

ROBT 403 Laboratory Report 1

**Daniil Filimonov, Abdirakhman Onabek, Kir
Smolyarchuk**

GitHub:

[https://github.com/Menerallka/FailTeamLab1/tree/
master](https://github.com/Menerallka/FailTeamLab1/tree/master)

1. Introduction to ROS (Robot Operating System)

The Robot Operating System (ROS) is an open-source, flexible framework for writing software for robots. Initially developed by Willow Garage in 2007, ROS has since become a standard in the robotics industry, supporting a vast range of robotic platforms and applications. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionalities, message-passing between processes, and package management.

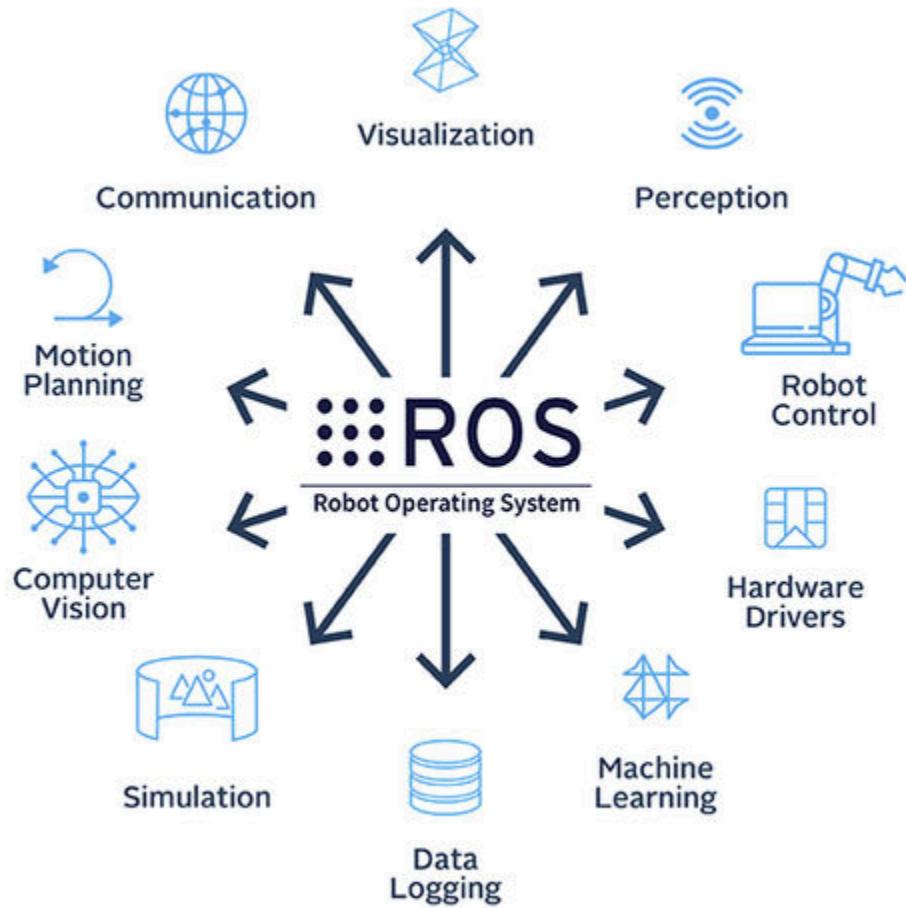


Figure 1. Versatility of ROS

ROS is designed to be highly modular, allowing developers to build complex robotic systems by connecting smaller, reusable components known as "nodes." Each node is responsible for a specific task, such as sensor data processing, motor control, or decision-making, and communicates with other nodes using a publish-subscribe messaging model.

A significant advantage of ROS is its extensive ecosystem, which includes a vast library of packages and tools developed by a global community of researchers and engineers. These resources enable rapid development and prototyping, making ROS an ideal choice for both academic research and commercial robotics applications.

