

ROS 2 Workshop

Marcelo Jacinto
(mjacinto@isr.tecnico.ulisboa.pt)



LARSyS

Laboratory of Robotics
and Engineering Systems

Motivation

ROS 2 is useful if you ever dream of ruling the world with an army of drones...



Why ROS 2

2 > 1



What is ROS 2

Not an operating system! (they use the name as clickbait)

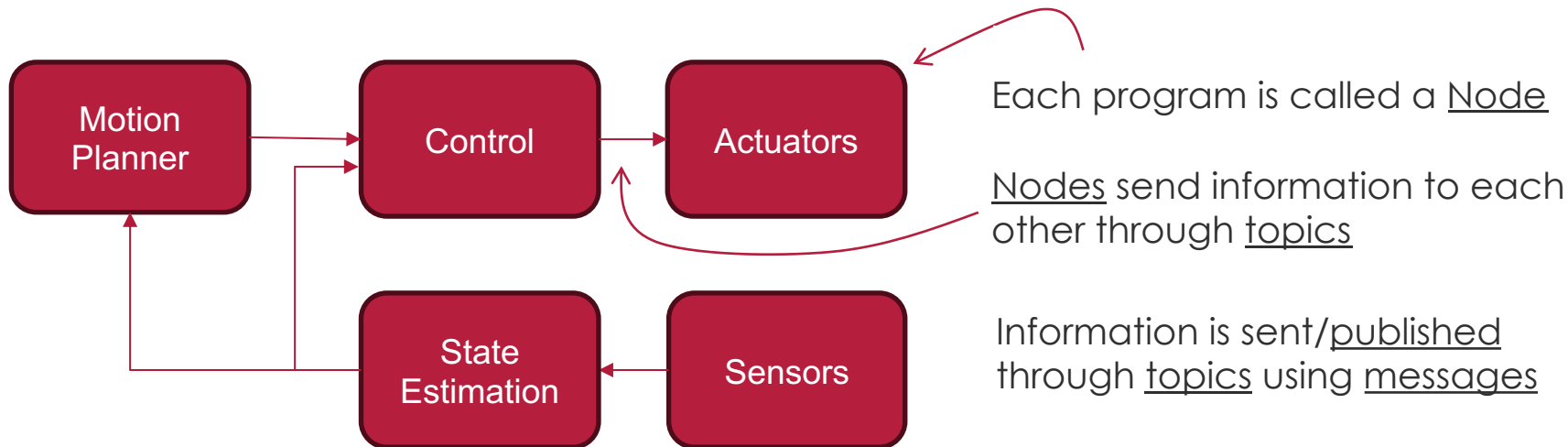
“The Robot Operating System (ROS) is a set of software libraries and tools for building robot applications.”

ROS 2 provides:

- a set of useful programs that run in your computer
- a communication system for programs to share data with each other

Communication Structure

Problem Formulation/Motivation: You want to design a Guidance, Control and Navigation (GNC) system for a drone, such that each block of the system is implemented as an individual program (written in Python or C++).



Installing ROS 2

Distro	Release date	Logo	EOL date
iron irwini	May 23rd, 2023		November 2024
Humble Hawksbill	May 23rd, 2022		May 2027
Galactic Geochelone	May 23rd, 2021		December 9th, 2022
Foxy Fitzroy	June 5th, 2020		June 20th, 2023

Follow the steps on the links below:

Ubuntu 22.04LTS

Official Documentation:

<https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html>

Ubuntu 20.04LTS

Official Documentation:

<https://docs.ros.org/en/foxy/Installation/Ubuntu-Install-Debians.html>

Installation tutorials on YouTube: <https://youtu.be/0aPbWsyENA8?t=122&si=CbwtHPBpd0aOQdl->

After Installing ROS 2

Run the following line on the terminal

```
source /opt/ros/humble/setup.bash >> ~/.bashrc
```

↪ foxy (if using Ubuntu 20.04LTS)

Tell your terminal where
ROS 2 is installed

Also install colcon build tools

```
sudo apt install python3-colcon-common-extensions
```

```
echo "source /usr/share/colcon_argcomplete/hook/colcon-argcomplete.bash" >> ~/.bashrc
```

```
echo "source /usr/share/colcon_cd/function/colcon_cd.sh" >> ~/.bashrc
```

```
echo "export _colcon_cd_root=/opt/ros/humble/" >> ~/.bashrc
```

↪ foxy (if using Ubuntu 20.04LTS)

Convenience tools such as
auto-complete in the terminal

Getting Started

Create a folder/workspace where the code will live.

```
mkdir drone_ws
```

Inside the workspace create a src folder/directory

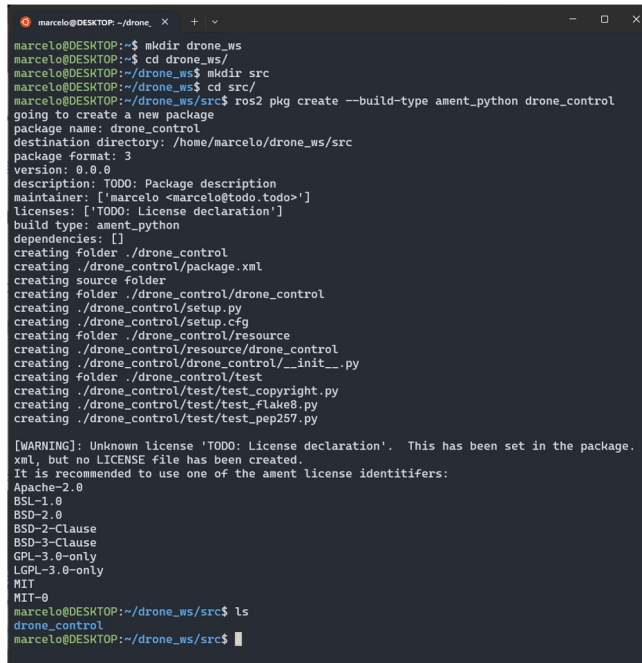
```
cd drone_ws
```

```
mkdir src
```

Inside the src directory create a Python code package

```
cd src
```

```
ros2 pkg create --build-type ament_python drone_control
```



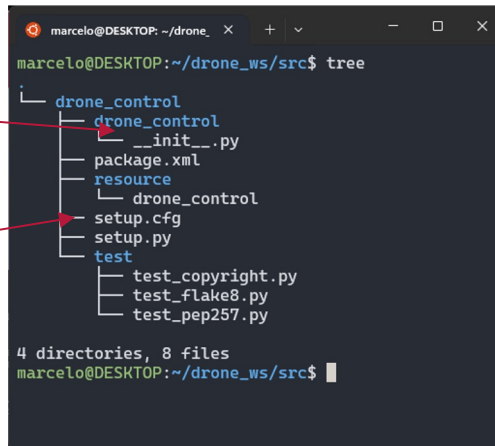
```
marcelo@DESKTOP: ~/drone_ws
marcelo@DESKTOP:~$ mkdir drone_ws
marcelo@DESKTOP:~$ cd drone_ws/
marcelo@DESKTOP:~/drone_ws$ mkdir src
marcelo@DESKTOP:~/drone_ws$ cd src/
marcelo@DESKTOP:~/drone_ws/src$ ros2 pkg create --build-type ament_python drone_control
going to create a new package
package name: drone_control
destination directory: /home/marcelo/drone_ws/src
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['marcelo <marcelo@todo.todo>']
licenses: ['TODO: License declaration']
build type: ament_python
dependencies: []
creating folder ./drone_control
creating ./drone_control/package.xml
creating source folder
creating folder ./drone_control/drone_control
creating ./drone_control/setup.py
creating ./drone_control/setup.cfg
creating folder ./drone_control/resource
creating ./drone_control/resource/drone_control
creating ./drone_control/drone_control/__init__.py
creating folder ./drone_control/test
creating ./drone_control/test/test_copyright.py
creating ./drone_control/test/test_flake8.py
creating ./drone_control/test/test_pep257.py

[WARNING]: Unknown license 'TODO: License declaration'. This has been set in the package.
xml, but no LICENSE file has been created.
It is recommended to use one of the ament license identifiers:
Apache-2.0
BSL-1.0
BSD-2.0
BSD-2-Clause
BSD-3-Clause
GPL-3.0-only
LGPL-3.0-only
MIT
MIT-0
marcelo@DESKTOP:~/drone_ws/src$ ls
drone_control
marcelo@DESKTOP:~/drone_ws/src$
```


Folder/Package Structure (Python)

Write your code here

Inform ROS 2 where
your “main” is in the
setup.py file



```
marcelo@DESKTOP: ~/drone_ws/src$ tree
.
├── drone_control
│   ├── drone_control
│   ├── __init__.py
│   ├── package.xml
│   └── resource
│       ├── drone_control
│       ├── setup.cfg
│       ├── setup.py
│       └── test
│           ├── test_copyright.py
│           ├── test_flake8.py
│           └── test_pep257.py
4 directories, 8 files
marcelo@DESKTOP: ~/drone_ws/src$
```

