

Lab 1: Introduction to ROS

CSCI 545

Team Roomba

Brandon Rathburn, Devayani S Krishnan, Lakshya Ahuja, Bilin Pattasseril, Rut Patel

Fall 2025

1 Intro

In this lab, you will be setting up your system and getting acquainted with ROS. This lab is very important since we will build upon this for subsequent labs. You will need to submit a lab report (in pdf) and two python files (for the publisher/subscriber exercise).

Submission format: Your submission should be based on your group's GitHub repository. Create a folder called **Lab1**. Files should be placed in that folder like this:

```
Lab1/report.pdf
Lab1/code/<files>
```

See Section 9 for more details about the report.

2 Working with your project repository

- a. First, login to your GitHub account and accept this assignment (click your name on the student list). <https://classroom.github.com/a/GIKQTnok>

Only one group member needs to link their account to GitHub Classroom. That account will serve as the main repo manager. Other members should fork this repo and merge their changes back. Credit will be shared across the group.

- b. Install a Linux virtual machine (VM) on your laptop/PC. We recommend to install Ubuntu 20.04 or newer (22.04 or 24.04) using VMware Fusion. Please refer to Section 3 for details.
- c. In your VM, clone your repository for this lab assignment.

```
git clone https://github.com/USC-Fall-2025-CSCI-545/{your_repo}.git
cd <your-repo>
```

- d. Make your changes as necessary.
- e. After you finish this assignment, commit all your changes to your repository.

```
git add <files to add>
git commit -m <Your commit message>
```

- f. Push your changes to the repository (we will only look at the main branch for grading).
- ```
git push origin main
```

### 3 Pre-requisites for this tutorial

#### a. Linux Machine

You can complete this lab assignment on either a physical Linux machine or a virtual machine. We recommend using Ubuntu 20.04, 22.04, or 24.04. The assignment has been successfully tested on Ubuntu 24.04 running on a MacBook.

If you do **not** have a real Linux machine, go to Step. b.

If you have access to a real Linux machine, go to Step. c..

#### b. Virtual Machine (VM)

We recommend using VMware Fusion (free) with Ubuntu 20.04, 22.04, or 24.04. Allocate at least 20 GB of disk space for the installation. Do **not** install the server edition, as this lab requires the graphical interface.

You may refer to any online tutorial, for example:

Tutorial for Windows users:<https://youtu.be/SgfrHKg81Qc?si=U0XNN04zLU0YUeam>

Tutorial for Mac users:[https://youtu.be/k4mznP\\_bl2M?si=czvJpjmFwnvIWipt](https://youtu.be/k4mznP_bl2M?si=czvJpjmFwnvIWipt)

#### c. Docker Image

In this assignment, we will use a Docker image so that you can skip configuring the ROS environment manually.

Install docker service on your Linux machine:

<https://docs.docker.com/engine/install/ubuntu/>

Extension reading material for the concept of docker image:

Notion: Docker Usage Tutorial

### 4 Create Docker Container for ROS

#### a. Download official docker image for ROS

**Open up a terminal**

Download the docker image

**docker pull ros:noetic-perception**

Check the downloaded docker image. Use

**docker images**

You will find a Docker image named 'ros:noetic-perception', which is approximately 3.77 GB in size, as shown in Fig. 1.

#### b. Create docker container

```
sudo docker run -it -v /etc/localtime:/etc/localtime:ro \
-v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY \
--shm-size 8g -e GDK_SCALE -e GDK_DPI_SCALE \
--network host --ipc=host -v /media:/media \
-v /data:/data -v /home:/home --user root \
--name my_ros ros:noetic-perception
```

```

zeyu@zeyu:~/PHD_LAB/Course/CSCI-545-Introduction-to-Robotics/lab-1-introduction-
to-ros-joshshanggu$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
zeyu_coding v0 10e6eabfa9ec 5 months ago 564MB
zeyu py101 47687923c043 6 months ago 224MB
ros noetic-perception e3c7e5698db4 4 years ago 3.77GB

```

Figure 1: Check existing docker images

After running the above command, you will be inside the container. Press ‘Ctrl+D’ to exit. Whenever you exit and stop the Docker container, you can check its status with ‘docker ps -a’. As shown in Fig. 2.

```

zeyu@zeyu:~/PHD_LAB/Course/CSCI-545-Introduction-to-Robotics/lab-1-introduction-
to-ros-joshshanggu$ sudo docker run -it -v /etc/localtime:/etc/localtime:ro -v /
tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY --shm-size 8g -e GDK_SCALE -e G
DK_DPI_SCALE --network host --ipc=host -v /media:/media -v /data:/data -v /home
:/home --user root --name zeyu_ros ros:noetic-perception
root@zeyu:/# ^C
root@zeyu:/# exit
zeyu@zeyu:~/PHD_LAB/Course/CSCI-545-Introduction-to-Robotics/lab-1-introduction-
to-ros-joshshanggu$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED
STATUS PORTS NAMES
21bfef7a1721 ros:noetic-perception "/ros_entrypoint.sh ..." 2 minutes ago
Exited (130) 2 seconds ago zeyu_ros

```

Figure 2: Create docker container.