2. Real-world applications

a. HERB (*Home Exploring Robot Butler*)

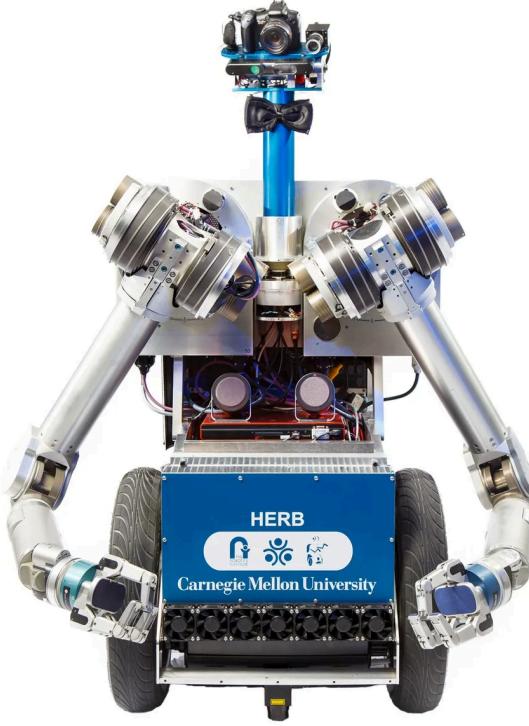


Figure 2. HERB Robot (Carnegie Mellon University)

Overview: HERB is a mobile manipulator robot developed by Carnegie Mellon University's Personal Robotics Lab. It is designed to assist with tasks in a domestic environment, such as fetching items, opening doors, and even serving food. HERB uses ROS for integrating its various sensors, planning algorithms, and manipulation tasks.

ROS Integration: HERB utilizes ROS to coordinate its perception, planning, and control systems. The robot's ability to interact with and manipulate objects in unstructured environments is heavily reliant on ROS's modular framework.

b. Husky UGV (*Unmanned Ground Vehicle*)



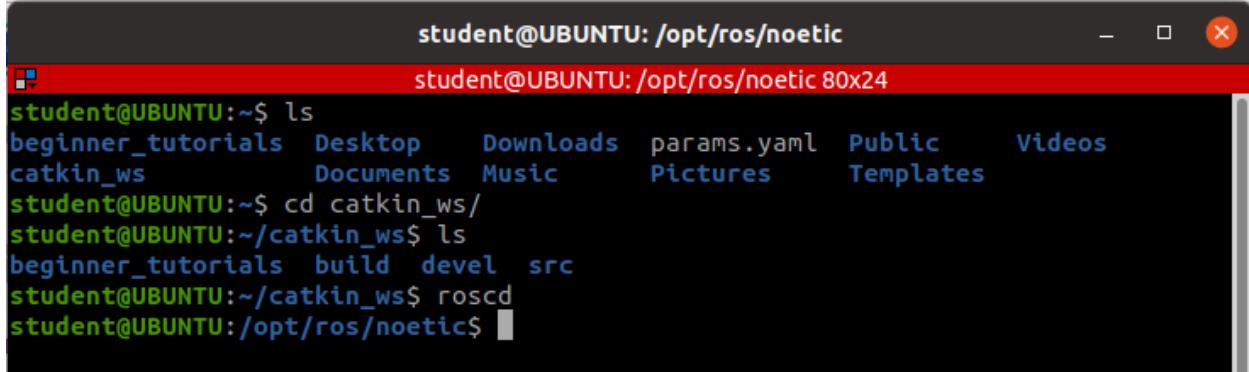
Figure 3. Husky UGV (Clearpath Robotics)

Overview: Husky is a rugged, all-terrain unmanned ground vehicle developed by Clearpath Robotics. It is designed for outdoor and industrial environments, where it can be used for research, mapping, and surveillance. Husky is known for its versatility and ability to carry heavy payloads.

ROS Integration: Husky uses ROS for its navigation, sensor integration, and control systems. The robot can be equipped with a variety of sensors and is commonly used in research projects that require autonomous navigation in challenging terrains.

