

Raspberry Pi - Instalacja biblioteki OpenCV

1.Wgrywanie systemu.

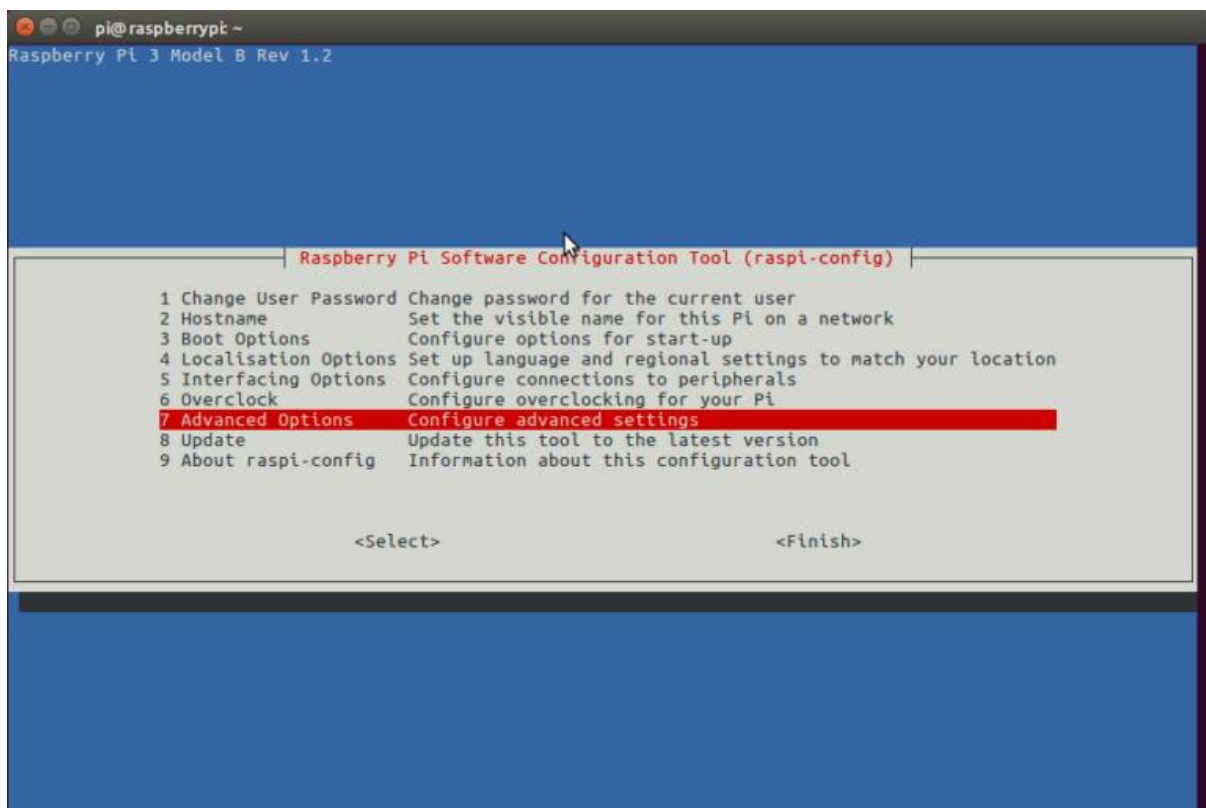
Najlepiej użyć Raspbiana, najmniej obciąża Raspberry Pi. Będziemy używali Pythona3, którego większość wersji raspbiana ma już preinstalowanego.

2.Instalacja odpowiednich narzędzi, pakietów i bibliotek.

