When working with ROS 2, you use "workspaces." The main one is called the underlay, and any others you create are called overlays. This helps you work with different versions of ROS 2 and different packages easily.

**Steps:**

**Source the Setup Files:** Every time you open a new shell, run this command to set up ROS 2:

source /opt/ros/jazzy/setup.bash

If the setup file is somewhere else, change the command to match its location.

**Add to Startup Script:** To avoid running the setup command every time, add it to your shell's startup script:

echo "source /opt/ros/jazzy/setup.bash" >> ~/.bashrc

**Check Environment Variables:** Make sure the ROS 2 environment variables are set up correctly by running:

printenv | grep -i ROS

**Set ROS\_DOMAIN\_ID:** If you need to set a domain ID, use this command:

export ROS\_DOMAIN\_ID=<your\_domain\_id>

(Arduino based)

Adding it to your startup script:

echo "export ROS\_DOMAIN\_ID=<your\_domain\_id>" >> ~/.bashrc

**Set ROS\_AUTOMATIC\_DISCOVERY\_RANGE:** To limit ROS 2 communication range, set the ROS\_AUTOMATIC\_DISCOVERY\_RANGE variable.

***Turtleism –***

Turtlesim is a basic simulator to help you learn ROS 2. It shows how ROS 2 works, which helps you understand what you'll do with real robots later.

**Key Tools:**

**ros2 Tool:** Manages and interacts with ROS 2 systems using various commands.

**rqt:** A GUI tool that provides a user-friendly way to interact with ROS 2, though you can do everything from the command line as well.

**Install Turtlesim:**

First, source your setup files in a new terminal.

Install Turtlesim with:

* + sudo apt update
  + sudo apt install ros-jazzy-turtlesim

**Check the installation:**

ros2 pkg executables turtlesim

This should list Turtlesim executables like turtlesim\_node.

**Start Turtlesim:**

*ros2 run turtlesim turtlesim\_node*

(Arduino based)

(A window with a turtle in the center will appear.)

**Controlling the Turtle:**

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

*ros2 run turtlesim turtle\_teleop\_key*

(Use the arrow keys to move the turtle and draw on the screen.)

**Install rqt:**

In a new terminal, install rqt and its plugins

*sudo apt update*

*sudo apt install '~nros-jazzy-rqt\*'*

Run it:

*rqt*

Select Plugins -> Services -> Service Caller from the menu.

**TO Use rqt:**

Use the refresh button to load all services.

Select the /spawn service to create a new turtle, turtle2, at coordinates like x = 1.0, y = 1.0.

Call the service to see the new turrtle in the window.

**To Set Pen Color:**

Use the / set\_pen service to change turtle1’s pen color.

Set r to 255 for red and width to 5. Call the service to update the pen.

**Close Turtlesim:**

Stop the progranm with Ctrl + C in the turtlesim\_node terminal

Turtlesim is a grasping way to understand the basics of ROS2. We were also shown it as a basis during the orientation.

***ROS 2 Nodes Summary***

**What are Nodes?**

In ROS 2, nodes are like small programs that each do one specific job, like controlling a motor or reading sensor data.Nodes can communicate with each other by sending and receiving data through topics, services, actions, or parameters and even sensor data

**Running Turtlesim:**

*ros2 run turtlesim turtlesim\_node*

(This opens the turtlesim window.)

**ListRunning Nodes:**

ros2 node list

(This shows the names of all running nodes, like /turtlesim.)

**Start Teleop Node:**

(IN another terminal)

ros2 run turtlesim turtle\_teleop\_key

(Run ros2 node list again, and now you’ll see)

/turtlesim

/teleop\_turtle

**Remapping Nodes:**

Remapping changes default node properties like names or topics.To rename the /turtlesim node, run:

*ros2 run turtlesim turtlesim\_node --ros-args --remap \_\_node:=my\_turtle*

(Run ros2 node list again to see)

/my\_turtle

/turtlesim

/teleop\_turtle

**Node Info:**

Get information about a node by running:

ros2 node info <node\_name>

For example, (ros2 node info /my\_turtle shows what topics, services, and actions the node uses.)

Nodes are basic units in ROS 2 that perform specific tasks.You used commands to run and manage nodes in the turtlesim package.

ros2 node list helps you find active nodes.

ros2 node info provides detailed info about nodes.

Understanding nodes and their connections is key to managing data flow in a robotics system.

***Topics in ROS-***

Topics in ROS are used for sending and receiving messages between nodes. Nodes can publish messages to a topic or subscribe to receive messages from a topic.

**Turtlesim eg:**

In the turtlesim simulation, topics are used for communication between nodes like turtlesim\_node and turtle\_teleop\_key.

**Publishers:**

1)A node that sends messages to a topic.

2)In the turtlesim example, the turtle\_teleop\_key node publishes velocity commands to the topic /turtle1/cmd\_vel.

**Subscribers:**

1)A node that receives messages from a topic.

2)The turtlesim\_node subscribes to the /turtle1/cmd\_vel topic to get the movement commands for the turtle.

**Instructions to explore Topics:**

To list all active topics:

*ros2 topic list*

To see information about a specific topic:

*ros2 topic info /turtle1/cmd\_vel*

To view messages being published to a topic:

*ros2 topic echo /turtle1/cmd\_vel*

Topics are essential for communication between nodes in ROS. Using commands like ros2 topic list, ros2 topic info, and ros2 topic echo, we can explore and understand the topic-based communication in ROS.

