the remainder of this tutorial I’ll be making the following assumptions:

1. You are working with a brand new, fresh install of Raspbian Buster (see the previous section to learn how to flash Buster to your microSD).
2. You are comfortable with the command line and Unix environments.
3. You have an SSH Pi. Alternatively, you could use a keyboard + mouse + screen.

Go ahead and insert your microSD into your Raspberry Pi and boot it up with a screen attached.

Once booted, configure your WiFi/ethernet settings to connect to the internet (you’ll need an internet connection to download and install required packages for OpenCV).

From there you can [use SSH](https://pyimagesearch.com/2019/07/01/remote-development-on-the-raspberry-pi-or-amazon-ec2/) as I have done, or go ahead and open a terminal.

The first step is to run,

raspi-config

and expand your filesystem:

As explained here, the physical RAM chip is used both by the CPU and the GPU. On a Raspberry Pi 2 or 3 default is 64 Mbyte allocated for the GPU. This can be somewhat small for vision projects. To increase the amount of memory for the GPU, use the following command. On a Raspberry Pi 4, there is 128 Mbyte given to the GPU. It is not necessary to change this at first.

**$ sudo raspi-config**

**Follow the next steps to modify the amount of GPU memory.**

and then select the “7 Advanced Options” menu item:

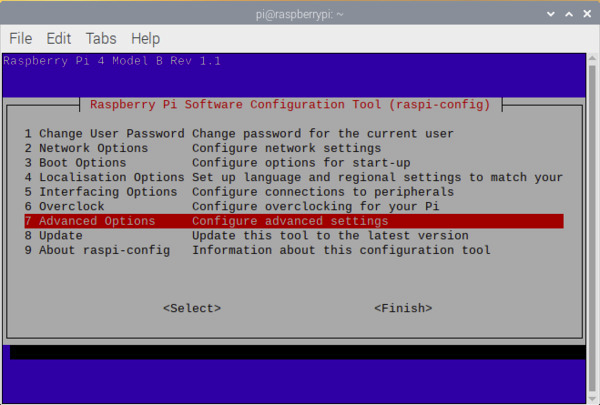


Figure 3: The `raspi-config` configuration screen for Raspbian Buster. Select `7 Advanced Options` so that we can expand our filesystem.

Followed by selecting “A1 Expand filesystem”:

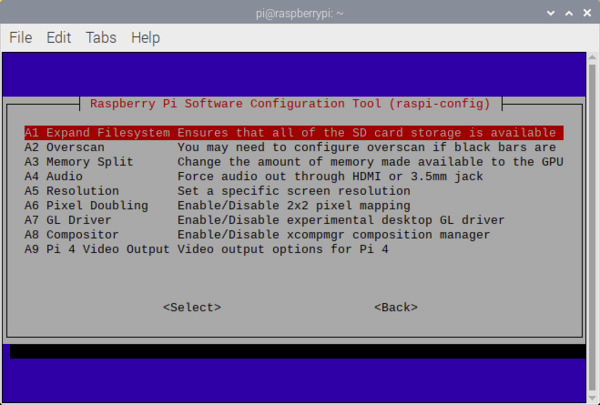


Figure 4: The A1 Expand Filesystem` menu item allows you to expand the filesystem on your microSD card containing the Raspberry Pi Buster operating system. Then we can proceed to install OpenCV 4.

\* Once prompted, you should select the first option, “A1 Expand File System”, hit

enter

  on your keyboard, arrow down to the “<Finish>” button, and then reboot your Pi — you may be prompted to reboot, but if you aren’t you can execute on the terminal

**$ sudo reboot**

After successful installing the Raspbian operating system, it is time to update and upgrade your system with the next commands in the terminal.



**$ sudo apt-get update**

**$ sudo apt-get upgrade**

We have removed the following software packages: BlueJ, Claws Mail, Greenfoot, LibreOffice, Mathematica, Minecraft, Node-RED, Scratch, Strach2, Sense HAT, SmartSim, and Sonic Pi. This action frees about 2.5 Gbytes. However, leave at any time *Thonny* on your system. In the beginning, we deleted it also in the urge for more memory. But somehow too many Python packages are then removed and OpenCV could no longer generate a proper Python library.

Once all the unnecessary packages are removed, two last instructions finish this action.

**$ sudo apt-get clean**

**$ sudo apt-get autoremove**

Check the python version on the Terminal

**$ python3**

**Python Recommended Version: 3.7.3(Default)**

### **Step #2: Compile OpenCV 4.2.0 from source(Begining from Scratch)**

### **Dependencies.**

The OpenCV software uses other third-party software libraries. These have to be installed first. Perhaps there are already installed, but that doesn't matter. Latest versions are always kept by the installation procedure. Install line for line the next packages.

