Understanding ROS 2 Topics

Estimated time to completion: **1 hour**

3.4 Define a Topic

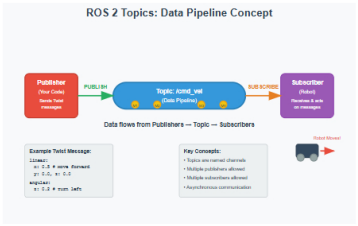
In the last section, you learned the basics of working with **Topics** in ROS 2: how to list them, extract key information, publish messages, and capture them.

Let's recap these concepts:

- You can send messages to a **Topic** (known as **publishing** to a Topic) and read messages from a **Topic** (known as **subscribing** to it).
- In **Example 3.3**, you published **Twist messages** (which represent velocities) to the **/cmd_vel** Topic.
- The robot, which was subscribed to this Topic, received the messages and moved accordingly.

So, how can we define what a **Topic** is?

Imagine a `Topic` as a pipeline that allows data to flow between different parts of the system, where messages are sent and received in a structured manner.



So far, you've only been working with topics through command-line tools. While this is very useful, especially for debugging potential issues with your robot, the most common way of working with topics is by creating **ROS 2 programs** that interact with them automatically.

In this section, you'll create a new **ROS 2 program** in Python to work with these topics.

- Exercise 3.1 -

TOPICS TO REVIEW
- ROS2 package creation
- ROS2 python scripts creation with custom ROS2 nodes
- ROS2 launch scripts creation
- Define entry points in the `setup.py` for ROS2 nodes

If you don't know how to do this, please return to unit2.

- **Create a new package to house all the code for this unit.**
  Following the guidelines from the previous unit, create a package named **mars_rover_tasks**.
  The dependencies for this package should include `rclpy`, `std_msgs`, `nav_msgs`, `sensor_msgs`, and `geometry_msgs`, as we will be using elements from these packages throughout the unit.
- **Set up the launch folder and configure the setup.py.**
  Create a **launch** folder inside the **mars_rover_tasks** package, and configure the **setup.py** so that launch files with the `.launch.py` extension are correctly recognized by the ROS2 system.
- **Create a Python script for the subscriber.**
  Inside the `scripts` folder (which should be named **mars_rover_tasks** by default), create a Python file named `subscriber_obstacle_detector.py`.
  - This file should define a custom ROS2 node called **ObstacleDetectorNode**.
  - For now, the script should only start the node, print a message confirming the node has started, and then terminate. No additional functionality is required at this point.
  - Set the entry point for this script as `subscriber_obstacle_detector_executable`.
- **Compile only the specific package.**
  To compile only the **mars_rover_tasks** package, use the following command:

▶ Execute in Terminal #1

In [ ]:

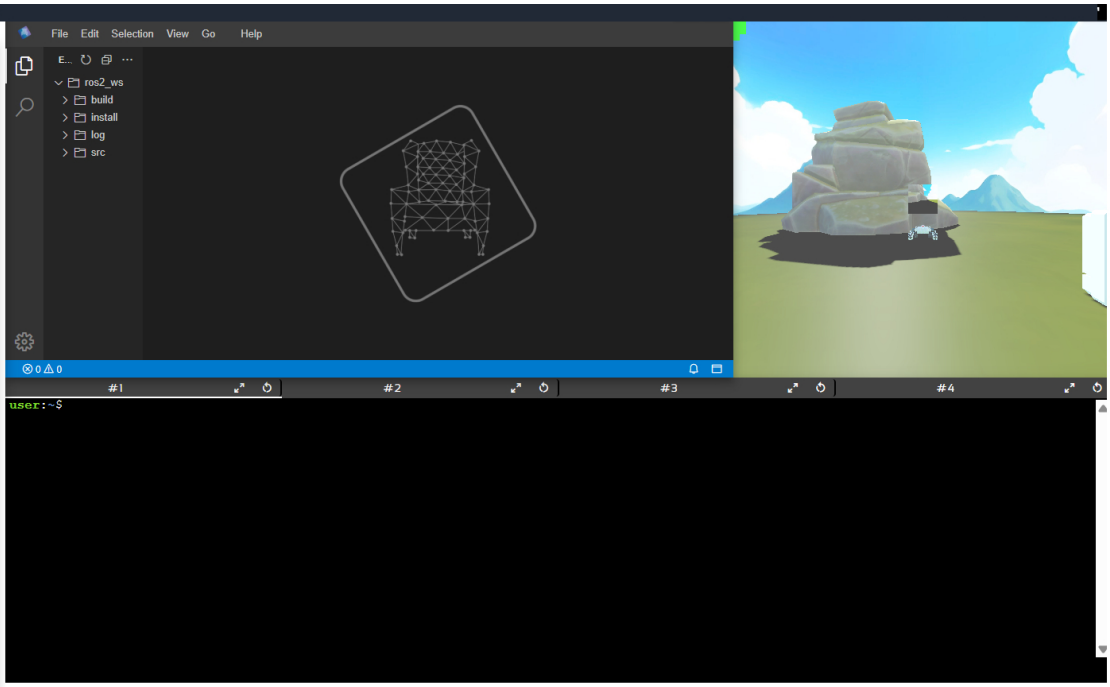
```
colcon build --packages-select mars_rover_tasks
```

- End of Exercise 3.1 -

- Notes -

You're probably asking yourself, **"How do I know which `interface` to add for each `topic` I want to use?"**

In response to your question, I propose that you use what you learned in the first section of this unit, namely `ros2 topic info <topic_name>`. You then know which `interface` works with which node, and, consequently, which one to include when creating a new package. Even if you want to know more, you could try the `ros2 interface` commands that you saw earlier in this unit.



- End of Notes -

- Expected Result -

▶ Execute in Terminal #1

In [ ]:

```
cd ~/ros2_ws
source install/setup.bash
ros2 run mars_rover_tasks subscriber_obstacle_detector_ex
```

Terminal #1 Output

```
[INFO] [1724775551.903318684] [obstacle_detector_node
```

- Expected Result -

- RECOMMENDATION -

PLEASE TRY TO COMPLETE THE EXERCISE BEFORE VIEWING THE SOLUTION.

- END RECOMMENDATION -

- Solution for Exercise 3.1 -

▶ Execute in Terminal #1

In [ ]:

```
cd ~/ros2_ws
source install/setup.bash
cd src
ros2 pkg create --build-type ament_python mars_rover_task
```

subscriber_obstacle_detector.py

In [ ]:

```
#!/usr/bin/env python

import rclpy
from rclpy.node import Node

class ObstacleDetectorNode(Node):
    def __init__(self, node_name="obstacle_detector_node"
        self._node_name = node_name
        super().__init__(self._node_name)
        self.get_logger().info(self._node_name +" Ready..

def main(args=None):
    rclpy.init(args=args)
    node = ObstacleDetectorNode()
    rclpy.shutdown()

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

setup.py

In [ ]:

```
from setuptools import find_packages, setup
import os
from glob import glob

package_name = 'mars_rover_tasks'

setup(
    name=package_name,
    version='0.0.0',
    packages=find_packages(exclude=['test']),
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        (os.path.join('share', package_name), glob('launc
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='user',
    maintainer_email='user@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'subscriber_obstacle_detector_executable = ma
```

```
        ],
    },
)
```

Execute in Terminal #1

In [ ]:

```
cd ~/ros2_ws
source install/setup.bash
colcon build --packages-select mars_rover_tasks
source install/setup.bash
```

Execute in Terminal #1

In [ ]:

```
cd ~/ros2_ws
source install/setup.bash
ros2 run mars_rover_tasks subscriber_obstacle_detector_ex
```

- End of Solution for Exercise 3.1 -

PLEASE HAVE THE EXERCISE 3.1 Done, otherwise you won't be able to continue.

24/09/2024