***Services in ROS –***

**What are Services?**

Services in ROS 2 allow nodes to send a reqquest and receive a response. They are used for tasks that need a reply after some processing, unlike topics which are used for continuous data,

**Using Services:**

In the Turtlesim package, services allow you to perform specific actions like spawning a new turtle, setting the pen colour, or resetting the program.

**Example Services in Turtlesim:**

**Spawn:**

Purpose**:** Create a new turtle at specific coordinates.

*ros2 service call /spawn turtlesim/srv/Spawn "{x: 2.0, y: 3.0, theta: 0.0, name: 'turtle2'}"*

**(**This command creates a new turtle named turtle2 at coordinates (2.0, 3.0).)

**Set Pen Service:**

Purpose**:** Change the color and width of the turtle's pen.

*ros2 service call /turtle1/set\_pen turtlesim/srv/SetPen "{r: 255, g: 0, b: 0, width: 5, off: 0}"*

**(**This command sets turtle1’s pen color to red (RGB: 255, 0, 0) and the width to 5.)

**Checking Available Services:**

To see all available services for a node, use:

*ros2 node info /turtlesim*

(This will list services like /clear, /spawn, and /turtle1/set\_pen)

Services in ROS 2 provide a request-response mechanism between nodes. In the Turtleisim example, services like spawning turtles, changing pen settings, and resetting the simulation demonstrate how services enable specific actions within a ROS 2 system.

***Parameters in ROS –***

**What are Parameters?**

Parameters in ROS 2 are used to configure nodes They can store settings like integer values, strings, or floats that nodes use to operate.

**Turtlesim Eg:**

**Setting Parameters**

Open a terminal and run:

*ros2 run turtlesim turtlesim\_node*

**Viewing Nodepara:**

List he parameters of the turtlesim\_node:

*ros2 param list /turtlesim*

(This shows the parameters available for the turtlesim\_node.)

**Getting Parametar Values:**

Get the value of a specific parameter, like background\_r (which sets the red component of the background color) -

*ros2 param get /turtlesim background\_r*

(This command returns the current value of the background\_r parameter.)

**Setting Parameter Values**

Change the value of the background\_r parameter to full red (255)

*ros2 param set /turtlesim background\_r 255*

Run the turtlesim node as shown above.

List all parameters of the turtlesim\_node:

*ros2 param list /turtlesim*

Check the current value of background\_r:

*ros2 param get /turtlesim background*\_r

Change background\_r to 255:

*ros2 param set /turtlesim background\_r 255*

Parameters in ROS 2 allow you to configure nodes at runtime. Use ros2 param list to see all parameters, ros2 param get to check values, and ros2 param set to update them. Parameters help in adjusting node behavior without modifying the code.

***ACTIONS in ROS –***

**What are Actions?**

Actions are a way for nodes to perform tasks that take time to complete and givve feedback during the process. They can do tasks like move the robot.

**Turtlesim Eg:**

Action Server:

In Turtlesim, the action server is used to rotate the turtle to a specific angle.

Eg action:

*/turtle1/rotate\_absolute.*

Action Client:

You can use an action client to send a request to th server.The client can specify the end resultl (e.g rotate the turtle to 90 degrees) and receive updates on progress.

Start the Turtlesim Node:

*ros2 run turtlesim turtlesim\_node*

Rotate Turtle Using an Action:

Use an action client to send a goal to the /turtle1/rotate\_absolute action server.This command tells the turtle to rotate to a specific angle and updates you on the progress.

Actions are useful for tasks that require ongoing feedback.In Turtlesim, the /turtle1/rotate\_absolute action server helps rotate the turtle to a specified angle and provides updates on the progress.

***Launching nodes –***

Start Turtlesim Node:

Open a terminal.

*ros2 run turtlesim turtlesim\_node*

(This opens the Turtlesim window with a turtle on the screen.)

List Running Nodes:

Open another terminal.

*ros2 node list*

((this shows all active nodes, including /turtlesim.)

Start Teleop:

Open a new terminal.

*ros2 run turtlesim turtle\_teleop\_key*

(This allows you to control the turtle with the arrow keys.)

remapping Node Names:

To rename the Turtlesim node:

*ros2 run turtlesim turtlesim\_node --ros-args --remap \_\_node:=my\_turtle*

(This changes the node name to /my\_turtle.)

Get Node Info**:**

To view details about a node:

*ros2 node info <node\_name>*

so,

ros2 node info /my\_turtle

***Recording and playing back data –***

Recording Data:

**Start Turtlesim:**

*ros2 run turtlesim turtlesim\_node*

**Record Data:**

(in a new terminal)

*ros2 bag record -o my\_bag /turtle1/pose /turtle1/cmd\_vel*

(This command starts recording the data from the /turtle1/pose and /turtle1/cmd\_vel topics into a bag file named my\_bag.)

Playing Back Data:

**Play Recorded Data:**

(in another terminal)

*ros2 bag play my\_bag*

(This command plays back the recorded data from the my\_bag file.)

Use *ros2 bag record* to save data from specific topics while Turtlesim is running. *Use ros2 bag play* to replay the recorded data later.