## 5 Create your own package

The simplest possible package might have a structure which looks like this:

```

my_package/
 CMakeLists.txt
 package.xml

```

Make a new package as follows:

- a. Run the container

Terminal:

```

docker start my_ros
docker exec -it my_ros /bin/bash

```

As shown in Fig. 3

```

zeyu@zeyu:~/PHD_LAB/Course/CSCI-545-Introduction-to-Robotics/lab-1-introduction-
to-ros-joshshanggu$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED
STATUS PORTS NAMES
21bfef7a1721 ros:noetic-perception "/ros_entrypoint.sh ..." 2 minutes ago
Exited (130) 2 seconds ago zeyu_ros
ef75fc519c9c 10e "/bin/bash" 5 months ago
Exited (130) 5 months ago zeyu_coding
5e66965c06c8 476 "/bin/bash" 6 months ago
Exited (130) 4 months ago zeyu_py101
zeyu@zeyu:~/PHD_LAB/Course/CSCI-545-Introduction-to-Robotics/lab-1-introduction-
to-ros-joshshanggu$ docker start 21b
21b
zeyu@zeyu:~/PHD_LAB/Course/CSCI-545-Introduction-to-Robotics/lab-1-introduction-
to-ros-joshshanggu$ docker exec -it 21b /bin/bash

```

Figure 3: Start the container.

b. Source ROS Environment

```

Source the ROS setup
source /opt/ros/noetic/setup.bash

```

c. Set up Catkin Workspace

```

Create and navigate to workspace
mkdir -p ~/catkin_ws/src cd ~/catkin_ws/src
Initialize the workspace
catkin_init_workspace

```

d. Create Your Package

```

Create the package with dependencies
catkin_create_pkg cs545_lab1 std_msgs rospy roscpp

```

This creates a package named `cs545_lab1` that lists `std_msgs`, `rospy`, and `roscpp` as dependencies. As shown in Fig. 4.

e. Build your Catkin workspace

```

Go back to workspace root
cd ~/catkin_ws
Build the workspace
catkin_make

```

As shown in Fig. 5.

f. After build succeeds, source the Workspace

```

Source the workspace setup

```

```

zeyu@zeyu:~/PHD_LAB/Course/CSCI-545-Introduction-to-Robotics/lab-1-introduction-
to-ros-joshshanggu$ docker exec -it 21b /bin/bash
root@zeyu:/# source /opt/ros/noetic/setup.bash
root@zeyu:/# mkdir -p ~/catkin_ws/src
root@zeyu:/# cd ~/catkin_ws/src
root@zeyu:~/catkin_ws/src# catkin_init_workspace
Creating symlink "/root/catkin_ws/src/CMakeLists.txt" pointing to "/opt/ros/noet-
ic/share/catkin/cmake/toplevel.cmake"
root@zeyu:~/catkin_ws/src# catkin_create_pkg cs545_lab1 std_msgs rospy roscpp
Created file cs545_lab1/package.xml
Created file cs545_lab1/CMakeLists.txt
Created folder cs545_lab1/include/cs545_lab1
Created folder cs545_lab1/src
Successfully created files in /root/catkin_ws/src/cs545_lab1. Please adjust the
values in package.xml.

```

Figure 4: Create your own package.

```

-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
-- ~~~~
-- ~~ traversing 1 packages in topological order:
-- ~~ - cs545_lab1
-- ~~~~
-- +++ processing catkin package: 'cs545_lab1'
-- ==> add_subdirectory(cs545_lab1)
-- Configuring done
-- Generating done
-- Build files have been written to: /root/catkin_ws/build
####
Running command: "make -j2 -l2" in "/root/catkin_ws/build"
####
root@zeyu:~/catkin_ws# source devel/setup.bash
root@zeyu:~/catkin_ws# echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
root@zeyu:~/catkin_ws#
root@zeyu:~/catkin_ws# rospack find cs545_lab1
/root/catkin_ws/src/cs545_lab1

```

Figure 5: Build workspace.

```

source devel/setup.bash
Add to bashrc for future sessions
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
Check if your package is found
rospack find cs545_lab1

```

```
List package dependencies
```

```
rospack depends cs545_lab1
```

Should return the name of your package, As shown in Fig. 5.

