

ASME ROS Workshop

Prerequisites

- Text editor
- Terminal emulator
- Github repository open in browser

Make Robot Hard.

What's ROS?

It's not an actual OS

**A set of software frameworks for robot
software development**

Why use ROS?

- A lot of prebuilt packages
- Modular
- Open-source

Nodes

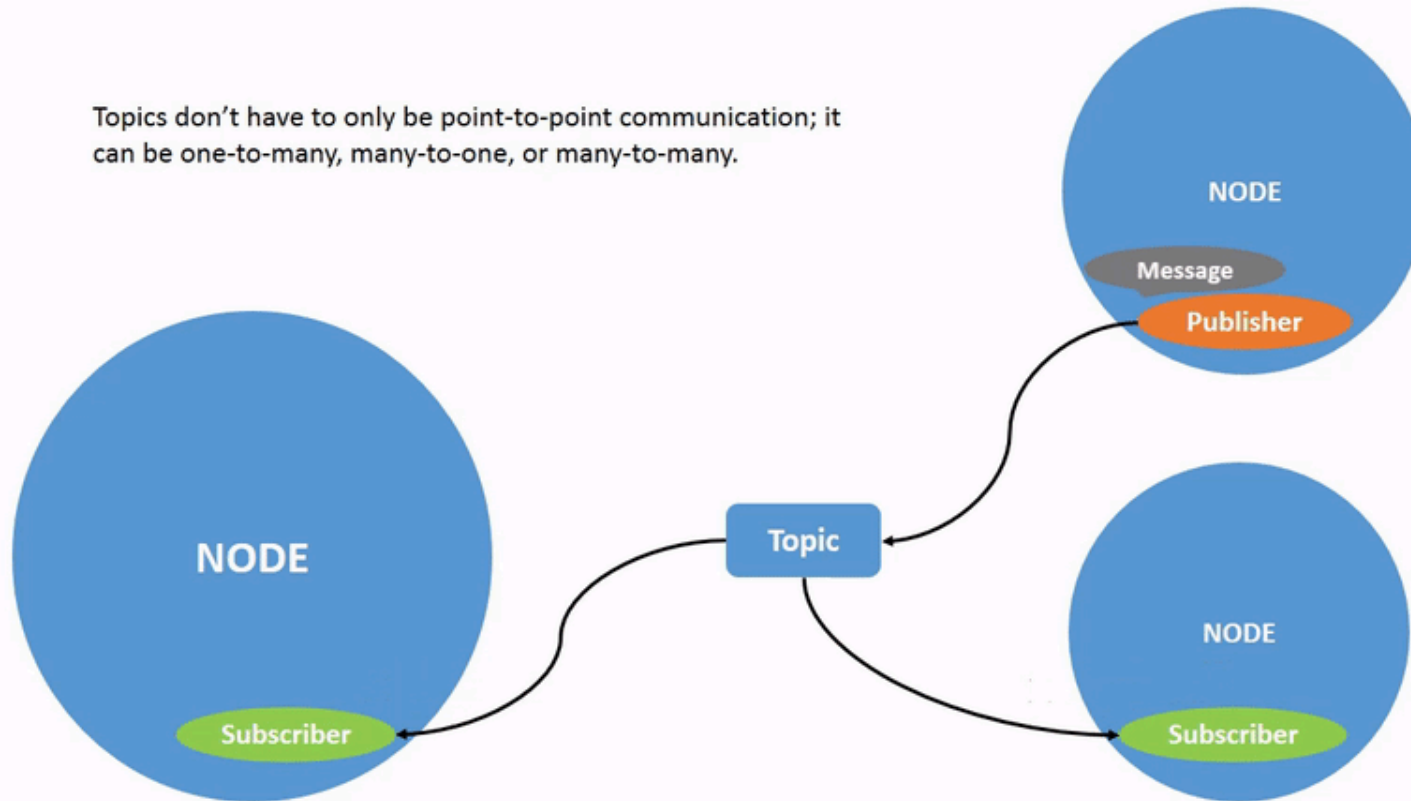
- An executable program or a running process that performs some kind of task
- A single node is responsible for a single, modular task

Ways to communicate

- Publisher - Subscriber
- Services
- Actions

Publisher - Subscriber

Topics don't have to only be point-to-point communication; it can be one-to-many, many-to-one, or many-to-many.