**$ sudo apt-get install build-essential cmake git unzip pkg-config**

**$ sudo apt-get install libjpeg-dev libpng-dev libtiff-dev**

**$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev**

**$ sudo apt-get install libgtk2.0-dev libcanberra-gtk\***

**$ sudo apt-get install libxvidcore-dev libx264-dev libgtk-3-dev**

**$ sudo apt-get install python3-dev python3-numpy python3-pip**

**$ sudo apt-get install python-dev python-numpy**

**$ sudo apt-get install libtbb2 libtbb-dev libdc1394-22-dev**

**$ sudo apt-get install libv4l-dev v4l-utils**

**$ sudo apt-get install libjasper-dev libopenblas-dev libatlas-base-dev libblas-dev**

**$ sudo apt-get install liblapack-dev gfortran**

**$ sudo apt-get install gcc-arm\***

**$ sudo apt-get install protobuf-compiler**

### **Download OpenCV:**

When all third-party software is installed, OpenCV itself can be downloaded. There are two packages needed; the basic version and the additional contributions. Check before downloading the latest version at <https://opencv.org/releases/>. If necessary, change the names of the zip files according to the latest version. After downloading, you can unzip the files. Please be aware of line wrapping in the text boxes. The two commands are starting with wget and ending with zip.

**$ cd ~**

**$ wget -O opencv.zip https://github.com/opencv/opencv/archive/4.2.0.zip**

**$ wget -O opencv\_contrib.zip https://github.com/opencv/opencv\_contrib/archive/4.2.0.zip**

**$ unzip opencv.zip**

**$ unzip opencv\_contrib.zip**

The next step is some administration. Rename your directories with more convenient names like opencv and opencv\_contrib. This makes live later on easier.

**$ mv opencv-4.2.0 opencv**

**$ mv opencv\_contrib-4.2.0 opencv\_contrib**

There is only one dependency for Python with OpenCV in a virtual environment and that is Numpy. Although you have previously installed this package, you must also install it within the environment, otherwise, CMake cannot compile

# with sudo!!!!

**$ sudo pip3 install numpy**

### **Build Make:**

Before we begin with the actual build of the library, there is one small step to go. You have to make a directory where all the build files can be located.

**$ cd ~/opencv/**

**$ mkdir build**

**$ cd build**

Now it is time for an important step. Here you tell CMake what, where and how to make OpenCV on your Raspberry. There are many flags involved. The most you will recognize. We save space by excluding any (Python) examples or tests.

There are only bare spaces before the -D flags, not tabs. Keep also a single space between the -D and the argument. Otherwise, CMake will misinterpreter the command. By the way, the two last dots are no typo. It tells CMake where it can find its CMakeLists.txt (the large recipe file); one directory up.

$ cmake -D CMAKE\_BUILD\_TYPE=RELEASE \

        -D CMAKE\_INSTALL\_PREFIX=/usr/local \

        -D OPENCV\_EXTRA\_MODULES\_PATH=~/opencv\_contrib/modules \

        -D ENABLE\_NEON=ON \

        -D ENABLE\_VFPV3=ON \

        -D WITH\_OPENMP=ON \

        -D BUILD\_TIFF=ON \

        -D WITH\_FFMPEG=ON \

        -D WITH\_TBB=ON \

        -D BUILD\_TBB=ON \

        -D BUILD\_TESTS=OFF \

        -D WITH\_EIGEN=OFF \

        -D WITH\_V4L=ON \

        -D WITH\_LIBV4L=ON \

        -D WITH\_QT=OFF \

        -D WITH\_VTK=OFF \

        -D OPENCV\_EXTRA\_EXE\_LINKER\_FLAGS=-latomic \

        -D OPENCV\_ENABLE\_NONFREE=ON \

        -D INSTALL\_C\_EXAMPLES=OFF \

        -D INSTALL\_PYTHON\_EXAMPLES=OFF \

        -D BUILD\_NEW\_PYTHON\_SUPPORT=ON \

        -D BUILD\_opencv\_python3=TRUE \

        -D OPENCV\_GENERATE\_PKGCONFIG=ON \

        -D BUILD\_EXAMPLES=OFF ..

If everything went well, CMake generates a report that looks something like this (for readability purposes we omitted most lines). Very crucial are the Python sections. If these are missing, OpenCV will not install proper Python libraries. In that case, usually CMake could not find the Python folders, or in the case of a virtual environment, a dependency such as numpy is probably not installed within the environment. We are experiencing these issues when removing too many packages from the operating system, as mentioned earlier. NEON and VFPV3 support must also be enabled. Certainly if you intend to build our deep learning examples. Check if v4l/v4l2 is available if your planning to use the Raspicam.

