robdos_sim Documentation

Release 0.1

Francisco J. Garcia R.

Aug 09, 2017

Contents

Simulations for robdos project, more info: http://www.robdosteam.com

CHAPTER 1

Contents:

Introduction

What is Robdos?

Robdos Team, an underwater robotics association, came up in 2014 from our concern as a group of students specialized in different fields about putting into practice all the skills and knowledge gathered during our university period. Marine, Industrail and Computer Engineering students from the Technical University of Madrid (Universidad Politecnica de Madrid), work hard on the development of an autonomous underwater platform.

Our main goal in the Association is to manufacture and develop an autonomous vehicle in order to attend international competitions, which are becoming more and more frequent over the years. Such events allow different universities and private teams show the progress made in their platforms throughout the years, and thus YUooperating with technological development in underwater robotics.

Install ROS and dependencies

Follow ros installation procedure:

http://wiki.ros.org/kinetic/Installation/Ubuntu

we can summarize the steps:

Open a command windows on ubuntu and run the following commands:

- Prepare ubuntu for installation:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /
↪etc/apt/sources.list.d/ros-latest.list'
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 0xB01FA116
sudo apt-get update
```

- Install ROS

For PC/Laptop we should install full desktop version:

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

For Raspberry PI 2 and Odroid we can install ROS-Base:

```
sudo apt-get install ros-kinetic-ros-base
```

Install rosdep:

```
sudo rosdep init
rosdep update
```

And finally prepare environment:

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

Compile project

First we need to install some dependencies and ROS packages:

```
sudo apt-get install libqwt-dev ros-kinetic-teleop-twist-joy  ros-kinetic-rviz-imu-
↪plugin python-smbus ros-kinetic-rqt-multiplot ros-kinetic-mavros*
```

Finally we create a workspace for project, clone github repository, install dependencies and compile it:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
git clone https://gitlab.com/SIMULACION_SUBMARINA/robdos_sim -b kinetic
cd ..
source devel/setup.bash
rosdep install robdos_sim
catkin_make
```

Test the project

Once the project has been compiled successfully, we have two different alternatives. can run a simulation that includes robot using a simple scenario.

```
cd ~/catkin_ws
source devel/setup.bash
roslaunch robdos_sim robdos_sim.launch
```

Launch the project

Once we have compiled and tested the project, we can launch the project in our computer or in the robot.

The first thing we must do is to connect a PS3 Sixaxis to out computer.

Then, if we want to launch it in our computer:

```
cd ~/catkin_ws
catkin_make_isolated
source devel/setup.bash
roscore
rosparam set joy_node/dev "/dev/input/js1"
rosrun joy joy_node
roslaunch robdos_sim robdos_state_machine.launch
```

On the other hand, if you want to control the robot, the first thing we must do is to connect our computer and the NUC that is on the AUV to the same network.

Secondly, we have to add the next lines into the file ~/.bashrc:

```
export ROS_IP=192.168.1.2
export ROS_MASTER_URI=http://192.168.1.101:11311/
```

Then, we must follow the next commands:

```
#Our computer
ping 192.168.1.101
ssh robdosteam@192.168.1.101
#Contraseña: ***********

#NUC computer
roscore
cd ~/robdos_ws
source devel_isolated/setup.bash
roslaunch robdos_state_machine on_board_architecture.launch

#Our computer
robdos_state_machine ground_architecture.launch

#It is not necessary to do the next commands but it allows us to change some PixHawk␣
↪values
rosservice call /mavros/set_stream_rate 0 10 1
rosservice call /mavros/cmd/arming 1
```

It is important to do the above steps in the order mentioned.

Robdos Architecture

This section explains how the code is structured and the usefulness of the interfaces that you can see when you launch the code.

So as to be able to understand and see the code, you must follow the steps explained in the previous section.

Code Structure

1. **Robdos Autonomous**

It includes the control of the AUV. It is used in both the real robot and the the simulates one. The inputs that we receive so that we can control the robot are: Odometry (Position and Orientation) and the waypoint desired. Last, it is important to mentioned that the linear and angular movement are done at the same time.

2. **Robdos Dependancies**

This package includes the dependancies (usb_cam, mavros, ...) necessary to compile the project. This package has been done in the case one of the packages gets upgraded and, this impedes us to be able to compile the project.

3. **Robdos Dynamics**

This is a package that we use in our as an alternative to the plugin to Gazebo that I will explain later. It receives the velocities we are sending to the thrusters of the submarine, transforms them to the forces and determines the position of the robot. Then, the node sends the position to Gazebo and it represents it.

4. **Robdos Safety**

This package checks all sensors from the AUV and acts in consequence of the value of those sensors. The sensors it reads are humidity, temperature and battery.

5. **Robdos Sim**

6. **Robdos State Machine**

This package includes the state machine that has been done with the tool Smach. We can move through the different states, depending on the mission we want to do.

7. **Robdos Station**

It shows a graphical user interface (GUI) that reads the value of the topics and monitors some of them.

8. **Robdos Teleop**

It converts the input of the joy into the input of the PixHawk. The buttoms we must press are mentioned above.

