# CSCI 545 Lab 1 - Report

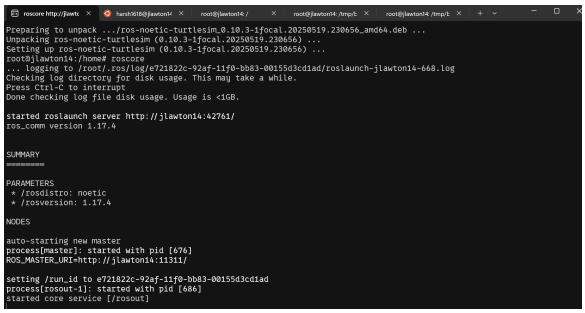
Jailbreakers

## ROS Exercises

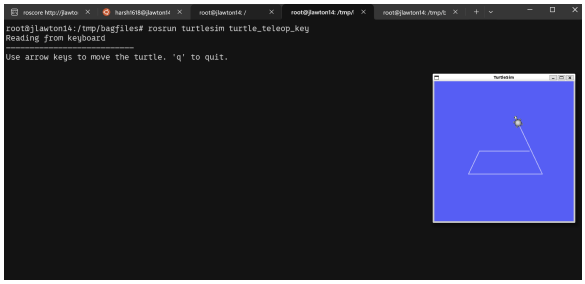
### Question 6 - Turtlesim Data Flow

The `/teleop_turtle` node publishes velocity commands on the `/turtle1/cmd_vel` topic, which the `/turtlesim_node` subscribes to. This shows the flow of data: key inputs are converted to velocity messages by the teleop node and sent to the turtlesim node, which moves the turtle accordingly.

### Question 6 - Screenshots




(a) Running the ROS Master with `roscore`.




(b) Running the keyboard teleoperation node.


(c) The RQT graph visualizing the connection between the `/teleop_turtle` publisher and the `/turtlesim` subscriber over the `/turtle1/cmd_vel` topic.

### Question 6 - Core ROS Commands

The following are fundamental command-line tools for interacting with the ROS system:

`rosservice` is used to call services, which are request/response style operations in ROS.

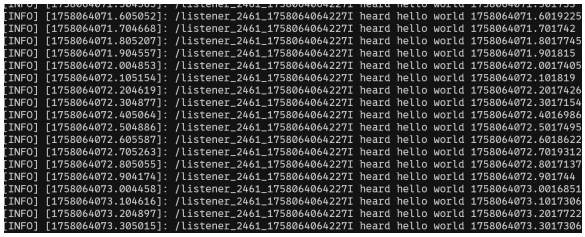
`rostopic` allows you to view, publish, and subscribe to topics, which are continuous data streams.

`rosparam` manages parameters on the parameter server, such as configuration values.

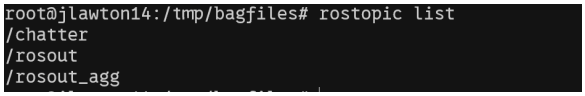
`rosbag` is used to record and replay topic data, which is useful for debugging or testing without live inputs.

### Question 7 - Publishers and Subscribers

This exercise demonstrates the fundamental **publish/subscribe model** in ROS by creating two communicating nodes. First, a publisher node is written in Python to broadcast messages of a specific type onto a named channel called a topic (e.g., `'chatter'`). Concurrently, a subscriber node is created to listen to that same topic. The subscriber uses a callback function that automatically executes to process each message it receives, typically by printing it to the console. To run the system, `roscore` is started first, followed by launching the publisher and subscriber nodes in separate terminals using `rosrun`. Successful communication is verified by observing the subscriber's output and using command-line tools like `rostopic list` and `rostopic echo` to inspect the topic's activity.




(a) Subscriber Logs.




(b) Validating the topic.

## Team Contributions

The five members of our team made equal contributions to this lab (20% each). While each member began by independently setting up their environment and working through the exercises, the final report was a collaborative effort. We collated our individual results, discussed our findings, and jointly wrote the final document.

## Resources Consulted

- Official documentation of ROS for more information on packages.
- ROS tutorial for writing a pub-sub model.
- ChatGPT for in-depth explanations and for some formatting help with LaTeX. We did not use any generative AI tool for code generation.