## Mise en vol du drone crazyflie dans l'arène du vol

### 1/Préparation:

Il est conseillé de lire +le tutoriel ros-tutoriel.pdf

-Montage

Suivre les instructions sur le site: https://www.bitcraze.io/getting-started-with-the-crazyflie-2-0/

-Pour vérifier si crazyflie fonctionne correctement, il est conseillé de vérifier son fonctionnement en téléchargeant **Crazyflie client** disponible pour Android et IPhone.

### 2/Installation de ROS

Le tutoriel officiel du ROS se trouve sur http://wiki.ros.org/kinetic/Installation/Ubuntu

Etape 1:

Cas de Raspberry PI(cas Ubuntu Mate)

- ◆ Si vous venez de terminer l'installation d'Ubuntu, il faut tout d'abord donner l'autorisation à Ubuntu de télécharger des logiciels depuis l'internet

Pour cela

Aller dans Paramètres de system->Logiciels & mises à jour ->Logiciel Ubuntu

- ☑ Cocher Logiciel libre et open source maintenue par Canonical(main)
- ☑ Cocher Logiciel libre et open source maintenue par la communauté(universe)
- ☑ Cocher Pilotes et périphériques (restircted)
- ☑ Cocher Logiciels restreins par des droits d'auteur ou des questions juridiques (multiverse)

Ouvrir dans terminal linux

Donner l'autorisation à l'ordinateur d'installer les packages provenant de package.ros.org

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -
sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

Paramétrage de serveur de clés

```
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 –recv-
key 421C365BD9FF1F717815A3895523BAEEB01FA116
```

Retour :

```
Executing: /tmp/tmp.J05ms8Ahf0/gpg.1.sh --keyserver
hkp://ha.pool.sks-keyservers.net:80
--recv-key
421C365BD9FF1F717815A3895523BAEEB01FA116
gpg: requesting key B01FA116 from hkp server ha.pool.sks-keyservers.net
gpg: key B01FA116: public key "ROS Builder <rosbuild@ros.org>" imported
gpg: Total number processed: 1
gpg: imported: 1
```

### Etape 2:

#### Mettre les packages à jour

```
sudo apt-get update
```

#### Retour:

```
Hit:1 http://ppa.launchpad.net/flexiondotorg/minecraft/ubuntu xenial
InRelease
Get:2 http://ppa.launchpad.net/ubuntu-mate-dev/welcome/ubuntu xenial
InRelease [18,1 kB]
Hit:3 http://ports.ubuntu.com xenial InRelease
xenial/main armhf Packages [672 B]
Fetched 3 326 kB in 5s (572 kB/s)
¡¡¡¡¡
Reading package lists... Done
```

#### Téléchargement/Installation de ROS (version complète)

```
sudo apt-get install ros-kinetic-desktop-full
```

Ceci dure longtemps dans Raspberry Pi 3 (environ 1 heure)

### Etape 3:

#### Initialiser rosdep

```
sudo rosdep init
rosdep update
```

Retour :

```
reading in sources list data from /etc/ros/rosdep/sources.list.d
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-
homebrew.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/python.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/ruby.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/releases/fuerte.yaml
Query rosdistro index
https://raw.githubusercontent.com/ros/rosdistro/master/index.yaml
Add distro "groovy"
Add distro "hydro"
Add distro "indigo"
Add distro "jade"
Add distro "kinetic"
Add distro "lunar"
updated cache in /home/vu/.ros/rosdep/sources.cache
```

Paramétrage du Bash

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc source ~/.bashrc
```

Installer rosinstall

```
sudo apt-get install python-rosinstall
```

Etape 4 :

Créer un Workspace-Ros

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
```

Copier tous les fichiers du code source qui se trouve dans cloud dans /catkin_ws/src

Installer les dépendances liées aux packages

```
$ rosdep install --from-path src --ignore-src
```

Compiler les packages

```
$ cd ~/catkin_ws/
$ catkin_make
```

Explication: **catkin_make** compile tous les packages trouvant dans src

Pour rendre tous les packages de /src exécutables, il faut compiler chaque fois lorsque l'on la modifie

Rafraichir le répertoire

```
$ cd ~/catkin_ws/
$ source devel/setup.bash
```

Très utile, il faut le répéter chaque fois qu'on refait le catkin_make/que l'on allume l'ordinateur/...