9. **Robdos Utils**

10. **Robdos Vision**

11. **Robdos Visualization**

Code format

So as to be able to have an standardized code, we must follow the rules mentioned in:

http://wiki.ros.org/PyStyleGuide

http://wiki.ros.org/CppStyleGuide

In addition, we have to:

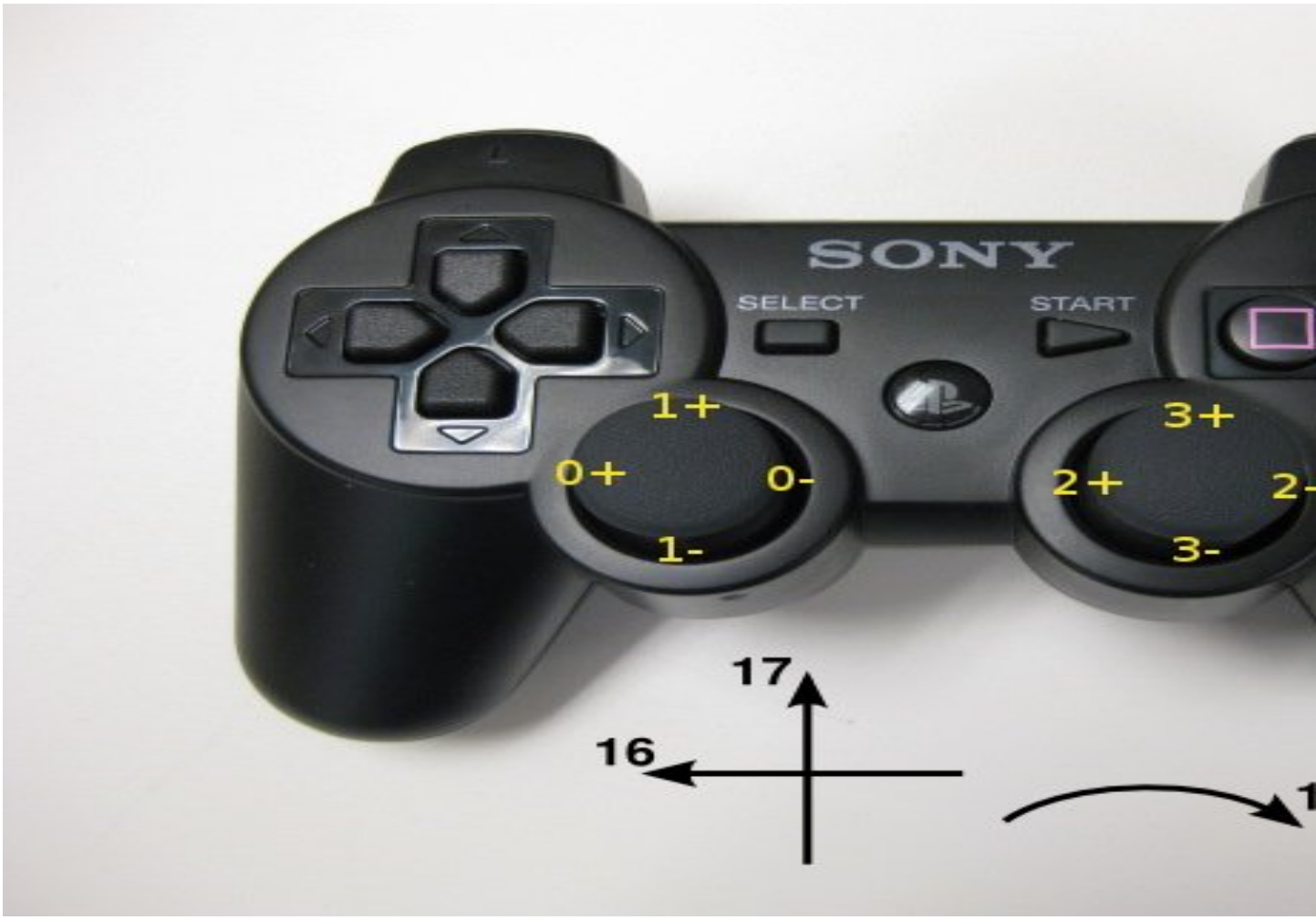
- Name topics as with /robdos/NameTopic
- Start package names with robdos