The terminal window displays the output of the 'tree' command for a ROS 2 package named 'drone_control'. The package structure is as follows:

- drone_control
 - drone_control
 - __init__.py
 - package.xml
 - resource
 - drone_control
 - setup.cfg
 - setup.py
 - test
 - test_copyright.py
 - test_flake8.py
 - test_pep257.py

4 directories, 8 files

Creating a Node

Python file
for node
+
Python file
for logic

Timer to periodically
compute the control signal

ROS 2 library imports

Reading parameters
dynamically

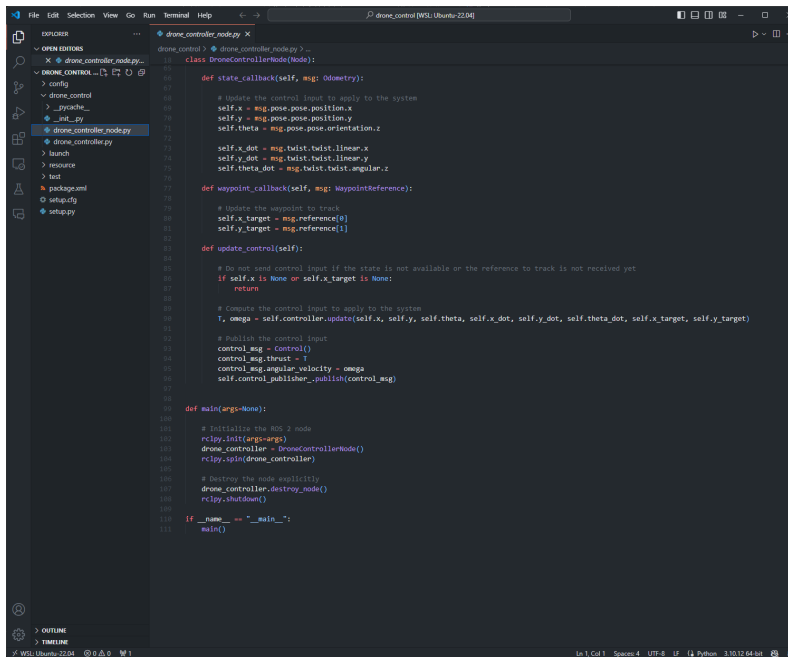
Creating the
publishers/subscribers

```

1 #!/usr/bin/env python3
2
3 # Import the ROS 2 libraries
4 import rclpy
5 from rclpy.node import Node
6 from rclpy.qos import qos_profile_sensor_data, qos_profile_system_default
7
8 # Import the ROS 2 message to received the current state of the drone
9 from nav_msgs.msg import Odometry
10
11 # Import my custom message to send the control to apply to the drone
12 from drone_msgs.msg import Control, WaypointReference
13
14 # Create a class that inherits the Node object
15 class DroneController(Node):
16
17     def __init__(self):
18         super().__init__('drone_simulation_node',
19                          allow_unDeclared_parameters=True,
20                          automatically_declare_parameters_from_overrides=True)
21
22         # Get the mass, inertia and frequency from the configurations
23         mass = self.get_parameter('drone_controller.mass').get_parameter_value().double_value
24         frequency = self.get_parameter('drone_controller.frequency').get_parameter_value().double_value
25
26         # Get the gains of the controller
27         kp_pos = self.get_parameter('drone_controller.position.kp').get_parameter_value().double_array_value
28         kd_pos = self.get_parameter('drone_controller.position.kd').get_parameter_value().double_array_value
29         kp_theta = self.get_parameter('drone_controller.attitude.kp').get_parameter_value().double_value
30         kd_theta = self.get_parameter('drone_controller.attitude.kd').get_parameter_value().double_value
31
32         # Use the parameters
33         self.get_logger().info('Drone mass: {}'.format(mass))
34         self.get_logger().info('Drone position gains: {}'.format(kp_pos, kd_pos))
35         self.get_logger().info('Drone attitude gains: {}'.format(kp_theta, kd_theta))
36         self.get_logger().info('Drone frequency: {}'.format(frequency))
37
38         # Create a object that dynamically simulates the dynamics of a drone
39         self.controller = Controller(kp_pos, kd_pos, kp_theta, kd_theta, mass)
40
41         # Get the current state of the system
42         self.x = None
43         self.y = None
44         self.theta = None
45         self.x_dot = None
46         self.y_dot = None
47         self.theta_dot = None
48
49         # Get the desired waypoint
50         self.x_target = None
51         self.y_target = None
52
53         # Publisher to the control inputs of the system
54         self.control_publisher = self.create_publisher(Control, 'input', qos_profile_sensor_data)
55
56         # Subscriber to the current state of the system and for the reference support for the controller to track
57         self.state_subscriber = self.create_subscription(Odometry, 'state', self.state_callback, qos_profile_system_default)
58         self.waypoint_subscriber = self.create_subscription(WaypointReference, 'waypoint', self.waypoint_callback, qos_profile_system_default)
59
60         # Create a timer that will generate the control signal of the simulation
61         self.timer = self.create_timer(1.0/frequency, self.update_control)

```

Creating a Node (2)



```

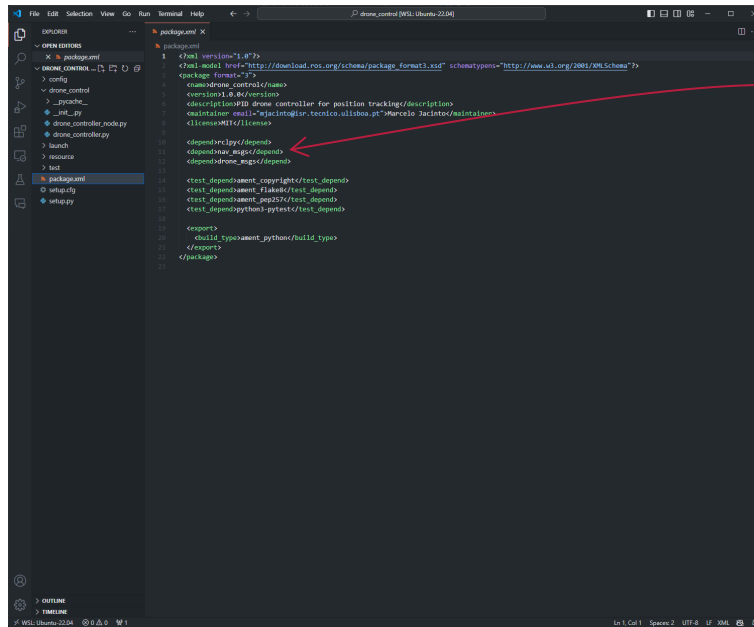
1  class DroneControllerNode(Node):
2
3      def state_callback(self, msg: Odometry):
4          # Update the control input to apply to the system
5          self.x = msg.pose.pose.position.x
6          self.y = msg.pose.pose.position.y
7          self.theta = msg.pose.pose.orientation.z
8
9          self.x_dot = msg.twist.twist.linear.x
10         self.y_dot = msg.twist.twist.linear.y
11         self.theta_dot = msg.twist.twist.angular.z
12
13     def waypoint_callback(self, msg: WaypointReference):
14         # Update the waypoint to track
15         self.x_target = msg.reference[0]
16         self.y_target = msg.reference[1]
17
18     def update_control(self):
19         # Do not send control input if the state is not available or the reference to track is not received yet
20         if self.x is None or self.x_target is None:
21             return
22
23         # Compute the control input to apply to the system
24         T, omega = self.controller.update(self.x, self.y, self.theta, self.x_dot, self.y_dot, self.theta_dot, self.x_target, self.y_target)
25
26         # Publish the control input
27         control_msg = Control()
28         control_msg.thrust = T
29         control_msg.angular_velocity = omega
30         self.control_publisher.publish(control_msg)
31
32     def main(args=None):
33         # Initialize the ROS 2 node
34         rclpy.init(args=args)
35         drone_controller = DroneControllerNode()
36         rclpy.spin(drone_controller)
37
38         # Destroy the node explicitly
39         drone_controller.destroy_node()
40         rclpy.shutdown()
41
42     if __name__ == "__main__":
43         main()

