



AI-Task

Manipulate with turtlesim package in ROS2

By

[Khalid Alharthy]

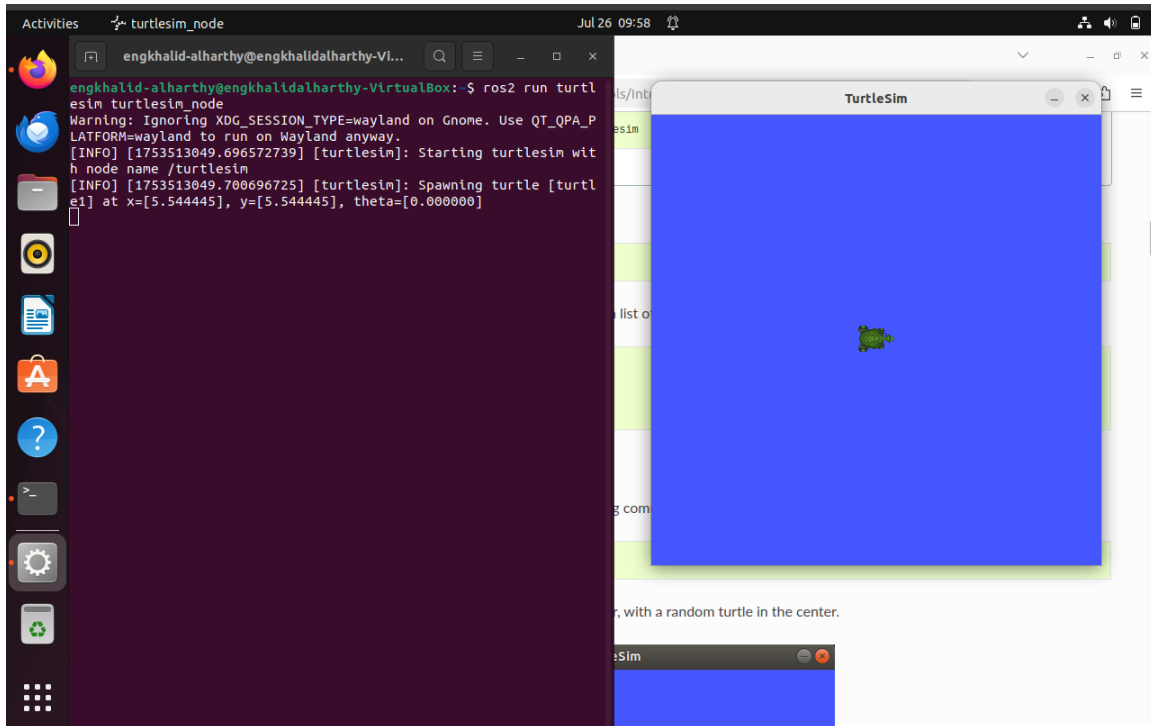
[July, 2025]

1. Using turtlesim, ros2, and rqt

1.1 Start Turtlesim

To start turtlesim, enter the following command in your terminal:

```
ros2 run turtlesim turtlesim_node
```