Joy

http://wiki.ros.org/ps3joy


SONY
SELECT
START
4
7
5
6
0
3
15
16
1
2


11
9
18
START
SELECT
SONY
DUALSHOCK 3
SIXAXIS
1 2 3 4

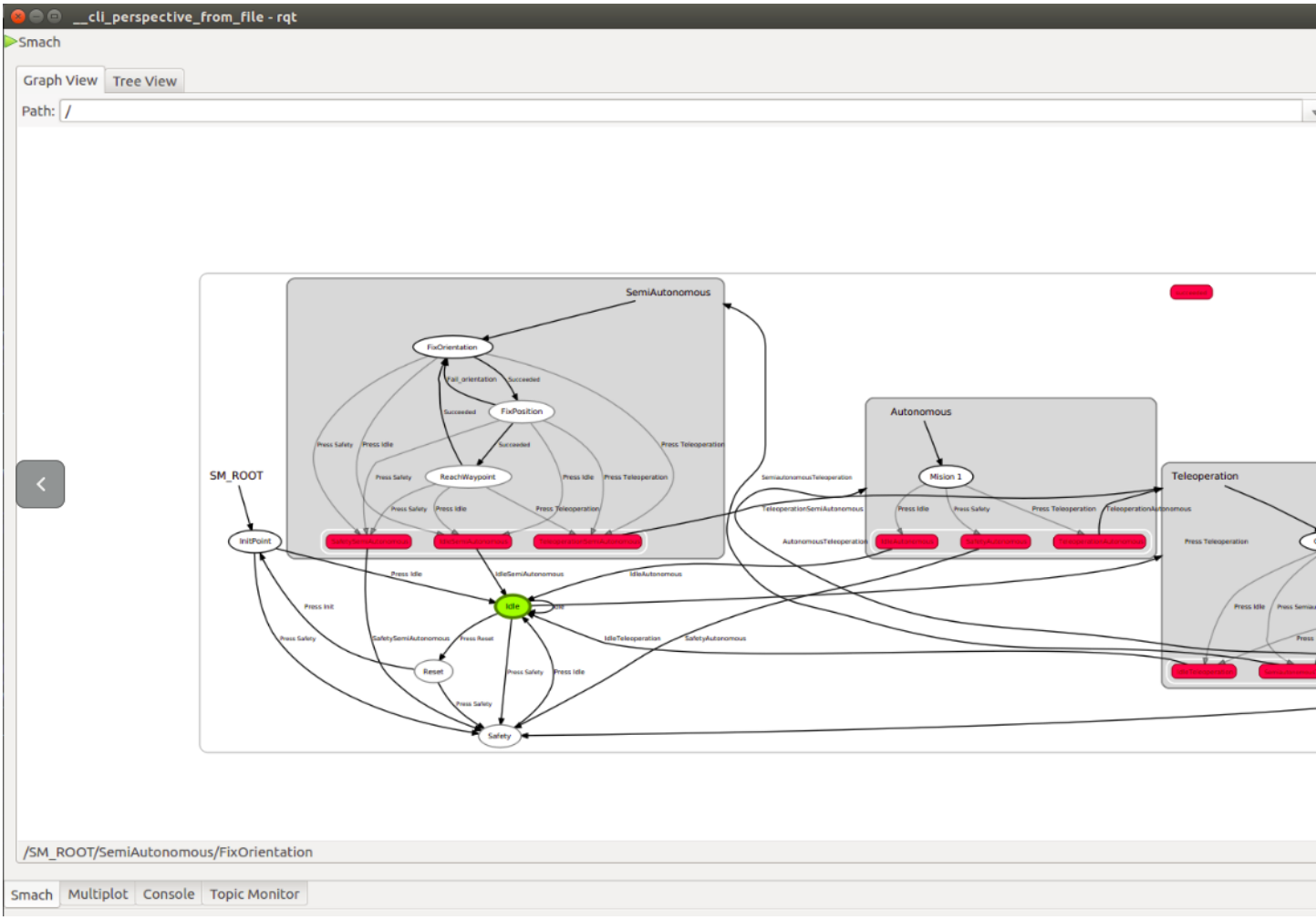


The possible commands we can use to control the robot are:

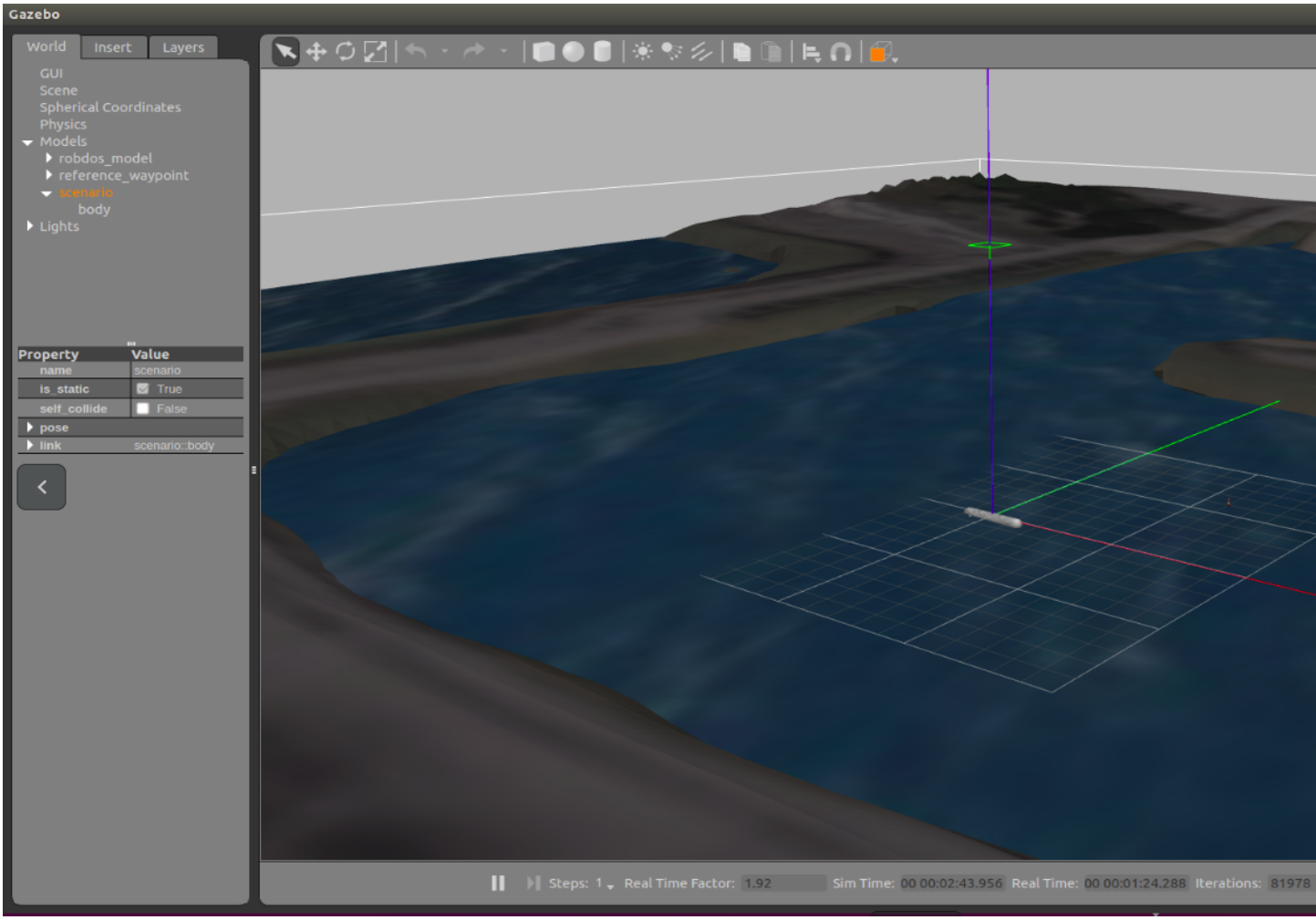
Buttoms	Consequence
3+	Forward
3-	Backward
0+	Yaw clockwise
0-	Yaw anti-clockwise
10 & 14	Arm
10 & 15	Disarm
10 & 12	Change teleop
8 & 9	Change semiautonomous

