Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Estado de México

Escuela de Ingeniería y Ciencias

Integración de Robótica y Sistemas Inteligentes

Grupo 501
Equipo 11 - La Voluntad

Manual de usuario:

Guía de Instalación Simulador PX4 con ROS/MAVROS

Profesor:

Dr. Alejandro Aceves López
Dr. Aldo Iván Aguilar Aldecoa
Dr. Miguel Ángel Galvez Zuñiga
Dr. Alf Kjartan Halvorsen
Dr. Francisco Javier Ortiz Cerecedo
Dr. Arturo Vargas Olivares

Alumno

Bruno Sánchez García	A01378960
Carlos Antonio Pazos Reyes	A01378262
Manuel Agustín Díaz Vivanco	A01379673

Atizapán de Zaragoza, México a 7 de junio de 2023

Guía para correr ROS en PX4 y Simuladores
Primero hay que instalar varias cosas. La instalación principal se puede seguir desde el enlace https://docs.px4.io/main/en/dev_setup/getting_started.html. Seguimos las instrucciones en https://docs.px4.io/main/en/dev_setup/dev_env_linux_ubuntu.html que indica cómo instalar todo en ubuntu.

Hay que tener instalado Ubuntu 18.04, ROS Melodic y MAVROS. Se puede optar por instalar mediante un archivo bash que se proporciona en este mismo enlace de PX4.

Para descargar el simulador PX4 se siguen los comandos:

```
Unset
git clone https://github.com/PX4/PX4-Autopilot.git --recursive
```

Dentro de la carpeta del repositorio clonado, hay que ejecutar:

```
Unset
bash ./PX4-Autopilot/Tools/setup/ubuntu.sh
```

Hay que instalar el software QGroundControl, siguiendo las instrucciones en https://docs.qgroundcontrol.com/master/en/getting_started/download_and_install.html en el apartado de Ubuntu Linux.
Se instala mediante los siguientes comandos:

```
Unset
sudo usermod -a -G dialout $USER
sudo apt-get remove modemmanager -y
sudo apt install gstreamer1.0-plugins-bad gstreamer1.0-libav
gstreamer1.0-gl -y
sudo apt install libqt5gui5 -y
sudo apt install libfuse2 -y
```

Estos instalan las librerías esenciales para correrlo y necesitan acceso y remover el modem manager para acceder a puerto serial de la máquina.
Hay que reiniciar la máquina después de estos comandos.
Luego hay que descargar la imagen de la aplicación.
NOTA: nos ha pasado que descargamos la imagen desde Daily Builds y otros enlaces pero la imagen no funciona, más bien descargar dentro del hiperenlace que aparece en esta página de instalación. Este enlace es https://d176tv9ibo4jno.cloudfront.net/latest/QGroundControl.AppImage. Si no funciona aún, pueden contactarnos y pasamos el archivo directamente.
Finalmente se puede instalar y correr la aplicación ejecutando este archivo de imagen:

```
Unset
chmod +x ./QGroundControl.AppImage
```

```
./QGroundControl.AppImage  (or double click)
```

Es muy probable que te pida alguna librería con versión específica, por lo que habrá que instalarlas conforme vaya solicitando.

Finalmente se procede a armar y correr un simulador de varios que tiene PX4.
Dentro de la carpeta del repositorio de PX4 (/PX4-Autopilot) hay que correr el comando:

```
Unset
make px4_sitl jmavsim
```

Arma y corre el simulador jmavsim, el cual necesita de java, por lo que si no se tiene instalada habrá que instalarlo.
Alternativamente se puede mandar armar y correr con Gazebo:

```
Unset
make px4_sitl gazebo
```

Y de la misma forma se puede solicitar armar y correr diferentes aeronaves y simuladores:

```
Unset
make [VENDOR_][MODEL][_VARIANT] [VIEWER_MODEL_DEBUGGER_WORLD]
```

Es muy poco probable que arme y corra a la primera, pero el mismo programa va indicando el por qué. Muy seguramente te pide instalar varias librerías de python, además que pide que sea en python3. Por ejemplo en alguna parte del error muerte algo como:

```
Unset
Failed to import jinja2: No module named 'jinja2'
You may need to install it using:
    pip3 install --user jinja2
```

En la sección de Troubleshooting recomiendan instalar las librerías:

```
Unset
pip3 install --user pyserial empty toml numpy pandas jinja2
pyyaml pyros-genmsg packaging
```

Dentro del enlace https://docs.px4.io/main/en/dev_setup/building_px4.html en la sección de hasta abajo vienen otros tips para troubleshooting que podrían ayudar, como errores de compilador en el que hay que desinstalar e instalar una versión de un compilador. También incluye los comandos básicos en caso de fallo general en el build:

```
Unset
git submodule update --recursive
make distclean
```

