Team	Meliksah Yorulmazlar
Members	- Meliksah Yorulmazlar
Assignment	Lab 2 Template

Highlights

- Github link
- Playing around with ros2 turtle
- Gaining familiarity with ros2

Encountered Problems with Solutions

I encountered many problems. I did not know how to use ros2 through docker. I reinstalled ros2 on macos via the terminal which took multiple hours. I was also able to successfully install rviz2 and gazebo. However, If I am not wrong the gazebo literally became outdated the next day as they were removing support for the gazebo I downloaded in January 2025. For many tasks I had to google what was going on and look through the documentation on my own. Overall, working just on my own was really time-consuming.

Execution of Lab Tasks

Task 1-setup a GitHub repository:
I did not have any issues executing task 1. The the link to my github repository is https://github.com/melikml/S25_RobotProgramming_Yorulmazlar . I have invited both the TA and the professor to my GitHub repository.

Task 2- Introducing ROS2 commands:

Starting Nodes:
1 action: ros2 pkg executables turtlesim
2 question: what are the executable nodes within the turtlesim package?
turtlesim draw_square
turtlesim mimic
turtlesim turtle_teleop_key
turtlesim turtlesim_node

3 action start a simulated turtle

4 question: what is the name and starting pose of this simulated turtle?
The name of the turtle is turtle1 and the x-coordinate is 5.44445 and the y-coordinate is 5.44445 and theta is zero.

5 note

6 note

7 action: screenshot

8 note
9 question: what happens if you try to drive the turtle out of the window?
For me on the telesim window, the turtle left the window.

Exploring the components:

1 action: write ros2 node list
2 question: how many nodes are running?
For me it said the following:
• /teleop_turtle
•/turtlesim
Therefore, 2 nodes are running.

Topics:
1 action: ros2 node info /turtlesim
2 note
3 question:
So the turtle subscribes to:
/parameter_events: rcl_interfaces/msg/ParameterEvent
/turtle1/cmd_vel: geometry_msgs/msg/Twist
And the turtle publishes to:
/parameter_events: rcl_interfaces/msg/ParameterEvent
/rosout: rcl_interfaces/msg/Log
/turtle1/color_sensor: turtlesim/msg/Color
/turtle1/pose: turtlesim/msg/Pose

For moving, the turtle subscribes to /turtle1/cmd_vel to take velocity commands and for tracking the turtle publishes /turtle1/pose to report its position

4 action

5 question:
The turtle node publishes to:

/parameter_events: rcl_interfaces/msg/ParameterEvent
/rosout: rcl_interfaces/msg/Log
/turtle1/cmd_vel: geometry_msgs/msg/Twist
The turtle node subscribes to :
/parameter_events: rcl_interfaces/msg/ParameterEvent

According to this the node listens for parameter changes/updates. It also publishes parameter events ,publishes log messages for debugging and monitoring, and it also publishes velocity commands

The associated topic names:

• /cmd_vel
• /rosout
• /parameter_events

Information sent:
/cmd_vel sends velocity commands
/rosout sends log messages
/parameter_events sends parameter changes/updates

Information Received:
/parameter_events lets the node receive parameter changes/updates

6-note

7- action

8-note

9 question:
For me it looks about the same. However, I think both publishers and subscribers are missing parameter_events. For publishers, I think that rosout is also missing.

10 action:

11 question:
Ros2 topic list only displays the topic names without any additional details
Ros2 topic list -t displays both the topic names and adds message types to understand the structure of message types.
(ros2) meliksahyorulmazlar@heffner-wl-80 ~ % ros2 topic list

```
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
(ros2) meliksahyorulmazlar@heffner-wl-80 ~ % 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]
```

12 action

13 question:

```
Type: geometry_msgs/msg/Twist
Publisher count: 1
Subscription count: 1
```

The action tells us the publisher and subscriber count.

It means that one node is publishing information and that one node is subscribing to commands.

However in the cmd_vel context, it probably means that one node is publishing velocity commands and that one node is subscribing to these commands to move the turtle.

14 note

15 action

16 action

17 question what happened?

When typed the command nothing happened. I waited some time thinking that it was going to need some time. Then after some time, I was like what happened and still nothing happened. Then I checked that the moment I pressed the arrow keys some output came out. It published some commands.

linear: x: 2.0 y: 0.0 z: 0.0 angular: x: 0.0 y: 0.0 z: 1.5

meant that the turtle moved 2 and turned 1.5.