Interfaces

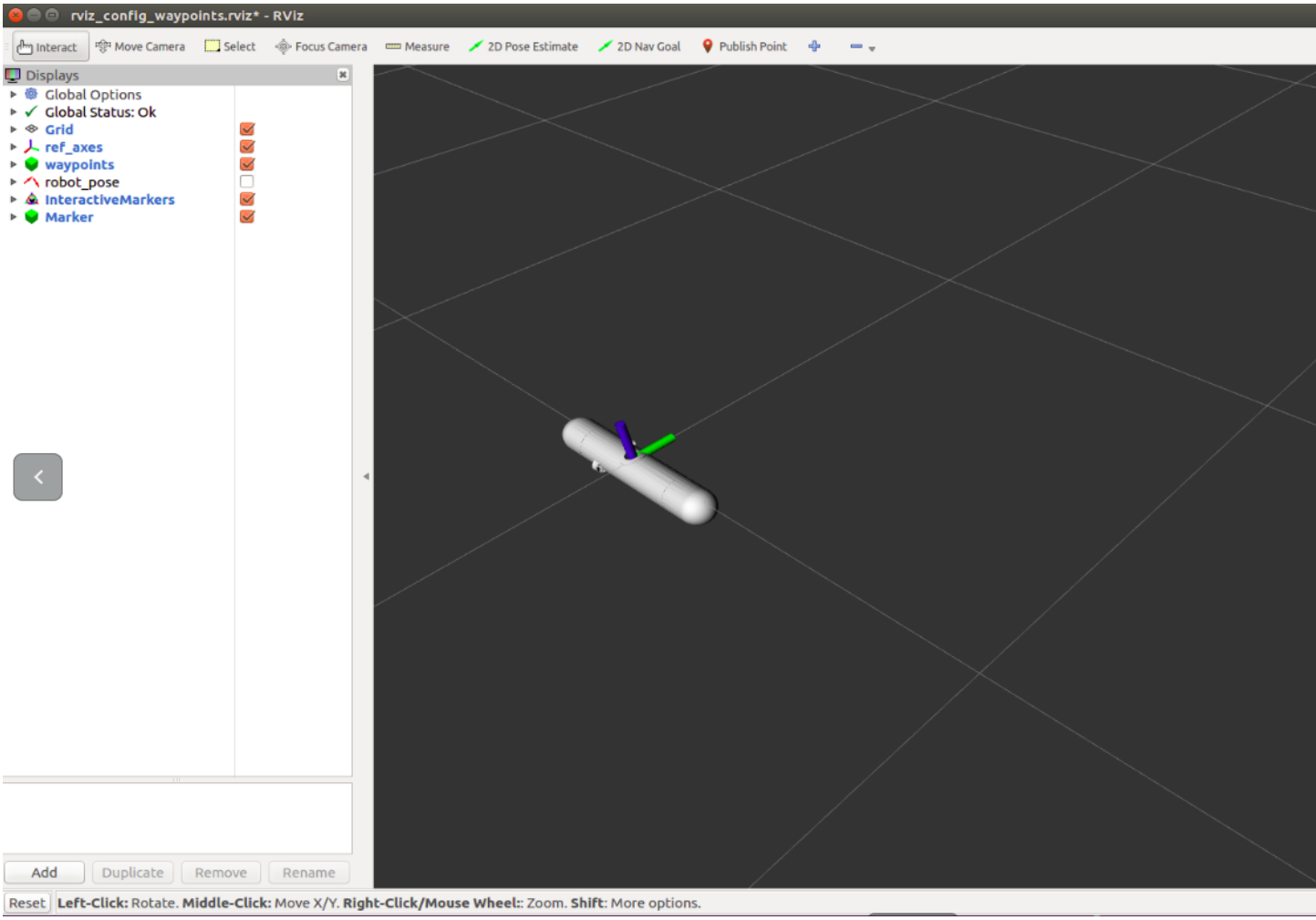
This is the interface that shows us the states available in the statemachine. The green one indicates the one we are at.