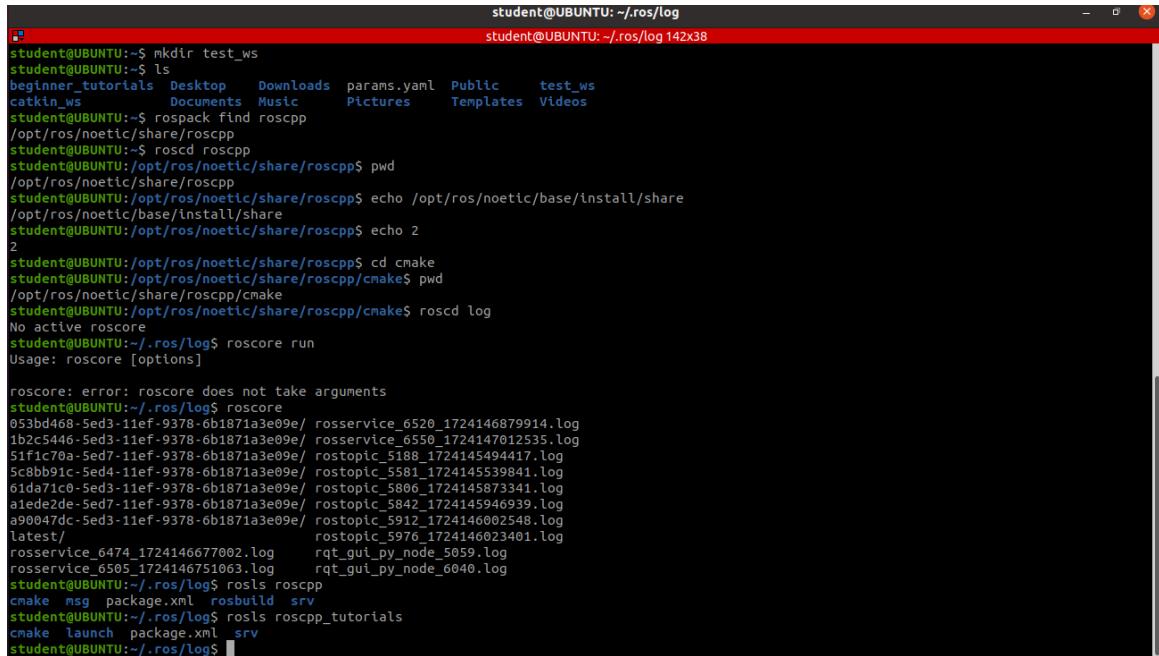
3. Task 0 - ROS Tutorials

In this task, we had to do a 16 step basic tutorial from the official ROS web-site. Figure array below shows completion of all tasks.



```
student@UBUNTU: /opt/ros/noetic
student@UBUNTU: /opt/ros/noetic 80x24
student@UBUNTU:~$ ls
beginner_tutorials Desktop Downloads params.yaml Public Videos
catkin_ws Documents Music Pictures Templates
student@UBUNTU:~$ cd catkin_ws/
student@UBUNTU:~/catkin_ws$ ls
beginner_tutorials build devel src
student@UBUNTU:~/catkin_ws$ roscd
student@UBUNTU:/opt/ros/noetic$
```