```

Subscriber callbacks
are called when a new
message arrives

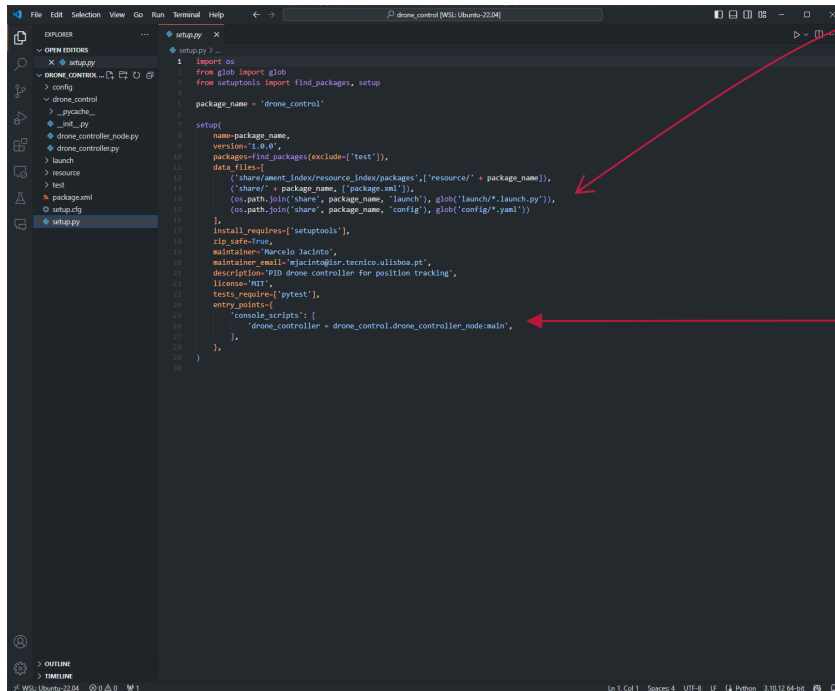
Function called by the
timer to compute and
publish the control
signal

Creating a Node (3)



Add the used
packages to the
package.xml file

Creating a Node (4)



```
1 import os
2 from glob import glob
3 from setuptools import find_packages, setup
4
5 package_name = 'drone_control'
6
7 setup(
8     name=package_name,
9     version='1.0.0',
10    packages=find_packages(exclude=['test']),
11    data_files=[
12        ('share/ament_index/resource_index/packages', ['resource/' + package_name]),
13        ('share/' + package_name, ['package.xml']),
14        (os.path.join('share', package_name, 'launch'), glob('launch/*_launch.py')),
15        (os.path.join('share', package_name, 'config'), glob('config/**.yaml')),
16    ],
17    install_requires=['setuptools'],
18    zip_safe=True,
19    maintainer='Marcelo Jacinto',
20    maintainer_email='mjacinto@tecnico.ulisboa.pt',
21    description='PID drone controller for position tracking',
22    license='MIT',
23    tests_require=['pytest'],
24    entry_points={
25        'console_scripts': [
26            'drone_controller = drone_control.drone_controller_node:main',
27        ],
28    },
29 )
```

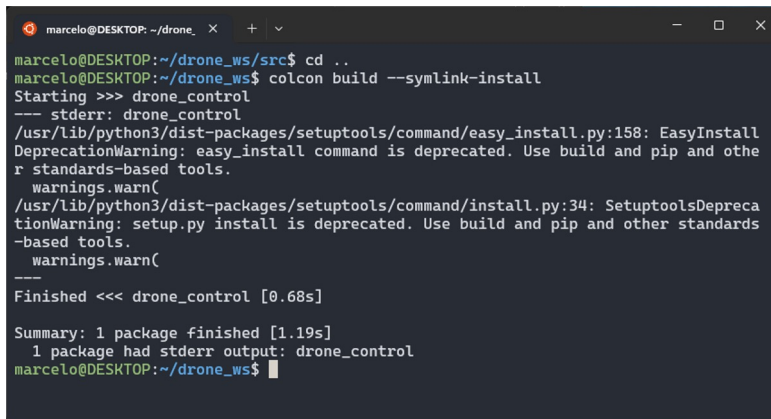
Index configuration
and launch directories
here

Setup your python
executable nodes here

Compiling/Indexing the Package

To compile all the packages, run (at the workspace level):

```
colcon build --symlink-install
```



```
marcelo@DESKTOP: ~/drone_ws/src$ cd ..
marcelo@DESKTOP: ~/drone_ws$ colcon build --symlink-install
Starting >>> drone_control
--- stderr: drone_control
/usr/lib/python3/dist-packages/setuptools/command/easy_install.py:158: EasyInstall
DeprecationWarning: easy_install command is deprecated. Use build and pip and other
standards-based tools.
  warnings.warn(
/usr/lib/python3/dist-packages/setuptools/command/install.py:34: SetuptoolsDepreca
tionWarning: setup.py install is deprecated. Use build and pip and other standards
-based tools.
  warnings.warn(
---
Finished <<< drone_control [0.68s]

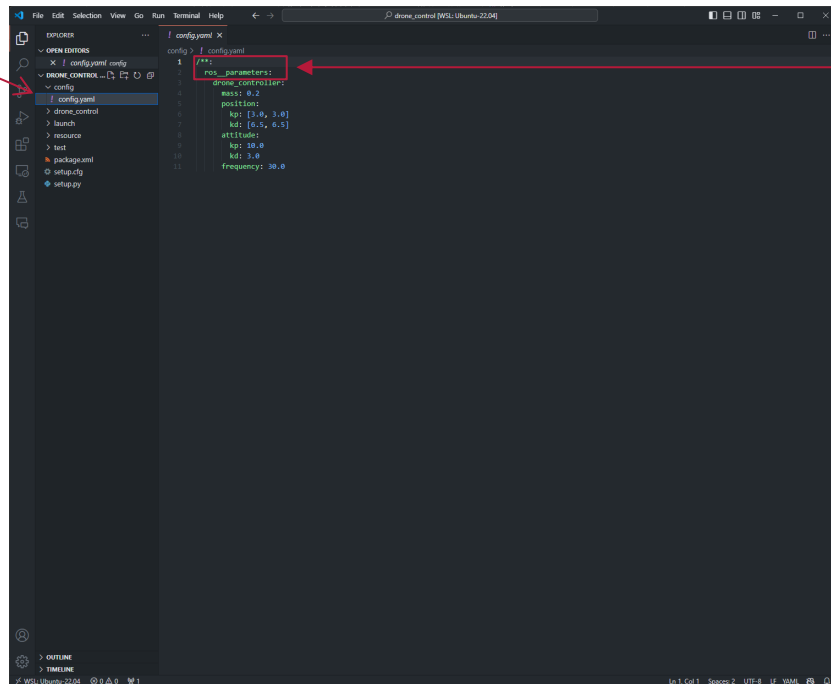
Summary: 1 package finished [1.19s]
1 package had stderr output: drone_control
marcelo@DESKTOP: ~/drone_ws$
```

If you get these warnings, just run the following lines

```
PYTHONWARNINGS="ignore:setup.py install is deprecated::setuptools.command.install" >> ~/.bashrc
export PYTHONWARNINGS >> ~/.bashrc
```

Dynamic Parameters

Create a configuration
directory with a YAML file



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure for a ROS2 package named 'drone_control'. The 'config' directory is expanded, showing a 'config.yaml' file. The code editor shows the contents of 'config.yaml', which is a YAML file defining parameters for a drone controller. The parameters are defined under the 'ros_parameters:' key, which is highlighted with a red box. The parameters include 'mass', 'position', 'kp', 'kI', 'attitude', 'kp', 'kI', and 'frequency'.

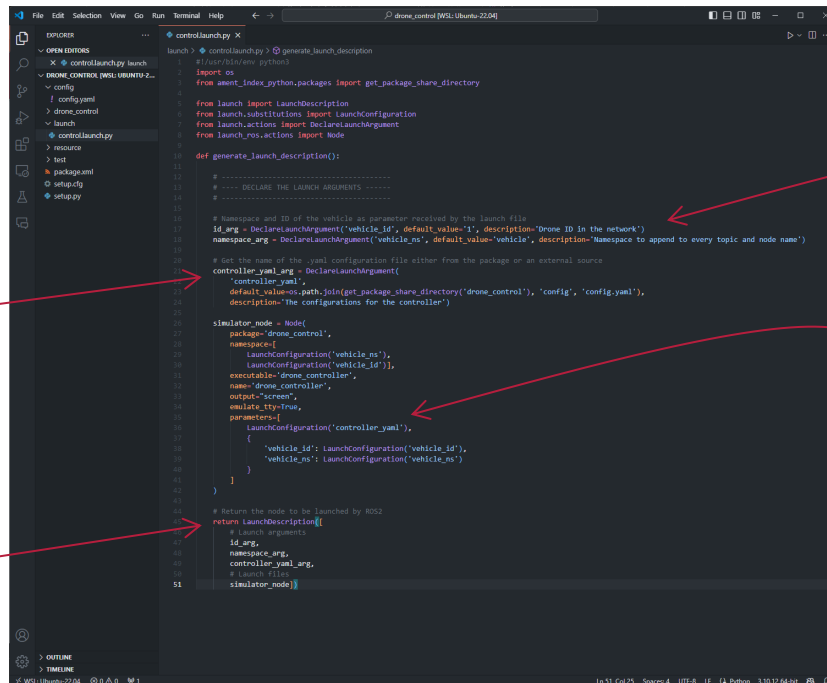
```
config:
  - config.yaml
  - drone_controller:
    mass: 0.2
    position:
      kp: [3.0, 3.0]
      kI: [6.5, 6.5]
    attitude:
      kp: 10.0
      kI: 3.0
    frequency: 30.0
```