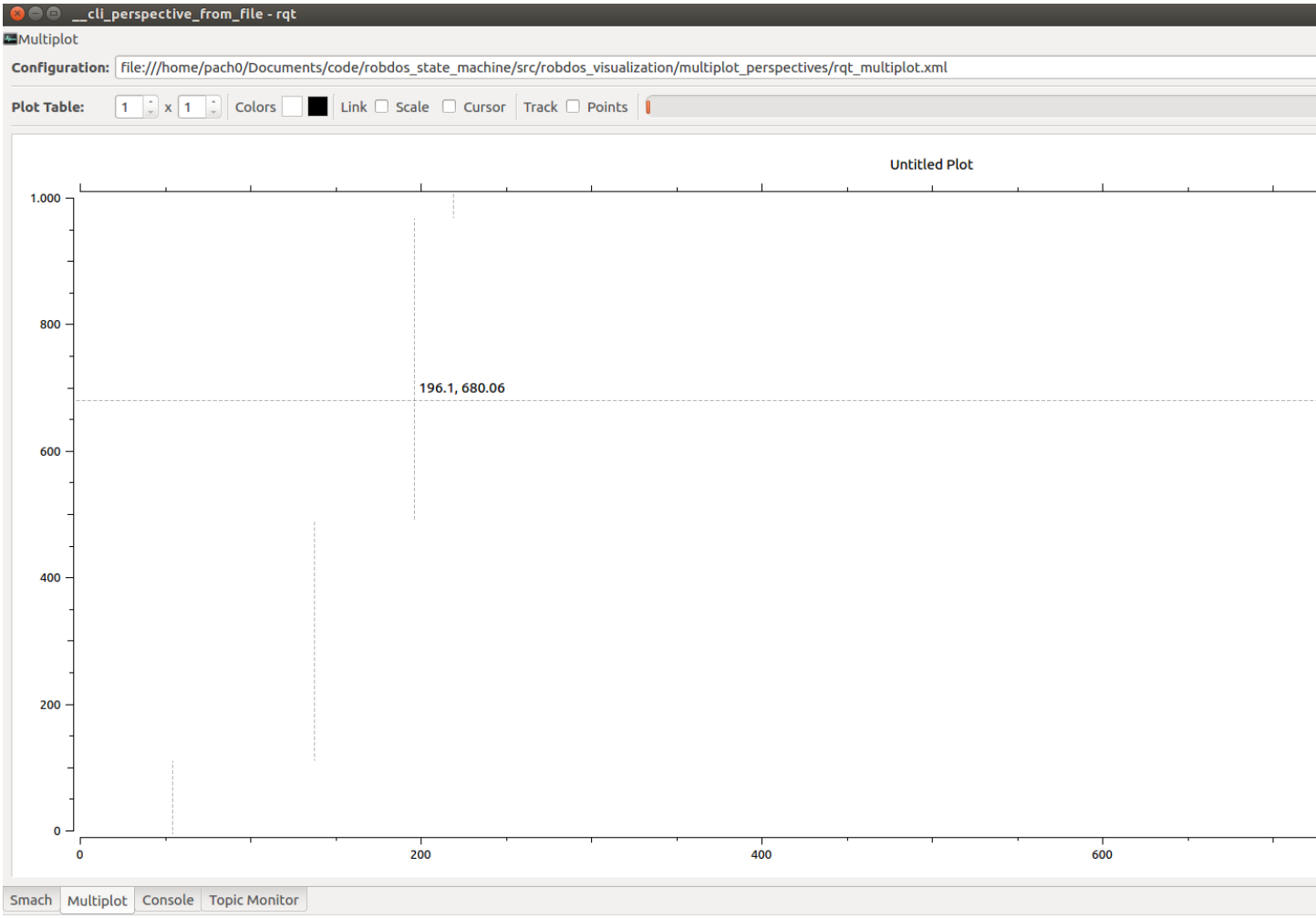
Gazebo is the interface that we use to represent the position of the robot when we are simulating it. We have also included a world similar to the one we will compete at.



