



SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE
DEPARTMENT OF ROBOTICS AND AUTOMATION ENGINEERING

ACADEMIC YEAR: 202425 SEM: V

ASSIGNMENT NO: 6

NAME: Akshaj Chainani
PRN: 22070127008
BATCH: RA1

Aim:

ROS Basics: Understanding of nodes, topics, services, actions, parameters, rqt, recording and playing back data with turtlesim.

Apparatus:

PC with ROS2 installed (Foxy, Galactic, or equivalent)
ROS2 environment set up with workspace
Turtlesim package installed
RQT package installed
Terminal access for executing ROS commands

Theory

1. Nodes

In ROS2, a node is a fundamental process that performs computation. Each node is designed to perform a specific task, such as publishing sensor data or controlling a robot. Nodes communicate with each other through topics, services, or actions.

Node Management Commands:

To list active nodes:

```
bash
ros2 node list
```

To get detailed information about a node:

```
bash
ros2 node info /<node_name>
```



SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

DEPARTMENT OF ROBOTICS AND AUTOMATION ENGINEERING

ACADEMIC YEAR: 202425

SEM: V

The screenshot shows a ROS 2 environment with a terminal window and a TurtleSim window. The terminal window displays the following output:

```
akshaj@akshaj-HP-Pavilion-Gaming-Laptop-15-ec2xxx:~$ ros2 run turtlesim turtlesim
[INFO] [1728824286.344267227]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.376733612]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.408138368]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.440632119]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.47267330]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.504797541]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.521297314]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.552704434]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.584183529]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.616504941]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.648989834]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.680477028]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.713614024]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.744299119]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.776814735]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.792994123]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.824282952]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.856584149]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.888883443]: [turtlesim]: Rotation goal completed successfully
[WARN] [1728824286.920184149]: [turtlesim]: Rotation goal received before a previous goal finished. Aborting previous goal
[INFO] [1728824288.744619937]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824288.777068861]: [turtlesim]: Rotation goal completed successfully
```

The TurtleSim window shows a turtle moving in a circle. The terminal window also displays the following output:

```
akshaj@akshaj-HP-Pavilion-Gaming-Laptop-15-ec2xxx:~$ ros2 run turtlesim turtle_t
akshaj@akshaj-HP-Pavilion-Gaming-Laptop-15-ec2xxx:~$ ros2 run turtlesim turtle_t
Use arrow keys to move the turtle.
Use G[B|V|C|D|E|R|T] keys to rotate to absolute orientations. 'F' to cancel a rotation. 'Q' to quit.
```

The screenshot shows a ROS 2 environment with a terminal window and a TurtleSim window. The terminal window displays the following output:

```
akshaj@akshaj-HP-Pavilion-Gaming-Laptop-15-ec2xxx:~$ ros2 run turtlesim turtlesim
[INFO] [1728824286.344267227]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.376733612]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.408138368]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.440632119]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.47267330]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.504797541]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.521297314]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.552704434]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.584183529]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.616504941]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.648989834]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.680477028]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.713614024]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.744299119]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.776814735]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.792994123]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.824282952]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.856584149]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.888883443]: [turtlesim]: Rotation goal completed successfully
[WARN] [1728824286.920184149]: [turtlesim]: Rotation goal received before a previous goal finished. Aborting previous goal
[INFO] [1728824288.744619937]: [turtlesim]: Rotation goal completed successfully
[INFO] [1728824288.777068861]: [turtlesim]: Rotation goal completed successfully
```

The TurtleSim window shows a turtle moving in a circle. The terminal window also displays the following output:

```
akshaj@akshaj-HP-Pavilion-Gaming-Laptop-15-ec2xxx:~$ ros2 run turtlesim turtle_t
akshaj@akshaj-HP-Pavilion-Gaming-Laptop-15-ec2xxx:~$ ros2 run turtlesim turtle_t
Use arrow keys to move the turtle.
Use G[B|V|C|D|E|R|T] keys to rotate to absolute orientations. 'F' to cancel a rotation. 'Q' to quit.
```



SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

DEPARTMENT OF ROBOTICS AND AUTOMATION ENGINEERING

ACADEMIC YEAR: 202425 SEM: V

2. Topics

Topics provide a means for nodes to communicate asynchronously. One node can publish data to a topic, while others can subscribe to the topic to receive the data. Topics are used to implement the publishsubscribe model in ROS2.