If you ever need to change dependencies for you package, you would have to edit the package.xml and CMakeLists.txt. See this tutorial page for more info.

## 6 ROS exercises

Note1: You will need multiple terminals for this section. Consider using programs to handle this use-case, such as tmux or Terminator.

Note2: Replace the screenshot with your own execution results.

a. Run turtlesim

(a) Terminal1:

```
Enter docker container:
docker exec -it my_ros /bin/bash
Install Turtlesim
First, source ROS environment
source /opt/ros/noetic/setup.bash
Update package lists
apt update
Install turtlesim package in the container
apt install -y ros-noetic-turtlesim
Run roscore service
roscore
```

As shown in Fig. 6.

A terminal window with a dark purple background and white text. The text shows the output of the 'roscore' command. It starts with 'started roslaunch server http://devayani-VMware-Virtual-Platform:37875/' and 'ros\_comm version 1.17.4'. Then it shows a 'SUMMARY' section with '=====' and a 'PARAMETERS' section with two entries: '\* /rostdistro: noetic' and '\* /rosversion: 1.17.4'. Below that is a 'NODES' section. The output continues with 'auto-starting new master', 'process[roscpp]: started with pid [597]', 'ROS\_MASTER\_URI=http://devayani-VMware-Virtual-Platform:11311/', 'setting /run\_id to 171c7d10-942c-11f0-b15d-000c296bf61f', 'process[roscpp-1]: started with pid [607]', and 'started core service [/roscpp]'.

Figure 6: Run roscore service.

(b) Terminal2:

Enter **host** terminal (**not** the container):  
Enable access to the X11 display server on Linux systems

**xhost +**

As shown in Fig. 7.

(c) Terminal3:

Enter docker container:  
**docker exec -it my\_ros /bin/bash**

```
devayani@devayani-VMware-Virtual-Platform:~$ xhost +
access control disabled, clients can connect from any host
devayani@devayani-VMware-Virtual-Platform:~$
```

Figure 7: X11 display.

```
Run turtlesim
roslaunch turtlesim turtlesim_node
```

You will see a pop up window showing a turtle at the center, As shown in Fig. 8.

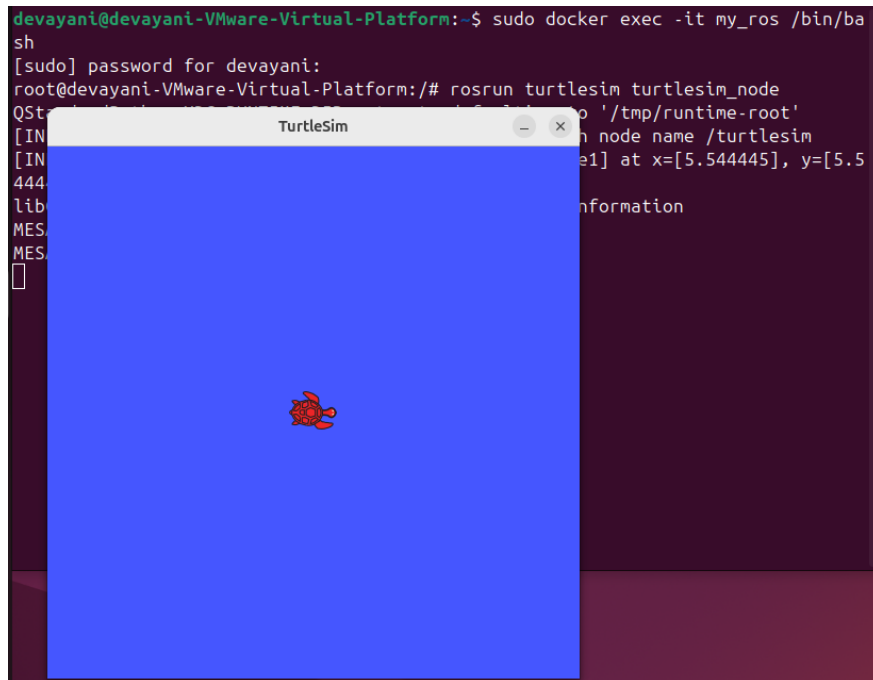


Figure 8: Run turtlesim.