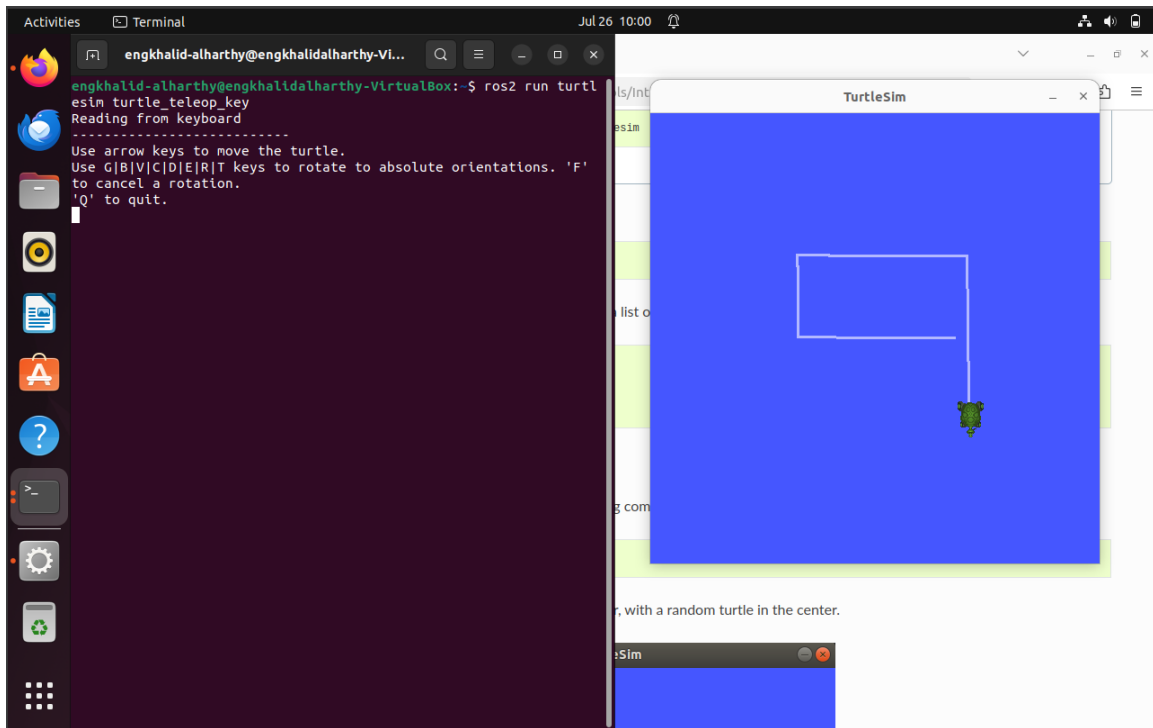
Under the command, you will see messages from the node. There you can see the default turtle's name and the coordinates where it spawns.

The simulator window should appear, with a random turtle in the center.

1.2 Use Turtlesim

Open a new terminal and source ROS 2 again. Now you will run a new node to control the turtle in the first node:

```
ros2 run turtlesim turtle_teleop_key
```



At this point you should have three windows open:

a terminal running `turtlesim_node`, a terminal running `turtle_teleop_key` and the `turtlesim` window. Arrange these windows so that you can see the `turtlesim` window but also have the terminal running `turtle_teleop_key` active so that you can control the turtle in `turtlesim`.

Use the arrow keys on your keyboard to control the turtle. It will move around the screen, using its attached “pen” to draw the path it followed so far.

You can see the nodes, and their associated topics, services, and actions, using the list subcommands of the respective commands:

ros2 node list

ros2 topic list

ros2 service list

ros2 action list

```
engkhalid-alharthy@engkhalidalharthy-VirtualBox: ~  
engkhalid-alharthy@engkhalidalharthy-VirtualBox:~$ ros2 node list  
/teleop_turtle  
/turtlesim  
engkhalid-alharthy@engkhalidalharthy-VirtualBox:~$ ros2 topic list  
/parameter_events  
/rosout  
/turtle1/cmd_vel  
/turtle1/color_sensor  
/turtle1/pose  
engkhalid-alharthy@engkhalidalharthy-VirtualBox:~$ ros2 service list  
/clear  
/kill  
/reset  
/spawn  
/teleop_turtle/describe_parameters  
/teleop_turtle/get_parameter_types  
/teleop_turtle/get_parameters  
/teleop_turtle/list_parameters  
/teleop_turtle/set_parameters  
/teleop_turtle/set_parameters_atomically  
/turtle1/set_pen  
/turtle1/teleport_absolute  
/turtle1/teleport_relative  
/turtlesim/describe_parameters
```

```
engkhalid-alharthy@engkhalidalharthy-VirtualBox: ~  
/clear  
/kill  
/reset  
/spawn  
/teleop_turtle/describe_parameters  
/teleop_turtle/get_parameter_types  
/teleop_turtle/get_parameters  
/teleop_turtle/list_parameters  
/teleop_turtle/set_parameters  
/teleop_turtle/set_parameters_atomically  
/turtle1/set_pen  
/turtle1/teleport_absolute  
/turtle1/teleport_relative  
/turtlesim/describe_parameters  
/turtlesim/get_parameter_types  
/turtlesim/get_parameters  
/turtlesim/list_parameters  
/turtlesim/set_parameters  
/turtlesim/set_parameters_atomically  
engkhalid-alharthy@engkhalidalharthy-VirtualBox:~$ ros2 action list  
/turtle1/rotate_absolute
```

