GUÍA PARA USO DE MANIPULADOR AÉREO MEDIANTE ROS

Se ha creado una guía para la instalación de ROS usando una maquina virtual y Ubuntu 20.04 LTS, se recomienda ver la guía de uso para su mejor desempeño.

1. Instalar Ubuntu en una máquina virtual de su preferencia, Virtual Box o VMWare.
2. Instalar ROS en Ubuntu

VERSIONES DE UBUTU Y ROS COMPATIBLES	
Ubuntu 16.04	ROS KINETIC
Ubuntu 18.04	ROS MELODIC
Ubuntu 20.04	ROS NOETIC

3. Añadir los repositorios de ROS a Ubuntu

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

sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net –recv
-key 0xB01FA116

sudo apt-get update
```

4. Instalar versión complete

```
sudo apt install ros-noetic-desktop-full
```

5. Configuración de entorno

```
source /opt/ros/noetic/setup.bash

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

source ~/.bashrc
```

6. Instalar dependencias para trabajar con paquetes

```
sudo apt install python3-rosdep python3-rosinstall python3-rosinsta
ll-generator python3-wstool build-essential
```

7. Inicializamos rosdep

```
sudo apt install python3-rosdep

sudo rosdep init

rosdep update
```

8. Creamos el área de trabajo (workspace)

```
mkdir -p ~/catkin_ws/src

cd ~/catkin_ws/src
```

9. Iniciar y compilar el workspace

```
rospack profile

roscd

catkin_init_workspace

cd ~/catkin_ws/

catkin_make

source devel/setup.bash
```

Se ha instalado correctamente ROS Noetic en Ubuntu versión 20.04, a continuación, se presentan comando para crear nuestro propio paquete.

```
cd ~/catkin_ws/src

catkin_create_pkg my_package std_msgs rospy roscpp
```

```
cd ~/catkin_ws

a) catkin_make →compila todo el workspace

b) catkin_make --pkg my_package  → compila sólo el paquete y todas las dependencias posibles

c) catkin_make --only-pkg-with-deps my_package → compila sólo el paquete y las dependencias mínimas
```

Para hacer uso de los paquetes del manipulador aéreo se debe descargar e introducir los archivos dados en el siguiente link, dentro de la carpeta src, para luego compilar todos los archivos.

Link: https://github.com/CristianKmas/Aerial-manipulator-with-open-manipulator-x.git

Estos archivos se deben instalar en las dos computadoras tanto en la Estación como la del UAV.

Para configurar la comunicación de ambas computadoras mediante ROS, realice en un terminal:

```
gedit .bashrc
```

Esto abrirá un archivo de texto en el cual se realiza los siguientes cambios.

Se debe cambiar la dirección IP de cada una de las computadoras que han sido asignadas por el Router, para saber que IP se le asigno a su computado en un terminal ejecute el comando ifconfig.

Para la Estacion debe ingresar la IP correspondiente a esa computadora tanto en el MASTER_URI como en el HOST_NAME.

GUÍA PARA USO DE MANIPULADOR AÉREO MEDIANTE ROS

```
128 export ROS_MASTER_URI=http://192.168.100.114:11311
129 export ROS_HOSTNAME=192.168.100.114
```

Mientras que en la computadora del UAV se debe ingresar la IP que se asigno a esa computadora solo en el HOSNAME y en el MASTER_URI se debe cambiar por la IP de la computadora Estación.

```
128 export ROS_MASTER_URI=http://192.168.100.114:11311
129 export ROS_HOSTNAME=192.168.100.115
```

Cerramos y guardamos cambios, además se debe reiniciar el terminal para poder lanzar el roscore con los cambios realizados.

Para la parte de obtención de datos de telemetría se usó los siguientes repositorios.

Onboard SDK 3.9

https://github.com/dji-sdk/Onboard-SDK.git

Onboard SDK ROS 3.8

https://github.com/dji-sdk/Onboard-SDK-ROS.git

Se debe instalar correctamente y teniendo en cuenta la versión que soporta su UAV. Si tiene un problema consulte la documentación en el siguiente enlace: https://developer.dji.com/onboard-sdk/documentation/development-workflow/environment-setup.html#linux-with-ros

Con todo instalado se debe realizar los siguientes comandos en el terminal de cada computador primero debe correr el ros master en la computadora Estacion mediante el comando **roscore**, luego se debe conectar el brazo mediante usb al computador del UAV y mediante terminal realizar los comandos.

