Table des matières

[Etape 0 : Mise à jour du <<firmware >> 2](#_Toc422122262)

[Etape 1: Installation des outils de développeur et de paquets 2](#_Toc422122263)

[Etape 2 : Installation des paquets nécessaires 2](#_Toc422122264)

[Pour la gestion des images de formats : JPEG, PNG, TIFF,. 2](#_Toc422122265)

[La librairie d’outils de développement GTK pour les interfaces graphiques (fenêtres,…). La librairie « highgui  » d’Opencv nécessite cette librairie. 2](#_Toc422122266)

[Pour la gestion des fichiers vidéos 3](#_Toc422122267)

[Librairie d’optimisation des opérations effectuées par la librairie Opencv 3](#_Toc422122268)

[Etape 3 : Installation du gestionnaire des paquets python. 3](#_Toc422122269)

[Etape 4 : Installation des outils nécessaires à la construction de l’environnement virtuel pour les projets python 3](#_Toc422122270)

[Virtualenv and VirtualenvWrapper 3](#_Toc422122271)

[Mise à jour du répertoire ~/.profile 3](#_Toc422122272)

[Recharger le fichier ./profile 3](#_Toc422122273)

[Création d’un environnement virtuel pour la vision par ordinateur (Computer vision) 3](#_Toc422122274)

[Etape 5 : Installation des outils de développement pour Python -2.7 (le numéro de version est à mettre à jour) 4](#_Toc422122275)

[Installation de python 4](#_Toc422122276)

[Installation de l’outil numpy pour la gestion de l’affichage des images en plusieurs dimensions 4](#_Toc422122277)

[Etape 6 : Installation d’Opencv 4](#_Toc422122278)

[Installation du répertoire Opencv 4](#_Toc422122279)

[Configuration du « build » 4](#_Toc422122280)

[Compilation 4](#_Toc422122281)

[Installation 5](#_Toc422122282)

[Etape 7 : Création d’un lien symbolique entre Opencv et l’outil « cv » (pour la création de l’environnement virtuel pour les projets Python) 5](#_Toc422122283)

[Etape 8 : Test de l’installation d’Opencv et de Python 5](#_Toc422122284)

[Etape 1 : Activer la camera 6](#_Toc422122285)

[Etape 2 : Tester la camera 7](#_Toc422122286)

[Etape 3 : Installation du module picamera 7](#_Toc422122287)

[Etape 3 : Programme de test de la camera 8](#_Toc422122288)

Installation d’Opencv et de python

Pour réaliser ce tutoriel je me suis totalement inspiré de celui trouvé sur ce [site](http://www.pyimagesearch.com/2015/02/23/install-opencv-and-python-on-your-raspberry-pi-2-and-b/).

Pour réaliser l’installation de la librairie Opencv et de python j’ai donc procédé aux étapes suivantes :

# Etape 0 : Mise à jour du <<firmware >>

$ sudo apt-get update

$ sudo apt-get upgrade

$ sudo rpi-update$ sudo apt-get update

$ sudo apt-get upgrade

$ sudo rpi-update

# Etape 1: Installation des outils de développeur et de paquets

Ça consiste à l’installation des outils « build-essential » et de « pkg-config » qui sont par défaut déjà installés. Mais pour être sûr de leur installation on les ajoute à notre comme « apt-get ».

$ sudo apt-get install build-essential cmake pkg-config

# Etape 2 : Installation des paquets nécessaires

Pour la gestion des images de formats : JPEG, PNG, TIFF,..

$sudo apt-get install libjpeg8-dev libtiff4-dev libjasper-dev libpng12-dev

### La librairie d’outils de développement GTK pour les interfaces graphiques (fenêtres,…). La librairie « highgui  » d’Opencv nécessite cette librairie.

$ sudo apt-get install libgtk2.0-dev

Pour la gestion des fichiers vidéos.