Use /** such that all the
parameters are available
to all nodes

Launch Files

Specify the yaml file
that contains
parameters

Return the node and
arguments to be
evaluated at runtime



```

1 launch > control_launch.py > generate_launch_description
2 #!/usr/bin/env python
3 import os
4 from ament_index_python.packages import get_package_share_directory
5
6 from launch import LaunchDescription
7 from launch.substitutions import LaunchConfiguration
8 from launch.actions import DeclareLaunchArgument
9 from launch_ros.actions import Node
10
11 def generate_launch_description():
12
13     # -----
14     # DECLARE THE LAUNCH ARGUMENTS -----
15     # -----
16
17     # Namespace and ID of the vehicle as parameter specified by the launch file
18     id_arg = DeclareLaunchArgument('vehicle_id', default_value='1', description='Drone ID in the network')
19     namespace_arg = DeclareLaunchArgument('vehicle_ns', default_value='vehicle', description='Namespace to append to every topic and node name')
20
21     # Get the name of the .yaml configuration file either from the package or an external source
22     controller_yaml_arg = DeclareLaunchArgument(
23         'controller_yaml',
24         default_value=os.path.join(get_package_share_directory('drone_control'), 'config', 'config.yaml'),
25         description='The configurations for the controller')
26
27     simulator_node = Node(
28         package='drone_control',
29         namespace=[
30             LaunchConfiguration('vehicle_ns'),
31             LaunchConfiguration('vehicle_id')],
32         executable='drone_controller',
33         name='drone_controller',
34         output='screen',
35         emulate_tty=True,
36         parameters=[
37             LaunchConfiguration('controller_yaml'),
38             {
39                 'vehicle_id': LaunchConfiguration('vehicle_id'),
40                 'vehicle_ns': LaunchConfiguration('vehicle_ns')
41             }
42         ])
43
44     # Return the node to be launched by ROS2
45     return LaunchDescription([
46         # Launch arguments
47         id_arg,
48         namespace_arg,
49         controller_yaml_arg,
50         # Launch files
51         simulator_node])

```

Define some parameters
directly in the launch file

Inject the parameters from
the yaml file in the node

Compiling/Indexing the Package (2)

To compile all the packages, run (at the workspace level):

```
colcon build --symlink-install
```



Important! Otherwise, every time a launch file or configuration files is changed, you must re-compile the code!

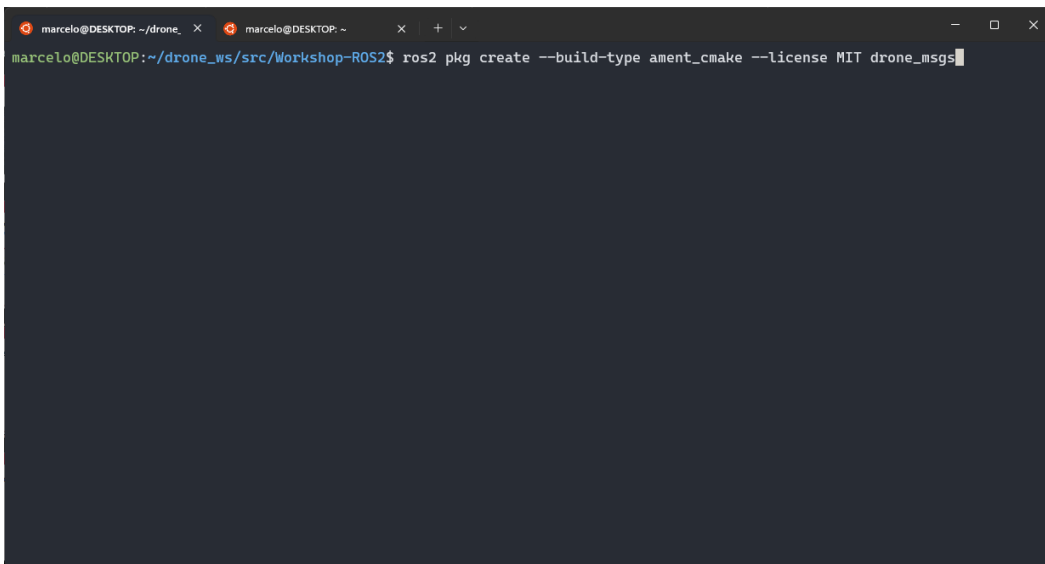
To compile a specific package, run (at the workspace level):

```
colcon build --symlink-install --packages-select <my_package_name>
```

To clear all the compiled packages, run (at the workspace level):

```
rm -rf install build install
```

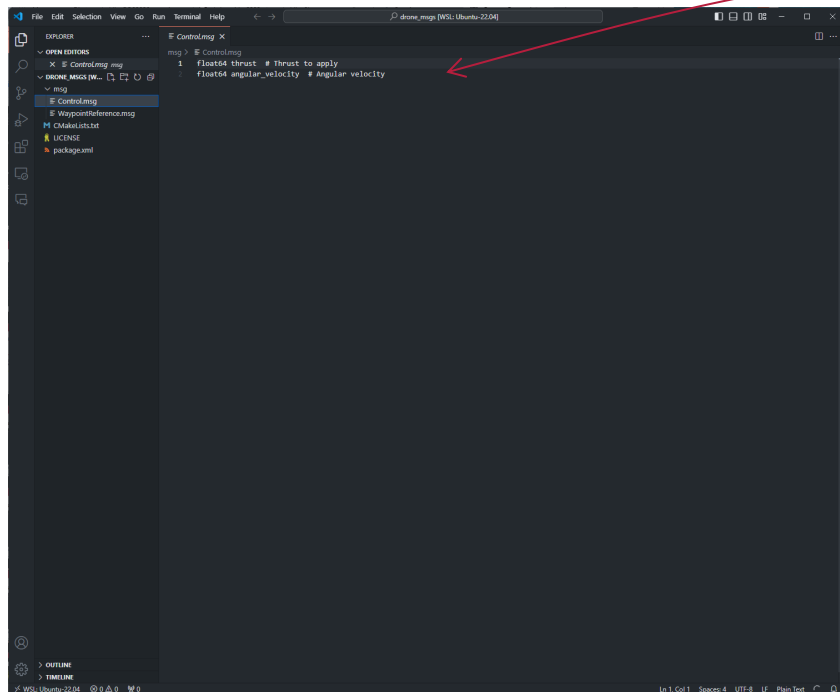
Creating a Messages Package



```
marcelo@DESKTOP: ~/drone_ws/src/Workshop-ROS2$ ros2 pkg create --build-type ament_cmake --license MIT drone_msgs
```

`ros2 pkg create --build-type ament_cmake --license MIT drone_msgs`

Creating a Messages Package (2)

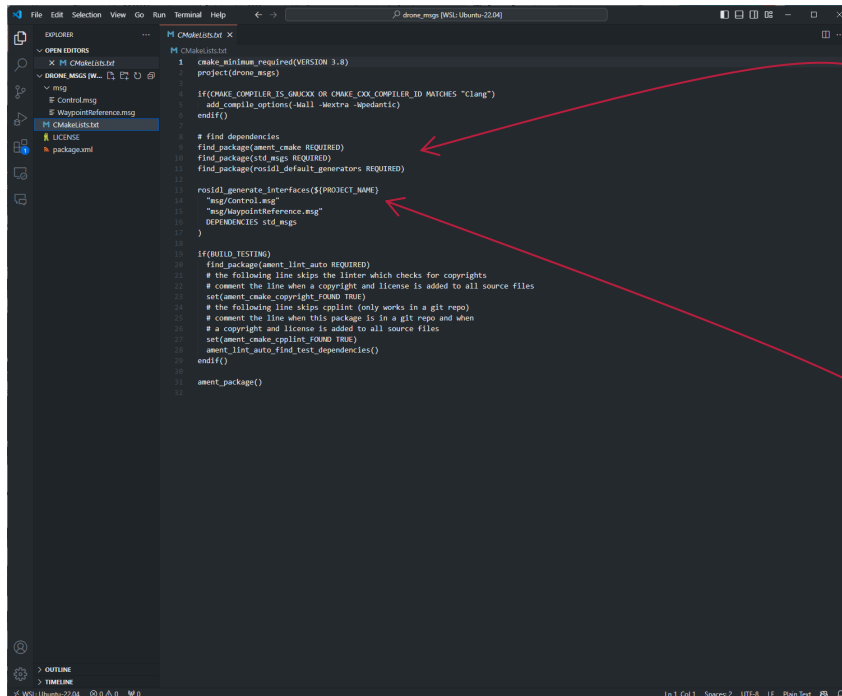


```
msg: Control.msg
1 float64 thrust # Thrust to apply
2 float64 angular_velocity # Angular velocity
```

The screenshot shows a code editor with a file named 'Control.msg' open. The file contains two lines of message definition: '1 float64 thrust # Thrust to apply' and '2 float64 angular_velocity # Angular velocity'. A red arrow points from the text 'Message definition' to the second line of the code.