Messages

std_msgs

std_msgs/String Message

File: `std_msgs/String.msg`

Raw Message Definition

```
string data
```

Compact Message Definition

```
string data
```

autogenerated on Mon, 28 Feb 2022 23:49:59

sensor_msgs/LaserScan Message

File: `sensor_msgs/LaserScan.msg`

Raw Message Definition

```
# Single scan from a planar laser range-finder
#
# If you have another ranging device with different behavior (e.g. a sonar
# array), please find or create a different message, since applications
# will make fairly laser-specific assumptions about this data

Header header          # timestamp in the header is the acquisition time of
                        # the first ray in the scan.
                        #
                        # in frame frame_id, angles are measured around
                        # the positive Z axis (counterclockwise, if Z is up)
                        # with zero angle being forward along the x axis

float32 angle_min      # start angle of the scan [rad]
float32 angle_max      # end angle of the scan [rad]
float32 angle_increment # angular distance between measurements [rad]

float32 time_increment  # time between measurements [seconds] - if your scanner
                        # is moving, this will be used in interpolating position
                        # of 3d points
float32 scan_time       # time between scans [seconds]

float32 range_min       # minimum range value [m]
float32 range_max       # maximum range value [m]

float32[] ranges         # range data [m] (Note: values < range_min or > range_max should be discarded)
float32[] intensities    # intensity data [device-specific units]. If your
                        # device does not provide intensities, please leave
                        # the array empty.
```

Compact Message Definition

```
std\_msgs/Header header
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

ROS Workspaces

- Directory containing ROS2 packages
- Sourcing the installation workspace to have the packages in that workspace available to you `source ./install/local_setup.bash`
- 3 types of workspaces
 - Python
 - C++
 - Python & C++

Python Workspace

```
ros_ws/
├── build/
│   └── ...
├── install/
│   └── ...
├── log/
│   └── ...
└── src/
    ├── my_package/
    │   ├── package.xml
    │   ├── resource/my_package
    │   ├── setup.cfg
    │   ├── setup.py
    │   └── my_package/
    │       └── package.py
```

C++ Workspace

```
ros_ws/  
├── build/  
│   └── ...  
├── install/  
│   └── ...  
├── log/  
│   └── ...  
└── src/  
    ├── CMakeLists.txt  
    ├── include/my_package/  
    ├── package.xml  
    └── src/  
        └── my_package.cpp
```

Python & C++

```
ros_ws/
├── build/
│   └── ...
├── install/
│   └── ...
├── log/
│   └── ...
└── src/
    ├── my_package/
    │   ├── CMakeLists.txt
    │   ├── package.xml
    │   ├── include/
    │   │   └── header_file.hpp
    │   ├── src/
    │   │   └── my_package.cpp
    │   ├── my_package/
    │   │   └── module_to_import.py
    │   └── scripts/
    │       └── my_package.py
```


What's a package?

- Organisational unit for ROS code
- Makes it easier to share your ROS work with others

Some Terminal Stuff

General Command Line Tools

```
cd <folder/path_to_folder> # Change Directory
cd ~/Documents # Example (~/ describes your home directory)

ls # List files

mv <path_to_file> <path_to_new_location> # Move
mv ~/Downloads/file_to_be_moved ~/Documents/ # Example

cp <path_to_file> <path_to_new_location> # Copies files or folders

mkdir <folder_name> # Make Directory a.k.a Creates a new folder
mkdir ~/new_folder/ # example

touch <file_name> # Creates a blank file
touch new_file_name # Example
```

NOTE: Use **CTRL+C** to interrupt a running process

Building a ROS2 package

```
mkdir -p /jackal_files/github_dir/ros_ws/src  
cd /jackal_files/github_dir/ros_ws/src  
source /opt/ros/foxy/setup.bash  
ros2 pkg create --build-type ament_python pub_sub
```

NOTE: pub_sub can be changed to any name you want

Minimal Publisher