18 action

19 question:
Movement commands were being published (Twist messages) and the output we saw from the terminal matched my expectations.

20 action

21 action

22 question:
I entered in ros2 topic hz /turtle1/pose and I got the following response:

```
average rate: 62.490
        min: 0.012s max: 0.020s std dev: 0.00107s window: 64
average rate: 62.535
        min: 0.009s max: 0.023s std dev: 0.00124s window: 127
average rate: 62.522
        min: 0.009s max: 0.023s std dev: 0.00110s window: 190
average rate: 62.516
        min: 0.009s max: 0.023s std dev: 0.00104s window: 253
average rate: 62.510
        min: 0.006s max: 0.025s std dev: 0.00125s window: 316
average rate: 62.507
```

I would the say that the simulator is running at a frequency of 62.5 per second.

23 action

24 note

25 action

26 question:
X,y,z refer to (in a turtlesim context):
Linear:
x means forwards or downwards (x>0: forward, x <0 downwards)
y means sideways (turtlesim only supports move in the x-direction so this is ignore)
z means up or down (turtlesim is 2-d so not really possible to move in the z-axis in 2-d)
angular:

x means rotation around the x-axis (not used in turtlesim, used in 3-d robots)
y means rotation around the y-axis (not used in turtlesim, used in 3-d robots)
z means turning left or right (z >0 turns left-> counterclockwise and z <0 turns right->clockwise)

27 note

Services:

1- Action
2 question:
The two ways are:
Ros2 service list -t
Ros2 service type /clear

3 question
The interface gets --- and the implications are that there are no inputs, no outputs and that it follows a simple execution model.

4 action

5 question what happened?
requester: making request: std_srvs.srv.Empty_Request()

response:
std_srvs.srv.Empty_Response()

The screen got cleared so all the previous drawings got removed.

6 Challenge:
Completed

7 note

Actions:
1 question:
ros2 service list
I think this was meant to be an action

2 action

3 question
The turtlesim node is the server and the client is teleop_turtle.

4 note

5 action
6note
7action
8 action

Task 3:
1 action
2 queston how did you do it?

The executed the following commands to draw the square

```
(ros2) meliksahyorulmazlar@heffner-wl-80 ~ % ros2 service call /turtle1/teleport_absolute
turtlesim/srv/TeleportAbsolute "{x: 2.0, y: 2.0, theta: 0.0}"

requester: making request: turtlesim.srv.TeleportAbsolute_Request(x=2.0, y=2.0, theta=0.0)

response:
turtlesim.srv.TeleportAbsolute_Response()

(ros2) meliksahyorulmazlar@heffner-wl-80 ~ % ros2 service call /turtle1/teleport_absolute
turtlesim/srv/TeleportAbsolute "{x: 6.0, y: 2.0, theta: 0.0}"

requester: making request: turtlesim.srv.TeleportAbsolute_Request(x=6.0, y=2.0, theta=0.0)

response:
turtlesim.srv.TeleportAbsolute_Response()

(ros2) meliksahyorulmazlar@heffner-wl-80 ~ % ros2 service call /turtle1/teleport_absolute
turtlesim/srv/TeleportAbsolute "{x: 6.0, y: 6.0, theta: 0.0}"

requester: making request: turtlesim.srv.TeleportAbsolute_Request(x=6.0, y=6.0, theta=0.0)

response:
turtlesim.srv.TeleportAbsolute_Response()

(ros2) meliksahyorulmazlar@heffner-wl-80 ~ % ros2 service call /turtle1/teleport_absolute
turtlesim/srv/TeleportAbsolute "{x: 2.0, y: 6.0, theta: 0.0}"

requester: making request: turtlesim.srv.TeleportAbsolute_Request(x=2.0, y=6.0, theta=0.0)

response:
turtlesim.srv.TeleportAbsolute_Response()

(ros2) meliksahyorulmazlar@heffner-wl-80 ~ % ros2 service call /turtle1/teleport_absolute
turtlesim/srv/TeleportAbsolute "{x: 2.0, y: 2.0, theta: 0.0}"
```

Basically my method was to bring the turtle to the bottom left corner facing eastwards, then go straight then face up go up, then turn left go straight and then face downwards and then return to the starting point.

Extended task 1

Action draw the star

Future

In the future, I will start on the assignments earlier and just take action earlier on the assignment. But completing the assignment felt really nice.

Collaboration Summary

Worked on the project on my own.

References