Figure 4. Setup ROS Environment (Tutorial 1)



```
student@UBUNTU:~$ mkdir test_ws
student@UBUNTU:~$ ls
beginner_tutorials Desktop Downloads params.yaml Public test_ws
catkin_ws Documents Music Pictures Templates Videos
student@UBUNTU:~$ rospack find roscpp
/opt/ros/noetic/share/roscpp
student@UBUNTU:~$ roscd roscpp
student@UBUNTU:/opt/ros/noetic/share/roscpp$ pwd
/opt/ros/noetic/share/roscpp
student@UBUNTU:/opt/ros/noetic/share/roscpp$ echo /opt/ros/noetic/base/install/share
/opt/ros/noetic/base/install/share
student@UBUNTU:/opt/ros/noetic/share/roscpp$ echo 2
2
student@UBUNTU:/opt/ros/noetic/share/roscpp$ cd cmake
student@UBUNTU:/opt/ros/noetic/share/roscpp$ cmake$ pwd
/opt/ros/noetic/share/roscpp/cmake
student@UBUNTU:/opt/ros/noetic/share/roscpp$ cmake$ roscore log
No active roscore
student@UBUNTU:./ros/log$ roscore run
Usage: roscore [options]

roscore: error: roscore does not take arguments
student@UBUNTU:./ros/log$ roscore
053bd468-5ed3-11ef-9378-6b1871a3e09e/ rosservice_6520_1724146879914.log
1b2c5446-5ed3-11ef-9378-6b1871a3e09e/ rosservice_6550_1724147012535.log
51fc70a-5ed7-11ef-9378-6b1871a3e09e/ rostopic_5188_1724145494417.log
5c8bb91c-5ed4-11ef-9378-6b1871a3e09e/ rostopic_5581_1724145539841.log
61da71c0-5ed3-11ef-9378-6b1871a3e09e/ rostopic_5806_1724145873341.log
a1ede2de-5ed7-11ef-9378-6b1871a3e09e/ rostopic_5842_1724145946939.log
a90047dc-5ed3-11ef-9378-6b1871a3e09e/ rostopic_5912_1724146002548.log
latest/ rostopic_5970_1724146023401.log
rosservice_6474_1724146677002.log rqt_gui_py_node_5059.log
rosservice_6505_1724146751063.log rqt_gui_py_node_6040.log
student@UBUNTU:./ros/log$ roslib roscpp
cmake msg package.xml rosbUILD srV
student@UBUNTU:./ros/log$ roslib roscpp_tutorials
cmake launch package.xml srV
student@UBUNTU:./ros/log$
```



```
student@UBUNTU:/opt/ros/noetic/share/roscpp_tutorials$ roscd turtle
roscd: No such package/stack 'turtle'
student@UBUNTU:/opt/ros/noetic/share/roscpp_tutorials$ roscd turtle
roscd: No such package/stack 'turtle'
student@UBUNTU:/opt/ros/noetic/share/roscpp_tutorials$ roscd turtle
turtle_actionlib/ turtlesim/ turtle_tf/ turtle_tf2/
student@UBUNTU:/opt/ros/noetic/share/roscpp_tutorials$ roscd turtlesim/
student@UBUNTU:/opt/ros/noetic/share/turtlesim$ roslibs
Display all 244 possibilities? (y or n)
student@UBUNTU:/opt/ros/noetic/share/turtlesim$ roslibs
```