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

### Librairie d’optimisation des opérations effectuées par la librairie Opencv

$ sudo apt-get install libatlas-base-dev gfortran

# Etape 3 : Installation du gestionnaire des paquets python.

$ wget https://bootstrap.pypa.io/get-pip.py

$ sudo python get-pip.py

# Etape 4 : Installation des outils nécessaires à la construction de l’environnement virtuel pour les projets python

### Virtualenv and VirtualenvWrapper

$ sudo pip install virtualenv virtualenvwrapper

### Mise à jour du répertoire ~/.profile

# virtualenv and virtualenvwrapper

export WORKON\_HOME=$HOME/.virtualenvs

source /usr/local/bin/virtualenvwrapper.sh

### Recharger le fichier ./profile

|  |  |
| --- | --- |
|  | $ source ~/.profile |

### Création d’un environnement virtuel pour la vision par ordinateur (Computer vision)

|  |  |
| --- | --- |
| $ mkvirtualenv cv | mkvirtualenv cv |

# Etape 5 : Installation des outils de développement pour Python -2.7 (le numéro de version est à mettre à jour)

Installation de python

|  |  |
| --- | --- |
|  | $ sudo apt-get install python2.7-dev |

### Installation de l’outil numpy pour la gestion de l’affichage des images en plusieurs dimensions

|  |  |
| --- | --- |
|  | $ pip install numpy |

# Etape 6 : Installation d’Opencv

### Installation du répertoire Opencv

|  |  |
| --- | --- |
|  | $ wget -O opencv-2.4.10.zip http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.10/opencv-2.4.10.zip/download  $ unzip opencv-2.4.10.zip  $ cd opencv-2.4.10 |

### Configuration du « build »

|  |
| --- |
| $ mkdir build  $ cd build  $ cmake -D CMAKE\_BUILD\_TYPE=RELEASE -D CMAKE\_INSTALL\_PREFIX=/usr/local -D BUILD\_NEW\_PYTHON\_SUPPORT=ON -D INSTALL\_C\_EXAMPLES=ON -D INSTALL\_PYTHON\_EXAMPLES=ON -D BUILD\_EXAMPLES=ON .. |

### Compilation

|  |  |
| --- | --- |
|  | $ make |

### Installation

$ sudo make install

$ sudo ldconfig

# Etape 7 : Création d’un lien symbolique entre Opencv et l’outil « cv » (pour la création de l’environnement virtuel pour les projets Python)

