Installing OpenCV 4.1.0 on Raspberry Pi Zero, 3 B/B+

- I used this article (May 2019) to install OpenCV 4.1.0. Basically need to download the OpenCV sourcecode and build it on the Rpi, making use of the hardware features of the Rpi, then install.
- https://medium.com/@aadeshshah/pre-installed-and-pre-configured-raspbian-with-opencv-4-1-0-for-raspberry-pi-3-model-b-b-9c307b9a993a
- Some notes:
  - In running the command “sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev”, take note libv4l-dev not 41!
  - Remember to restore back the original swapsize after installation.
  - For Rpi Zero just use “Make” instead of “Make –j4” since the processor is only single core.
  - It takes more than 12 hours to build the codes for Rpi Zero, we can pause whenever we want by pressing Ctrl C and continue from where we stop by issuing the “Make” command.

Increasing Raspberry Pi Camera Frame Rate with Python and OpenCV

- Use *PiCamera* libraries, native Raspberry Pi camera driver, see https://picamera.readthedocs.io/en/release-1.13/index.html
- Use threading, and make the capturing of camera frame non-blocking by running it in a thread.
- See https://www.pyimagesearch.com/2015/12/28/increasing-raspberry-pi-fps-with-python-and-opencv/

Experiment: Powering Up Raspberry Pi Zero-W With Bench Top Power Supply

- Condition: Headless, e.g. no display and keyboard connected. Booting up with desktop GUI. With 5MP camera module activated, and running *RealVNC* server and OpenCV2 routines.
- Supply: 4.95V.
- Current consumption: 0.15-0.27A average.
- Recommend a power source that can supply a minimum of 0.35A, considering that we are going to add two white LEDs which would draw 20mA each.

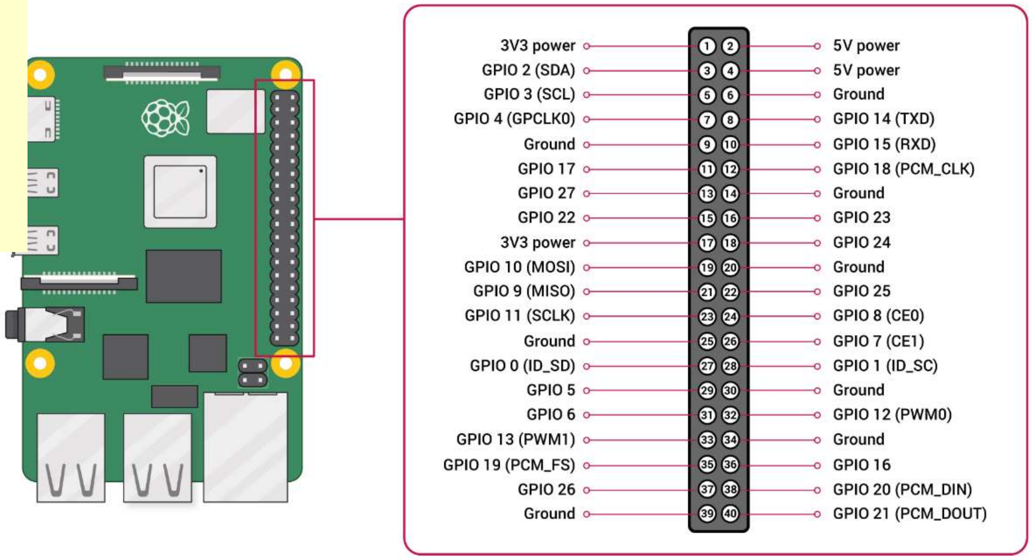
Experiment: Powering Up Raspberry Pi 3B With Bench Top Power Supply

- Condition: Headless, e.g. no display and keyboard connected. Booting up with desktop GUI. With 5MP camera module activated, and running *RealVNC* server and OpenCV2 routines.
- Supply: 5.05V.
- Current consumption: 0.50-0.70A average.
- Recommend a power source that can supply a minimum of 1.00A, considering that we are going to add two white LEDs which would draw 20mA each.

Using Raspberry Pi GPIO Pins

- From Rpi B+ onwards (to current Rpi 4), all boards has a 40-pins GPIO header which is mapped to GPIO0 port in the processor.
- For python we use the *gpiozero* library to access the pins.
- See https://www.raspberrypi.org/documentation/usage/gpio/
- For software documentation using python see https://www.raspberrypi.org/documentation/usage/gpio/python/README.md and https://gpiozero.readthedocs.io/en/stable/recipes.html

Note: in the terminal,
We can also run the command *pinout* to list the GPIO pins Assignment for the board.