Message definition

Creating a Messages Package (3)



```
1 cmake_minimum_required(VERSION 3.8)
2 project(drone_msgs)
3
4 if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
5   add_compile_options(-Wall -Wextra -Wpedantic)
6 endif()
7
8 # find dependencies
9 find_package(ament_cmake REQUIRED)
10 find_package(std_msgs REQUIRED)
11 find_package(rosidl_default_generators REQUIRED)
12
13 rosidl_generate_interfaces(${PROJECT_NAME}
14   "msg/Control.msg"
15   "msg/WaypointReference.msg"
16   DEPENDENCIES std_msgs
17 )
18
19 if(BUILD_TESTING)
20   find_package(ament_lint_auto REQUIRED)
21   # the following line skips the linter which checks for copyrights
22   # comment the line when a copyright and license is added to all source files
23   set(ament_cmake_copyright_FOUND TRUE)
24   # the following line skips cpplint (only works in a git repo)
25   # comment the line when this package is in a git repo and when
26   # a copyright and license is added to all source files
27   set(ament_cmake_cpplint_FOUND TRUE)
28   ament_lint_auto_find_test_dependencies()
29 endif()
30
31 ament_package()
```

Import the base
packages that the
messages will use

List the messages
to be compiled

The screenshot shows the VS Code editor with the `package.xml` file of the `std_msgs` package open. The file content is as follows:

```
<?xml version="1.0"?>
<url model="http://download.ros.org/schema/package-format1.xsd" schematypens="http://www.w3.org/2001/XMLSchema"?>
  <package format="1">
    <name>std_msgs</name>
    <version>1.0.0</version>
    <description>Custom message definitions for the ROS 2 Workshop</description>
    <maintainer email="jaci@intsigis.technico.uilbooa.pl">Marcelo Jacinto</maintainer>
    <license>MIT</license>

    <buildtool_depend>ament_cmake</buildtool_depend>

    <depend>std_msgs</depend>
    <buildtool_depend>rosidl_default_generators</buildtool_depend>
    <exec_depend>rosidl_default_runtime</exec_depend>
    <member_of_group>rosidl_interface_packages</member_of_group>

    <test_depend>ament_lint_auto</test_depend>
    <test_depend>ament_lint_common</test_depend>

    <export>
      <build_type>ament_cmake</build_type>
    </export>
  </package>
</url>
```

A red arrow points to the `<buildtool_depend>ament_cmake</buildtool_depend>` line, which is the subject of the text in the adjacent block.

Import the base packages that the messages will use

Messages Provided by ROS 2

std_msgs Message Documentation

- msg/MultiArrayLayout
- msg/UInt16MultiArray
- msg/Int16
- msg/UInt32MultiArray
- msg/Int8
- msg/MultiArrayDimension
- msg/Empty
- msg/UInt8MultiArray
- msg/String
- msg/Int32
- msg/UInt16
- msg/Int32MultiArray
- msg/UInt64MultiArray
- msg/Float64
- msg/Int8
- msg/Int16MultiArray
- msg/ColorRGBA
- msg/Float32MultiArray
- msg/Int64MultiArray
- msg/UInt32
- msg/Header
- msg/Byte
- msg/Float64MultiArray
- msg/Int64
- msg/Float32
- msg/ByteMultiArray
- msg/UInt64
- msg/Char
- msg/Bool
- msg/Int8MultiArray

autogenerated on Oct 09 2020 00:02:33

sensor_msgs Message Documentation

- msg/PointCloud
- msg/Imu
- msg/CompressedImage
- msg/PointCloud2
- msg/TimeReference
- msg/Range
- msg/Illuminance
- msg/PointField
- msg/MagneticField
- msg/CameraInfo
- msg/JoyFeedbackArray
- msg/JoyFeedback
- msg/NavSatFix
- msg/ChannelFloat32
- msg/Temperature
- msg/Joy
- msg/MultiDOFJointState
- msg/LaserEcho
- msg/RelativeHumidity
- msg/NavSatStatus
- msg/FluidPressure
- msg/BatteryState
- msg/RegionOfInterest
- msg/JointState
- msg/LaserScan
- msg/Image
- msg/MultiEchoLaserScan

sensor_msgs Service Documentation

- srv/SetCameraInfo

autogenerated on Oct 09 2020 00:02:33

geometry_msgs Message Documentation

- msg/PoseWithCovariance
- msg/Vector3Stamped
- msg/Pose
- msg/InertiaStamped
- msg/TransformStamped
- msg/Twist
- msg/AccelWithCovariance
- msg/QuaternionStamped
- msg/Transform
- msg/WrenchStamped
- msg/Accel
- msg/Polygon
- msg/Pose2D
- msg/PoseStamped
- msg/AccelStamped
- msg/AccelWithCovarianceStamped
- msg/Quaternion
- msg/Point
- msg/Inertia
- msg/Vector3
- msg/PolygonStamped
- msg/PoseArray
- msg/PointStamped
- msg/Point32
- msg/Wrench
- msg/TwistStamped
- msg/TwistWithCovarianceStamped
- msg/TwistWithCovariance
- msg/PoseWithCovarianceStamped

autogenerated on Oct 09 2020 00:02:33

nav_msgs Message Documentation

- msg/MapMetaData
- msg/GridCells
- msg/Odometry
- msg/OccupancyGrid
- msg/Path

nav_msgs Service Documentation

- srv/GetPlan
- srv/SetMap
- srv/GetMap

autogenerated on Oct 09 2020 00:02:33

map_msgs Message Documentation

- msg/PointCloud2Update
- msg/ProjectedMap
- msg/OccupancyGridUpdate
- msg/ProjectedMapInfo

map_msgs Service Documentation

- srv/SetMapProjections
- srv/ProjectedMapsInfo
- srv/SaveMap
- srv/GetPointMapROI
- srv/GetPointMap
- srv/GetMapROI

autogenerated on Oct 09 2020 00:02:33

Official Documentation:

<https://docs.ros2.org/foxy/api>

https://docs.ros2.org/foxy/api/std_msgs/index-msg.html

https://docs.ros2.org/foxy/api/sensor_msgs/index-msg.html

https://docs.ros2.org/foxy/api/geometry_msgs/index-msg.html

https://docs.ros2.org/foxy/api/nav_msgs/index-msg.html

https://docs.ros2.org/foxy/api/map_msgs/index-msg.html

Terminal Tools

Terminal Tools

```
marcelo@DESKTOP:~$ ros2
(base) marcelo@DESKTOP:~$ ros2
usage: ros2 [-h] [--use-python-default-buffering] Call 'ros2 <command> -h' for more detailed usage. ...

ros2 is an extensible command-line tool for ROS 2.

options:
  -h, --help            show this help message and exit
  --use-python-default-buffering
                        Do not force line buffering in stdout and instead use the python default buffering, which might be
                        affected by PYTHONUNBUFFERED/-u and depends on whatever stdout is interactive or not

Commands:
  action                Various action related sub-commands
  bag                   Various rosbag related sub-commands
  component             Various component related sub-commands
  daemon               Various daemon related sub-commands
  doctor               Check ROS setup and other potential issues
  interface             Show information about ROS interfaces
  launch               Run a launch file
  lifecycle             Various lifecycle related sub-commands
  multicast            Various multicast related sub-commands
  node                 Various node related sub-commands
  param                Various param related sub-commands
  pkg                  Various package related sub-commands
  run                  Run a package specific executable
  security              Various security related sub-commands
  service              Various service related sub-commands
  topic                Various topic related sub-commands
  wtf                  Use 'wtf' as alias to 'doctor'

Call 'ros2 <command> -h' for more detailed usage.
(base) marcelo@DESKTOP:~$
```

Listing all the available commands:

`ros2`

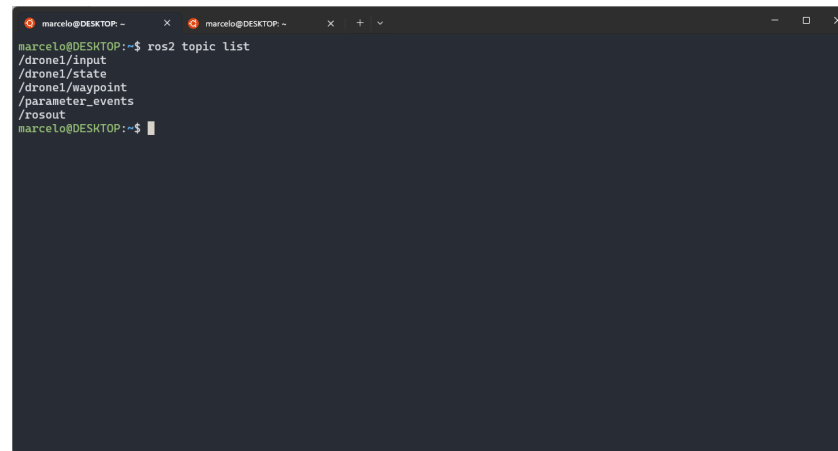
Terminal Tools

```
marcelo@DESKTOP: ~  
marcelo@DESKTOP:~$ ros2 pkg  
usage: ros2 pkg [-h] Call 'ros2 pkg <command> -h' for more detailed usage. ...  
  
Various package related sub-commands  
  
options:  
  -h, --help            show this help message and exit  
  
Commands:  
  create                Create a new ROS 2 package  
  executables           Output a list of package specific executables  
  list                  Output a list of available packages  
  prefix                Output the prefix path of a package  
  xml                   Output the XML of the package manifest or a specific tag  
  
Call 'ros2 pkg <command> -h' for more detailed usage.  
marcelo@DESKTOP:~$
```

Each command can have multiple options. For example:

`ros2 pkg`

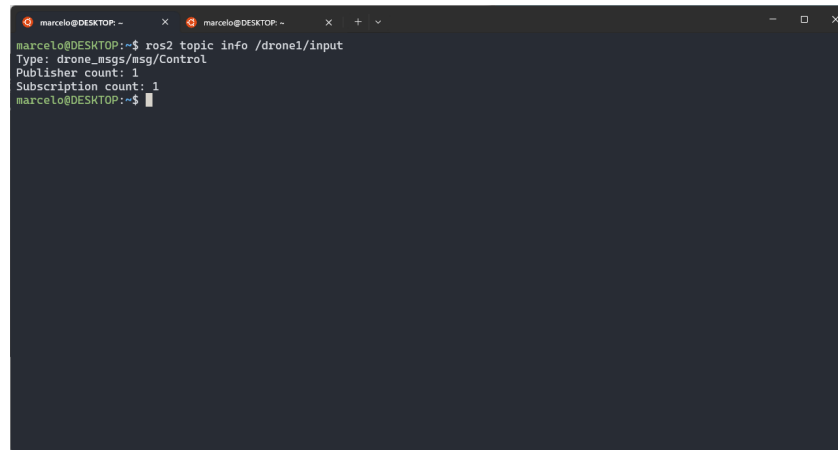
List all Topics Published/Subscribed



```
marcelo@DESKTOP: ~  
marcelo@DESKTOP:~$ ros2 topic list  
/drone1/input  
/drone1/state  
/drone1/waypoint  
/parameter_events  
/rosout  
marcelo@DESKTOP:~$
```

`ros2 topic list`

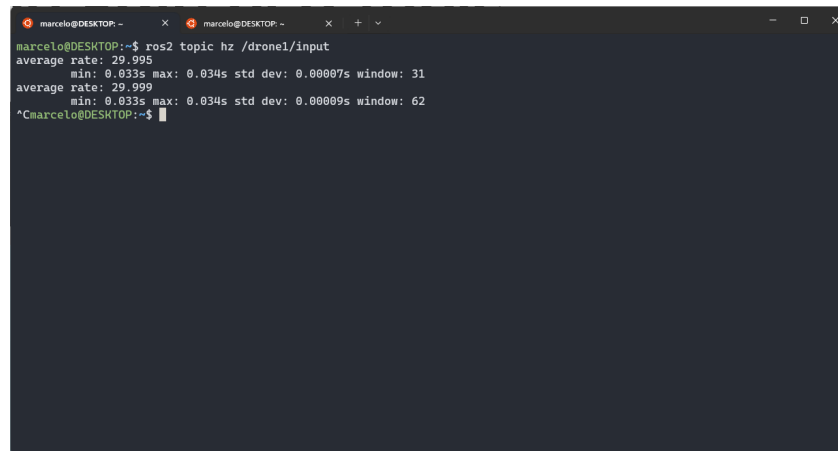
Get Information About a Topic



```
marcelo@DESKTOP: ~  
marcelo@DESKTOP:~$ ros2 topic info /drone1/input  
Type: drone_msgs/msg/Control  
Publisher count: 1  
Subscription count: 1  
marcelo@DESKTOP:~$
```

`ros2 topic info /<name_of_topic>`

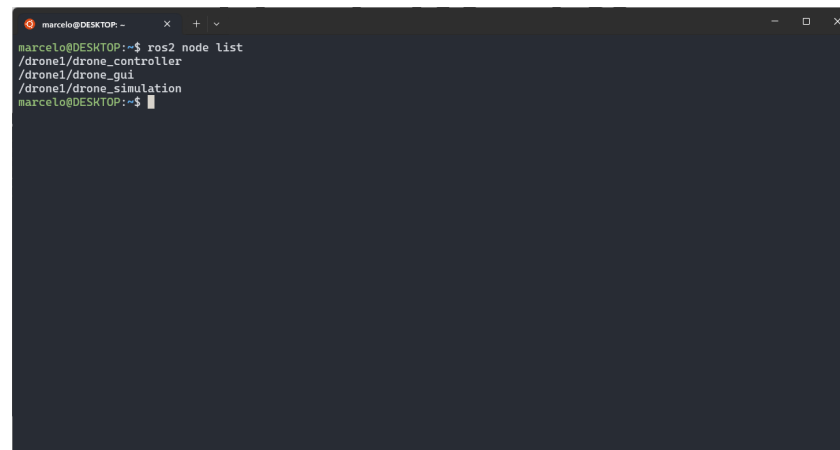
Get the Rate of a Topic



```
marcelo@DESKTOP: ~  
marcelo@DESKTOP:~$ ros2 topic hz /drone1/input  
average rate: 29.995  
  min: 0.033s max: 0.034s std dev: 0.00007s window: 31  
average rate: 29.999  
  min: 0.033s max: 0.034s std dev: 0.00009s window: 62  
^Cmarcelo@DESKTOP:~$
```

`ros2 topic hz /<name_of_topic>`

Get all the Nodes Running



```
marcelo@DESKTOP: ~  
marcelo@DESKTOP:~$ ros2 node list  
/drone1/drone_controller  
/drone1/drone_gui  
/drone1/drone_simulation  
marcelo@DESKTOP:~$
```

`ros2 node list`

Saving Data using Bags

```
marcelo@DESKTOP: ~$ ros2 bag record -a -o saved_data
[INFO] [1713790849.889880752] [rosbag2_recorder]: Press SPACE for pausing/resuming
[INFO] [1713790849.896627554] [rosbag2_storage]: Opened database 'saved_data/saved_data_0.db3' for READ_WRITE.
[INFO] [1713790849.898787341] [rosbag2_recorder]: Listening for topics...
[INFO] [1713790849.898790466] [rosbag2_recorder]: Event publisher thread: Starting
[INFO] [1713790849.899703498] [rosbag2_recorder]: Subscribed to topic '/rosout'
[INFO] [1713790849.900443762] [rosbag2_recorder]: Subscribed to topic '/parameter_events'
[INFO] [1713790849.900859508] [rosbag2_recorder]: Subscribed to topic '/events/write_split'
[INFO] [1713790849.903447808] [rosbag2_recorder]: Subscribed to topic '/drone1/waypoint'
[INFO] [1713790849.908415627] [rosbag2_recorder]: Subscribed to topic '/drone1/state'
[INFO] [1713790849.908475507] [rosbag2_recorder]: Recording...
[INFO] [1713790850.009505728] [rosbag2_recorder]: Subscribed to topic '/drone1/input'
```