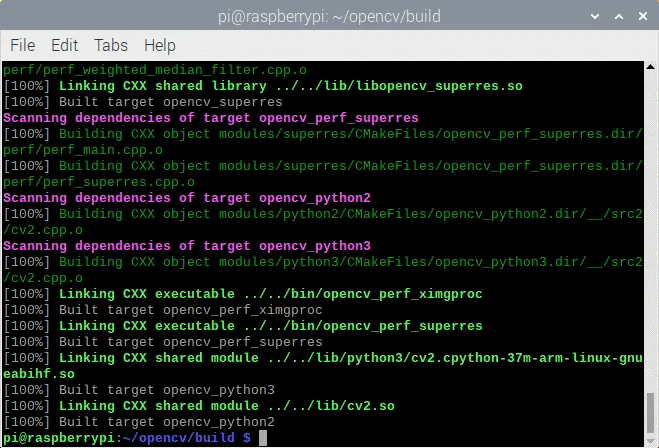
### Make.

Now everything is ready for the great build. This takes a lot of time. Be very patient is the only advice here. Don't be surprised if at 99% your build seems to be crashed. That is 'normal' behaviour. Even when your CPU Usage Monitor gives very low ratings like 7%. In reality, your CPU is working so hard it has not enough time to update these usage numbers correctly.

You can speed things up with four cores working simultaneously (make -j4). On a Raspberry Pi 4, it takes just over an hour to build the whole library. Sometimes the system crashes for no apparent reason at all at 99% or even 100%. In that case, restart all over again, as explained at the end of this page, and rebuild with make -j1.

Probably you get a whole bunch of warnings during the make. Don't pay to much attention to it. They are generated by subtle differences in template overload functions due to little version differences. So take coffee and a good book for reading, and start building with the next command.

**$ sudo make -j4**



Now to complete, install all the generated packages to the database of your system with the next commands.

**$ sudo make install**

**$ sudo ldconfig**

**# cleaning (frees 300 KB)**

**$ make clean**

**$ sudo apt-get update**

After open cv installed we can check the versions on the terminal like below

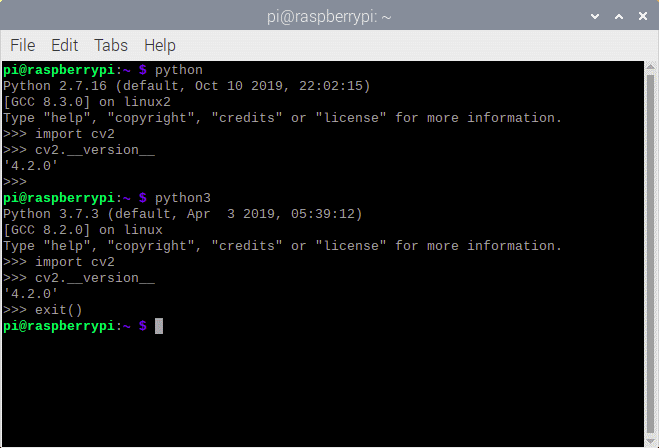


Figure:python,opencv versions checking

### **Step #3:**

**Installing other Packages as well for face Recognition**

**1.$sudo pip3 install numpy**

**2.$sudo pip3 install scipy**

**3.$sudo pip3 install**

**4.$sudo pip3 matplotlib**

**5.$sudo** **pip3 install face-recognition**

**6.sudo pip3 install dlib**

**7.****$sudo pip3 install acapture**

**8.$sudo pip3 install tensorflow**

**9.$sudo pip3 install keras**

### **Step #3:**

**The Realtime\_Face\_recogntion\_DAIOTS folder contains 6 files 1.Dataset: Need to create dataset folder and it contains and creates person wise folders like mohan\_1,kalyan\_2,Ravi\_3,Raju\_4 etc.**

2.videocaptureasync.py

3.welcome.py After running this file voice will be activated(Attendance successful) when any known person entering into the camera

4.welcome1.py

After running this file voice will be activated(intruder) when any unknown person entering into the camera

5.Face\_Train.py:After running this file to Generate and Save face encodings using numpy

$python3 Face\_Train.py

6.Face\_Main.py:After running the file compare the face encodings with database and Faces will be recognized

$python3 Face\_Daiots\_Main.py Setups for Raspberry Pi

To use Raspberry Pi camera or USB camera module

#cap = cv2.VideoCapture(0).start() //For webcam, comment it if using Raspberry Pi Camera module cap = cv2.VideoCapture(1).start() //For Raspberry Pi Camera module, comment it if using webcam

To use USB webcam

cap = cv2.VideoCapture(-1).start() //For webcam #cap = cv2.VideoCapture(usePiCamera=True).start() //For Raspberry Pi Camera

Setups for any other PC

To use default webcam

cap = cv2.VideoCapture.start()

To use external webcam

cap = cv2.VideoCapture.start().start() #If system has more than one webcam use the 'src' number accordingly