Al correr cada aplicación en una terminal cada uno, se debe hacer la conexión automática entre estos, si esto falla significa que hay alguna configuración faltante entre las aplicaciones, pero según la instalación esta configuración debe ya estar hecha por default. Para probar que sirve, correr el comando en la terminal donde se tenga la simulación en PX4:

```
Unset
pxh> commander takeoff
```

Este comando debe hacer que el dron haga hover indefinidamente. Se debe ver en el simulador el dron en hover, y en el QGroundControl su estatus actualizado que el dron ha sido armado y se encuentra activo.
Para volver a bajar el dron a suelo:

```
Unset
pxh> commander land
```

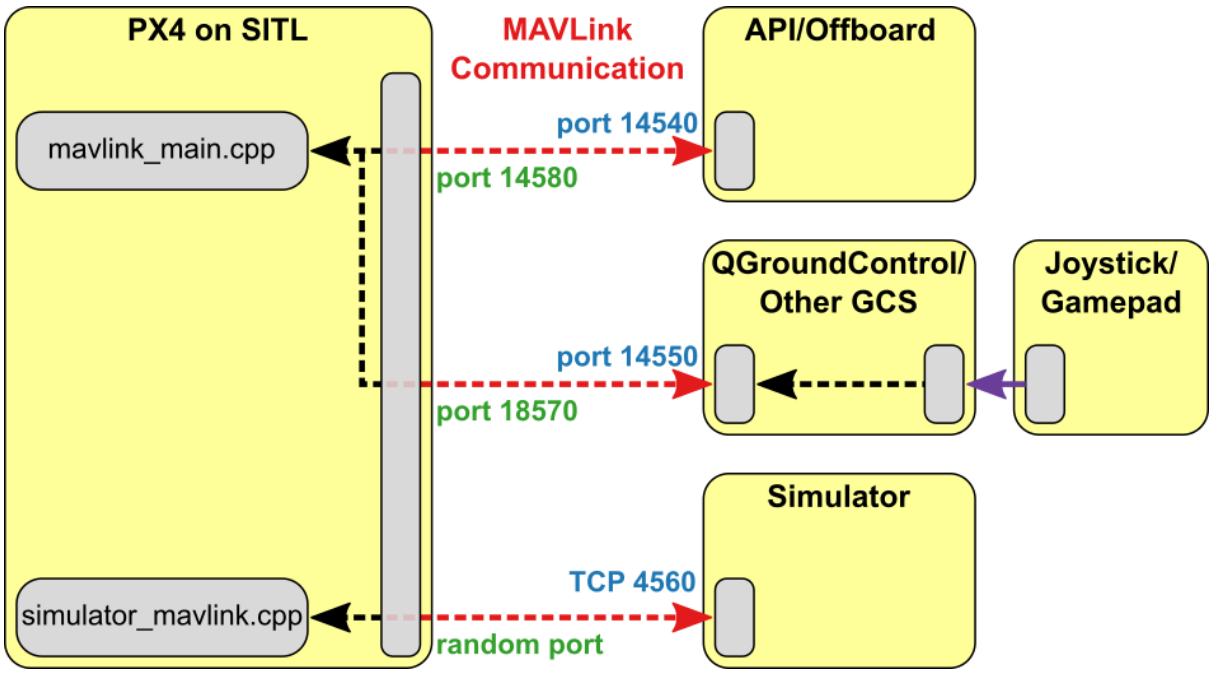
Y para finalizar la simulación simplemente CTRL + C, o alternativamente:

```
Unset
pxh> shutdown
```

Una vez que se puedan correr en simultáneo QGroundControl y algún simulador PX4, ya se puede pasar a control con ROS.

Ejemplo de control con ROS
Para controlar el dron con PX4 mediante ROS, hay que tener instalado MAVROS, que se encarga de hacer la conexión mediante un bridge MAVLINK.

A grandes rasgos, el PX4 tiene el firmware de autopilot con ArduPilot, que se conecta mediante serial a la Odroid que cuenta con ROS y MAVROS/MAVLINK. Se hace el puente y se pueden comunicar para controlar el vuelo del dron.
En simulación es como se muestra:



El ejemplo proviene siguiendo el enlace: https://docs.px4.io/main/en/simulation/ros_interface.html. La imagen explica el puerto y conexiones entre los elementos. El QGroundControl nos ayuda a visualizar y controlar las misiones y vuelos del dron al igual que tiene muchas configuraciones diferentes dependiendo lo que se desee. El PX4 hace la conexión con el simulador, el QGroundControl y la ejecución con ROS mediante MAVLINK.