`ros2 bag record -a -o <bag_name>`

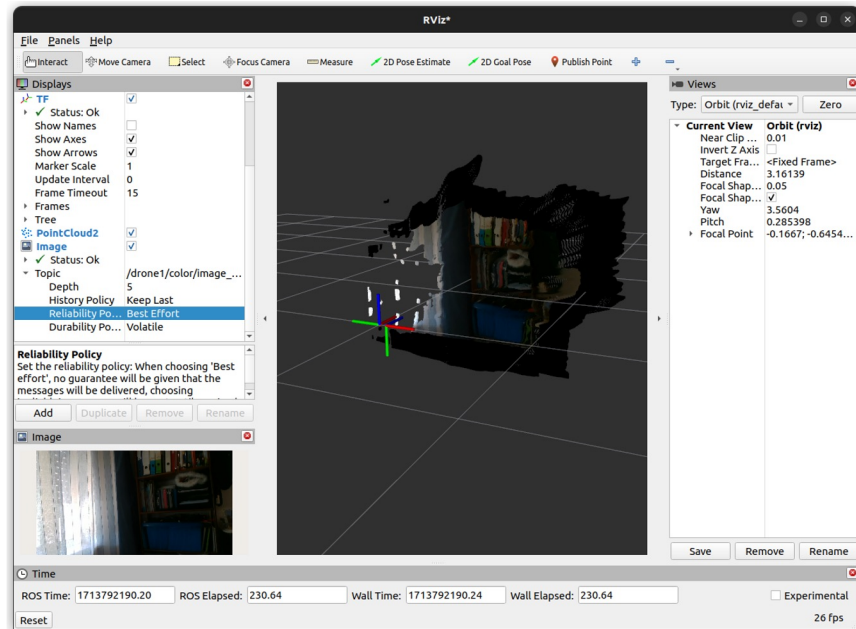
Playing Back the Bag Data

```
marcelo@DESKTOP: ~$ ros2 bag record -a -o saved_data
[INFO] [1713790849.889880752] [rosbag2_recorder]: Press SPACE for pausing/resuming
[INFO] [1713790849.896627554] [rosbag2_storage]: Opened database 'saved_data/saved_data_0.db3' for READ_WRITE.
[INFO] [1713790849.898787341] [rosbag2_recorder]: Listening for topics...
[INFO] [1713790849.898798466] [rosbag2_recorder]: Event publisher thread: Starting
[INFO] [1713790849.899703498] [rosbag2_recorder]: Subscribed to topic '/rosout'
[INFO] [1713790849.900443762] [rosbag2_recorder]: Subscribed to topic '/parameter_events'
[INFO] [1713790849.900859508] [rosbag2_recorder]: Subscribed to topic '/events/write_split'
[INFO] [1713790849.903447808] [rosbag2_recorder]: Subscribed to topic '/drone1/waypoint'
[INFO] [1713790849.908415627] [rosbag2_recorder]: Subscribed to topic '/drone1/state'
[INFO] [1713790849.908475507] [rosbag2_recorder]: Recording...
[INFO] [1713790850.009505728] [rosbag2_recorder]: Subscribed to topic '/drone1/input'
```

`ros2 bag play <bag_name>`

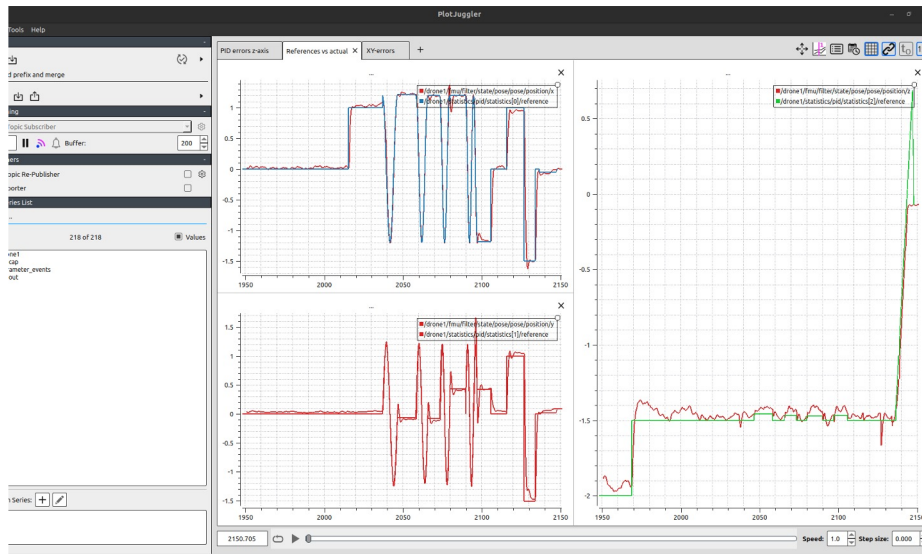
Visualization Tools

RVIZ 2



Executing: `rviz2`

Plotjuggler



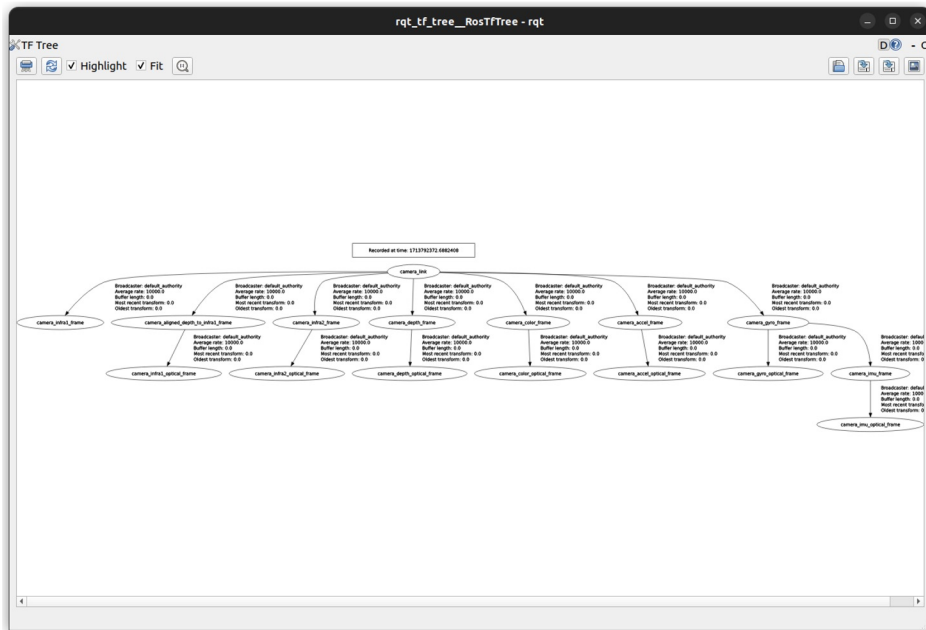
Installing:

```
sudo apt install ros-humble-plotjuggler
```

Executing:

```
ros2 run plotjuggler plotjuggler
```

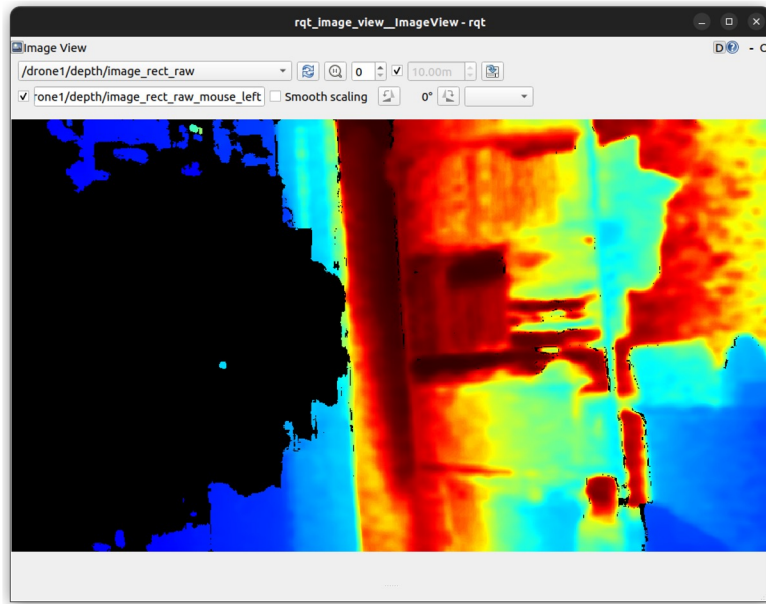
Transforms (TF2)



