TUTORIAL

ROBOTIC OPERATING SYSTEM (ROS)

This small tutorial is dedicated to guiding the user to install ROS and necessary packages in order to use, manipulate and control the Crazyflie nano-quadcopter with the VICON camera system.

ROS is an open-source platform (BSD license) with a growing community (just like Arduino and Raspberry Pi). It provides libraries, tools, device drivers, hardware abstraction to interact and develop robotic applications. ROS collaborates different robotic softwares to create a complex and robust robot behavior. For example, to make a robot pick an object from one place to another place in a building, many different sub-tasks must be combined. One group can design the mapping of indoor environment, another group may use computer vision for recognizing the object, and another group may look into the navigation problem. ROS gives these kinds of collaboration a good platform.

Website: www.ros.org

I. Installation of ROS

1. Introduction

For now, ROS only supports Linux-based OS. We can install it on laptop, computer or Raspberry Pi. ROS has many Distributions such as Indigo, Hydro, Fuerte or Kinetic. Here are the officially supported Linux distros that the Kinetic ROS version supports:

	32-bit (i386)	**64-bit (amd64)**	**armhf (ARM-based)**
Wily (Ubuntu 15.10)	X	X	
Xenial (Ubuntu 16.04 LTS)	X	X	X
Jessie (Debian 8)		X	X

(The other versions of Ubuntu can also be used but it's not confirmed, the user should search on the Internet to make sure that the other versions are stable with ROS).

In this tutorial, I will guide you step by step on how to install ROS Kinetic version. The other versions are quite the same and the user can find it easily on the Installing part on the website.

You may also find the supported robots (which mean there are some available ROS packages for these robots) on this website: http://robots.ros.org/.

2. Steps to install

The official instruction for ROS installation is available on website:
http://wiki.ros.org/kinetic/Installation/Ubuntu

Download and flash the image for Ubuntu 16.04 MATE Operating System here:

https://ubuntu-mate.org/download/
https://www.raspberrypi.org/documentation/installation/installing-images/

It is recommended to use at least SD card with 16 GB for ROS usage.

Otherwise, you can flash an image (Ubuntu 16.04 MATE OS) with ROS pre-installed on this website:
http://www.german-robot.com/2017-03%20Ubuntu%20Mate%2016.04%20+%20ROS%20Kinetic.img.zip

or ***the Raspberry Pi 3 Ubuntu with ROS*** part in this website:
http://www.german-robot.com/2016/05/26/raspberry-pi-sd-card-image/

2.1 Configure Ubuntu repositories

Allow **restricted, universe, multiverse** for Ubuntu repositories:

1. Open ***System Settings***

2. Click on ***Software and Updates***

3. In Ubuntu Software tab, tick the box:

- Software restricted by copyright or legal issues (multiverse).
- Community-maintained free and open-source software (universe).
- Proprietary drivers for devices (restricted).

2.2 Set up sources.list and keys.

1. Open the terminal on Ubuntu.

2. Allow your computer to clone, download, access or upgrade the ROS packages from ***package.ros.org***. Type on the terminal:

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

3. Set up the keyserver. Type on the terminal:

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

Return result:

```
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
```

You can substitute *hkp://ha.pool.sks-keyservers.net:80* with *hkp://pgp.mit.edu:80* or *hkp://keyserver.ubuntu.com:80* if there is any problem connecting to the keyserver.

2.3 Download ROS

1. Make sure all the packages are up-to-date by typing on the terminal:

```
sudo apt-get update
```

Return result:

```
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
```

2. Install ROS (full version):

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

This step will take normally about 30 - 45 minutes to finish.

2.4 Initialize rosdep

This enables you to easily install the system and package dependencies for the source (package) you want to compile. This is also required to run some components on ROS. Type on the terminal:

```
sudo rosdep init
rosdep update
```

Return result:

```
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
```

Add ROS environment variables to bash session automatically when a new shell is launched:

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

2.6 Get rosinstall

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

This will take 5-10 minutes.

2.7 Create a ROS workspace

ROS workspace is in a folder named ***catkin_ws***. This workspace allows you to modify, build (compile, make) and install all the independent ROS packages at once.