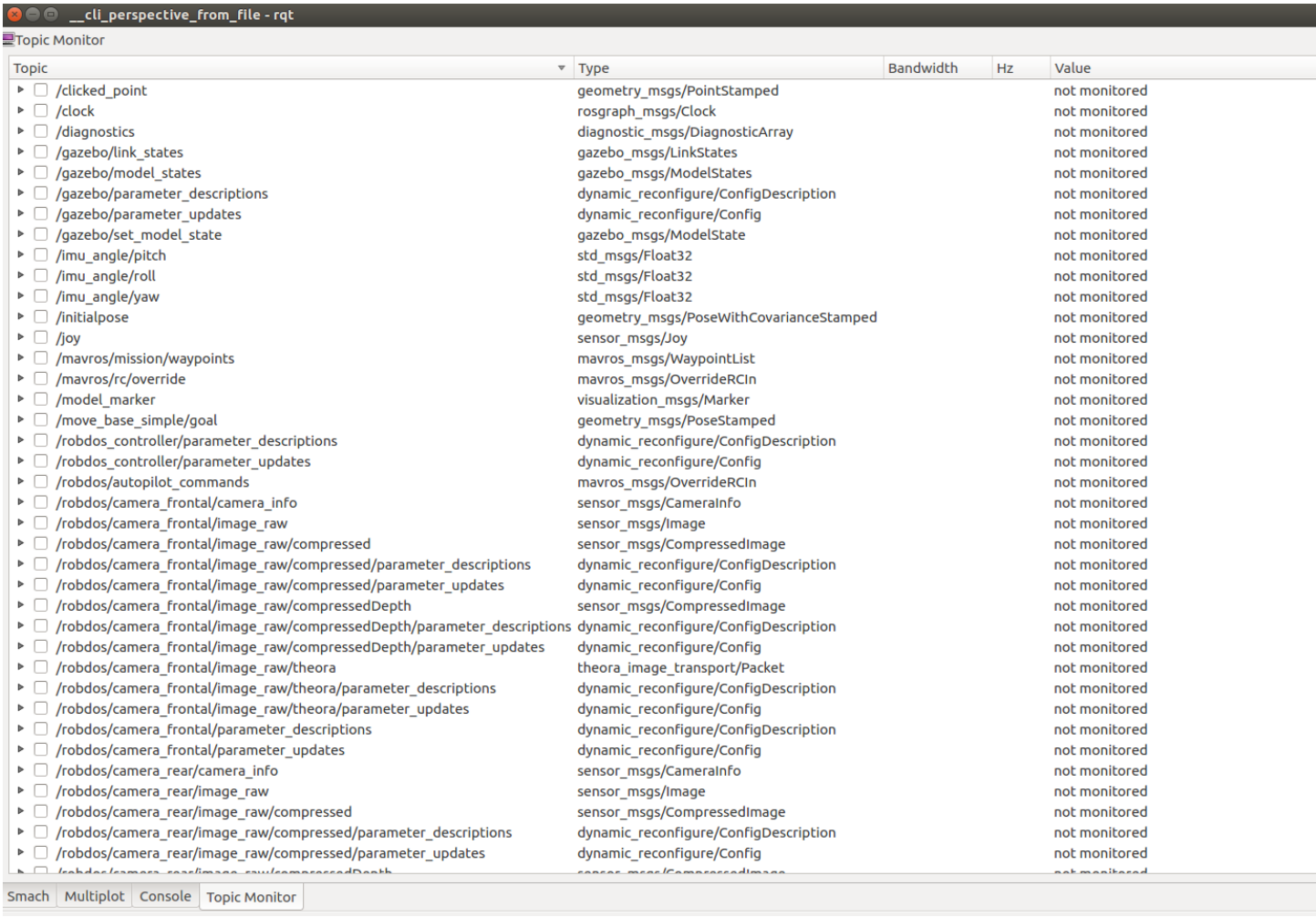
We use Rviz to send the robot to a point (with the ball shown in the image) and, also, to see the position of the robot.



It shows us different graphs about how different variables change with the time.



It indicates all topics and the value they are having when we launch the project.



Launch files

The launch available in the code are:

The ones that we use in order to simulate or control the real robot are:

- Simulated robot:

```
robdos_gazebo_dynamics.launch
robdos_simulated_dynamics.launch
```

- Real Robot:

List of topics

Topics of the PixHawk:

Topics	msg	Description
/mavlink/from	mavros_msgs::Mavlink	
/mavlink/to	mavros_msgs::Mavlink	

Continued on next page

Table 1.1 – continued from previous page

Topics	msg	Description
/mavros/battery	mavros__msgs::BatteryStatus	
/mavros/cam_imu_sync/cam_imu_stamp	mavros__msgs::CamIMUStamp	
/mavros/extended_state	mavros__msgs::ExtendedState	
/mavros/global_position/compass_hdg	std_msgs::Float64	
/mavros/global_position/global	sensor_msgs::NavSatFix	
/mavros/global_position/local	nav_msgs::Odometry	
/mavros/global_position/raw/fix	sensor_msgs::NavSatFix	
/mavros/global_position/raw/gps_vel	geometry_msgs::TwistStamped	
/mavros/global_position/rel_alt	std_msgs::Float64	
/mavros/hil_controls/hil_controls	mavros__msgs::HilControls	
/mavros/imu/atm_pressure	sensor_msgs::FluidPressure	
/mavros/imu/data	sensor_msgs::Imu	
/mavros/imu/data_raw	sensor_msgs::Imu	
/mavros/imu/mag	sensor_msgs::MagneticField	
/mavros/imu/temperature	sensor_msgs::Temperature	
/mavros/local_position/odom	nav_msgs::Odometry	
/mavros/local_position/pose	geometry_msgs::PoseStamped	
/mavros/local_position/velocity	geometry_msgs::TwistStamped	
/mavros/manual_control/control	mavros__msgs::ManualControl	
/mavros/mission/waypoints	mavros_msgs::WaypointList	
/mavros/radio_status	mavros__msgs::RadioStatus	
/mavros/rc/in	mavros__msgs::RCIn	
/mavros/rc/out	mavros__msgs::RCOut	
/mavros/rc/override	mavros_msgs::OverrideRCIn	
/mavros/setpoint_accel/accel	geometry_msgs::Vector3Stamped	
/mavros/setpoint_position/local	geometry_msgs::PoseStamped	
/mavros/setpoint_raw/attitude	mavros__msgs::AttitudeTarget	
/mavros/setpoint_raw/global	mavros__msgs::GlobalPositionTarget	
/mavros/setpoint_raw/local	mavros__msgs::PositionTarget	
/mavros/setpoint_raw/target_attitude	mavros__msgs::AttitudeTarget	
/mavros/setpoint_raw/target_global	mavros__msgs::PositionTarget	
/mavros/setpoint_raw/target_local	mavros__msgs::PositionTarget	
/mavros/setpoint_velocity/cmd_vel	geometry_msgs::TwistStamped	
/mavros/state	mavros__msgs::State	
/mavros/time_reference	sensor_msgs::TimeReference	
/mavros/vfr_hud	mavros__msgs::VFR_HUD	
/mavros/wind_estimation	geometry_msgs::TwistStamped	