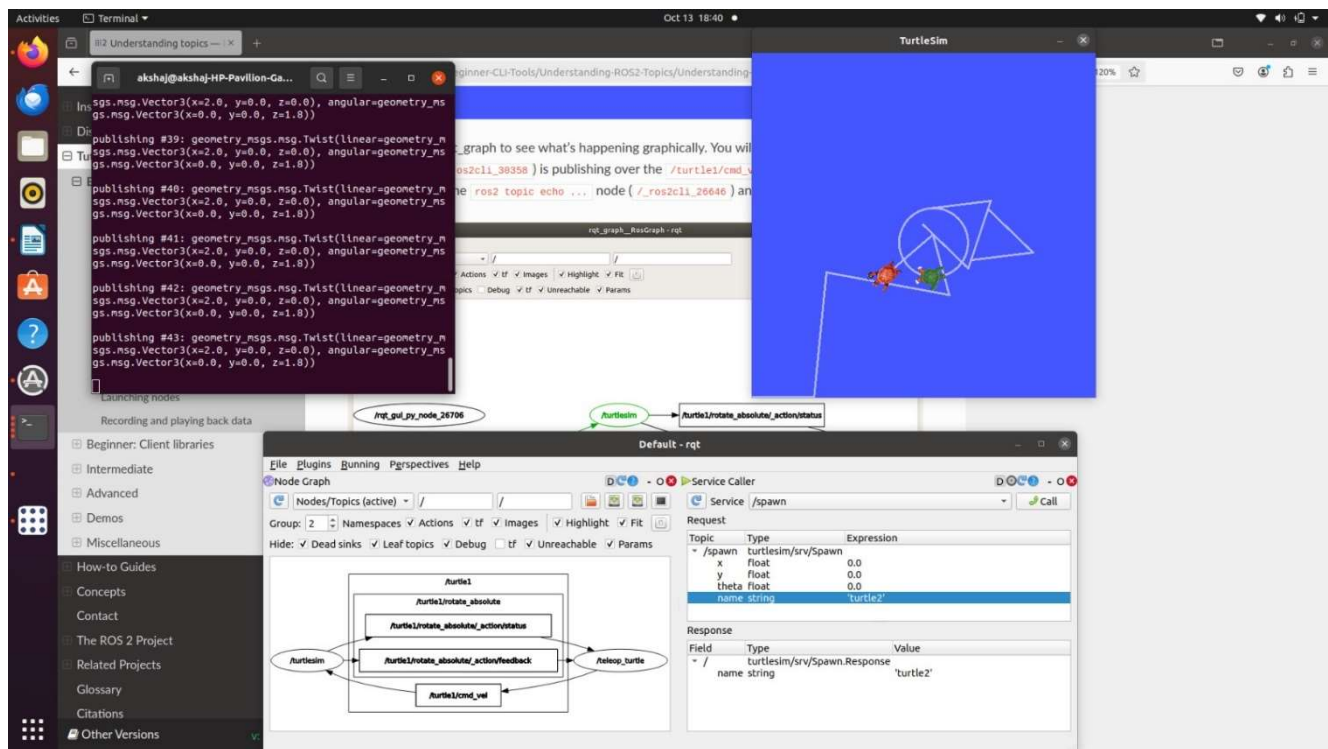
Topic Management Commands:

To list all topics:

```
bash
ros2 topic list
```

To display messages published to a specific topic:

```
bash
ros2 topic echo <topic_name>
```



3. Services

Services in ROS2 follow a clientserver architecture. A client node sends a request to the server node, and the server processes the request and sends a response back. Services are ideal for tasks that need immediate responses and are not continuous.

Service Management Commands:

To list available services:

```
bash
ros2 service list
```

To call a specific service:

```
bash
ros2 service call <service_name> <service_type>
```



SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE
DEPARTMENT OF ROBOTICS AND AUTOMATION ENGINEERING

ACADEMIC YEAR: 202425 SEM: V

4. Actions

Actions are similar to services but are designed for longrunning tasks. They provide feedback to the client during execution and allow the client to cancel or preempt the goal if necessary.