1.3 Install rqt

Open a new terminal to install rqt and its plugins:

```
sudo apt update
```

```
sudo apt install '~nros-humble-rqt*'
```

To run rqt:

```
rqt
```

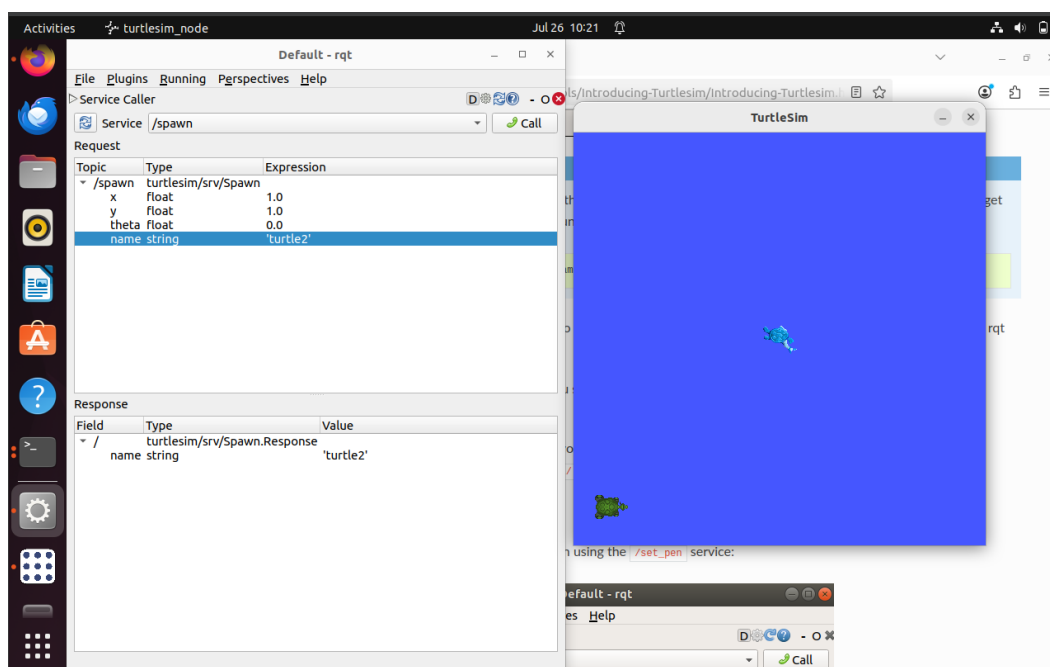
1.4 Use rqt

When running rqt for the first time, the window will be blank. No worries; just select **Plugins > Services > Service Caller** from the menu bar at the top.

Let's use rqt to call the `/spawn` service. You can guess from its name that `/spawn` will create another turtle in the turtlesim window.

Give the new turtle a unique name, like `turtle2`, by double-clicking between the empty single quotes in the **Expression** column.

You can see that this expression corresponds to the value of **name** and is of type **string**. Next enter some valid coordinates at which to spawn the new turtle, like `x = 1.0` and `y = 1.0`.



To spawn turtle2, you then need to call the service by clicking the **Call** button on the upper right side of the rqt window.