(d) Terminal4:

Enter docker container:

```
docker exec -it my_ros /bin/bash
```

Use the arrow keys to move the turtlebot.

```
roslaunch turtlesim turtle_teleop_key
```

As shown in Fig. 9 and Fig. 10.



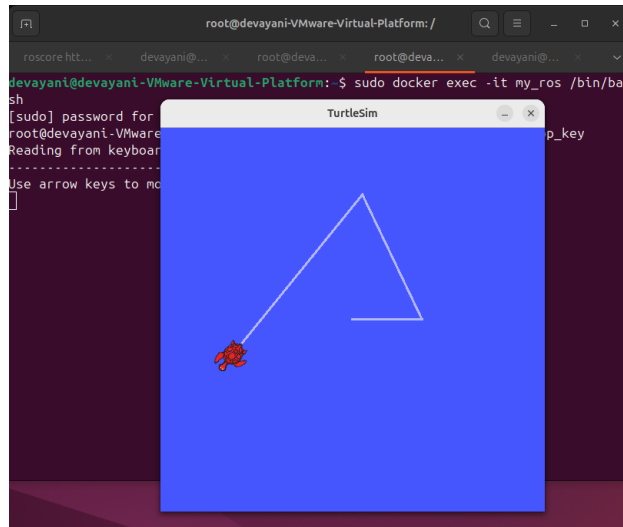


Figure 9: Teleoperation.

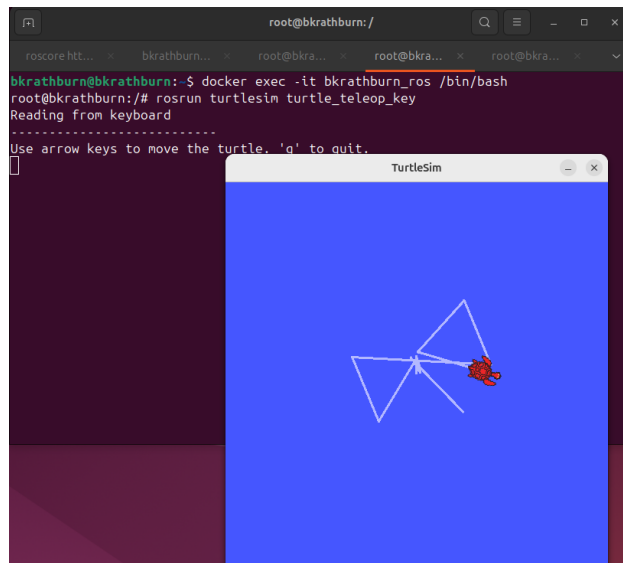


Figure 10: Teleoperation Example 2

b. Terminal5: Find topics using rostopics

Enter docker container:

`docker exec -it my_ros /bin/bash`

(a) Get list of topics with `rostopic -v`, as shown in Fig. 11

```

bkrathburn@bkrathburn:~$ docker exec -it bkrathburn_ros /bin/bash
root@bkrathburn:/# rostopic -v
rostopic is a command-line tool for printing information about ROS Topics.

Commands:
 rostopic bw display bandwidth used by topic
 rostopic delay display delay of topic from timestamp in header
 rostopic echo print messages to screen
 rostopic find find topics by type
 rostopic hz display publishing rate of topic
 rostopic info print information about active topic
 rostopic list list active topics
 rostopic pub publish data to topic
 rostopic type print topic or field type

Type rostopic <command> -h for more detailed usage, e.g. 'rostopic echo -h'

```

Figure 11: rostopic -v results

- (b) Install 'rqt' in the container  
`apt update && apt install -y ros-noetic-rqt-graph`
- (c) Run `rqt_graph` to see a graphical representation of the nodes. Explain what you see in 1-3 sentences.  
 As shown in Fig. 12.