```
import rclpy
from rclpy.node import Node

from std_msgs.msg import String
```

```
class MinimalPublisher(Node):  
  
    def __init__(self):  
        super().__init__('minimal_publisher')  
        # TODO: Replace <topic_name> with desired topic name  
        self.publisher_ = self.create_publisher(String, '<topic_name>', 10)  
        # TODO: Replace <period> with desired timer period in seconds  
        timer_period = <period> # seconds  
        self.timer = self.create_timer(timer_period, self.timer_callback)  
        self.i = 0
```

```
def timer_callback(self):  
    msg = String()  
    # TODO: Replace <custom_msg> with desired message or uncomment the line below  
    # msg.data = 'Hello World: %d' % self.i  
    msg.data = '<custom_msg> %d' % self.i  
    self.publisher_.publish(msg)  
    self.get_logger().info('Publishing: "%s"' % msg.data)  
    self.i += 1
```

```
def main(args=None):
    rclpy.init(args=args)

    minimal_publisher = MinimalPublisher()

    rclpy.spin(minimal_publisher)

    # Destroy the node explicitly
    # (optional – otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    minimal_publisher.destroy_node()
    rclpy.shutdown()

if __name__ == "__main__":
    main()
```


Minimal Subscriber

```
import rclpy
from rclpy.node import Node

from std_msgs.msg import String
```

```
class MinimalSubscriber(Node):

    def __init__(self):
        super().__init__("minimal_subscriber")
        # TODO: replace '<topic_name>' with desired topic name
        self.subscription = self.create_subscription(
            String, "<topic_name>", self.listener_callback, 10
        )
        self.subscription # prevent unused variable warning

    def listener_callback(self, msg):
        self.get_logger().info('I heard: "%s"' % msg.data)
```

```
def main(args=None):
    rclpy.init(args=args)

    minimal_subscriber = MinimalSubscriber()

    rclpy.spin(minimal_subscriber)

    # Destroy the node explicitly
    # (optional – otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    minimal_subscriber.destroy_node()
    rclpy.shutdown()

if __name__ == "__main__":
    main()
```

Copy code to the package directory

```
cd pub_sub/pub_sub  
cp ~/jackal_files/github_dir/part_1/python_scripts/* ./
```

Modifying setup.py

Copy the file into the package directory

```
cd /jackal_files/github_dir/  
cp part_1/setup.py ros_ws/src/pub_sub/
```

Or add these lines to the `setup.py` file in the package directory

```
import os # Added
from glob import glob # Added
from setuptools import setup
```

```
entry_points={
    "console_scripts": [
        "minimal_publisher = pub_sub.minimal_publisher:main", # Added
        "minimal_subscriber = pub_sub.minimal_subscriber:main", # Added
    ],
}
```

Building the ROS Workspace

```
cd /jackal_files/github_dir/ros_ws/  
colcon build
```

```
ros_ws/  
├── build/  
├── install/  
├── log/  
└── src/
```

Running the nodes

Using ros2 run

```
source /opt/ros/foxy/setup.bash
source ./install/local_setup.bash
ros2 run pub_sub minimal_publisher.py
```

NOTE: To run the subscriber just replace minimal_publisher with minimal_subscriber

Executing it directly

```
cd /jackal_files/github_dir/ros_ws/src/pub_sub/pub_sub
python3 minimal_publisher.py
```

NOTE: Same thing applies here

What are launch files?

- Used to run many nodes at the same time using one command `ros2 launch ...`

Writing launch files

- Python
- YAML
- XML

Python

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription(
        [
            Node(
                package="pub_sub",
                executable="minimal_publisher",
            ),
            Node(
                package="pub_sub",
                executable="minimal_subscriber",
            ),
        ]
    )
```

YAML

```
launch:
```

- node:
 - pkg: "pub_sub"
 - exec: "minimal_publisher"
- node:
 - pkg: "pub_sub"
 - exec: "minimal_subscriber"

XML

```
<launch>  
  <node pkg="pub_sub" exec="minimal_publisher"/>  
  <node pkg="pub_sub" exec="minimal_subscriber"/>  
</launch>
```

Copy the launch folder

```
cp -r /jackal_files/github_dir/part_1/launch/ /jackal_files/github_dir/ros_ws/src/pub_sub/
```

Edit setup.py file in the package directory

```
data_files=[
    ('share/ament_index/resource_index/packages',
     ['resource/' + package_name]),
    ('share/' + package_name, ['package.xml']),
    (os.path.join('share', package_name, 'launch'), glob(os.path.join('launch', '*launch.[pxy][yma]*'))) # Add
],
```