Figure 5. Navigating the ROS Filesystem (Tutorial 2)

```

student@UBUNTU:~/catkin_ws$ . ~/catkin_ws/devel/setup.bash
student@UBUNTU:~/catkin_ws$ rospack depends1 beginner_tutorials2
[rospack] Error: no such package beginner_tutorials2
student@UBUNTU:~/catkin_ws$ cd ~
student@UBUNTU:~$ rospack depends1 beginner_tutorials2
[rospack] Error: no such package beginner_tutorials2
student@UBUNTU:~$ rospack depends beginner_tutorials2
[rospack] Error: no such package beginner_tutorials2
student@UBUNTU:~$ rospack
cflags-only-I depends-indent depends-on1 help libs-only-other plugins vcs
cflags-only-other depends-manifests depends-why langs list profile vcs0
depends depends-msgsrv export libs-only-l libs-only-L list-duplicates rosdep
depends1 depends-on fnd list-names rosdep0
student@UBUNTU:~$ rospack depends1 beginner_tutorials2
roscpp
rospy
std_msgs
student@UBUNTU:~$ rospack depends1 beginner_tutorials2
[rospack] Error: no such package beginner_tutorials2
student@UBUNTU:~$ roscd beginner_tutorials2
roscd: No such package/stack 'beginner_tutorials2'
student@UBUNTU:~$ 

student@UBUNTU:~$ catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
usage: catkin_create_pkg [-h] [--meta] [-s [SYS_DEPS [SYS_DEPS ...]]] [-b [BOOST_COMPS [BOOST_COMPS ...]]] [-V PKG_VERSION] [-D DESCRIPTION]
                         [-l LICENSE] [-a AUTHOR] [-m MAINTAINER] [-r ROSDISTRO]
                         name [dependencies [dependencies ...]]
catkin_create_pkg: error: File exists: /home/student/beginner_tutorials/package.xml
student@UBUNTU:~$ catkin_create_pkg beginner_tutorials2 std_msgs rospy roscpp
Created file beginner_tutorials2/package.xml
Created file beginner_tutorials2/CMakeLists.txt
Created folder beginner_tutorials2/include/beginner_tutorials2
Created folder beginner_tutorials2/src
Successfully created files in /home/student/beginner_tutorials2. Please adjust the values in package.xml.
student@UBUNTU:~$ cd catkin_ws/
student@UBUNTU:~/catkin_ws$ catkin_make
Command 'catking_make' not found, did you mean:
  command 'catkin_make' from deb catkin (0.8.0-1ubuntu2)
Try: apt install <deb name>
student@UBUNTU:~/catkin_ws$ catkin make
Base path: /home/student/catkin_ws
Source space: /home/student/catkin_ws/src
Build space: /home/student/catkin_ws/build
Devel space: /home/student/catkin_ws/devel
Install space: /home/student/catkin_ws/install
#####
##### Running command: "make cmake_check_build_system" in "/home/student/catkin_ws/build"
#####
#####

```

```

student@UBUNTU:~/catkin_ws/src/beginner_tutorials$ rospack depends1 rospy
genpy
roscpp
rosgraph
rosgraph_msgs
roslib
std_msgs
student@UBUNTU:~/catkin_ws/src/beginner_tutorials$ rospack depends beginner_tutorials
cpp_common
rostime
roscpp_traits
roscpp_serialization
catkin
genmsg
genpy
message_runtime
gencpp
geneus
gennodejs
genlisp
message_generation
rosbuild
rosconsole
std_msgs
rosgraph_msgs
xmlrpcpp
roscpp
rosgraph
ros_environment
rospack
roslib
rospy

student@UBUNTU:~$ roscl beginner_tutorials/
student@UBUNTU:~/catkin_ws/src/beginner_tutorials$ cat package.xml
<?xml version="1.0"?>
<package format="2">
  <name>beginner_tutorials</name>
  <version>0.0.0</version>
  <description>The beginner_tutorials package</description>

  <!-- One maintainer tag required, multiple allowed, one person per tag -->
  <!-- Example: -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <maintainer email="student@todo.todo">student</maintainer>

  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <license>TODO</license>

  <!-- Url tags are optional, but multiple are allowed, one per tag -->
  <!-- Optional attribute type can be: website, bugtracker, or repository -->
  <!-- Example: -->
  <!-- <url type="website">http://wiki.ros.org/beginner_tutorials</url> -->

  <!-- Author tags are optional, multiple are allowed, one per tag -->
  <!-- Authors do not have to be maintainers, but could be -->
  <!-- Example: -->
  <!-- <author email="jane.doe@example.com">Jane Doe</author> -->

  <!-- The *depend tags are used to specify dependencies -->
  <!-- Dependencies can be catkin packages or system dependencies -->
  <!-- Examples: -->