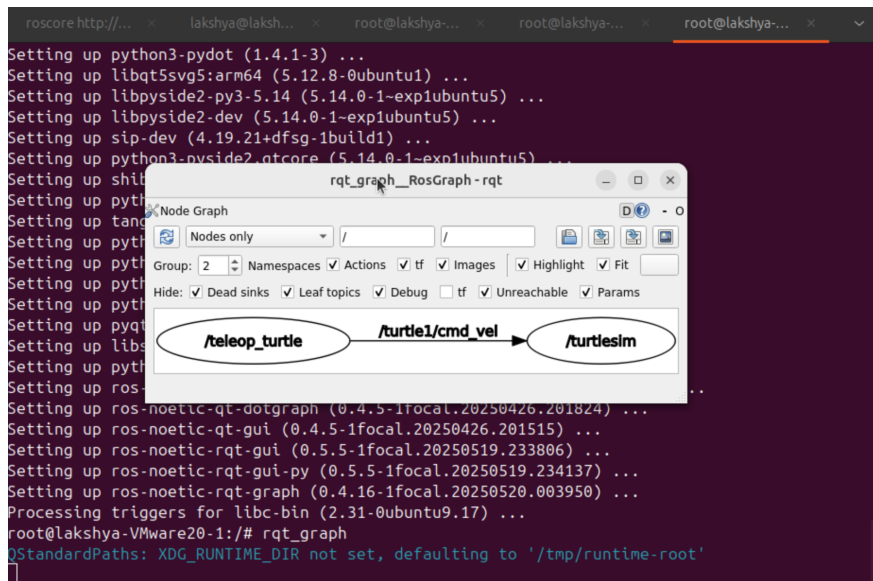


Figure 12: rqt\_graph output.

The `rqt_graph` window shows a simple ROS setup: the `/teleop_turtle` node sends movement commands over the `/turtle1/cmd_vel` topic to the `/turtlesim` node. This means the keyboard teleop node is publishing velocity messages, and the `turtlesim` node is subscribing and moving the turtle accordingly.

- (d) Get some info from a rostopic. (Shown in Fig. 13)

```

root@lakshya-VMware20-1:/# rostopic info /turtle1/cmd_vel
Type: geometry_msgs/Twist

Publishers:
* /teleop_turtle (http://lakshya-VMware20-1:45273/)

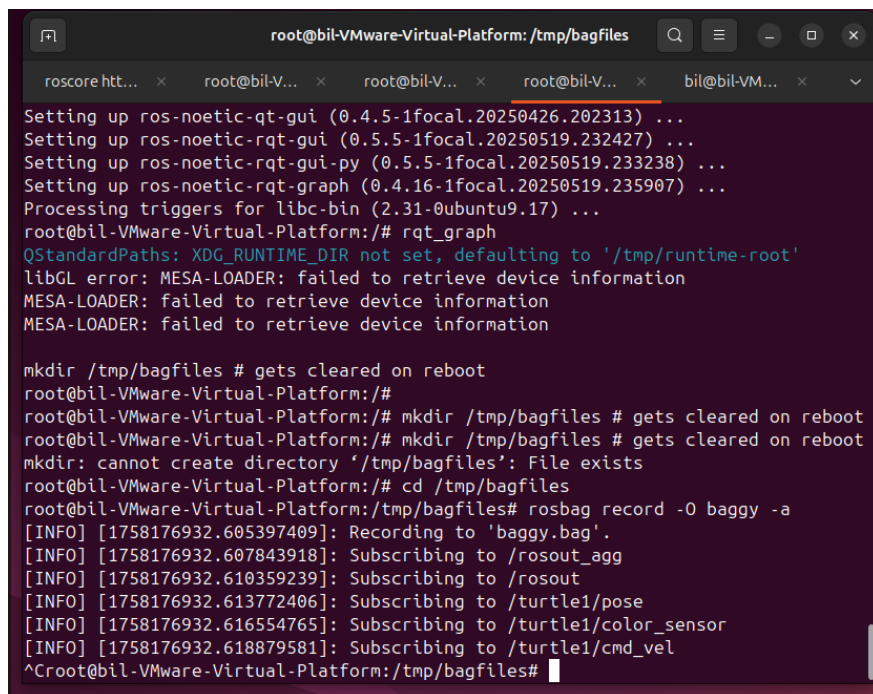
Subscribers:
* /turtlesim (http://lakshya-VMware20-1:45419/)

```

Figure 13: rostopic info output.