More info: http://wiki.ros.org/catkin/workspaces

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

The folder (workspace) is empty at first time. There is no ROS packages in the ***src*** folder. We will build the workspace:

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

Note: we need to do the ***catkin_make*** command above whenever:

- We clone a new ROS package to the workspace, or when we add/modify any source files (.cpp/.hpp/.py/.srv/.msg ….) of any ROS package so it will build, compile and make these changes.

Now we have build and devel folder inside the catkin_ws. Inside the devel folder, we also have some setup files *.sh.

After that, we will source the new setup **.sh* file so the changes or new ROS packages you made will be avaialble for use:

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

Note: we need to source the new setup files *.sh like above whenever:

- After we do the catkin_make.
- We start or restart the laptop.

3. Simple tests to make sure ROS is installed successfully

Open the terminal and try some basic ROS command like in this website:

http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem

II. Basic information about ROS environment

You can find all the basic tutorials of ROS on this website:

http://wiki.ros.org/ROS/Tutorials

ROS package programs (often called ***node*** in ROS) are written in Python or C++. The user controls all the ROS programs (nodes) through terminal.

Basic components the user needs to understand to manipulate the source files easier and effectively:

1. Command to run a program (node) in ROS:

- ***rosrun***: execute what is inside a program (node) http://wiki.ros.org/rosbash#rosrun
- ***roslaunch***: execute a *.launch file (which can launch many nodes at the same time in a file). http://wiki.ros.org/ROS/Tutorials/UsingRqtconsoleRoslaunch

2. Message and Service (*.msg and *.srv files) (http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv):

3. Subscriber and Publisher (http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29)

4. tf: This can help the user to deal with coordinate transformation related problems (often in multi-robot applications) http://wiki.ros.org/tf/Tutorials.

5. rviz: 3D visualization http://wiki.ros.org/rviz.

6. rqt: GUI tools for ROS http://wiki.ros.org/rqt.

III. Package Installation for the use of Crazyflie and VICON

1. Packages for Crazyflie and VICON

1.1 Cloning

The user needs to clone 2 packages:

- Package ***crazyflie_ros*** (http://wiki.ros.org/crazyflie).
- Package ***hector_quadrotor*** (http://wiki.ros.org/hector_quadrotor). This is for the teleoperation of Crazyflie using Joystick.

There are many different packages to interact with VICON or OptiTrack. Here we will use the package ***vrpn_client_ros*** made by ***clearpathrobotics***.

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

Clone the package into the ***src*** file of ***catkin_ws*** by:

```
$ cd ~/catkin_ws/src/
$ git clone https://github.com/whoenig/crazyflie_ros.git
$ git clone https://github.com/tu-darmstadt-ros-pkg/hector_quadrotor.git
$ git clone https://github.com/clearpathrobotics/vrpn_client_ros.git
```

Then the user need to catkin_make to build and compile all the packages:

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

To be able to use the Crazyradio, we need to set the udev permissions as followed:

1. Allow the use of Crazyradio without being root. Type in the terminal:

```
sudo groupadd plugdev
sudo usermod -a -G plugdev <username>
```

2. Create a file named `/etc/udev/rules.d/99-crazyradio.rules` by this way:

```
$ cd /etc/udev/rules.d
$ sudo gedit 99-crazyradio.rules
```

3. This will create and open a blank file named *99-crazyradio.rules* in the directory */etc/udev/rules.d*. Just copy this into the file:

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

4. Similarly, create a file named `/etc/udev/rules.d/99-crazyflie.rules` by this way:

```
$ cd /etc/udev/rules.d
$ sudo gedit 99-crazyflie.rules
```

5. Copy this into the file:

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

6. Restart your laptop so that the permissions will be activated.

Instruction link: https://github.com/bitcraze/crazyflie-lib-python#setting-udev-permissions

1.2 Modify vrpn_client_ros package to use with VICON

1. Locate the vrpn_client_ros package in the directory ***~/catkin_ws/src***. Open the file ***sample.launch*** in launch folder.