If the service call was successful, you should see a new turtle (again with a random design) spawn at the coordinates you input for **x** and **y**.

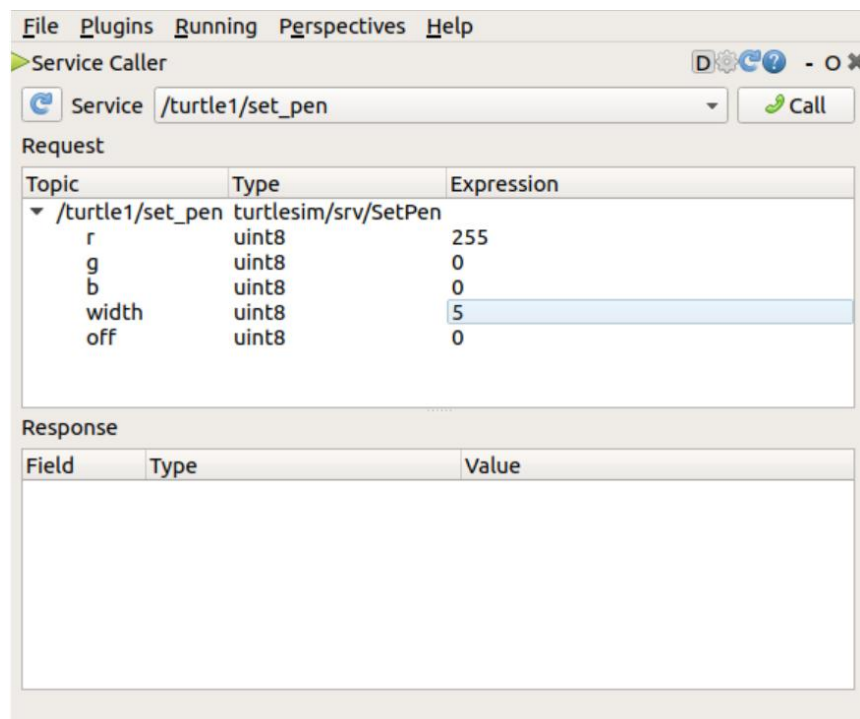
If you refresh the service list in rqt, you will also see that now there are services related to the new turtle, /turtle2/..., in addition to /turtle1/....

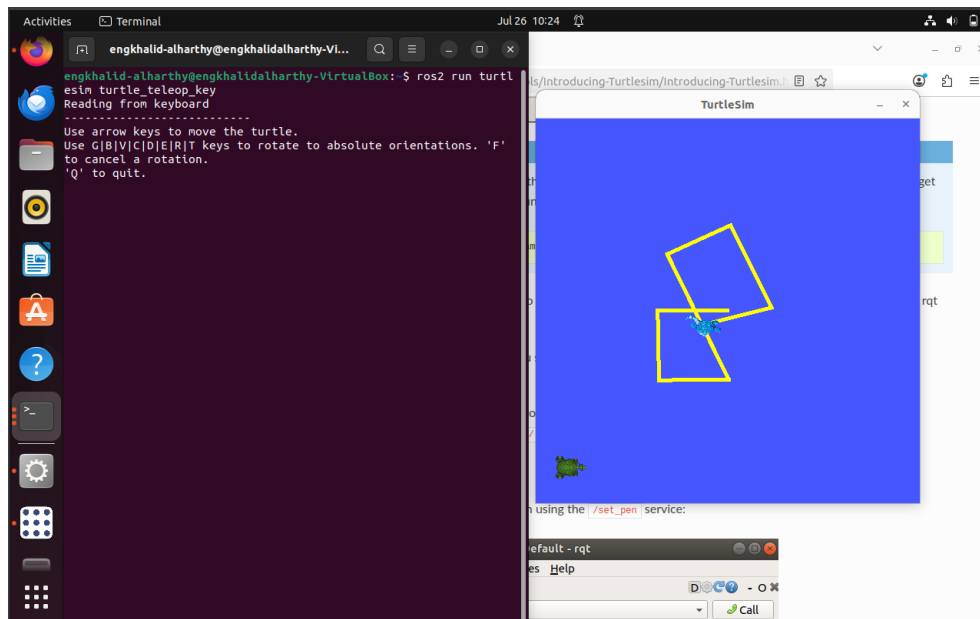
Now let's give turtle1 a unique pen using the /set_pen service:

The values for **r**, **g** and **b**, which are between 0 and 255, set the color of the pen turtle1 draws with, and **width** sets the thickness of the line.

To have turtle1 draw with a distinct red line, change the value of **r** to 255, and the value of **width** to 5. Don't forget to call the service after updating the values.

If you return to the terminal where turtle_teleop_key is running and press the arrow keys, you will see turtle1's pen has changed.





You've probably also noticed that there's no way to move turtle2. That's because there is no teleop node for turtle2.

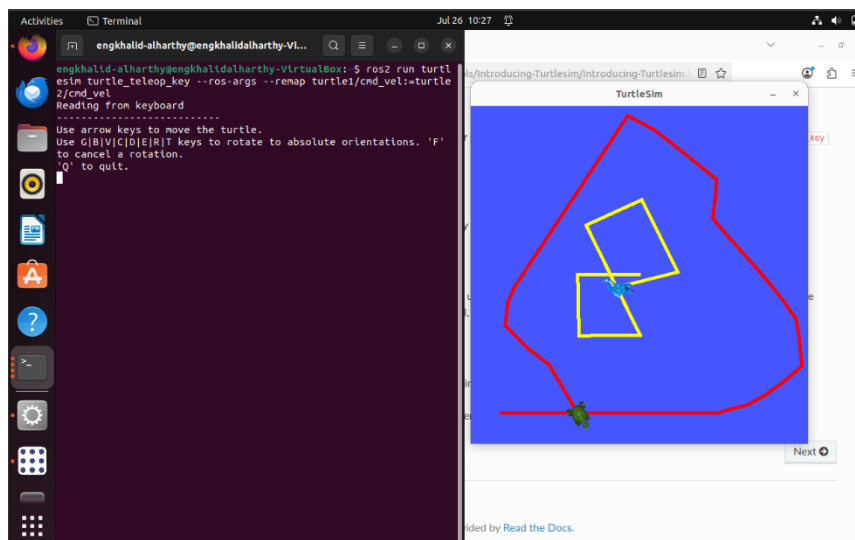
1.5 Remapping

You need a second teleop node in order to control `turtle2`. However, if you try to run the same command as before, you will notice that this one also controls `turtle1`. The way to change this behavior is by remapping the `cmd_vel` topic.

In a new terminal, source ROS 2, and run:

```
ros2 run turtlesim turtle_teleop_key --ros-args --remap
turtle1/cmd_vel:=turtle2/cmd_vel
```

Now, you can move turtle2 when this terminal is active, and turtle1 when the other terminal running `turtle_teleop_key` is active



2. Understanding topics

ROS 2 breaks complex systems down into many modular nodes. Topics are a vital element of the ROS graph that act as a bus for nodes to exchange messages.

A node may publish data to any number of topics and simultaneously have subscriptions to any number of topics.