c. Record data using **rosbag record**

- (a) Terminal5: `mkdir /tmp/bagfiles # gets cleared on reboot`
- (b) `cd /tmp/bagfiles`
- (c) `rosbag record -O baggy -a`
- (d) Move turtlebot using arrow keys in Terminal4
- (e) After recording for a few seconds, kill (Ctrl-C) the rosbag record in Terminal5



```

root@bil-VMware-Virtual-Platform: /tmp/bagfiles
roscore htt... x root@bil-V... x root@bil-V... x root@bil-V... x bil@bil-VM... x
Setting up ros-noetic-qt-gui (0.4.5-1focal.20250426.202313) ...
Setting up ros-noetic-rqt-gui (0.5.5-1focal.20250519.232427) ...
Setting up ros-noetic-rqt-gui-py (0.5.5-1focal.20250519.233238) ...
Setting up ros-noetic-rqt-graph (0.4.16-1focal.20250519.235907) ...
Processing triggers for libc-bin (2.31-0ubuntu9.17) ...
root@bil-VMware-Virtual-Platform:/# rqt_graph
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
libGL error: MESA-LOADER: failed to retrieve device information
MESA-LOADER: failed to retrieve device information
MESA-LOADER: failed to retrieve device information

mkdir /tmp/bagfiles # gets cleared on reboot
root@bil-VMware-Virtual-Platform:/#
root@bil-VMware-Virtual-Platform:/# mkdir /tmp/bagfiles # gets cleared on reboot
root@bil-VMware-Virtual-Platform:/# mkdir /tmp/bagfiles # gets cleared on reboot
mkdir: cannot create directory '/tmp/bagfiles': File exists
root@bil-VMware-Virtual-Platform:/# cd /tmp/bagfiles
root@bil-VMware-Virtual-Platform:/tmp/bagfiles# rosbag record -O baggy -a
[INFO] [1758176932.605397409]: Recording to 'baggy.bag'.
[INFO] [1758176932.607843918]: Subscribing to /rosout_agg
[INFO] [1758176932.610359239]: Subscribing to /rosout
[INFO] [1758176932.613772406]: Subscribing to /turtle1/pose
[INFO] [1758176932.616554765]: Subscribing to /turtle1/color_sensor
[INFO] [1758176932.618879581]: Subscribing to /turtle1/cmd_vel
^Croot@bil-VMware-Virtual-Platform:/tmp/bagfiles#

```

Figure 14: rosbag record

d. Play it back using **rosbag play**

- a. Kill (Ctrl-C) the turtle teleop key in Terminal4
- b. `cd /tmp/bagfiles`
- c. `rosbag info baggy.bag`

d. rosbag play baggy.bag

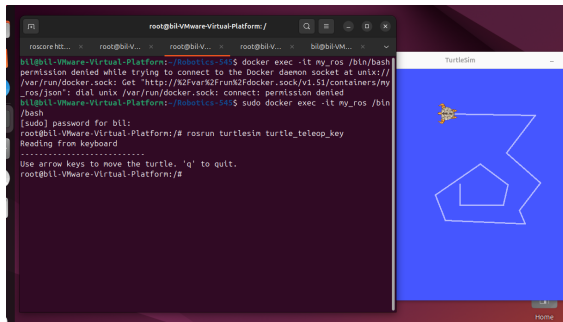


Figure 15: before

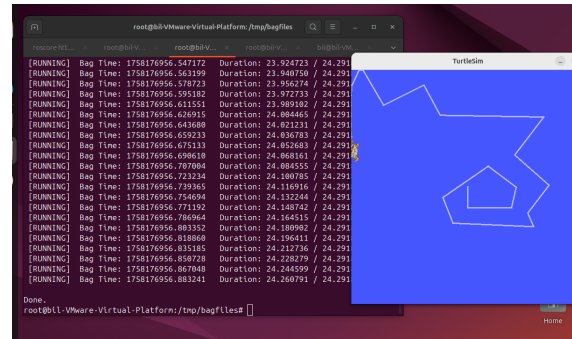


Figure 16: after

e. Explain in 3-5 sentences the difference between rosservice, rostopic, rosparams, and rosbag.

- **rosservice** is used to make one-time requests to a ROS node, for example asking a node to reset or perform a specific action.
- **rostopic** works with the live message streams (topics) where nodes continuously publish or subscribe to data, such as sensor readings or velocity commands.
- **rosparam** lets you store and manage configuration values on the ROS parameter server so that nodes can read or change settings while running.
- **rosbag** records the messages being published on topics and can later replay them, allowing you to capture and reproduce system data for testing or analysis.

## 7 Publisher and Subscriber Topics

This section requires you to write code to create a publisher and subscriber for a topic. Refer to the ROS tutorial slides presented in class or the official ROS tutorials for more information on this. You may need to use multiple terminals.