```
roslaunch open_manipulator_controller open_manipulator_controller.launch
```

En el terminal se presenta el siguiente resultado satisfactorio

```
port_name and baud_rate are set to /dev/ttyUSB0, 1000000
Joint Dynamixel ID : 11, Model Name : XM430-W350
Joint Dynamixel ID : 12, Model Name : XM430-W350
Joint Dynamixel ID : 13, Model Name : XM430-W350
Joint Dynamixel ID : 14, Model Name : XM430-W350
Gripper Dynamixel ID : 15, Model Name :XM430-W350
[INFO] Succeeded to init /open_manipulator_controller
```

Ya tenemos conectado el brazo y distribuido median una red LAN con ROS, ahora debemos conectar el UAV a la computadora del UAV e introducir otro comando mediante terminal para usar la SDK del DJI MATRICE 210.

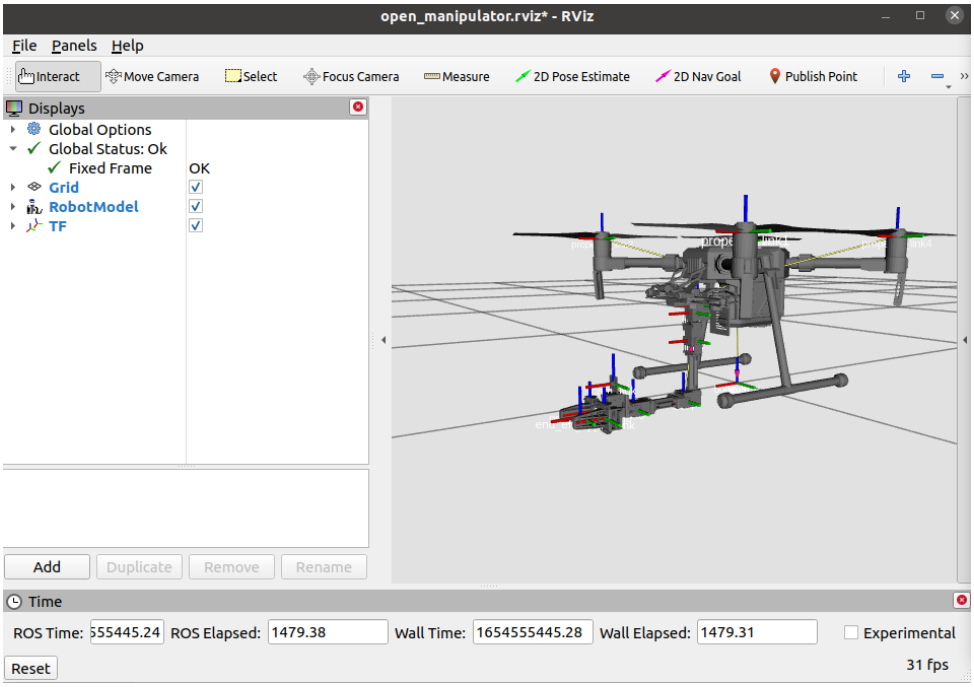
```
roslaunch dji_sdk sdk.launch
```

con esto finalizamos en la computadora del UAV, ahora para la visualización y control del brazo debemos dirigirnos a la computadora Estación e introducir los comandos.

Para visualización en RVIZ

```
roslaunch open_manipulator_description open_manipulator_rviz.launch
```

Se obtiene el resultado

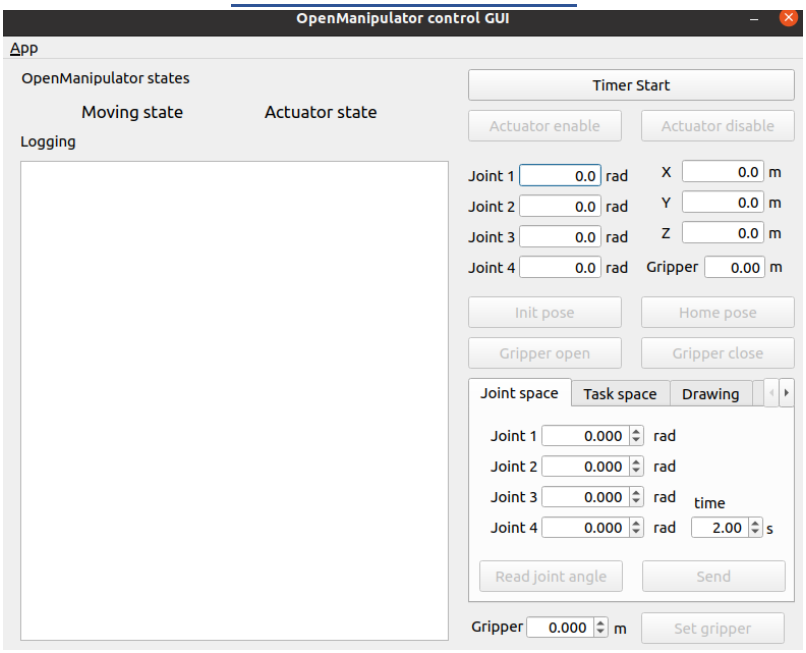


Para el control del brazo mediante una interfaz grafica