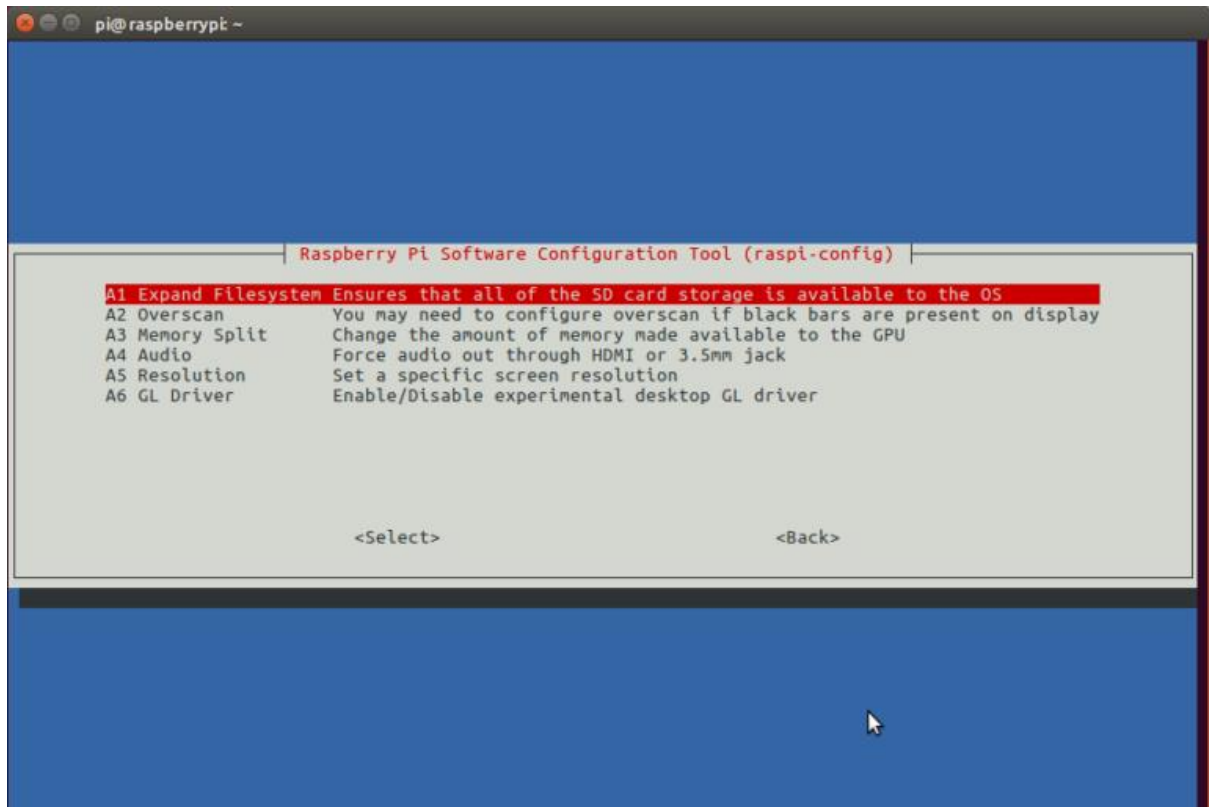
(poszczególne etapy w kroku drugim zajmują całkiem dużo czasu, instalacje niektórych bibliotek może potrwać nawet parę godzin)

a) Zmieńmy rozszerzenie systemu plików, aby wykorzystać całe dostępne miejsce na naszej karcie micro-SD

- wpisujemy: ***sudo raspi-config***
- wybieramy Advanced Options(2.1)



- następnie wybieramy Expand Filesystem(2.2)



2.2

zatwierdzamy enterem i rebootujemy raspberry komendą: **sudo reboot**

b) Instalacja niezbędnych narzędzi, pakietów i bibliotek seria komend.

- **sudo apt-get update && sudo apt-get upgrade**
- **sudo apt-get install build-essential cmake pkg-config**
- **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 libgtk2.0-dev libgtk-3-dev**
- **sudo apt-get install libatlas-base-dev gfortran**
- **sudo apt-get install python3-dev**

c) Ściągnięcie biblioteki OpenCV (wersje `opencv` oraz `opencv_contrib` muszą się zgadzać)

- `cd ~`
- `wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip`
- `unzip opencv.zip`
- `wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.1.0.zip`
- `unzip opencv_contrib.zip`

d) Instalacja i stworzenie wirtualnego środowiska.

Dobłą praktyką pisania kodu w Pythonie jest wykorzystywanie wirtualnych środowisk do projektów, aby nie mieszały się zależności z różnych projektów.