Action Management Commands:

To list available actions:

```
bash
ros2 action list
```

To send a goal to an action server:

```
bash
ros2 action send_goal <action_name> <action_type> <values>
```

5. Parameters

Parameters are used to configure nodes dynamically. These parameters can be integers, strings, booleans, or lists, and they control the behavior of nodes without the need to restart them.

Parameter Management Commands:

To list parameters for a node:

```
bash
ros2 param list
```

To get the value of a parameter:

```
bash
ros2 param get <node_name> <param_name>
```

To set a parameter:

```
bash
ros2 param set <node_name> <param_name> <value>
```

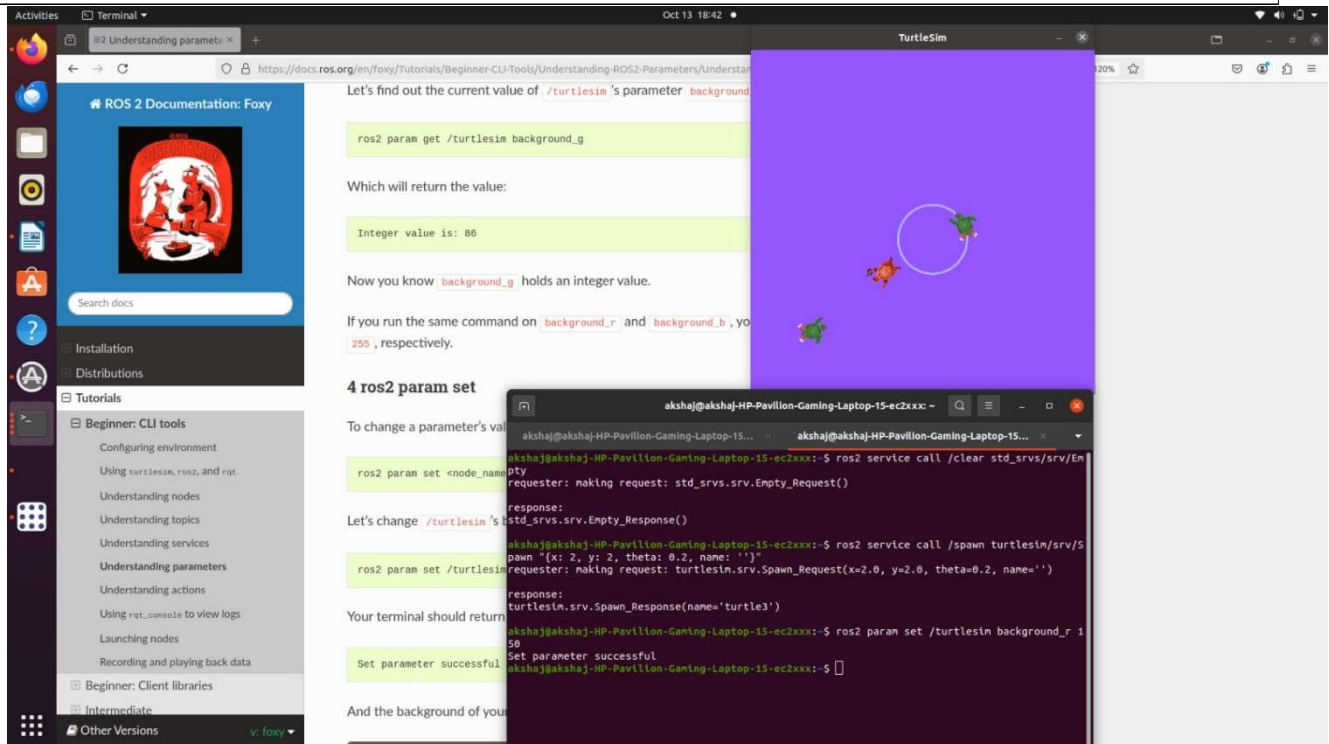


SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

DEPARTMENT OF ROBOTICS AND AUTOMATION ENGINEERING

ACADEMIC YEAR: 202425

SEM: V