```
roslaunch open_manipulator_control_gui open_manipulator_control_gui.launch
```

Se obtiene el resultado con el cual se puede conectar y empezar a controlar el brazo robótico.

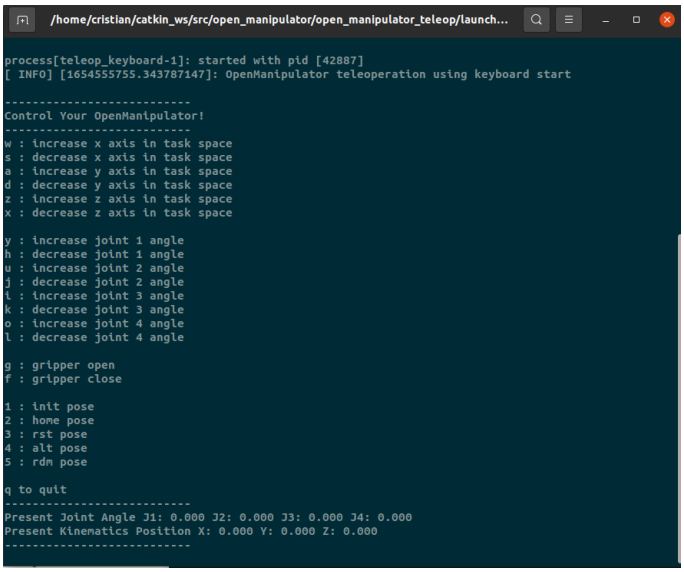
GUÍA PARA USO DE MANIPULADOR AÉREO MEDIANTE ROS



Para la parte de control del brazo mediante teclado se lo hace con el comando

```
roslaunch open_manipulator_teleop open_manipulator_teleop_keyboard.launch
```

Se obtiene el resultado en el cual se puede interactuar con el brazo.



Finalmente, para suscribirnos a la telemetría del UAV usamos dos nodos uno para el brazo y otro para el UAV mismo que mostrara la trayectoria tanto del brazo como del UAV.

Para subscripción de brazo

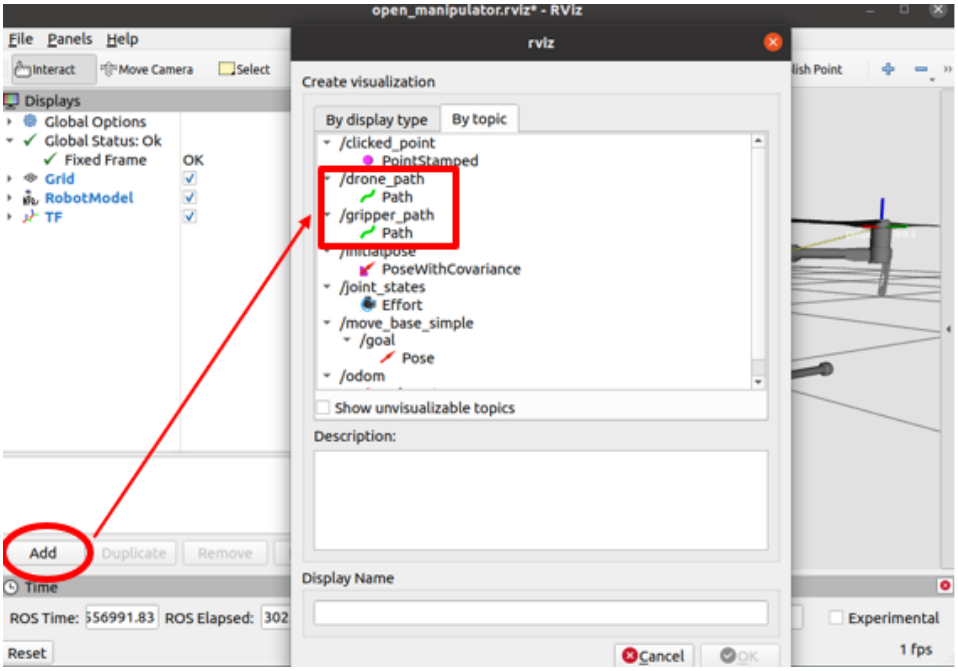
```
rosrun open_manipulator_description gripper_controller_paths.py
```

Para suscripción al UAV

GUÍA PARA USO DE MANIPULADOR AÉREO MEDIANTE ROS

```
rosrun odometria drone_paths.py
```

Para la visualización de estos dos nodos de subscripción se debe agregar en RVIZ como se muestra en la siguiente imagen.



Algo que hay que recalcar es que podemos ver nuestros nodos y todas conexiones que intervienen al lanzar cada comando antes mencionado con el cual se puede observar y analizar si estamos realizando correctamente.

```
rosrun rqt_graph rqt_graph
```

Obteniendo el siguiente resultado.