- ✧ Le Ram de Raspberry est très petit, il est conseillé de configurer une espace mémoire swap avant d'effectuer catkin_make (Pas besoin en cas d'un PC portable)
- ✧ Il n'est pas nécessaire de télécharger le package hector_quadrotor

Remarque----------

Pour effectuer le swap

Le tutoriel en ligne testé: https://www.digitalocean.com/community/tutorials/how-to-add-swap-space-on-ubuntu-16-04

---

*Si aucune erreur s'affiche, alors l'installation est terminée.*

Etape 5:

Il faut désormais configurer les permissions pour que les programmes de crazyflie et de crazyradio puissent être exécutés avec la permission root. Pour le faire:

```
sudo groupadd plugdev
sudo usermod -a -G plugdev <username>
$ cd /etc/udev/rules.d
$ sudo gedit 99-crazyradio.rules
```

Copier

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="1915", ATTRS{idProduct}=="7777",
MODE="0664", GROUP="plugdev"
```

dans l'éditeur du texte. Sauvegarder.

Dans le même répertoire :

```
$ sudo gedit 99-crazyflie.rules
```

Copier

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="0483", ATTRS{idProduct}=="5740",
MODE="0664", GROUP="plugdev"
```

Dans l'éditeur du texte. Sauvegarder.

Redémarrer Raspberry PI

**********

Cas de Linux sur PC portable: Il faut assurer que le python intégré est de version 2.7.

Pour vérifier, taper

```
python -V
```

Si la version n'est pas la bonne, il faut installer conda.

Le tutoriel d'installation se trouve sur https://conda.io/docs/user-guide/install/linux.html

Une fois installé:

```
conda create -n myenv python=2.7
source activate myenv
```

Il est important de **recompiler** les fichiers sous cet environnement virtuel. (et donc installer les libraires et dépendances manquantes pour assurer la compilation)

**Chaque fois que l'on ouvre un nouvel terminal** il faut activer l'environnement virtuelle

```
source activate myenv
```

**********

Remarque conernant l'installtion----------

Les fichiers sous cloud sont issues de la modification de package

- Package ***crazyflie_ros*** (http://wiki.ros.org/crazyflie).
- Package ***vrpn_client_ros*** (http://wiki.ros.org/vrpn_client_ros).

On peut aussi donc les récupérer et les modifier

Dans ***vrpn_client_ros/launch/sapmle.launch***

Remplacer

server: $(arg server)