Skip this step if you copied the `setup.py` file earlier

Rebuild the workspace

```
cd /jackal_files/github_dir/ros_ws/  
colcon build
```

Running the launch file

```
source /opt/ros/foxy/setup.bash  
source ./install/local_setup.bash  
ros2 launch pub_sub launch.py
```

NOTE: file extension depends on which launch file you want to use
(launch.xml/launch.yaml)

Simulations

Why

- Expensive and time consuming to test on physical robot
- Jackal is an open source, robotic platform which is built on ROS and Gazebo
- Gazebo is a simulation software not made specifically for ROS but can be used with it

Commands to run sim and teleop

Move the jackal_ws into the github directory

```
mv /jackal_files/jackal_ws/ /jackal_files/github_dir  
cd /jackal_files/github_dir/jackal_ws  
rm -rf build/ install/ log/  
colcon build
```

Launch the jackal simulator

```
source /opt/ros/foxy/setup.bash  
source ./install/local_setup.bash  
ros2 launch jackal_gazebo jackal_world.launch.py
```

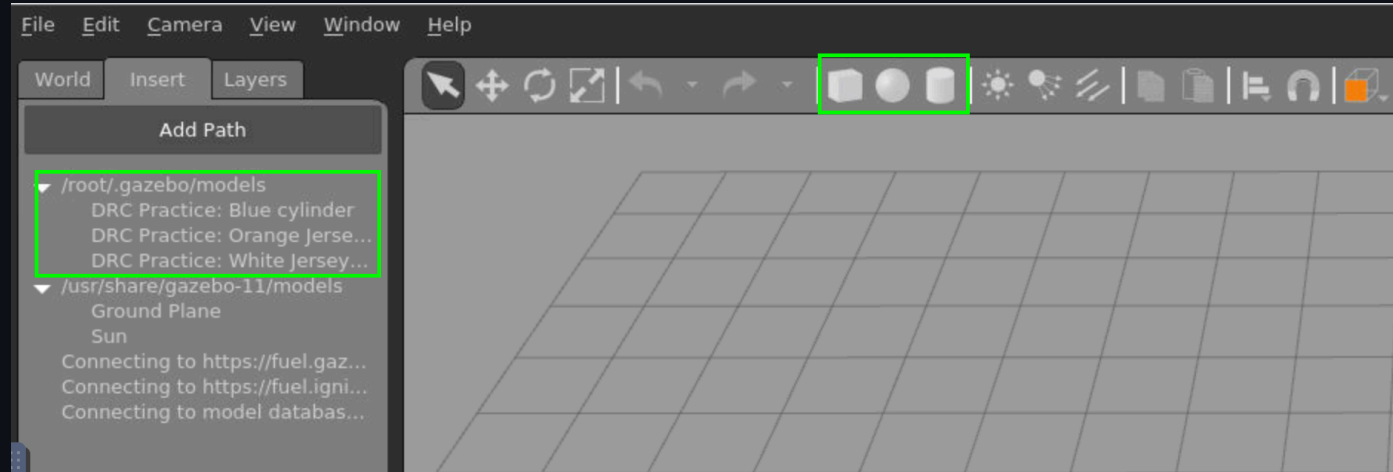
Run the pre-built teleop node

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

Build map in gazebo

1. Launch gazebo

2. Place models or shapes



3. Then save the file by navigating to `File > Save World As` and name it `custom_world.world`.

Then save it at this directory

```
/jackal_files/jackal_ws/src/jackal_simulator/jackal_gazebo/worlds/
```

Use newly built map

Copy the launch file and rename it

```
cd /jackal_files/jackal_ws/src/jackal_simulator/jackal_gazebo/launch/  
cp jackal_world.launch.py custom_world.launch.py
```

Edit the launch file

```
def generate_launch_description():  
  
    world_file = PathJoinSubstitution(  
        [FindPackageShare('jackal_gazebo'),  
        'worlds',  
        'custom_world.world'], # Changed  
    )
```

Rebuild the workspace

```
cd jackal_files/github_dir/jackal_ws/  
colcon build
```