3V3 power 1 2 5V power
GPIO 2 (SDA) 3 4 5V power
GPIO 3 (SCL) 5 6 Ground
GPIO 4 (GPCLK0) 7 8 GPIO 14 (TXD)
Ground 9 10 GPIO 15 (RXD)
GPIO 17 11 12 GPIO 18 (PCM_CLK)
GPIO 27 13 14 Ground
GPIO 22 15 16 GPIO 23
3V3 power 17 18 GPIO 24
GPIO 10 (MOSI) 19 20 Ground
GPIO 9 (MISO) 21 22 GPIO 25
GPIO 11 (SCLK) 23 24 GPIO 8 (CE0)
Ground 25 26 GPIO 7 (CE1)
GPIO 0 (ID_SD) 27 28 GPIO 1 (ID_SC)
GPIO 5 29 30 Ground
GPIO 6 31 32 GPIO 12 (PWM0)
GPIO 13 (PWM1) 33 34 Ground
GPIO 19 (PCM_FS) 35 36 GPIO 16
GPIO 26 37 38 GPIO 20 (PCM_DIN)
Ground 39 40 GPIO 21 (PCM_DOUT)

Auto start Applications in Raspbian Desktop (1 of 2)

- There are a few ways to auto start software in Debian Linux upon boot up. Here I want to start my python application after the windows desktop loaded.
- One of the simplest is to edit the */etc/rc.local* file using *sudo nano* command and insert the command line to execute the application. However I discovered this method will run the application in the */etc/rc.local* file before the desktop is loaded. So if the application requires desktop support like creating windows etc., then this method will not work.
- After searching online, I found a youtube video which shows how to start an application on boot up **after** the desktop is loaded. https://www.youtube.com/watch?v=zeB5TimDNj8
- This method edits a file called *autostart* in the folder */etc/xdg/lxsession/LXDE-pi* , where the following line is added as the last entry to the file: @/usr/bin/python3 *your_python_file_with_path*

Note: can use the text editor *nano* to edit the autostart file: *sudo nano autostart*

Auto start Applications in Raspbian Desktop (2 of 2)

- Important: Whether we edit the file /etc/rc.local or autostart, it is important to bear in mind that the python codes must be error free.
- Else during the interpretation the python application will abort, and the program will not start up!

Python Codes to Shutdown RPi from External Switch (1 of 2)

- Let’s assume the filename for the codes below is “ShutDownButton.py”.

```
from gpiozero import Button
from signal import pause
import os

def shutdown():
        os.system("sudo poweroff")

psw = Button(23, hold_time=2)  # Set GPIO23 as button switch, active low, hold time
                               # 2 seconds.
psw.when_held = shutdown
pause()                        # Run in the background.
```