Replace the line:

```
server: $(arg server)
```

with

```
server: 139.124.59.124
```

This is the IP address of the VICON system in the flying arena of the Biorobotic laboratory.

2. Open file ***vrpn_client_ros.cpp*** in the ***src*** folder in vrpn_client_ros package. Modify as followed:

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]));`

What we are trying to do here is to change the sign of translation and rotation of x and y axis, so it will match with the coordinate definition of VICON.

3. In the demo launch file in crazyflie_demo package (Ex: hover_vicon.launch or waypoint_vicon.launch), we need to mofidy:

change

```
<arg name="uri" default="radio://0/90/2M" />
<arg name="frame" default="/vicon/crazyflie/crazyflie" />
```

to

```
<arg name="uri" default="radio://0/70/2M" />
<arg name="frame" default="/crazyflie" />
```

and change

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

to

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

4. Do the catkin_make and source again.

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

Note:
- To be able to use VICON, remember ***connect with Internet*** via an Ethernet cable first to enable TCP/IP communication.
- Please make sure that the Crazyflie object (reflective markers) defined on the VICON software is enabled.

1.3 Bugs

1. Error when ***catkin_make***:

```
CMake Error at /opt/ros/hydro/share/catkin/cmake/catkinConfig.cmake:72 (find_package):
  Could not find a configuration file for package hector_pose_estimation.

  Set hector_pose_estimation_DIR to the directory containing a CMake
  configuration file for hector_pose_estimation.  The file will have one of
  the following names:

    hector_pose_estimationConfig.cmake
    hector_pose_estimation-config.cmake
```

or

```
> CMake Error at
> vrpn_client_ros/CMakeLists.txt:5
> (find_package):   By not providing
> "FindVRPN.cmake" in CMAKE_MODULE_PATH
> this project has   asked CMake to find
> a package configuration file provided
> by "VRPN", but   CMake did not find
> one.
>
>   Could not find a package
> configuration file provided by "VRPN"
> with any of   the following names:
>
>     VRPNConfig.cmake
>     vrpn-config.cmake
```

Explanation:

You are lacking the dependency packages for hector_quadrotor or vrpn_client_ros packages, so it cannot be built.

Solution:

Run this on the terminal to install all the dependencies:

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

2. Hovering and Waypoint following

1. ***catkin_make*** and ***source*** first if you haven't done this

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

Remember to connect the Crazyradio and the joystick before running the program.

2. For the hovering at 2 meters, type this command:

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

After the program has launched, type the blue button **x** on the joystick to take off. Then, push the green button **o** to land.

To end the hovering program, type Ctrl+C

3. For the waypoint navigation, get a copy of the waypoint_vicon.launch and copy it to the demo folder in crazyflie_demo package. Then, type this command:

```
roslaunch crazyflie_demo waypoint_vicon.launch
```

After the program has launched, type the blue button **x** on the joystick to take off. Then, push the green button **o** to land.

The waypoints are defined in the ***demo1.py*** file In the scripts folder in ***crazyflie_demo*** package. The syntax for each waypoint is:

[x position , y position , z position , yaw angle , wait time to next waypoint]

Example: [0 , 0 , 2 , 0 , 2] means go to the position (x,y,z) = (0,0,2) at yaw angle 0 and stay there 2 seconds.

4. To control the Crazyflie with joystick, type this command:

```
roslaunch crazyflie_demo teleop_xbox360.launch uri:=radio://0/70/2M
```

3. Off-board PID Controller

To modify the Kp, Ki, Kd of the controller, the user will go to the ***crazyflie2.yaml*** file in the ***config*** folder of the ***crazyflie_controller*** package.

To interfere in the controller, the user can modify the ***controller.cpp*** and ***pid.hpp*** files in the ***src*** folder of the ***crazyflie_controller*** package.

IV. Install Arduino rosserial_arduino package

1. Install package

Please install the Arduino IDE first.