Make sure you're at the root of the workspace `jackal_ws/`

Launch the new file

```
source ./install/local_setup.bash  
ros2 launch jackal_gazebo custom_world.launch.py
```

Debugging

Looking at topics and messages

```
ros2 topic echo <topic_name>  
ros2 topic hz <topic_name>  
ros2 topic info <topic_name>
```


geometry_msgs/Twist

```
ros2 topic echo /cmd_vel
```

```
linear:  
  x: 0.5  
  y: 0.0  
  z: 0.0  
angular:  
  x: 0.0  
  y: 0.0  
  z: 0.0
```

sensor_msgs/LaserScan

```
ros2 topic echo /front/scan
```

```
header:
  stamp:
    sec: 1541
    nanosec: 197000000
  frame_id: front_laser
angle_min: -2.3561899662017822
angle_max: 2.3561899662017822
angle_increment: 0.006554075051099062
time_increment: 0.0
scan_time: 0.0
range_min: 0.10000000149011612
range_max: 30.0
ranges:
- 3.672072410583496
- ...
```

nav_msgs/Odometry

```
ros2 topic echo --no-arr /odom
```

Header

```
header:  
  stamp:  
    sec: 1712668434  
    nanosec: 829049376  
  frame_id: odom  
child_frame_id: base_link
```

Pose

```
pose:
  pose:
    position:
      x: 2.1441852005195785
      y: 1.408813177853442
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.06435134457900855
      w: 0.9979273041914796
  covariance: '<array type: double[36]>'
```

Twist

```
twist:  
  twist:  
    linear:  
      x: 3.7634551129476595e-06  
      y: 0.0  
      z: 0.0  
    angular:  
      x: 0.0  
      y: 0.0  
      z: -1.3571082599241658e-08  
  covariance: '<array type: double[36]>'
```

```
ros2 topic hz /front/scan
```

```
average rate: 49.326
```

```
    min: 0.019s max: 0.023s std dev: 0.00080s window: 51
```