Executing:

```
ros2 run rqt_tf_tree rqt_tf_tree
```

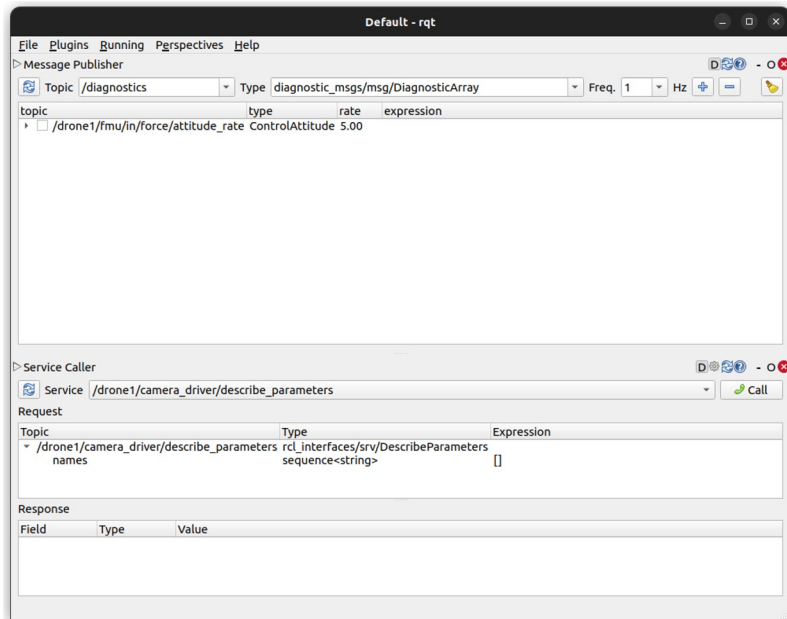
RQT Image View



Executing:

```
ros2 run rqt_image_view rqt_image_view
```

RQT GUI



Executing:

```
ros2 run rqt_gui rqt_gui
```

ROS ID

- By default, all the ROS 2 nodes run with domain ID 0
- To avoid interference between different groups of computers running ROS 2 on the same network, a different domain ID should be set for each group

```
export ROS_DOMAIN_ID=0
```

- By default, all the ROS 2 nodes broadcast to the local network

```
export ROS_LOCALHOST_ONLY=0
```

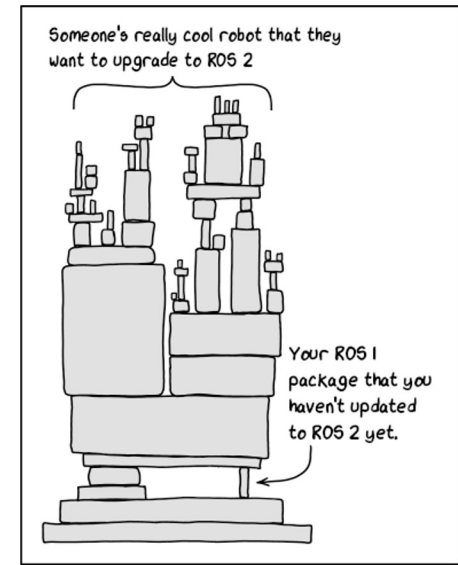
Installing External Libraries

Installing packages follows this standard:

```
sudo apt install ros-<version>-<name-of-package>
```

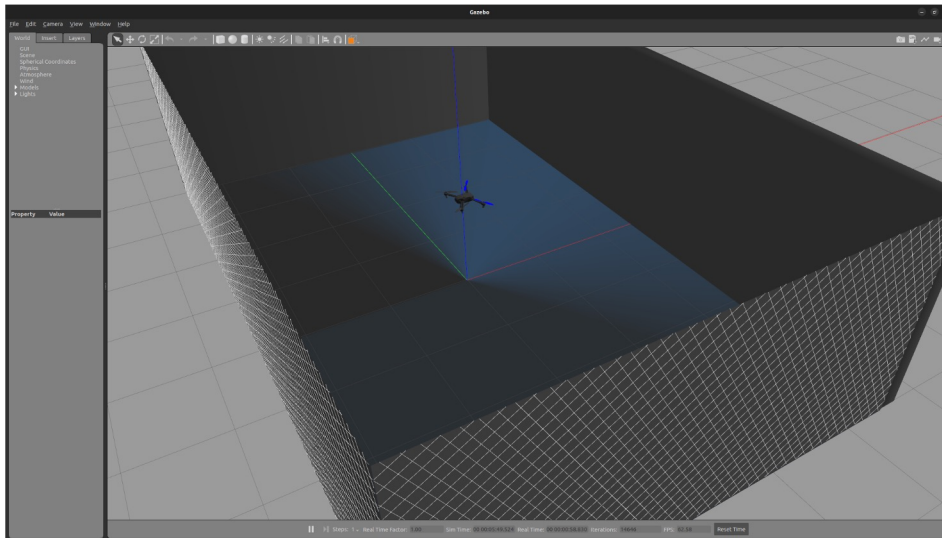
```
marcelo@DESKTOP: ~  
marcelo@DESKTOP: ~  
marcelo@DESKTOP: ~  
marcelo@DESKTOP: ~$ sudo apt install ros-humble-vision-msgs  
[sudo] password for marcelo:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following packages were automatically installed and are no longer required:  
  cuda-12-0 cuda-cccl-12-0 cuda-command-line-tools-12-0 cuda-compiler-12-0 cuda-cudart-12-0 cuda-cudart-dev-12-0  
  cuda-cuobjdump-12-0 cuda-cupti-12-0 cuda-cupti-dev-12-0 cuda-cuxxfilt-12-0 cuda-demo-suite-12-0  
  cuda-documentation-12-0 cuda-driver-dev-12-0 cuda-gdb-12-0 cuda-libraries-12-0 cuda-libraries-dev-12-0  
  cuda-nsight-12-0 cuda-nsight-compute-12-0 cuda-nsight-systems-12-0 cuda-nvcc-12-0 cuda-nvdisasm-12-0  
  cuda-nvml-dev-12-0 cuda-nvprof-12-0 cuda-nvprune-12-0 cuda-nvrtc-12-0 cuda-nvrtc-dev-12-0 cuda-nvtx-12-0  
  cuda-nvvp-12-0 cuda-openssl-12-0 cuda-openssl-dev-12-0 cuda-profiler-api-12-0 cuda-runtime-12-0 cuda-sanitizer-12-0  
  cuda-toolkit-12-0 cuda-toolkit-12-0-config-common cuda-tools-12-0 cuda-visual-tools-12-0 gds-tools-12-0  
  libcublas-12-0 libcublas-dev-12-0 libcufft-12-0 libcufft-dev-12-0 libcufft-12-0 libcufft-dev-12-0 libcufft-12-0  
  libcurand-dev-12-0 libcusolver-12-0 libcusolver-dev-12-0 libcusparse-12-0 libcusparse-dev-12-0 liblvm13 libnpp-12-0  
  libnpp-dev-12-0 libnvjitlink-12-0 libnvjitlink-dev-12-0 libnvjpeg-12-0 libnvjpeg-dev-12-0 libnvvm-samples-12-0  
  nsight-compute-2022.4.0 nsight-compute-2022.4.1 nsight-systems-2022.4.2  
Use 'sudo apt autoremove' to remove them.  
The following NEW packages will be installed:  
  ros-humble-vision-msgs  
0 upgraded, 1 newly installed, 0 to remove and 9 not upgraded.  
Need to get 150 kB of archives.  
After this operation, 2,105 kB of additional disk space will be used.  
0% [Connecting to packages.ros.org]
```

Example for the vision_msgs package



3D Simulators

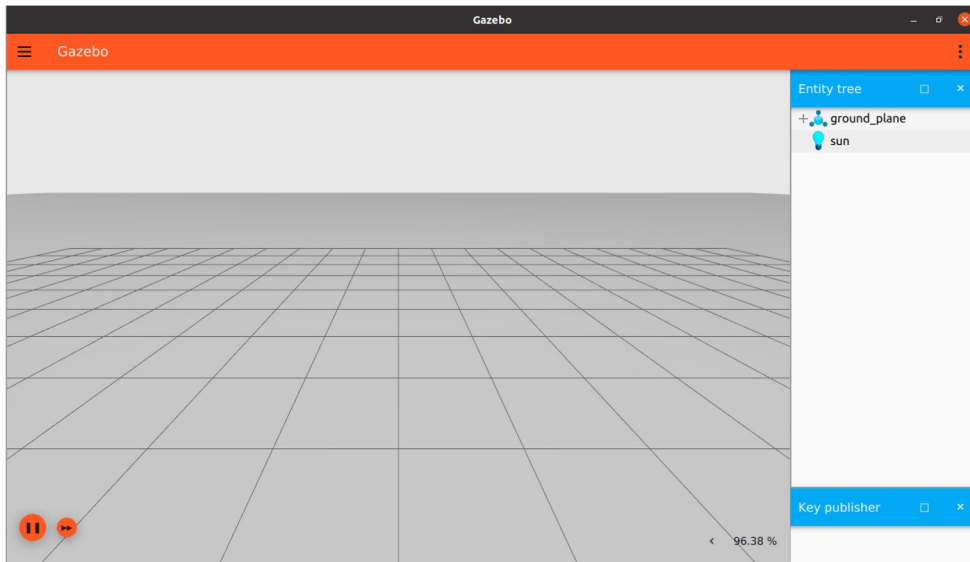
Simulators (Gazebo Classic)



Installing

```
sudo apt install ros-humble-gazebo-ros-pkgs
```

Simulators (Gazebo Garden)



Installing

```
sudo apt-get install ros-humble-ros-gz
```

Simulators (Pegasus Simulator)