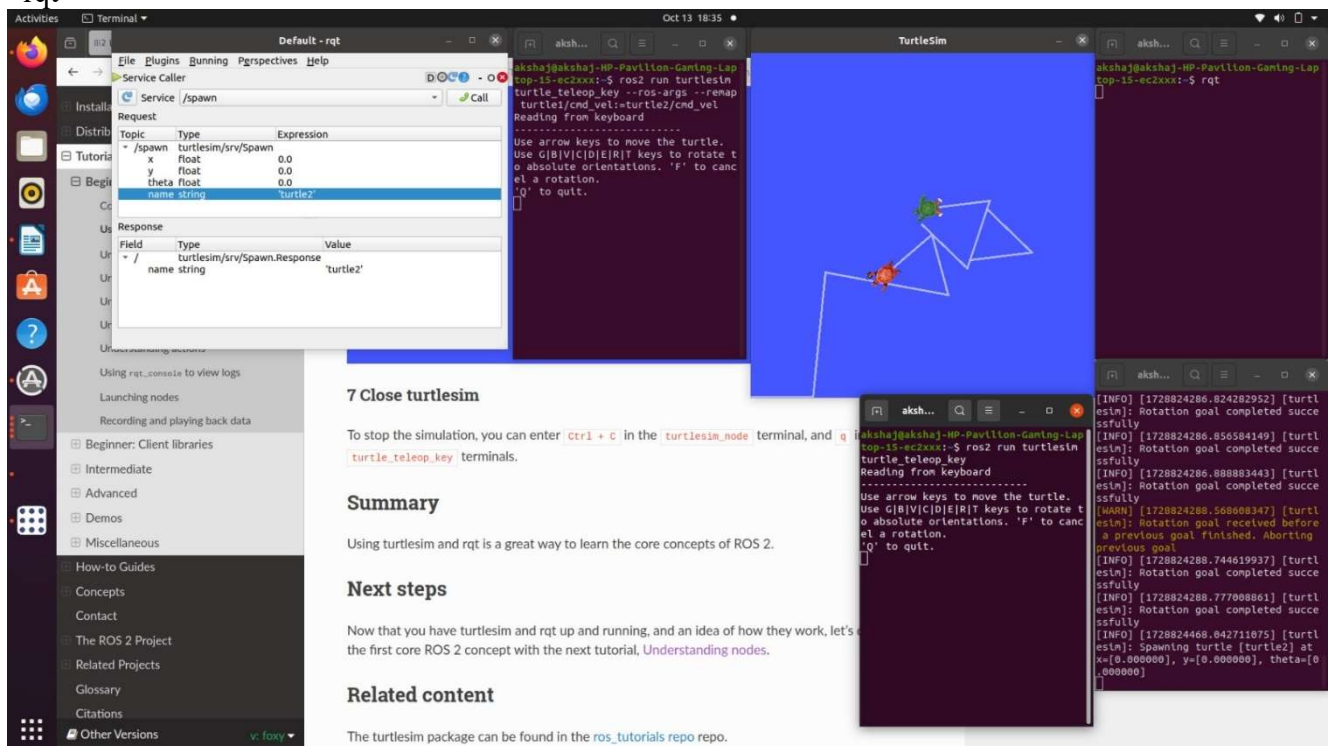


6. RQT

RQT is a graphical tool that provides a userfriendly interface for introspecting and visualizing ROS2 systems. It allows for the inspection of nodes, topics, services, and parameters in a graphical format.

Command to run RQT:

```
bash
rqt
```





SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

DEPARTMENT OF ROBOTICS AND AUTOMATION ENGINEERING

ACADEMIC YEAR: 202425

SEM: V

Activities | rqt

Oct 13 18:38

Default - rqt

File | Plugins | Running | Perspectives | Help

Node Graph

Nodes only

Group: 2 | Namespaces | Actions | tf | Images | Highlight | Fit

Hide: Dead sinks | Leaf topics | Debug | Unreachable | Params

Diagram showing nodes: `/turtle1`, `/turtle1/rotate_absolute/action/feedback`, `/turtle1/rotate_absolute/action/status`, `/turtle1/cmd_vel`, and `/rqt_graph`.