- a. Write a publisher, using the skeleton code provided in `publisher_exercise.py`  
(<https://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29>)
- b. Move the file to `~/catkin_ws/src/cs545_lab1/scripts/`
- c. `apt update`  
`apt install python3-catkin-tools`
- d. `cd ~/catkin_ws`  
`catkin_make`
- e. Make the publisher executable  
`chmod +x ~/catkin_ws/src/cs545_lab1/scripts/publisher_exercise.py`
- f. Run `roscore` to start ROS
- g. Run your publisher: `roslaunch cs545_lab1 publisher_exercise.py`

```
root@rutpatel:~/catkin_ws/src# roslaunch cs545_lab1 talker.py
[INFO] [1758253361.291168]: talker started on host=rutpatel, rate=50 Hz, tag=Team Roomba
[INFO] [1758253361.292392]: {"header":{"seq":1,"stamp":1758253361.292293},"node":"/talker_353_1758253361175","host":"rutpatel","tag":"Team Roomba","note":"Latency low, curiosity high."}
[INFO] [1758253361.311490]: {"header":{"seq":2,"stamp":1758253361.3112814},"node":"/talker_353_1758253361175","host":"rutpatel","tag":"Team Roomba","note":"Safety first, then autonomy."}
[INFO] [1758253361.331479]: {"header":{"seq":3,"stamp":1758253361.3312619},"node":"/talker_353_1758253361175","host":"rutpatel","tag":"Team Roomba","note":"East or West, CS 545 is the best."}
[INFO] [1758253361.351719]: {"header":{"seq":4,"stamp":1758253361.3513155},"node":"/talker_353_1758253361175","host":"rutpatel","tag":"Team Roomba","note":"Stay curious. Stay calibrated."}
[INFO] [1758253361.371476]: {"header":{"seq":5,"stamp":1758253361.3712616},"node":"/talker_353_1758253361175","host":"rutpatel","tag":"Team Roomba","note":"Sensors up, noise down."}
[INFO] [1758253361.391492]: {"header":{"seq":6,"stamp":1758253361.3912768},"node":"/talker_353_1758253361175","host":"rutpatel","tag":"Team Roomba","note":"Robots do it deterministically. Mostly."}
```

Figure 17: Program 1: Terminal output of the publisher node publishing host name, team name, random pre-defined robotic quotes, sequence counter and system time.

- h. Verify that the topic is being published using `rostopic list` and `rostopic echo`

```
^Croot@rutpatel:~/catkin_ws/src/cs545_lab1# rostopic list
/chat
/roscout
/roscout_agg
root@rutpatel:~/catkin_ws/src/cs545_lab1# rostopic echo chatter
data: {"header":{"seq":6396,"stamp":1758253489.1912403},"node":"/talker_353_1758253361175","host":"rutpatel","tag":"Team Roomba","note":"Safety first, then autonomy."}

data: {"header":{"seq":6397,"stamp":1758253489.2112956},"node":"/talker_353_1758253361175","host":"rutpatel","tag":"Team Roomba","note":"Sensors up, noise down."}

data: {"header":{"seq":6398,"stamp":1758253489.2317147},"node":"/talker_353_1758253361175","host":"rutpatel","tag":"Team Roomba","note":"East or West, CS 545 is the best."}

data: {"header":{"seq":6399,"stamp":1758253489.251265},"node":"/talker_353_1758253361175","host":"rutpatel","tag":"Team Roomba","note":"East or West, CS 545 is the best."}

data: {"header":{"seq":6400,"stamp":1758253489.2712386},"node":"/talker_353_1758253361175","host":"rutpatel","tag":"Team Roomba","note":"East or West, CS 545 is the best."}
```

Figure 18: Terminal output confirming the topic being published: "chatter" using `rostopic list` and messages being published to the topic "chatter" using `rostopic echo`.

- i. Write a subscriber, using the skeleton code provided in `subscriber_exercise.py` (<https://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29>)

```
root@rutpatel:~/catkin_ws/src/cs545_lab1# roslaunch cs545_lab1 listener.py
[INFO] [1758253406.232793]: heard seq=2248 from rutpatel tag=Team Roomba lat=1.5ms note='Safety first, then autonomy.'
[INFO] [1758253406.252790]: heard seq=2249 from rutpatel tag=Team Roomba lat=1.4ms note='East or West, CS 545 is the best.'
[INFO] [1758253406.273137]: heard seq=2250 from rutpatel tag=Team Roomba lat=1.7ms note='East or West, CS 545 is the best.'
[INFO] [1758253406.293272]: heard seq=2251 from rutpatel tag=Team Roomba lat=1.8ms note='East or West, CS 545 is the best.'
[INFO] [1758253406.312695]: heard seq=2252 from rutpatel tag=Team Roomba lat=1.4ms note='Safety first, then autonomy.'
[INFO] [1758253406.333039]: heard seq=2253 from rutpatel tag=Team Roomba lat=1.7ms note='Stay curious. Stay calibrated.'
[INFO] [1758253406.352231]: heard seq=2254 from rutpatel tag=Team Roomba lat=0.9ms note='Sensors up, noise down.'
[INFO] [1758253406.373589]: heard seq=2255 from rutpatel tag=Team Roomba lat=2.3ms note='Stay curious. Stay calibrated.'
```