El ejemplo se centra en una simulación de hover en Gazebo, usando un código de Python con ROS.
Primero hay que correr la aplicación QGroundControl en una terminal:

```
Unset
./QGroundControl.AppImage
```

En otra terminal correr el simulador de gazebo:

```
Unset
make px4_sitl gazebo
```

Ahora, hasta ahora no hay enlace con ROS, por lo que si se buscan los tópicos, no se han levantado. Es ahora cuando hay que usar MAVROS. En otra terminal hay que usar el archivo launch:

```
Unset
roslaunch mavros px4.launch fcu_url:="udp://:14540@127.0.0.1:14557"
```

El parámetro fcu_url es para señalar la url con especificación de IP/puertos de uso para hacer la conexión. Para la simulación con un dron único se puede usar sólamente localhost, por lo que la IP es 127.0.0.1. Esto levantará la información del dron en tópicos prefijo */mavros/…* por lo que ya se pueden usar y visualizar.
TIP: para visualizar en tiempo real y con facilidad los muchos tópicos que levanta MAVROS, usar en una terminal el comando rqt (si ya se tiene instalado) y abrir un Plugins/Topics/Topic Monitor, que permite visualizar todos los tópicos activos simultáneamente y de forma fácil junto con toda la información como tipo de mensaje, frecuencia, ancho de banda y el valor en tiempo real.

El código usado para el hover se tomó del proporcionado por la página de PX4: https://docs.px4.io/main/en/ros/mavros_offboard_python.html. Este mismo enlace sugiere crear dentro del workspace, una carpeta de scripts y launch donde se creen el código python y un archivo launch para lanzar el nodo respectivamente. Se puede correr de esa forma. Hay que tener cuidado con los PATH hacia Gazebo y hacia el repositorio de PX4-Autopilot para que corra.
Alternativamente se puede simplemente ejecutar el código de python en una terminal y se ejecutará en el simulador. Por ejemplo:

```
Unset
python offb_node.py
```

El código debe hacer que el dron se mantenga en hover a 2 metros del suelo, dentro de un modo de vuelo OFFBOARD, y el dron armado. Esta información se debe reflejar en los tópicos, por ejemplo el tópico */mavros/state* debe mostrar en *mode* el string ‘OFFBOARD’.
Cancelar la ejecución del código hace que el dron entre un modo de vuelo automático y descienda a aterrizar.

HELP: Algunos videos de ayuda para hacer la prueba con ROS:

- **Instalación completa y ejecución de código python: https://www.youtube.com/watch?v=r5GEO2Zvs54&t=326s**
- **Instalación e Implementación PX4/MAVROS (Parte 1): https://www.youtube.com/watch?v=jBTikChu02E**
- **Instalación e Implementación PX4/MAVROS (Parte 2): https://www.youtube.com/watch?v=rxt0aBnBeJI**

Modos Vuelo

La ejecución del código Python utiliza el servicio de armado de mavros, y el servicio de set_mode para armar el dron y cambiar el modo de vuelo respectivamente.

Con MAVROS, se puede hacer conexión con Ardupilot, referenciado como APM, y para levantar los tópicos y el enlace se usa el roslaunch mavros apm.launch. Nosotros hacemos conexión con PX4 por lo que se levanta con roslaunch mavros px4.launch.

Dependiendo de ello se tienen diferentes modos de vuelo, como se especifica aquí: http://wiki.ros.org/mavros/CustomModes#PX4_native_flight_stack.

El estado del dron se especifica en el tópico */mavros/state* con un mensaje del tipo *mavros_msgs/State* que se compone de:

- std_msgs/Header header
- bool connected
- bool armed
- bool guided
- bool manual_input
- string mode
- uint8 system_status

En el código, mediante el servicio */mavros/set_mode/* se especifica que se desea un *custom mode* OFFBOARD. Y mediante el servicio */mavros/cmd/amring* se manda armar el dron para poder volar. Esto se debe ver reflejado en el state del dron en la parte de *armed* y *mode*.

El servicio *set_mode* incluye varios modos predefinidos, que se pueden solicitar mediante su id, según su referencia en: http://docs.ros.org/en/noetic/api/mavros_msgs/html/srv/SetMode.html. Y los modos de vuelo en el estado del dron se pueden ver en: http://docs.ros.org/en/noetic/api/mavros_msgs/html/msg/State.html.

El system_status refiere al estado del robot:

Value	Field Name	Description
0	MAV_STATE_UNINIT	Uninitialized system, state is unknown.
1	MAV_STATE_BOOT	System is booting up.
2	MAV_STATE_CALIBRATING	System is calibrating and not flight-ready.
3	MAV_STATE_STANDBY	System is grounded and on standby. It can be launched any time.
4	MAV_STATE_ACTIVE	System is active and might be already airborne. Motors are engaged.

Value	Field Name	Description
5	MAV_STATE_CRITICAL	System is in a non-normal flight mode. It can however still navigate.
6	MAV_STATE_EMERGENCY	System is in a non-normal flight mode. It lost control over parts or over the whole airframe. It is in mayday and going down.
7	MAV_STATE_POWEROFF	System just initialized its power-down sequence, will shut down now.
8	MAV_STATE_FLIGHT_TERMINATION	System is terminating itself.