Service Caller

Service: /spawn

Request

Topic	Type	Expression
/spawn	turtlesim/srv/Spawn	
x	float	0.0
y	float	0.0
theta	float	0.0
name	string	"turtle2"

Response

Field	Type	Value
/	turtlesim/srv/Spawn.Response	
name	string	"turtle2"

Terminal

akshaj@akshaj-HP-Pavilion-Ga... | Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding

```
in: sgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=1.0))
De: publishing #39: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=1.0))
publishing #40: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=1.0))
publishing #41: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=1.0))
publishing #42: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=1.0))
publishing #43: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=1.0))
```

Launching nodes

Recording and playing back data

Beginner: Client libraries

Intermediate

Advanced

Demos

Miscellaneous

How-to Guides

Concepts

Contact

The ROS 2 Project

Related Projects

Glossary

Citations

Other Versions

Diagram showing nodes: `/turtle1`, `/turtle1/rotate_absolute`, `/turtle1/rotate_absolute/action/status`, `/turtle1/rotate_absolute/action/feedback`, `/rqt_graph`, and `/rqt_gui_py_node_26706`.

Service Caller

Service: /spawn

Request

Topic	Type	Expression
/spawn	turtlesim/srv/Spawn	
x	float	0.0
y	float	0.0
theta	float	0.0
name	string	"turtle2"

Response

Field	Type	Value
/	turtlesim/srv/Spawn.Response	
name	string	"turtle2"