```

Figure 6. Creating a ROS Package (Tutorial 3)

```

student@UBUNTU: ~/catkin_ws
student@UBUNTU: ~/catkin_ws 142x38
student@UBUNTU:~/catkin_ws/src/beginner_tutorials$ source /opt/ros/noetic/setup.bash
student@UBUNTU:~/catkin_ws/src/beginner_tutorials$ cd ~
student@UBUNTU:~$ cd catkin_ws/
student@UBUNTU:~/catkin_ws$ ls scr
ls: cannot access 'scr': No such file or directory
student@UBUNTU:~/catkin_ws$ ls src
beginner_tutorials CMakeLists.txt coolpack
student@UBUNTU:~/catkin_ws$ catking_make

Command 'catking_make' not found, did you mean:

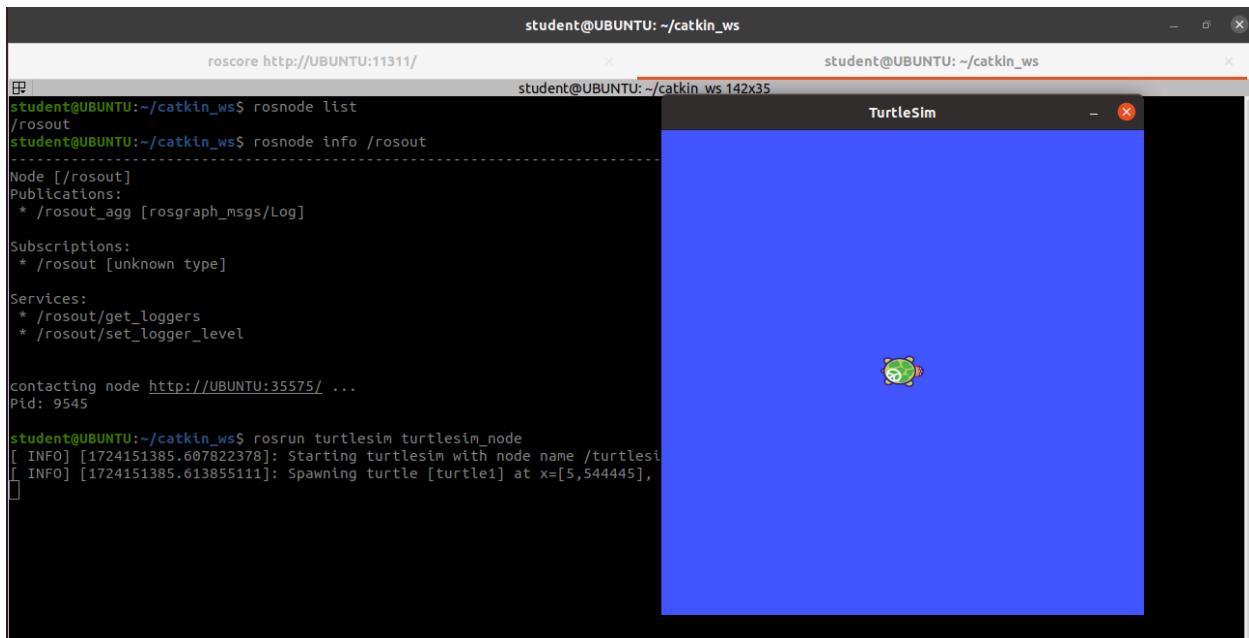
  command 'catkin_make' from deb catkin (0.8.0-1ubuntu2)

Try: apt install <deb name>

student@UBUNTU:~/catkin_ws$ catkin_make
Base path: /home/student/catkin_ws
Source space: /home/student/catkin_ws/src
Build space: /home/student/catkin_ws/build
Devel space: /home/student/catkin_ws/devel
Install space: /home/student/catkin_ws/install
#####
##### Running command: "make cmake_check_build_system" in "/home/student/catkin_ws/build"
#####
#####
##### Running command: "make -j4 -l4" in "/home/student/catkin_ws/build"
#####
student@UBUNTU:~/catkin_ws$ ls
beginner_tutorials build devel src
student@UBUNTU:~/catkin_ws$ 