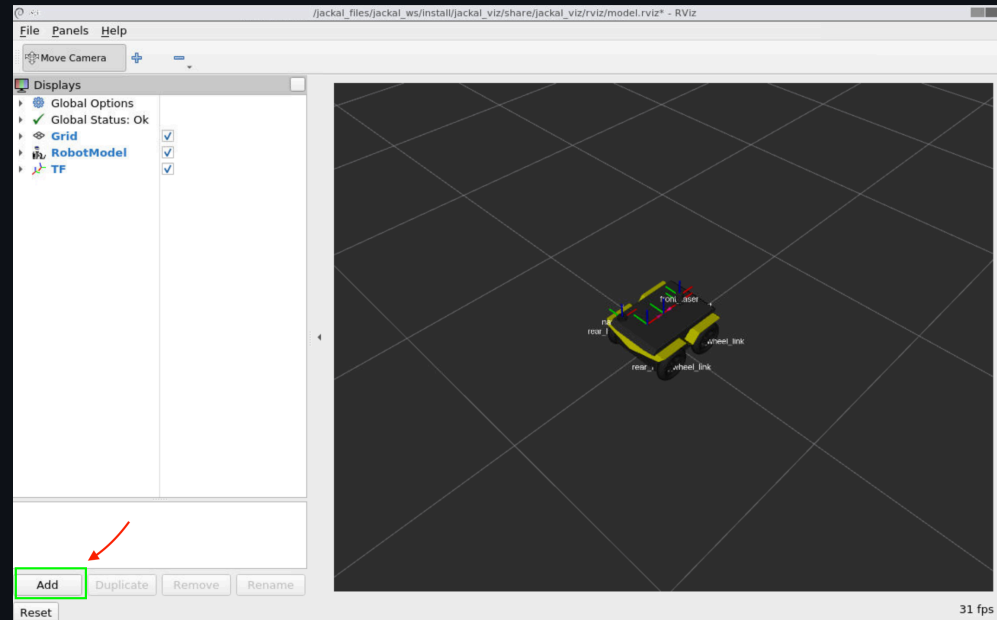
```
ros2 topic info /front/scan
```

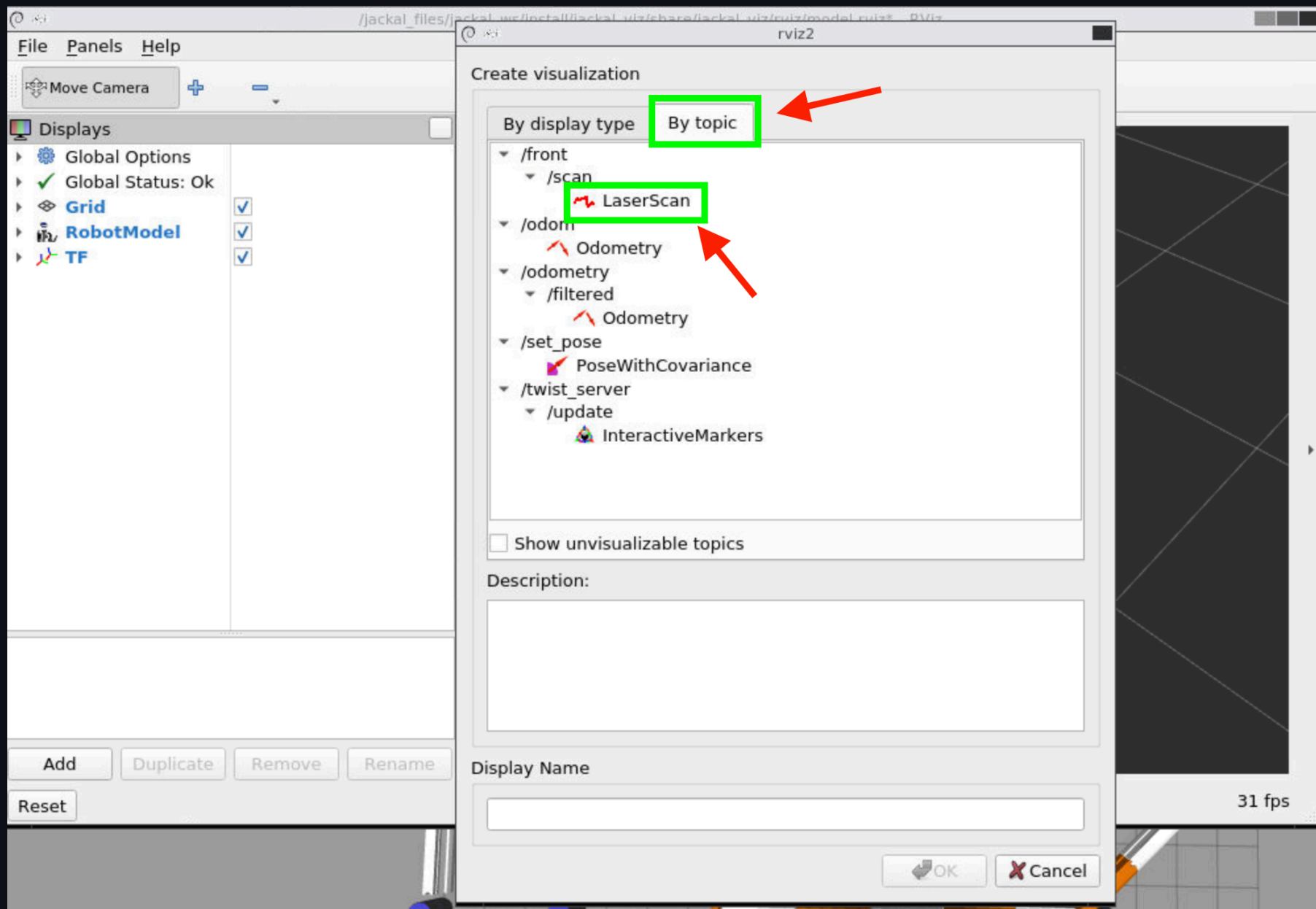
```
Type: sensor_msgs/msg/LaserScan  
Publisher count: 1  
Subscription count: 0
```

Visualize LaserScan in Rviz

Jackal View Model

```
ros2 launch jackal_viz view_model.launch.py
```





Examples

Local Planners

- Heuristic-based planning
- Rule-based planning

wall_follow

Run wall_follow in jackal sim

```
python3 wall_follow.py
```

Additional resources (and References)

[ros2 \(foxy\) tutorials](#)

| Note ros2 foxy is EOL

[jackal](#)

F1Tenth Team