7.15. Questions

1. Ros 2 nodes are nodes/executables that can talk to each other and perform different functions. They communicate with each other through topics.

2. Ros 2 topics are the methods which nodes communicate with each other. You can publish a topic and subscribe to a topic, which essentially means that you can talk and listen through topics.

3. Ros 2 workspaces are saved as a folder locally on your computer and you have all the nodes and topics in that workspace, as well as all the other packages.

4. ROS services implement these **request-response type of communications**. They consist of two message types: One for requesting data, and one for the response.

5. Services are based on a call-and-response model versus the publisher-subscriber model of topics. Topics allow nodes to subscribe to data streams and get continual updates, services only provide data when they are specifically called by a client.

6. You need to source the setup files, source /path/to/ros2_ws/install/setup.bash, and to automate it you can put nano ~/.bashrc.

7. Your path is inside the workspace and then inside the node, and it looks like /ros2_ws/src/my_robot_controller/.

8. The rgt_graph would produce a diagram having two nodes that have “talker” and “listener” labeled on them and there is an arrow between them labeled “chatter”.

9. After setting the workspace and everything else up, you would run ros2 run node_tester in the terminal.

10. source install/setup.bash
    ros2 run my_robot_controller my_custom_node

11. You need to open the python file and you need to put two lines of code and they have two curly braces and they spin the node and

12. import rclpy
    from rclpy.node import Node
    You also might need to import String, Parameter, and Timer

13. Publisher: topic name, message type, and queue size
    Subscriber: topic name, message type, and callback function

14. Because if you don’t add spin it only runs one time and if you add spin it will run continuously.

15. A call back is when you call a function back to implement it. It is passed as an argument to another function and is intended to be "called back" at a later time

16. Use rostopic list to get a list of the ros topics and you can track a specific topic using rostopic echo (topic name).

17. First check if the topic exists using rostopic list and then monitor the activity using rostopic echo, and lastly use **tab.rosmsg show geometry_msgs/Pose** to inspect the message structure.

18. Use echo to first check if the topic exists and then check the message type by entering **rostopic type (topic name)**.

19. You should google the error and then ask classmates and then ask the TAs and then ask the instructor.

20. test_node = my_robot_controller.my_first_node:main
    Executable name: test_node
    Package name: my_robot_controller
    Node name: my_first_node

21. You put /.py in the terminal and it asks you if you want to continue and it opens up a text file except its not a text file its a python file and then you edit and then you save it

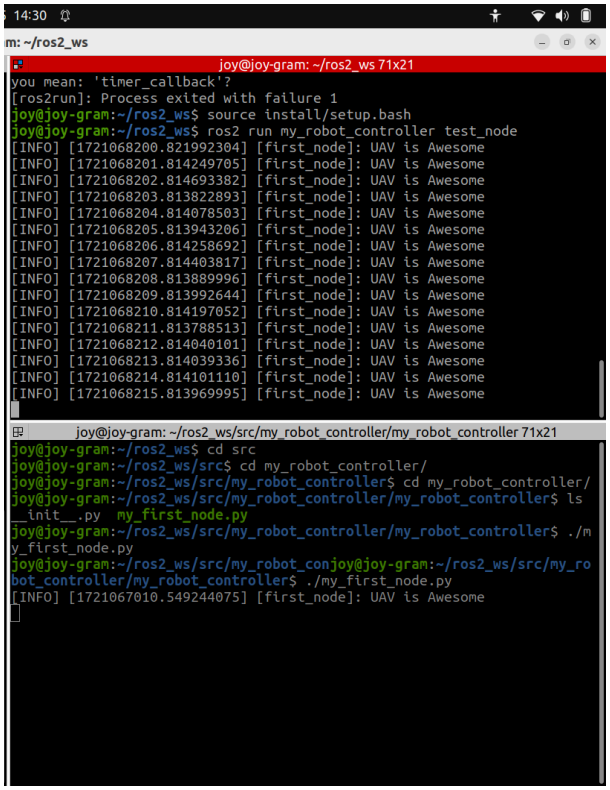
22. It adds executable permissions.

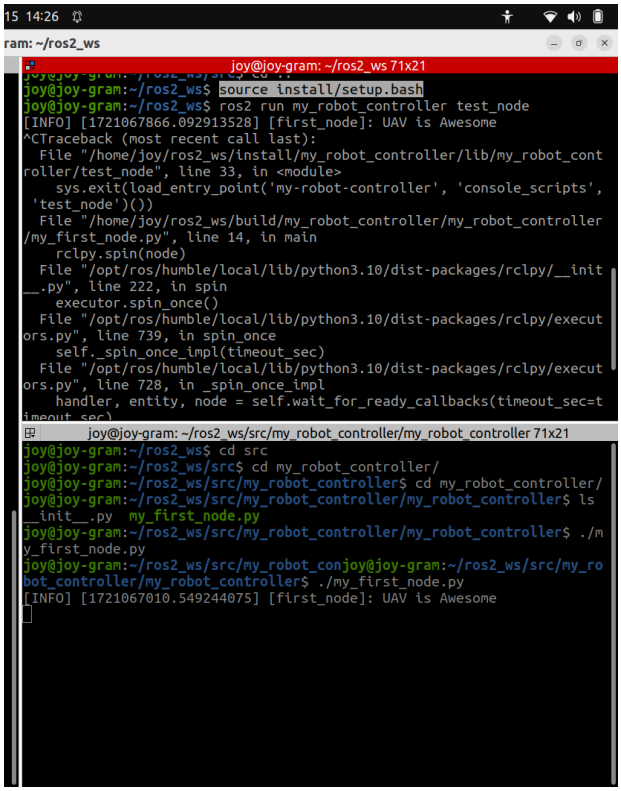
23. A src folder sources the node and it is necessary because it contains the source code for the node???

