# **Interfacing the Simulation to ROS**

The simulated autopilot starts a second MAVLink interface on port 14557. Connecting MAVROS to this port allows to receive all data the vehicle would expose if in real flight.

## **Launching MAVROS**

If an interface to ROS is wanted, the already running secondary MAVLink instance can be connected to ROS via [mavros](http://dev.px4.io/ros-mavros-offboard.html). To connect to a specific IP (fcu\_url is the IP / port of SITL), use a URL in this form:

roslaunch mavros px4.launch fcu\_url:="udp://:14540@192.168.1.36:14557"

To connect to localhost, use this URL:

roslaunch mavros px4.launch fcu\_url:="udp://:14540@127.0.0.1:14557"

## **Launching Gazebo with ROS wrappers**

In case you would like to modify the Gazebo simulation to integrate sensors publishing directly to ROS topics e.g. the Gazebo ROS laser plugin, Gazebo must be launched with the appropriate ROS wrappers.

There are ROS launch scripts available to run the simulation wrapped in ROS:

* [posix\_sitl.launch](https://github.com/PX4/Firmware/blob/master/launch/posix_sitl.launch): plain SITL launch
* [mavros\_posix\_sitl.launch](https://github.com/PX4/Firmware/blob/master/launch/mavros_posix_sitl.launch): SITL and MAVROS

To run SITL wrapped in ROS the ROS environment needs to be updated, then launch as usual:

(optional): only source the catkin workspace if you compiled MAVROS or other ROS packages from source

cd <Firmware\_clone>

make posix\_sitl\_default gazebo

source ~/catkin\_ws/devel/setup.bash // (optional)

source Tools/setup\_gazebo.bash $(pwd) $(pwd)/build\_posix\_sitl\_default

export ROS\_PACKAGE\_PATH=$ROS\_PACKAGE\_PATH:$(pwd)

export ROS\_PACKAGE\_PATH=$ROS\_PACKAGE\_PATH:$(pwd)/Tools/sitl\_gazebo

roslaunch px4 posix\_sitl.launch

Include one of the above mentioned launch files in your own launch file to run your ROS application in the simulation.

### **What's happening behind the scenes**

(or how to run it manually)

no\_sim=1 make posix\_sitl\_default gazebo

This should start the simulator and the console will look like this

[init] shell id: 46979166467136

[init] task name: px4

\_\_\_\_\_\_ \_\_ \_\_ \_\_\_

| \_\_\_ \ \ \ / / / |

| |\_/ / \ V / / /| |

| \_\_/ / \ / /\_| |

| | / /^\ \ \\_\_\_ |

\\_| \/ \/ |\_/

Ready to fly.

INFO LED::init

729 DevObj::init led

736 Added driver 0x2aba34001080 /dev/led0

INFO LED::init

742 DevObj::init led

INFO Not using /dev/ttyACM0 for radio control input. Assuming joystick input via MAVLink.

INFO Waiting for initial data on UDP. Please start the flight simulator to proceed..

Now in a new terminal make sure you will be able to insert the Iris model through the Gazebo menus, to do this set your environment variables to include the appropriate sitl\_gazebo folders.

cd <Firmware\_clone>

source Tools/setup\_gazebo.bash $(pwd) $(pwd)/build\_posix\_sitl\_default

Now start Gazebo like you would when working with ROS and insert the Iris quadcopter model. Once the Iris is loaded it will automatically connect to the px4 app.

roslaunch gazebo\_ros empty\_world.launch world\_name:=$(pwd)/Tools/sitl\_gazebo/worlds/iris.world