par

server: 139.124.59.124

Dans ***crazyflie_ros/crazyflie_demo***

Remplacer

<arg name="uri" default="radio://0/90/2M" />

<arg name="frame" default="/vicon/crazyflie/crazyflie" />

Par

<arg name="uri" default="son uri />

<arg name="frame" default="/nom d'objet dans Vicon " />

Remplacer

<include file="$(find vicon_bridge)/launch/vicon.launch"/>

Par

<include file="$(find vrpn_client_ros)/launch/sample.launch"/>

Dans ***vrpn_client_ros.cpp*** *du package* ***vrpn_client_ros***

| Find these lines | Replace them with these lines |
|---|---|
| tracker->pose_msg_.pose.position.x = tracker_pose.pos[0]; tracker->pose_msg_.pose.position.y = tracker_pose.pos[1]; | tracker->pose_msg_.pose.position.x = tracker_pose.pos[0]*-1; tracker->pose_msg_.pose.position.y = - tracker_pose.pos[1]*-1; |
| tracker->pose_msg_.pose.orientation.x = tracker_pose.quat[0]; tracker->pose_msg_.pose.orientation.y = tracker_pose.quat[1]; | tracker->pose_msg_.pose.orientation.x = - tracker_pose.quat[0]*-1; tracker->pose_msg_.pose.orientation.y = tracker_pose.quat[1]*-1; |
| tracker->transform_stamped_.transform.translation.x = tracker_pose.pos[0]; tracker->transform_stamped_.transform.translation.y = tracker_pose.pos[1]; | tracker->transform_stamped_.transform.translation.x = tracker_pose.pos[0]*-1; tracker->transform_stamped_.transform.translation.y = tracker_pose.pos[1]*-1; |
| tracker->transform_stamped_.transform.rotation.x = tracker_pose.quat[0]; tracker->transform_stamped_.transform.rotation.y = tracker_pose.quat[1]; | tracker->transform_stamped_.transform.rotation.x = tracker_pose.quat[0]*-1; tracker->transform_stamped_.transform.rotation.y = tracker_pose.quat[1]*-1; |
| tracker->twist_msg_.twist.linear.x = tracker_twist.vel[0]; tracker->twist_msg_.twist.linear.y = tracker_twist.vel[1]; | tracker->twist_msg_.twist.linear.x = tracker_twist.vel[0]*-1; tracker->twist_msg_.twist.linear.y = tracker_twist.vel[1]*-1; |
| tf2::Quaternion(tracker_twist.vel_quat[0], tracker_twist.vel_quat[1], tracker_twist.vel_quat[2], tracker_twist.vel_quat[3])); | tf2::Quaternion(tracker_twist.vel_quat[0]*-1, tracker_twist.vel_quat[1]*-1, tracker_twist.vel_quat[2], tracker_twist.vel_quat[3])); |
| tracker->accel_msg_.accel.linear.x = tracker_accel.acc[0]; tracker->accel_msg_.accel.linear.y = tracker_accel.acc[1]; | tracker->accel_msg_.accel.linear.x = tracker_accel.acc[0]*-1; tracker->accel_msg_.accel.linear.y = tracker_accel.acc[1]*-1; |
| tf2::Quaternion(tracker_accel.acc_quat[0], tracker_accel.acc_quat[1], tracker_accel.acc_quat[2], tracker_accel.acc_quat[3])); | tf2::Quaternion(tracker_accel.acc_quat[0]*-1, tracker_accel.acc_quat[1]*-1, tracker_accel.acc_quat[2], tracker_accel.acc_quat[3])); |

Le package Hector_Quadrator ne sert à rien.

---

Tout l'installation et paramétrage sont terminés.

Rafraichir le répertoire

```
$ cd ~/catkin_ws/
$ source devel/setup.bash
```

## 3/Mise en pratique

**Test de fonctionnement 1:**

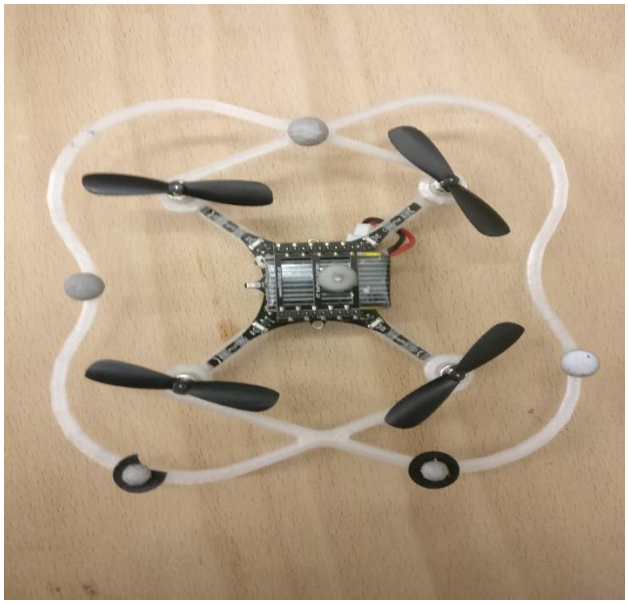
Pour contrôler crazyflie avec un joystick il faudrait lance le programme avec

```
roslaunch crazyflie_demo teleop_xbox360.launch uri:=radio://0/80/250K
```

**Fin test**

Avant de mettre en vol dans arène du vol, il faudrait mettre les marqueurs rafraichissant autour du drone pour qu'il puisse être reconnu par le système VICON

La disposition suivante est conseillée



- ✧ Il faut veiller que les marqueurs réfléchissent bien sous Vicon (pas de clignotement)
- ✧ La disposition doit être asymétrique (5-6 marqueurs)
- ✧ Il ne faut pas mettre 2 marqueurs côte à côte

Paramétrer le crazyflie dans le logiciel Vicon
Brancher une câble Ethernet et assurer que le Raspberry PI est connecté. Si Vicon n'est pas reconnu, paramétrer manuellement une adresse IP.