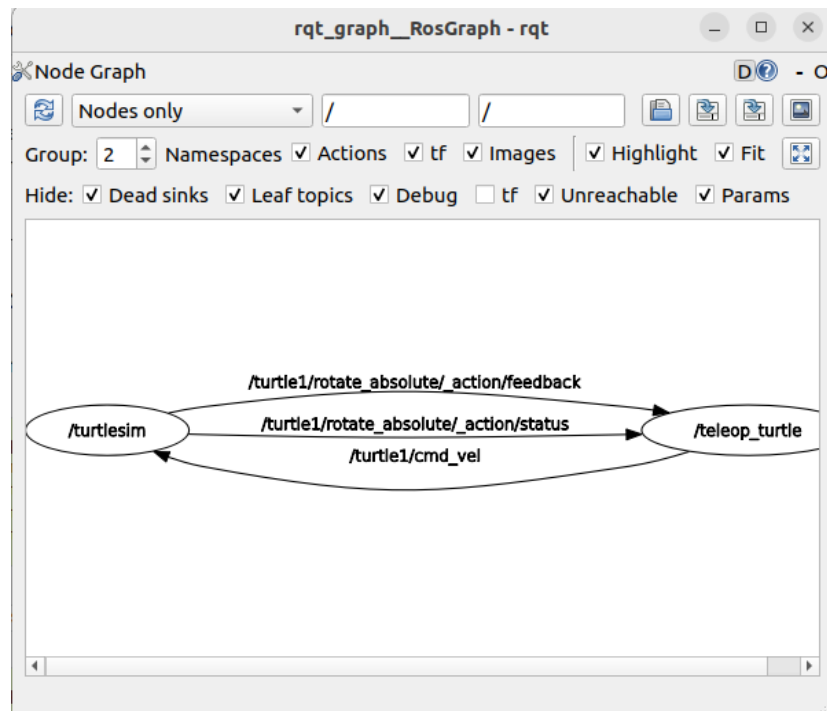
Topics are one of the main ways in which data is moved between nodes and therefore between different parts of the system.

2.1 rqt_graph

Throughout this tutorial, we will use `rqt_graph` to visualize the changing nodes and topics, as well as the connections between them.

To run `rqt_graph`, open a new terminal and enter the command:

```
rqt_graph
```



You should see the above nodes and topic, as well as two actions around the periphery of the graph (let's ignore those for now). If you hover your mouse over the topic in the center, you'll see the color highlighting like in the image above.

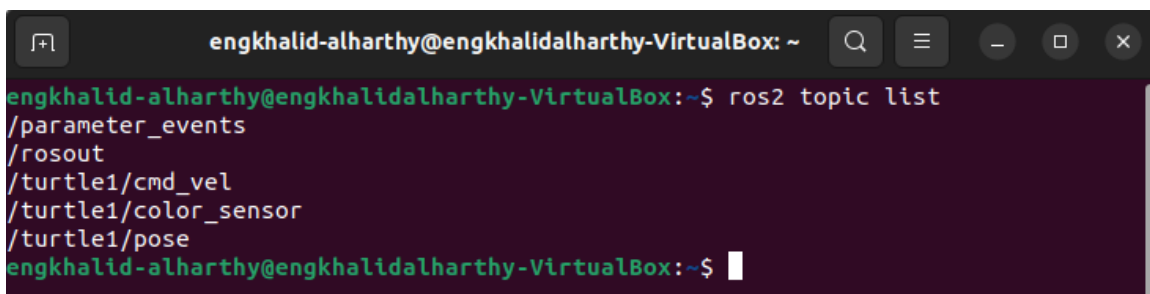
The graph is depicting how the `/turtlesim` node and the `/teleop_turtle` node are communicating with each other over a topic. The `/teleop_turtle` node is publishing data (the keystrokes you enter to move the turtle around) to the `/turtle1/cmd_vel` topic, and the `/turtlesim` node is subscribed to that topic to receive the data.

The highlighting feature of `rqt_graph` is very helpful when examining more complex systems with many nodes and topics connected in many different ways.

`rqt_graph` is a graphical introspection tool. Now we'll look at some command line tools for introspecting topics.