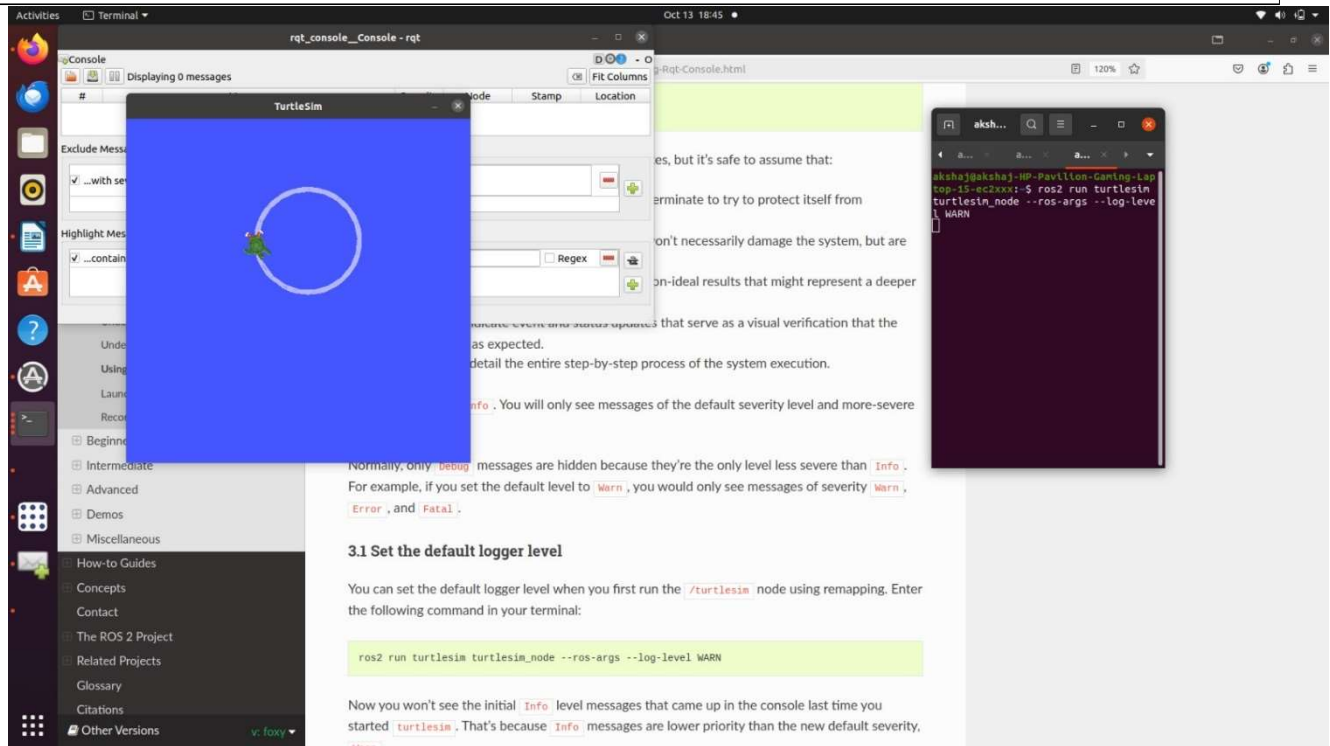
TurtleSim



SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

DEPARTMENT OF ROBOTICS AND AUTOMATION ENGINEERING

ACADEMIC YEAR: 202425 SEM: V



7. Recording and Playing Back Data

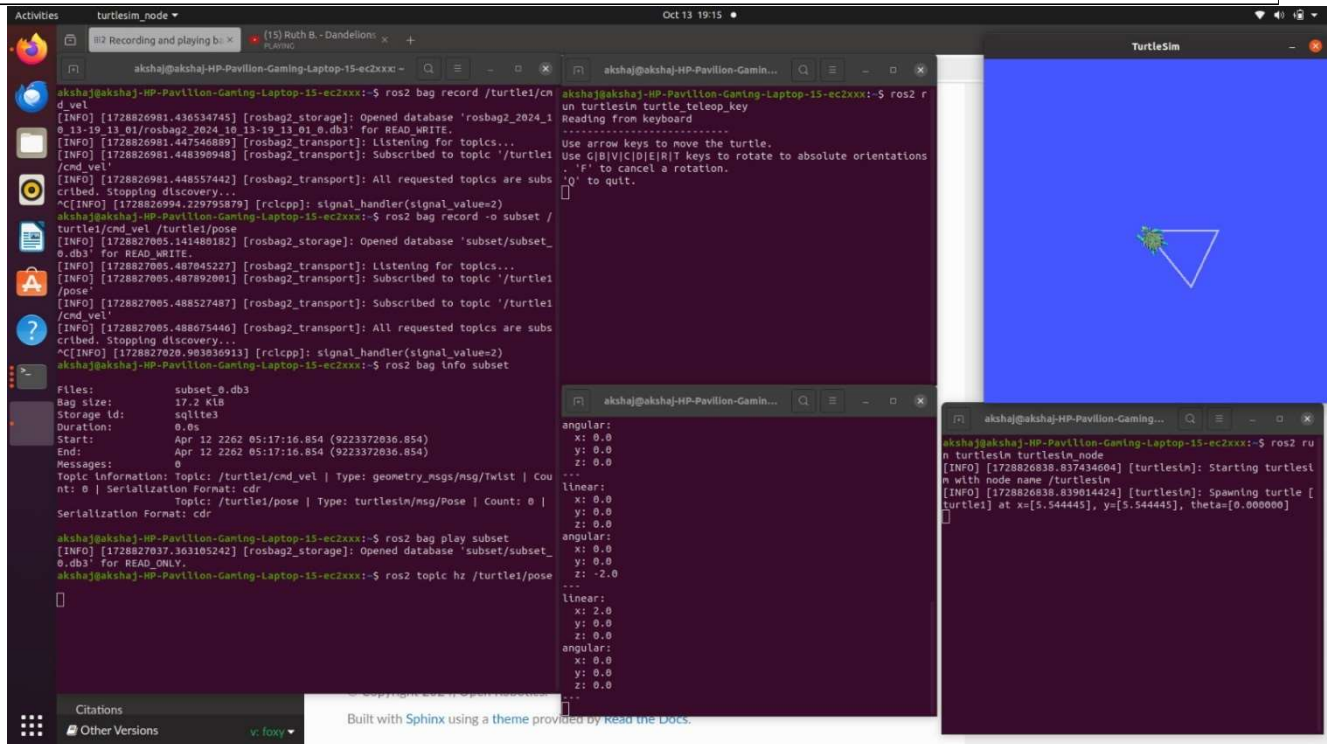
ROS2 allows recording data from topics using `ros2 bag`. This tool is useful for debugging, testing, and sharing experiments. The recorded data can be replayed to reproduce the results.

Command to record data:

```
bash
ros2 bag record <topic_name>
```

Command to play back recorded data:

```
bash
ros2 bag play <bag_file_name>
```



Basic Commands for Each Topic

Nodes

ros2 node list: Lists all active nodes in the system.

ros2 node info <node_name>: Displays detailed information about a specific node, including its publishers, subscribers, services, and actions.

Topics

ros2 topic list: Lists all active topics.

`ros2 topic echo <topic name>`: Displays messages published to a topic in realtime.

`ros2 topic pub <topic name> <msg type> <args>`: Publishes messages to a topic.

Services

ros2 service list: Lists all available services.

`ros2 service call <service name> <service type>`: Calls a service and passes necessary arguments.

Actions

ros2 action list: Lists all actions available in the ROS graph.