Othe topics:

Topics	msg	Description
/diagnostics	diagnostic_msgs::DiagnosticArray	
/joy	sensor_msgs::Joy	
/rosout	rosgraph_msgs::Log	

Robdos Git

This section explains the git repositories Robdos owns.

Our Git repositories

https://gitlab.com/ROBDOS_TEAM/robdos_ws

https://github.com/francisc0garcia/robdos_documentation

How to commit

```
    git pull
git status
git add .
git commit -m "NameCommit"
git push
```

Other git commands

*Create a new branch

```
git branch NameBranch
```

*Move from one branch to another

```
git checkout NameBranch
```

Format commit

The commits must include the descriptions of what it has been done. The extension will depend on how things it has been changed.

Robdos robot tutorials

This section explain some examples of how to use the gazebo simulation and ROS in order to play with the robdos robot and develop some extensions.

this section assume you already have installed robdos project.

Play bag files

In order to understand how variables changes inside the robot, let's play with a recorded bag files.

Remember download first the bag files in your computer and change the path into launch file (robdos_sim_play_bags.launch)

```
cd ~/catkin_ws
source devel/setup.bash
roslaunch robdos_sim robdos_sim_play_bags.launch
```

Once the launch file have started, a RVIZ window will open and show the movement of robot.

Waypoints using gazebo and RVIZ

You can execute a simple demo using waypoints (mavros_msgs/WaypointList), a simulated environment in gazebo and RVIZ.

```
cd ~/catkin_ws
source devel/setup.bash
roslaunch robdos_sim robdos_waypoints.launch
```

This command will open 2 main windows: a RVIZ viewer for waypoint visualization and a gazebo window with a simulated robot that execute the proper velocity commands.

The previous demo can be summarized in the following processes:

- Waypoint publisher

A python script (waypoint_publisher.py), it publishes a topic /mavros/mission/waypoints (type mavros_msgs/WaypointList) with a set of points to be reached by the robot.

it generates also a set of markers using topic /robdos/makers_waypoints (type visualization_msgs/Marker) for visualization using RVIZ.

- Waypoint controller

A python script (waypoint_controller.py), it publishes a velocity command for mavlink using topic /mavros/rc/out (type mavros_msgs/RCOut).

It subscribes for odometry information (position/orientation) to /robdos/odom (type nav_msgs/Odometry) (from gazebo simulation but also possible to subscribe for real odometry topic)

It also subscribes for waypoint list /mavros/mission/waypoints in order to perform a simple controller.

This demo only considers the first waypoint, but it will include the other coming soon.

- Gazebo plugin (gazebo_sub_drive)

In order to perform a realistic simulation of the robot, we have develop gazebo plugin that publishes simulated odometry and subscribes for desired velocity commands (mavros_msgs/RCOut).

This node convert RCOut values into valid angular velocity for thrusters, then, the Lift-Drag plugin and gazebo computes the resulting force over the robot.

Regression tests

This section includes the tests that ensure that our new code does not have any conflict with the old one.

Robdos lessons

Ardusub

So as to download the Ardusub code:

```
sudo apt-get -qq -y install git
git clone https://github.com/bluerobotics/ardusub.git
cd ardusub
git submodule update -- init -- recursive
Tools/scripts/install-prereqs- ubuntu.sh -y
. ~/.profile
```

```
cd ArduSub/
make px4-v2
```

Configure Pixhawk with QGroundControl

QGroundControl is a tool that is used as a control computer for drones and other robots. It is prepared to connected to the PixHawk and it allows you to modify its firmware easily and calibrate it.

1. **Instalation**

Access the next page in order to download the program.

https://s3-us-west-2.amazonaws.com/qgroundcontrol/latest/QGroundControl.AppImage

Then, follow the next commands:

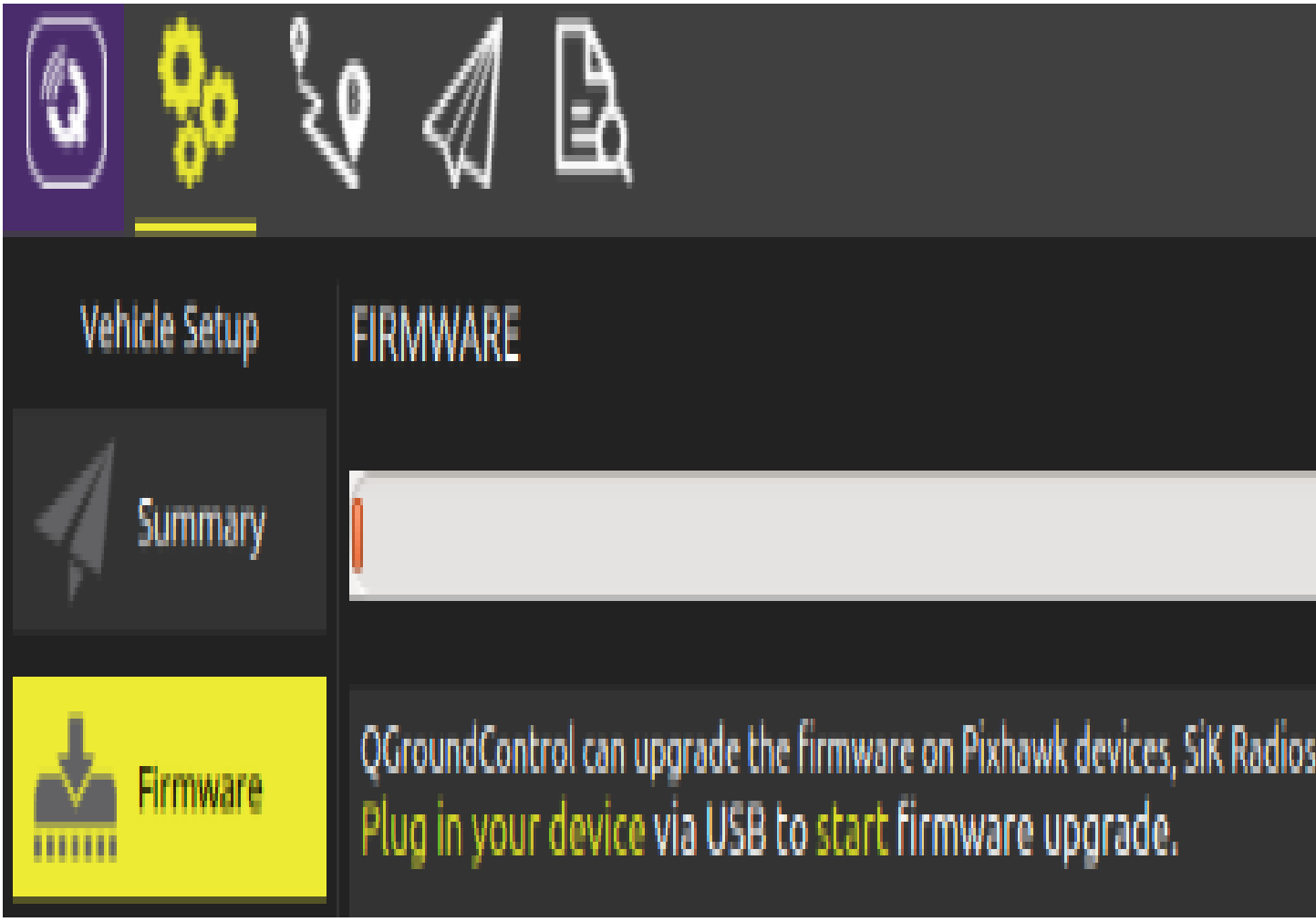
```
sudo chmod +x QgroundControl.AppImage
sudo usermod -a -G dialout <USER>
sudo apt-get remove modemmanager
./QgroundControl.AppImage
```

All the steps mentioned and how you can install QGroundControl can be found in the next link:

https://docs.qgroundcontrol.com/en/getting_started/download_and_install.html

2. **Load Firmware**

In order to load the firmware, the first thing we must do is to open the QGroundControl and go to "Settings -> Firmware".



After we connect the PixHawk, we will be asked about what firmware you want to upload. In our case we have to have to click "Custom" and choose our compiled workspace.

If we can see a blinking green light and nothing appears in the QGroundControl interface when we have connected the PixHawk to our computer, we have to disconnect and connect again.

3. **Calibration**

Every time that we upload a new firmware, the best thing is to recalibrate the PixHawk so as to prevent ourselves from future problems. In order to calibrate the PixHawk we must go to "Settings -> Sensors". In that path we should be able to see the sensors there are available and a green point if they are calibrated. If this is not the case, you can press on top and click, on the side, to the bottom "Calibration". In the main screen we will see the calibration instructions.

Here there is a video that shows how we can calibrate the PixHawk. Even if it uses another program, the steps are the same ones:

https://www.youtube.com/watch?v=uH2iCRA9G7k (Minute 16:44)

4. **Set parameters**

When we compile we do not say what kind of submarine we have so it is necessary to go to "Settings -> Frame" and choose the position of the thrusters you want. In Robdos Team we are using SimpleROV2.

As a verification that everything is working well, we can press in the top bottom that seems like a paper plane. It should appeared a localization and a lecture of some of the sensors and it should allow us to arm and disarm the PixHawk. If everything is well calibrated, we should be able to arm without any problems.

If we have changed the parameters in the PixHawk, it is necessary to restart it so they work right (This is because some of them are loaded at the beginning of the program and any other modification does not affect them.)

Contact us

- If you want more info about the project:
  - Francisco J. Garcia R. - garciar@rhrk.uni-kl.de
  - Javier Fernandez - javier.f3rnand3z@gmail.com
- robdos simulation license:

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

This project is released under GPLv2 license, please consider that external plugins may have a different type of license.

Developed by:

- Francisco J. Garcia R. - garciar@rhrk.uni-kl.de
- Javier Fernandez - javier.f3rnand3z@gmail.com