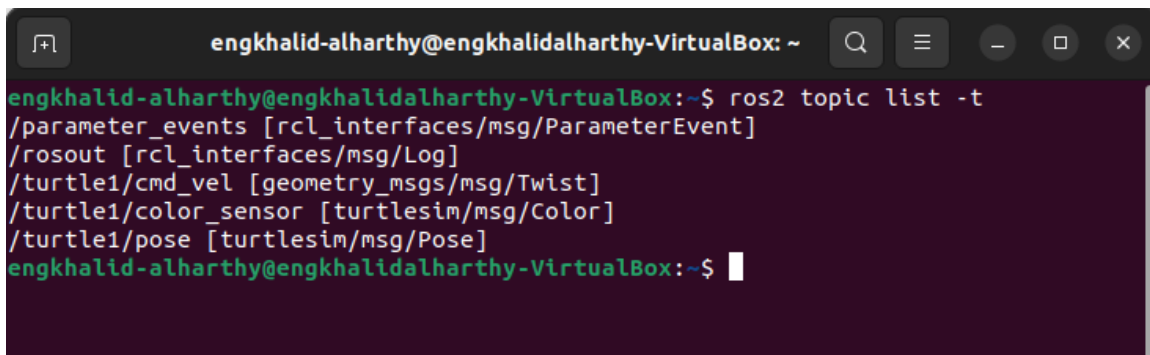
2.2 `ros2 topic list`

Running the `ros2 topic list` command in a new terminal will return a list of all the topics currently active in the system:



```
engkhalid-alharthy@engkhalidalharthy-VirtualBox: ~  
engkhalid-alharthy@engkhalidalharthy-VirtualBox:~$ ros2 topic list  
/parameter_events  
/rosout  
/turtle1/cmd_vel  
/turtle1/color_sensor  
/turtle1/pose  
engkhalid-alharthy@engkhalidalharthy-VirtualBox:~$
```

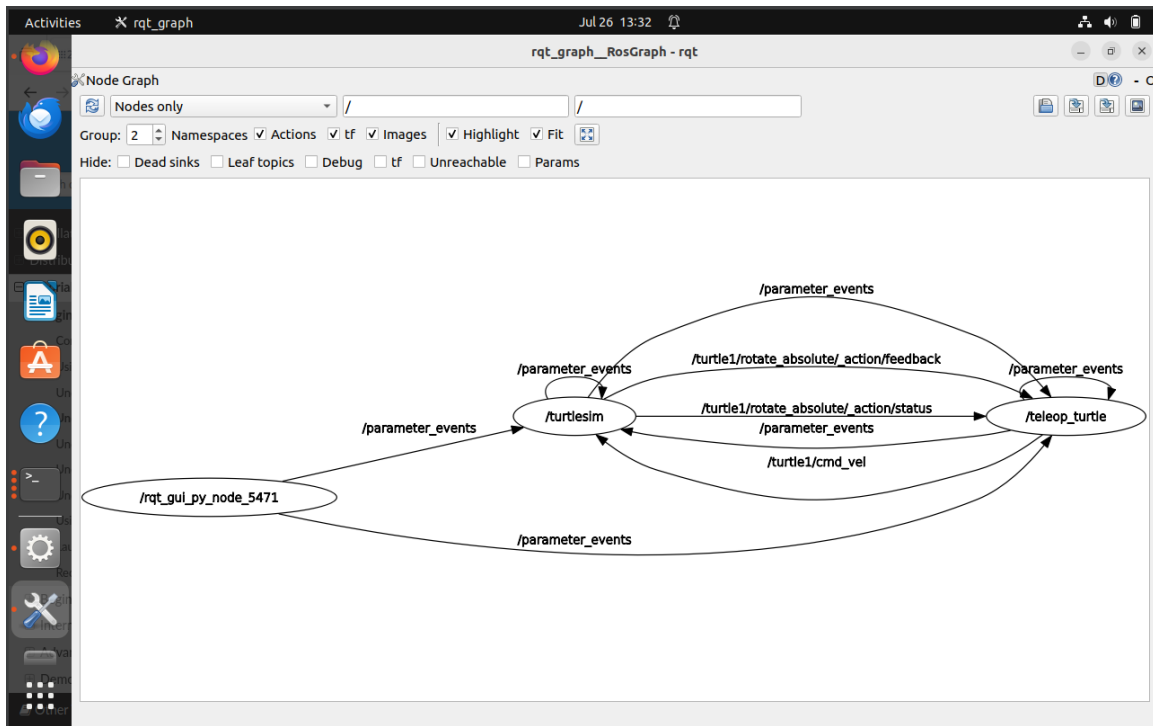
`ros2 topic list -t` will return the same list of topics, this time with the topic type appended in brackets:



```
engkhalid-alharthy@engkhalidalharthy-VirtualBox: ~  
engkhalid-alharthy@engkhalidalharthy-VirtualBox:~$ ros2 topic list -t  
/parameter_events [rcl_interfaces/msg/ParameterEvent]  
/rosout [rcl_interfaces/msg/Log]  
/turtle1/cmd_vel [geometry_msgs/msg/Twist]  
/turtle1/color_sensor [turtlesim/msg/Color]  
/turtle1/pose [turtlesim/msg/Pose]  
engkhalid-alharthy@engkhalidalharthy-VirtualBox:~$
```