`ros2 action send goal <action name> <action type> <values>`: Sends a goal to an action server.

Parameters

ros2 param list: Lists parameters of a node.

ros2 param set <node name> <param name> <value>: Sets a parameter for a node.

`ros2 param get <node name> <param name>`: Retrieves the current value of a parameter.



SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

DEPARTMENT OF ROBOTICS AND AUTOMATION ENGINEERING

ACADEMIC YEAR: 202425

SEM: V

RQT

rqt: Launches the RQT graphical interface to visualize ROS systems.

Recording and Playback

ros2 bag record <topic_name>: Records data from a topic.

ros2 bag play <bag_file_name>: Plays back recorded data.

Snapshots

1. Turtlesim Setup

The screenshot shows a Linux desktop environment with the following windows:

- Turtlesim:** A window showing a turtle on a blue field. The turtle is positioned at the center of the field.
- Terminal:** A terminal window showing the command `ros2 run turtlesim turtlesim` and the output of the `turtle_teleop_key` command. The output shows the turtle's position and orientation.
- RQT:** A window showing the ROS graph. The graph displays the `turtlesim` node and its associated topics and services.

The terminal output shows the following messages:

```
[INFO] [1728824286.344267227] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.376733612] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.408138308] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.440632119] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.472367330] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.504797541] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.521297314] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.552704434] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.584103529] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.610584941] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.640989834] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.680477028] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.713614024] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.744299119] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.776814735] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.792994123] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.824282952] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.856584149] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.888884443] [turtlesim]: Rotation goal completed successfully
[WARN] [1728824286.906068347] [turtlesim]: Rotation goal received before a previous goal finished. Aborting previous goal
[INFO] [1728824286.944619937] [turtlesim]: Rotation goal completed successfully
[INFO] [1728824286.977088861] [turtlesim]: Rotation goal completed successfully
```



SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

DEPARTMENT OF ROBOTICS AND AUTOMATION ENGINEERING

ACADEMIC YEAR: 202425

SEM: V

2. RQT Interface

The screenshot shows the RQT interface with the 'Service Caller' window open. The 'Request' tab is selected, showing the 'spawn' service from the 'turtlesim/srv/Spawn' package. The 'Response' tab shows the 'Spawn_Response' with the 'name' field set to 'turtle2'. The 'TurtleSim' window shows a 2D environment with two turtles, one red and one green, on a blue background. The terminal window shows the command `ros2 run turtlesim turtle_teleop_key` and the output of the service call.

7 Close turtlesim

To stop the simulation, you can enter `ctrl + c` in the `turtlesim_node` terminal, and `q` in the `turtle_teleop_key` terminals.

Summary

Using turtlesim and rqt is a great way to learn the core concepts of ROS 2.

Next steps

Now that you have turtlesim and rqt up and running, and an idea of how they work, let's move on to the first core ROS 2 concept with the next tutorial, Understanding nodes.

Related content

The turtlesim package can be found in the [ros_tutorials](#) repo repo.

3. Parameter Setting

The screenshot shows the RQT interface with the 'Parameter Editor' window open. The 'background_g' parameter is selected, and its value is set to 86. The 'TurtleSim' window shows the same 2D environment with two turtles. The terminal window shows the command `ros2 param get /turtlesim background_g` and the output of the parameter setting.

4 ros2 param set

To change a parameter's value, you can use the `ros2 param set` command. For example, to set the `background_g` parameter to 86, you would run:

```
ros2 param set /turtlesim background_g 86
```

Let's change `/turtlesim`'s `background_r` parameter to 255, respectively.

Your terminal should return:

```
Set parameter successful
```

And the background of your turtlesim window should change to red.



SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

DEPARTMENT OF ROBOTICS AND AUTOMATION ENGINEERING

ACADEMIC YEAR: 202425

SEM: V

The screenshot displays the ROS 2 documentation page for 'Understanding Parameters' and the TurtleSim interface. The documentation includes the following text:

Let's find out the current value of `/turtlesim`'s parameter `background`

```
ros2 param get /turtlesim background_g
```

Which will return the value:

```
Integer value is: 86
```

Now you know `background_g` holds an integer value.

If you run the same command on `background_r` and `background_b`, you will get `255`, respectively.