Test de fonctionnement 2: (Hovering/Waypoint)

L'algorithme de hovering permet le drone d'atteindre au point (x,y,z) donné par l'utilisateur et se stabiliser dans cette position
Avec x=0, y=0,z=2, on lance avec

```
roslaunch crazyflie_demo hover_vicon.launch x:=0 y:=0 z:=2
```

Appuyer X, quand le drone devient stable appuyer A enfin ctr+c.
Fin test hovering

L'algorithme waypoint permet au drone de parcourir prédéfini, on lance avec

```
roslaunch crazyflie_demo waypoint_vicon.launch
```

Appuyer X, quand le drone finit sa trajectoire appuyer A enfin ctr+c.

Fin test waypoint

Remarque----------

Pour effecteur le décollage, il faut appuyer X.
Pour effectuer l'atterrissage, il faut appuyer A.
Pour arrêter le programme, if faut appuyer ctr+c.
La trajectoire est définie dans ***demo1.py*** de package ***crazyflie_demo***
La syntaxe pour chaque waypoint est définie de manière suivante
[x,y, z, angle de lacet, temps d'attente]

A propos de uri:

- ✧ Il est l'adresse de chaque drone
- ✧ On peut trouver l'uri en branchant le drone par port USB

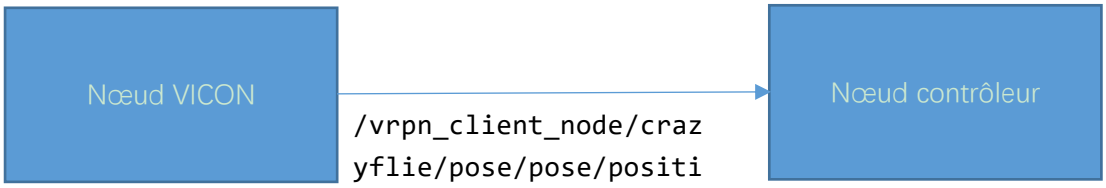
```
rosrun crazyflie_tools scan
```

---

Exploitation de trajectoire:

Principe:

Pendant l'exécution de l'algorithme waypoint, le nœud de Vicon échange des donnes avec le nœud de contrôleur du drone




Nous allons utiliser les commandes de Linux pour créer une fichier texte qui enregistre la trajectoire, ce fichier se trouvera dans /catkin_ws

Il faut tout d'abord lancer le programme et poser le crazyflie dans l'arène du vol.

```
roslaunch crazyflie_demo waypoint_vicon.launch
```

Ensuite, pour enregistrer

```
cd ~/catkin_ws/
rostopic echo /vrpn_client_node/crazyflie/pose/pose/position>essai.txt
```

- ■ Une fois le vol ait terminé, on appuiera ctr+c pour arrêter le programme.

Pour visualiser la trajectoire,

On pourra utiliser le programme Matlab suivant :

```
%--------------------------------extraction de ros
fileID = fopen('essai.txt','r');
C = textscan(fileID,'x: % f \n y: % f \n z: % f \n---\n');
scatter3(C{1},C{2},C{3})
fclose(fileID);
%--------------------------------waypoints
hold on
fileID = fopen('demo1.txt','r');
C = textscan(fileID,'[% f , % f, % f, % f, % f],');
scatter3(C{1},C{2},C{3})
```

essai.txt contient la trajectoire enregistré du drone.
demo1.txt contient les waypoints.

## Synthèse de la trajectoire

Il est possible de synthétiser une trajectoire de vol a partir d'une équation paramétrique

- ✧ Il est conseillé que les cordonnes de |x| et |y| ne dépassent pas 0.5
- ✧ On définit 0.1s comme la durée entre deux waypoints consécutifs

On peut ensuite les sauvegarder dans le format directement lisible par le programme
Par exemple :

```
a = 0.5;
c = 1;
t = 5*pi:0.1:10*pi;
x = a*sin(t);
y = a*cos(t);
z = t/(5*pi*c);
figure(1)
scatter3(x, y, z);
xlabel('x'); ylabel('y'); title('Circula helix');
fileID = fopen('hex.txt','w');
l=length(t);
for i=1:1:l
    fprintf(fileID,'[% f , % f, % f, % f, %
f],\n',x(i),y(i),z(i),0,0.1);
end
fclose(fileID);
```

Remarque----------

On remplacera dans le temps d'attente de la première ligne (0.1) en 8 .
Ceci permettra le drone de se stabiliser avant de poursuivre son chemin.