The tutorial for this package is in this website: http://wiki.ros.org/rosserial_arduino/Tutorials

1. Install ***rosserial_arduino*** package. Type on the terminal:

```
$ cd ~/catkin_ws/
$ sudo apt-get install ros-kinetic-rosserial-arduino
$ sudo apt-get install ros-kinetic-rosserial
$ catkin_make
$ source devel/setup.bash
```

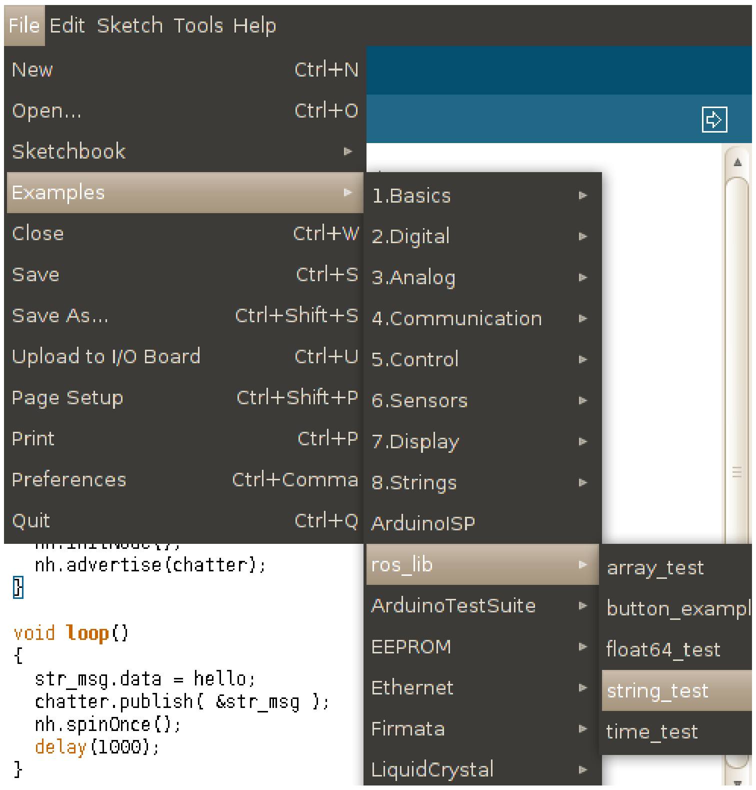
2. Install ***ros_lib*** to the Arduino IDE (containing some ros functions to be used with IDE):

Delete the ros_lib first (if it is existed) and create a new ros_lib:

```
  $ cd ~/<sketchbook>/libraries
  $ rm -rf ros_lib
  $ rosrun rosserial_arduino make_libraries.py .
```

Normally, <*sketchbook*> is named Arduino.

3. Restart your IDE, check if you see the ros_lib in File → Example or not. If you see it, your installation is successful:



```
  nh.advertise(chatter);
}

void loop()
{
  str_msg.data = hello;
  chatter.publish( &str_msg );
  nh.spinOnce();
  delay(1000);
}
```

4. We should ensure that the USB cable connection between the Arduino and Ubuntu is recognized by doing the udev permission as following:

a. Add yourself to ***dialout*** using:

```
$ sudo gpasswd --add <username> dialout
```

b. Find the idVendor and idProduct of the Arduino:

```
$ lsusb
```

You must see something with similar syntax (but not the same) to this:

```
...............
Bus 002 Device 005: ID 0043:2341 Atmel Arduino Mega ......
...............
```

For the ID 0043:2341 above, the idVendor = 0043 and the idProduct = 2341.

c. Create an udev permission:

```
$ cd /etc/udev/rules.d/
$ sudo gedit 97-arduino.rules
```

Copy in the empty file opened (replacce the #### with idVendor and idProduct):

```
SUBSYSTEMS=="usb", ACTION=="add", ATTRS{idVendor}=="####",
ATTRS{idProduct}=="####", MODE="0666", SYMLINK+="arduino
arduino_$attr{serial}", GROUP="dialout",
```

d. Update the udev permission and restart

```
  $ sudo udevadm control --reload-rules && sudo service udev restart && sudo
udevadm trigger
```

The laptop should recognize the connection now (the ***/dev/ttyACM0*** port is recognized in the Arduino IDE).

Instruction link:
http://www.clearpathrobotics.com/assets/guides/ros/Driving%20Husky%20with%20ROSSerial.html