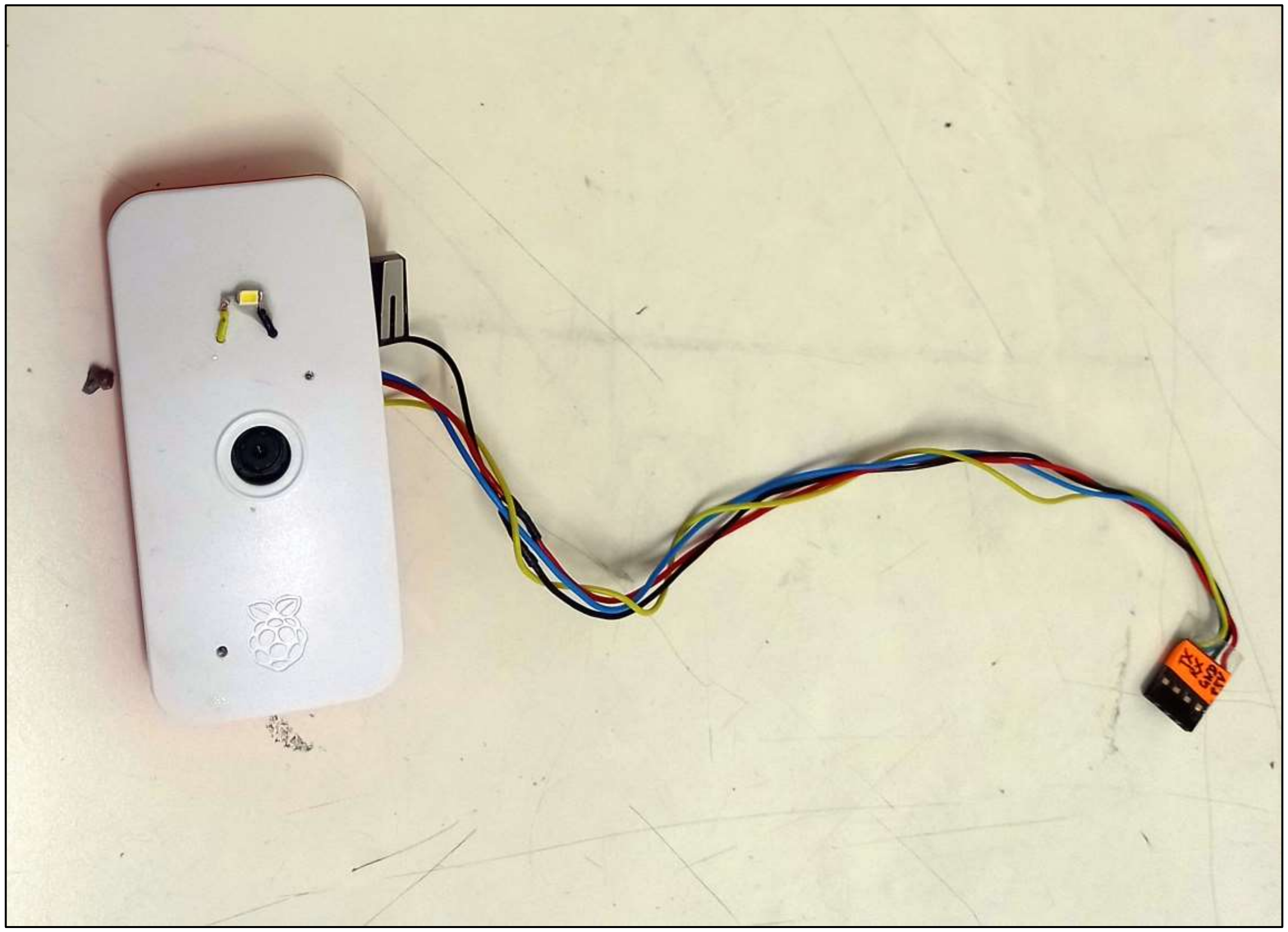
Python Codes to Shutdown Rpi from External Switch (2 of 2)

- We should then run the codes on boot up.
- This can be done by adding the following line in the file */etc/rc.local* before the *exit 0* line:
- Python3 /home/pi/Python/ShutDownButton.py &
- The ampersand behind the instruction is to make the codes above to be non-blocking, else RPi will hang.
- A screen as seen from the text editor Nano is shown below:

Remote Monitoring with VNC

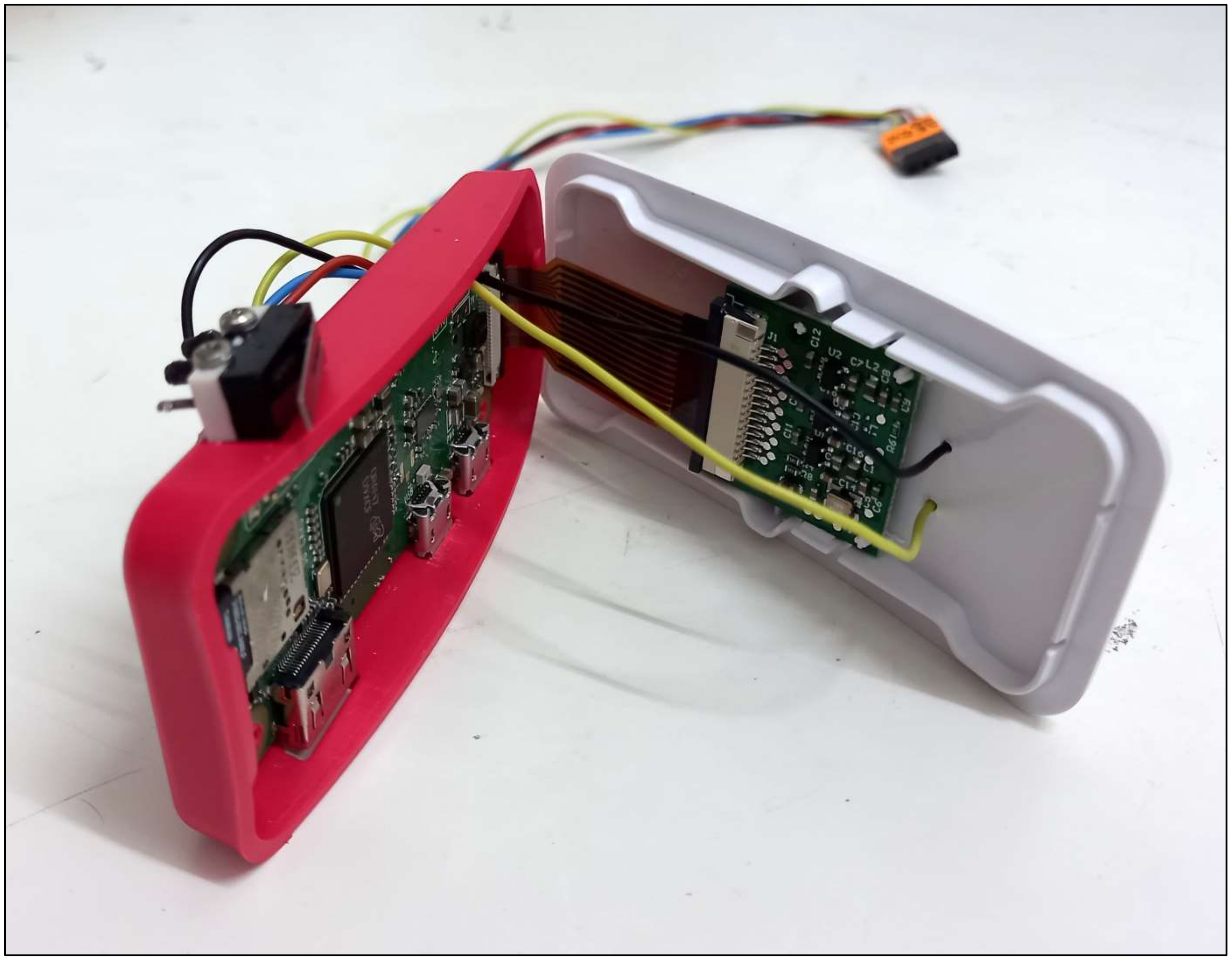
- Enable VNC server on Raspberry Pi and start the server at the start of desktop. Also run the Raspberry Pi in ‘headless’ mode, e.g. no display and keyboard and mouse.
- The VNC server will create a virtual display on the Pi RAM memory.
- This way we can connect to the Raspberry Pi using WiFi and monitor the applications on the Pi in real time.

Test: Raspberry Pi Zero W with Camera and Enclosure, with LED and External Shutdown Switch (1 of 2)



LED – GPIO12
Shutdown – GPIO23
TX – GPIO14
RX – GPIO15

Test 1: Raspberry Pi Zero W with Camera and Enclosure, with LED and External Shutdown Switch (2 of 2)



Test 2: Raspberry Pi 3A+ with Camera and Enclosure

Progress – 12 May 2021

- Settle with using Raspberry Pi 3A+ as the single-board computer, the Pi Zero W is simply too slow and cannot handle image resolution of QVGA.
- The current software version 0.39, uses Open CV2 V4.1.0 libraries, contains 4 modules:
  - Camera driver using *picamera* with threading, or standard Pi camera driver.
  - Serial port routine using *Pyserial* with threading.
  - Face recognition (Haar Cascade) and tracking routines.
  - Sparse optical flow routines using Lucas-Kanade approach.
- Thus far the face recognition and optical flow routines are working properly after a few weeks of efforts and debugging. For optical flow initially I faced the issue of the *calcOpticalFlowPyrLK()* function triggering exception every now and then when the robot move or vibrate.
- Upon investigation the exception is due to the feature points set return by the *calcOpticalFlowPyrLK() or goodFeaturesToTrack()* functions return empty sets (of points) because the algorithm cannot find any features or corners to track in the image. The issue is solved by pausing the optical flow function whenever we cannot find any features or corners.

Progress – 12 May 2021 Cont…

- At the moment only 1 image processing algorithm can be executed at any time, e.g. either face recognition with tracking, or sparse optical flow.
- Also from experiments, the sparse optical flow is not very good at detecting motion along y axis, although x-axis is ok. This is due to the robot platform which vibrates, and also the head vibrates.
- Effect to improve the robot dynamics and reduce the vibration will be carried out in the coming weeks.