4 ros2 param set

To change a parameter's value, use the `ros2 param set` command.

```
ros2 param set <node_name> <parameter_name> <value>
```

Let's change `/turtlesim`'s `background_g` parameter to `1`.

```
ros2 param set /turtlesim background_g 1
```

Your terminal should return:

```
Set parameter successfully
```

And the background of your TurtleSim window will change to green.

The TurtleSim window shows a green background with two turtles, one orange and one green, moving around.

4. Recording and Playback

The screenshot displays the ROS 2 documentation page for 'Recording and Playback' and the TurtleSim interface. The documentation includes the following text:

Let's record the current state of the `turtlesim` node.

```
ros2 bag record --output=/tmp/bag --topics /turtlesim
```

After running the command, a `bag` file will be created in the `/tmp` directory.

To play back the recorded data, use the `ros2 bag play` command.

```
ros2 bag play /tmp/bag
```

The TurtleSim window shows a blue background with a white line representing the path of the turtles.

The documentation also includes a 'Related content' section with links to 'ros2 bag' and 'ros2 bag2'.



SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

DEPARTMENT OF ROBOTICS AND AUTOMATION ENGINEERING

ACADEMIC YEAR: 202425

SEM: V

```
akshaj@akshaj-HP-Pavillon-Gaming-Laptop-15-ec2xxx:~$ ros2 bag record /turtle1/cmd_vel
[INFO] [1728826981.436534745] [rosbag2_storage]: Opened database 'rosbag2_2024_10_13-19_13_01/rosbag2_2024_10_13-19_13_01_0.db3' for READ_WRITE.
[INFO] [1728826981.447546889] [rosbag2_transport]: Listening for topics...
[INFO] [1728826981.448390948] [rosbag2_transport]: Subscribed to topic '/turtle1/cmd_vel'
[INFO] [1728826981.448557442] [rosbag2_transport]: All requested topics are subscribed. Stopping discovery...
^C[INFO] [1728826994.229795879] [rcicpp]: signal_handler(signal_value=2)
akshaj@akshaj-HP-Pavillon-Gaming-Laptop-15-ec2xxx:~$ ros2 bag record -o subset /turtle1/cmd_vel /turtle1/pose
[INFO] [1728827005.141480182] [rosbag2_storage]: Opened database 'subset/subset_0.db3' for READ_WRITE.
[INFO] [1728827005.487045227] [rosbag2_transport]: Listening for topics...
[INFO] [1728827005.487892081] [rosbag2_transport]: Subscribed to topic '/turtle1/pose'
[INFO] [1728827005.488527487] [rosbag2_transport]: Subscribed to topic '/turtle1/cmd_vel'
[INFO] [1728827005.488675446] [rosbag2_transport]: All requested topics are subscribed. Stopping discovery...
^C[INFO] [1728827020.983836913] [rcicpp]: signal_handler(signal_value=2)
akshaj@akshaj-HP-Pavillon-Gaming-Laptop-15-ec2xxx:~$ ros2 bag info subset
Files: subset_0.db3
Bag size: 17.2 KiB
Storage id: sqlite3
Duration: 0.0s
Start: Apr 12 22:02 05:17:16.854 (9223372036.854)
End: Apr 12 22:02 05:17:16.854 (9223372036.854)
Messages: 0
Topic information: Topic: /turtle1/cmd_vel | Type: geometry_msgs/Twist | Count: 0 | Serialization format: cdr
Topic: /turtle1/pose | Type: turtlesim/pose | Count: 0 | Serialization format: cdr
akshaj@akshaj-HP-Pavillon-Gaming-Laptop-15-ec2xxx:~$ ros2 bag play subset
[INFO] [1728827037.363105242] [rosbag2_storage]: Opened database 'subset/subset_0.db3' for READ_ONLY.
akshaj@akshaj-HP-Pavillon-Gaming-Laptop-15-ec2xxx:~$ ros2 topic hz /turtle1/pose
angular:
  x: 0.0
  y: 0.0
  z: 0.0
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -2.0
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
...
```

Learning Outcomes

- Gained a deep understanding of ROS2 nodes, topics, services, actions, and parameters.
- Developed proficiency in using RQT to visualize and interact with ROS2 systems.
- Learned how to record and replay msgs data using ros2 bag, which is essential for debugging and testing in ROS.
- Practiced controlling the Turtlesim simulation using both the terminal and graphical tools.