$ cd ~/.virtualenvs/cv/lib/python2.7/site-packages/

$ ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so

$ ln -s /usr/local/lib/python2.7/site-packages/cv.py cv.py

# Etape 8 : Test de l’installation d’Opencv et de Python

$ workon cv

$ python

>>> import cv2

>>> cv2.\_\_version\_\_

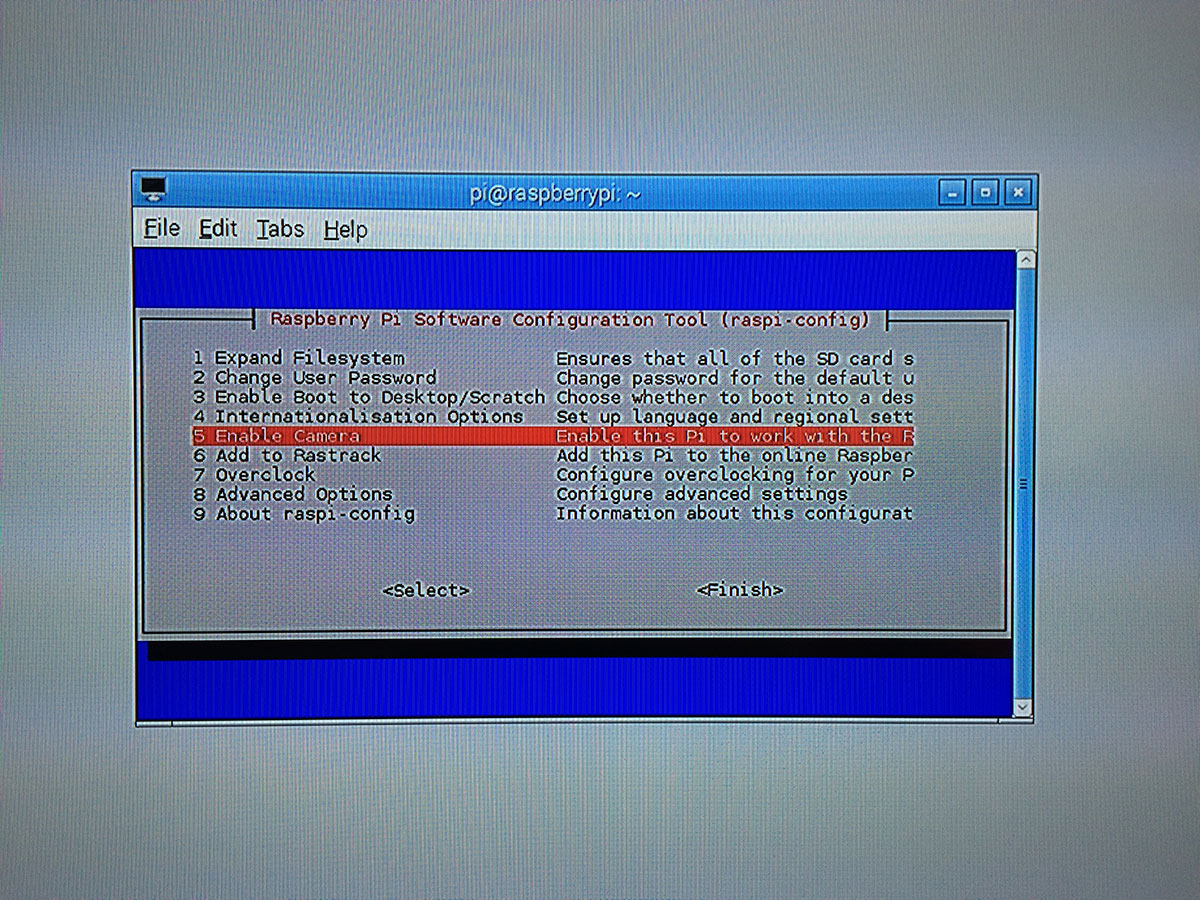
'2.4.10'

Accéder à la caméra

Après avoir installé la camera dans ce tutoriel nous utilisons une « Raspberry Pi camera ». Vous trouverez les instructions qui suivront depuis ce [site](http://www.pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/).

# Etape 1 : Activer la camera

$ sudo raspi-config

[](http://www.pyimagesearch.com/wp-content/uploads/2015/03/raspi_config.jpg)

# Etape 2 : Tester la camera

* Faire une capture Cette commande démarre le module caméra puis prend une capture et la stocke dans le fichier output.jpg

$ raspistill -o output.jpg

# Etape 3 : Installation du module picamera

Afin de pouvoir accéder aux flux vidéo de notre camera avec python il est nécessaire d’installer le module « picamera »

Avant d’installer ce module il est necessaire d’activer l’environnement virtuel ( cv , à faire à chaque fois avant l’ exécution des programmes python )

$ source ~/.profile

$ workon cv

Installation du module picamera

|  |  |
| --- | --- |
|  | $ pip install "picamera[array]" |

# Etape 3 : Programme de test de la camera

# import the necessary packages

from picamera.array import PiRGBArray

from picamera import PiCamera

import time

import cv2

# initialize the camera and grab a reference to the raw camera capture

camera = PiCamera()

camera.resolution = (640, 480)

camera.framerate = 32

rawCapture = PiRGBArray(camera, size=(640, 480))

# allow the camera to warmup

time.sleep(0.1)

# capture frames from the camera

for frame in camera.capture\_continuous(rawCapture, format="bgr", use\_video\_port=True):

# grab the raw NumPy array representing the image, then initialize the timestamp

# and occupied/unoccupied text

image = frame.array

# show the frame

cv2.imshow("Frame", image)

key = cv2.waitKey(1) & 0xFF

# clear the stream in preparation for the next frame

rawCapture.truncate(0)

# if the `q` key was pressed, break from the loop

if key == ord("q"):

break