- `sudo pip3 install virtualenv virtualenvwrapper`
- `sudo rm -rf ~/.cache/pip`

następnie na koniec pliku `~/.profile` należy dopisać następujące linie:

```
export WORKON_HOME=$HOME/.virtualenvs
```

```
source /usr/local/bin/virtualenvwrapper.sh
```

np. za pomocą nast. poleceń:

- `echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile`
- `echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile`

następnie aby wymusić ponowne załadowanie .pliku `profile` używamy:

- `source ~/.profile`

teraz tworzymy wirtualne środowisko:

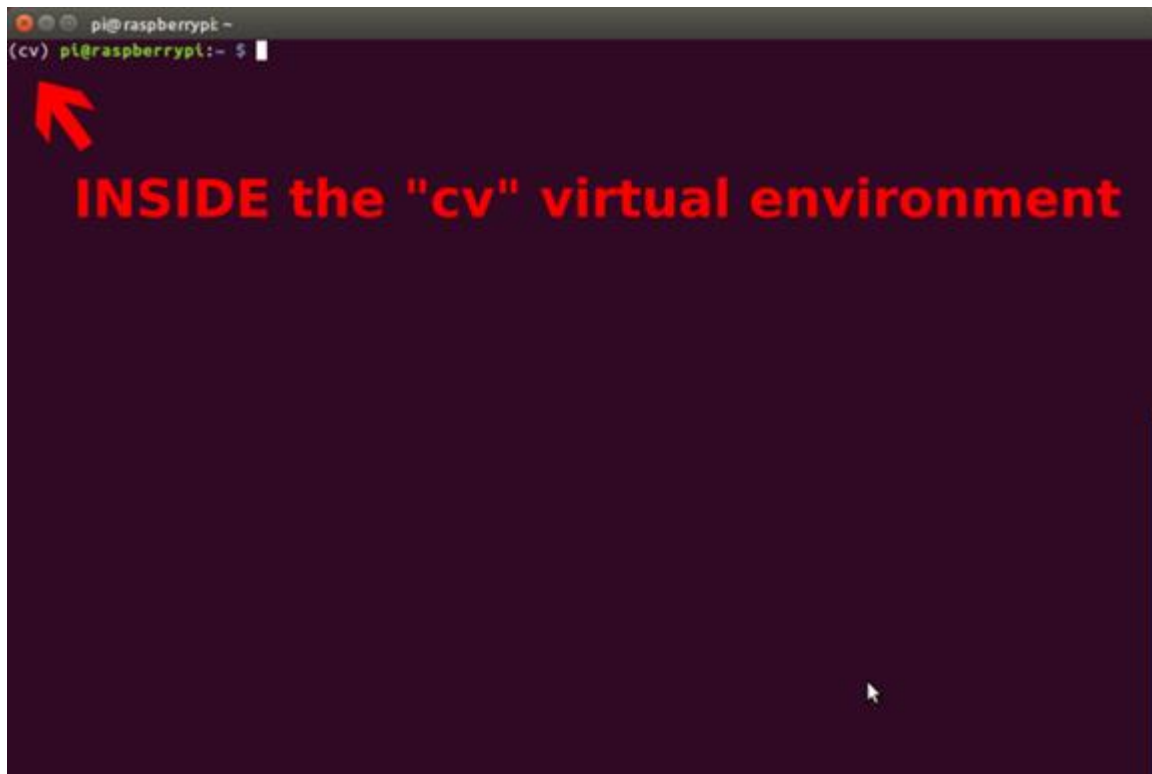
- `mkvirtualenv cv -p python3`

e) Jesteśmy gotowi do używania wirtualnego środowiska przełączamy się na nie za pomocą poleceń.

- `source ~/.profile`
- `workon cv`

f) Instalacja kolejnych niezbędnych bibliotek(kroki wykonujemy wewnątrz wirtualnego środowiska).

upewniamy się, że jesteśmy wewnątrz wirtualnego środowiska ([2.3](#))



2.3

następnie wywołujemy polecenia:

- `pip3 install numpy`

kompilacja i instalacja openCV:

- `cd ~/opencv-3.3.0/`
- `mkdir build`
- `cd build`
- `cmake -D CMAKE_BUILD_TYPE=RELEASE \`
`-D CMAKE_INSTALL_PREFIX=/usr/local \`
`-D ENABLE_PRECOMPILED_HEADERS=OFF \`
`-D INSTALL_PYTHON_EXAMPLES=ON \`

```
-D OPENCV_EXTRA_MODULES_PATH=~/.opencv_contrib-3.3.0/modules \
-D BUILD_EXAMPLES=ON ..
```

Należy koniecznie pamiętać o opcji `ENABLE_PRECOMPILED_HEADERS=OFF`, bez niej podczas budowania projektu ujrzymy masę warningów i errorów o brakach bibliotek i plików.