These attributes, particularly the type, are how nodes know they're talking about the same information as it moves over topics.

If you're wondering where all these topics are in rqt_graph, you can uncheck all the boxes under **Hide:**



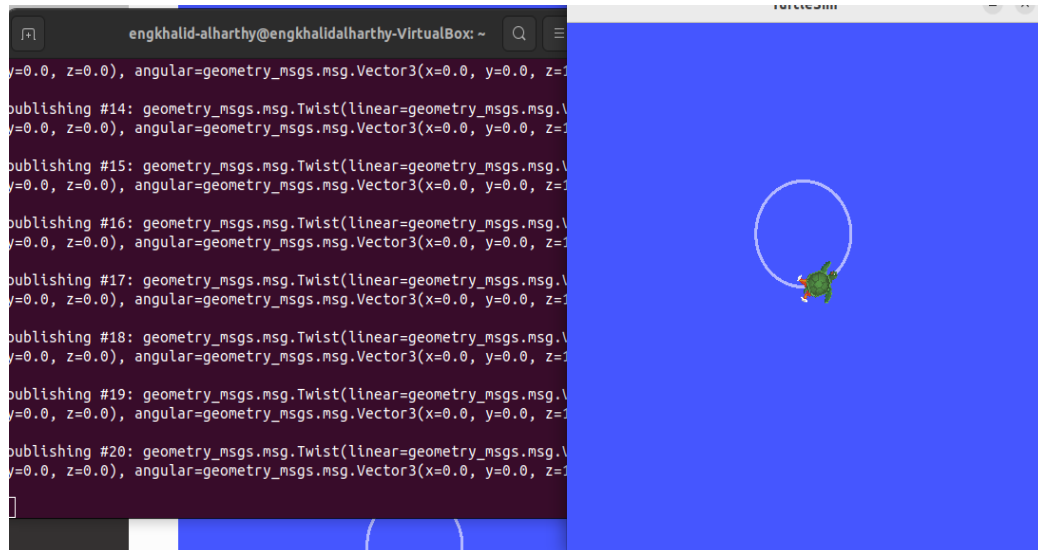
2.3 ros2 topic pub

The turtle (and commonly the real robots which it is meant to emulate) require a steady stream of commands to operate continuously. So, to get the turtle moving, and keep it moving, you can use the following command. It's important to note that this argument needs to be input in YAML syntax. Input the full command like so:

```
ros2 topic pub /turtle1/cmd_vel geometry_msgs/msg/Twist  
"{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"
```

With no command-line options, `ros2 topic pub` publishes the command in a steady stream at 1 Hz.

At times you may want to publish data to your topic only once (rather than continuously). To publish your command just once add the `--once` option.



This is just some ros2 turtlesim package operation and you can see more at the website :

<https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Introducing-Turtlesim/Introducing-Turtlesim.html>