Figure 19: Terminal output showing the subscriber node listening to the topic "chatter"

- j. Do the necessary steps to build and run the subscriber. Make sure you are able to receive the message and are printing it out.

```
^Croot@lakshya-VMware20-1:~/catkin_ws/src/cs545_lab1/scripts# roslaunch cs545_lab1 talker.py
[INFO] [1758178046.924061]: Publishing numbers: 2,8
[INFO] [1758178047.926200]: Publishing numbers: 7,6
[INFO] [1758178048.925832]: Publishing numbers: 7,4
[INFO] [1758178049.926988]: Publishing numbers: 2,7
[INFO] [1758178050.927063]: Publishing numbers: 6,8
[INFO] [1758178051.930429]: Publishing numbers: 9,6
[INFO] [1758178052.926519]: Publishing numbers: 1,0
[INFO] [1758178053.926677]: Publishing numbers: 3,3
[INFO] [1758178054.927017]: Publishing numbers: 8,4
[INFO] [1758178055.927177]: Publishing numbers: 10,2
[INFO] [1758178056.926902]: Publishing numbers: 2,5
[INFO] [1758178057.927097]: Publishing numbers: 8,9
```

Figure 20: Program 2: Terminal output of the talker node publishing random numbers.

```
^Croot@lakshya-VMware20-1:~/catkin_ws/src/cs545_lab1/scripts# rosrun cs545_lab1 listener.py
[INFO] [1758178046.927441]: 2 + 8 = 10
[INFO] [1758178047.931433]: 7 + 6 = 13
[INFO] [1758178048.930545]: 7 + 4 = 11
[INFO] [1758178049.932441]: 2 + 7 = 9
[INFO] [1758178050.931998]: 6 + 8 = 14
[INFO] [1758178051.968749]: 9 + 6 = 15
[INFO] [1758178052.933921]: 1 + 0 = 1
[INFO] [1758178053.934026]: 3 + 3 = 6
[INFO] [1758178054.931589]: 8 + 4 = 12
[INFO] [1758178055.935261]: 10 + 2 = 12
[INFO] [1758178056.932737]: 2 + 5 = 7
[INFO] [1758178057.933482]: 8 + 9 = 17
```

Figure 21: Terminal output of the listener node showing the sums received.

## 8 Additional Questions

As a very rough guideline, we anticipate that for most teams, the answers to each question below will be approximately one paragraph long (or about 4-5 bullet points). However, your answers may be shorter or longer if you believe it is necessary.

### 8.1 Resources Consulted

**Question:** Please describe which resources you used while working on the assignment. You do not need to cite anything directly part of the class (e.g., a lecture, the CSCI 545 course staff, or the readings from a particular lecture). Some examples of things that could be applicable to cite here are: (1) did you get help from a classmate *not* part of your lab team; (2) did you use external resources such as LLMs (in any capacity); (3) did you use someone's code (again, for someone *not* part of your lab team)? When you write your answers, explain not only the resources you used but HOW you used them. If you believe your team did not use anything worth citing, *you must still state that in your answer* to get full credit.

**Answer:** In terms of consulted resources, we did not refer to other classmates for any reason (outside of our lab group) for assistance or code. We did use rospy documentation for Python code, referring to the ROS Wiki  
<https://wiki.ros.org/rospy/Overview/Logging>.

### 8.2 Team Contributions

**Question:** Please describe below the contributions for each team member to the overall lab. *Furthermore, state a (rough) percentage contribution for each member.* For example, in a team of 4, did each team member contribute roughly 25% to the overall effort for the project?

**Answer:** Every person in the group worked on the lab assignment together (and on their own machines). We are attaching the output of two machines for diversity and to avoid redundancy in the screenshots, but all machines achieved the same outputs. Everyone in our group contributed 20% to the overall effort of the project. The python code for publisher and subscriber was written with the help of contributions from all the 5 team members. Each person in the team edited the python files, discussing and implementing their own ideas in it. The lab report was worked on by all the members of the group.

## 9 Report

Please complete Questions 6, 7, and 8. Include the necessary screenshots of your results. Write a report comprising of all the textual answers required in this lab. The report should be named **report.pdf**). *Your report must also answer BOTH questions in Section 8. Include both in the report and make it **very easy** for us to find.*