Następnie możemy przejść do kompilacji. Wykonujemy to komendą:

- **make (-j2/j4)**

Opcjonalne parametry deklarują ilość rdzeni na, których zostanie wykonana kompilacja. Skróci to znacznie czas wykonania zadania, lecz może doprowadzić do jej niepoprawnego zakończenia.

Na początku sugeruję użycie komendy z parametrem **-j4**, jeśli kompilacja jednak nie skończy się z pozytywnym wynikiem należy uruchomić ją na jednym rdzeniu:

- **make clean**
- **make**

Po zakończeniu kompilacji instalujemy openCV:

- **sudo make install**
- **sudo ldconfig**

Następnie aby ułatwić importowanie biblioteki do projektu warto zmienić jej nazwę, aktualną nazwę znajdziemy poleceniem (3.5 odnosi się do wersji pythona, której używamy być może należy zmienić ten parametr) :

- **ls -l /usr/local/lib/python3.5/site-packages/**

nazwa może wyglądać mniej więcej tak:

cv2.cpython-35m-arm-linux-gnueabi.so chcemy zamienić ją na *cv2.so*

Zmienić nazwę możemy w nast. sposób:

- **cd /usr/local/lib/python3.5/site-packages/**
- **sudo mv nazwa_pliku cv2.so**

Następnie musimy stworzyć dowiązania symboliczne naszych wiązań biblioteki openCV do wirtualnego środowiska za pomocą poleceń:

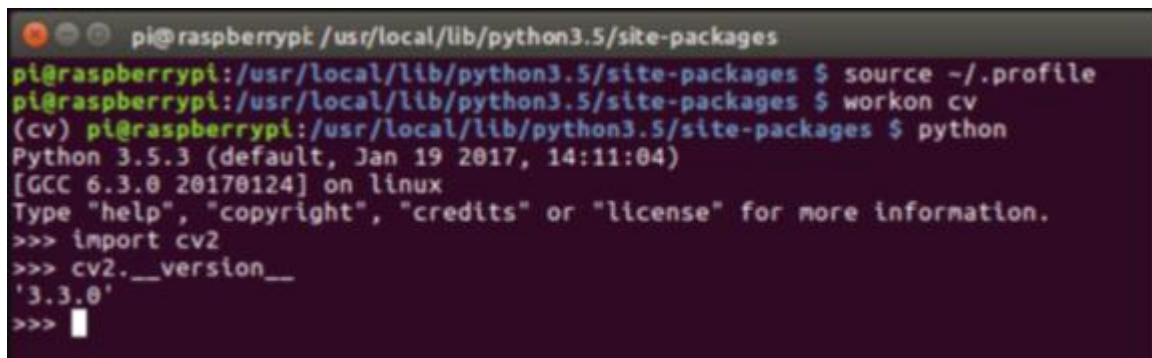
- **cd ~/.virtualenvs/cv/lib/python3.5/site-packages/**
- **ln -s /usr/local/lib/python3.5/site-packages/cv2.so cv2.so**

g) Testowanie czy poprawnie zainstalowaliśmy bibliotekę (ciągle jesteśmy w stworzonym przez nas wirtualnym środowisku).

wpisujemy:

- **python3**
- **import cv2**
- **cv2.__version__**

powinna nam się ukazać wersja openCV jak na (2.4)

A terminal window on a Raspberry Pi showing the process of setting up a virtual environment for OpenCV. The user is in the directory /usr/local/lib/python3.5/site-packages. They run 'source ~/.profile' and 'workon cv' to activate the environment. Then they run 'python' to start the Python interpreter. The interpreter shows it's Python 3.5.3 on Linux. The user then enters 'import cv2' and 'cv2.__version__', which returns '3.3.0'.

```
pi@raspberrypi: /usr/local/lib/python3.5/site-packages
pi@raspberrypi:/usr/local/lib/python3.5/site-packages $ source ~/.profile
pi@raspberrypi:/usr/local/lib/python3.5/site-packages $ workon cv
(cv) pi@raspberrypi:/usr/local/lib/python3.5/site-packages $ python
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.3.0'
>>> 
```

2.4

sources:

<https://www.pyimagesearch.com/2016/04/18/install-guide-raspberry-pi-3-raspbian-jessie-opencv-3/>

<https://docs.opencv.org/>