24. You create a ros package by first setting up a workspace and then you set up a package and then you put nodes and topics in, it contains init_py and a python file, and the specific command in the terminal is **ros2 pkg create my_robot_controller –build-type ament python.**

25. It allows using symbolic links instead of copying files to the ROS 2 folders during the installation, which can increase efficiency and make the process easier.

26. Bonus: Create a Ros node that will log : “ UAV is Awesome”?


14:30
m: ~/ros2_ws
joy@joy-gram: ~/ros2_ws 71x21
you mean: 'timer_callback'?
[ros2run]: Process exited with failure 1
joy@joy-gram:~/ros2_ws$ source install/setup.bash
joy@joy-gram:~/ros2_ws$ ros2 run my_robot_controller test_node
[INFO] [1721068200.821992304] [first_node]: UAV is Awesome
[INFO] [1721068201.814249705] [first_node]: UAV is Awesome
[INFO] [1721068202.814693382] [first_node]: UAV is Awesome
[INFO] [1721068203.813822893] [first_node]: UAV is Awesome
[INFO] [1721068204.814078503] [first_node]: UAV is Awesome
[INFO] [1721068205.813943206] [first_node]: UAV is Awesome
[INFO] [1721068206.814258692] [first_node]: UAV is Awesome
[INFO] [1721068207.814403817] [first_node]: UAV is Awesome
[INFO] [1721068208.813889996] [first_node]: UAV is Awesome
[INFO] [1721068209.813992644] [first_node]: UAV is Awesome
[INFO] [1721068210.814197052] [first_node]: UAV is Awesome
[INFO] [1721068211.813788513] [first_node]: UAV is Awesome
[INFO] [1721068212.814040101] [first_node]: UAV is Awesome
[INFO] [1721068213.814039336] [first_node]: UAV is Awesome
[INFO] [1721068214.814101110] [first_node]: UAV is Awesome
[INFO] [1721068215.813969995] [first_node]: UAV is Awesome
joy@joy-gram: ~/ros2_ws/src/my_robot_controller/my_robot_controller 71x21
joy@joy-gram:~/ros2_ws$ cd src
joy@joy-gram:~/ros2_ws/src$ cd my_robot_controller/
joy@joy-gram:~/ros2_ws/src/my_robot_controller$ cd my_robot_controller/
joy@joy-gram:~/ros2_ws/src/my_robot_controller/my_robot_controller$ ls
__init__.py my_first_node.py
joy@joy-gram:~/ros2_ws/src/my_robot_controller/my_robot_controller$ ./m
y_first_node.py
joy@joy-gram:~/ros2_ws/src/my_robot_conjoy@joy-gram:~/ros2_ws/src/my_ro
bot_controller/my_robot_controller$ ./my_first_node.py
[INFO] [1721067010.549244075] [first_node]: UAV is Awesome


15 14:26
ram: ~/ros2_ws
joy@joy-gram: ~/ros2_ws 71x21
joy@joy-gram:~/ros2_ws$ source install/setup.bash
joy@joy-gram:~/ros2_ws$ ros2 run my_robot_controller test_node
[INFO] [1721067866.092913528] [first_node]: UAV is Awesome
^CTraceback (most recent call last):
File "/home/joy/ros2_ws/install/my_robot_controller/lib/my_robot_cont
roller/test_node", line 33, in <module>
sys.exit(load_entry_point('my-robot-controller', 'console_scripts',
'test_node')())
File "/home/joy/ros2_ws/build/my_robot_controller/my_robot_controller
/my_first_node.py", line 14, in main
rclpy.spin(node)
File "/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/__init
__.py", line 222, in spin
executor.spin_once()
File "/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/execut
ors.py", line 739, in spin_once
self._spin_once_impl(timeout_sec)
File "/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/execut
ors.py", line 728, in _spin_once_impl
handler, entity, node = self.wait_for_ready_callbacks(timeout_sec=t
joy@joy-gram: ~/ros2_ws/src/my_robot_controller/my_robot_controller 71x21
joy@joy-gram:~/ros2_ws$ cd src
joy@joy-gram:~/ros2_ws/src$ cd my_robot_controller/
joy@joy-gram:~/ros2_ws/src/my_robot_controller$ cd my_robot_controller/
joy@joy-gram:~/ros2_ws/src/my_robot_controller/my_robot_controller$ ls
__init__.py my_first_node.py
joy@joy-gram:~/ros2_ws/src/my_robot_controller/my_robot_controller$ ./m
y_first_node.py
joy@joy-gram:~/ros2_ws/src/my_robot_conjoy@joy-gram:~/ros2_ws/src/my_ro
bot_controller/my_robot_controller$ ./my_first_node.py
[INFO] [1721067010.549244075] [first_node]: UAV is Awesome

The code:

```python
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node

class MyNode(Node):

    def __init__(self):
        super().__init__("first_node")
        self.counter_ = 0
        self.create_timer(1.0, self.timer_callback)

    def timer_callback(self):
        self.get_logger().info("UAV is Awesome" + str(self.counter_))
        self.counter_ += 1

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

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