```

Figure 7. Building a ROS Package (Tutorial 4)



```
roscore http://UBUNTU:11311/
roscore http://UBUNTU:11311/ 142x38
student@UBUNTU:~/catkin_ws$ roscore
... logging to /home/student/.ros/log/a85981cc-5ee2-11ef-ac10-05499d1f1c8e/roslaunch-UBUNTU-9527.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://UBUNTU:35081/
ros_comm version 1.16.0

SUMMARY
=====
PARAMETERS
* /rosdistro: noetic
* /rosversion: 1.16.0

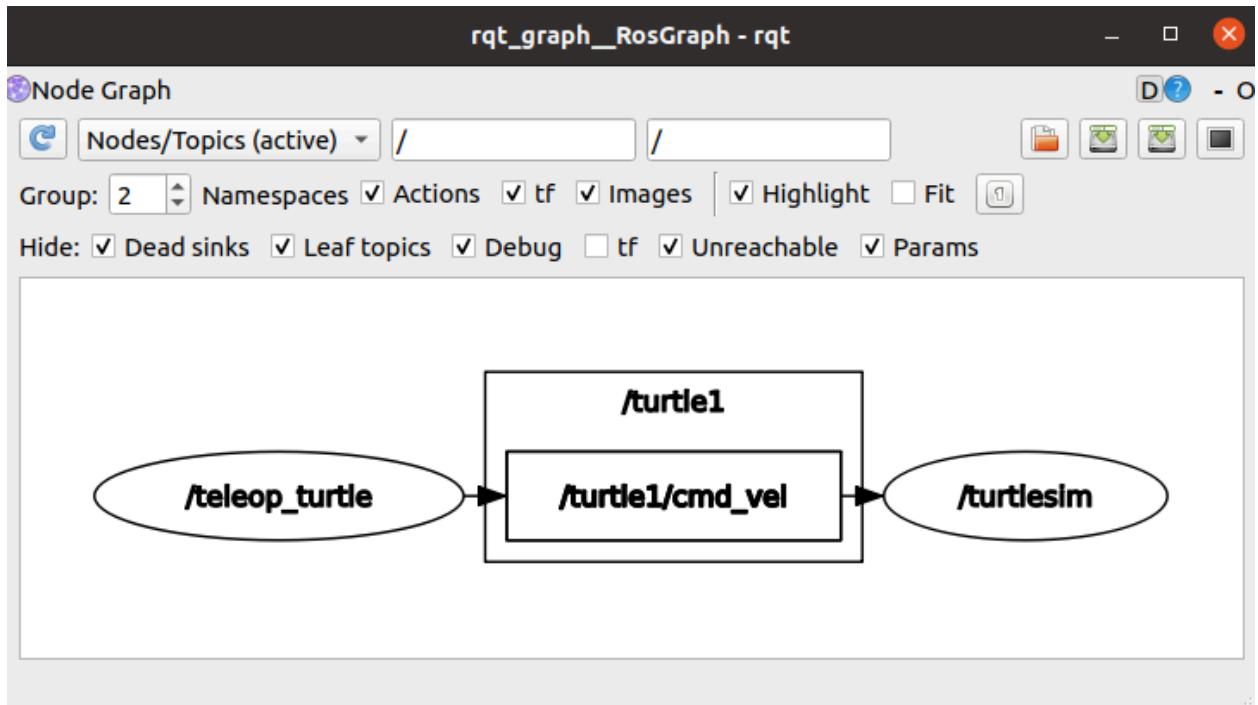
NODES

auto-starting new master
process[master]: started with pid [9535]
ROS_MASTER_URI=http://UBUNTU:11311/

setting /run_id to a85981cc-5ee2-11ef-ac10-05499d1f1c8e
process[rosout-1]: started with pid [9545]
started core service [/rosout]
```

```
student@UBUNTU:~/catkin_ws$ rosnode ping /turtlesim
rosnode: node is [/turtlesim]
pinging /turtlesim with a timeout of 3.0s
xmlrpc reply from http://UBUNTU:38651/ time=0.487089ms
xmlrpc reply from http://UBUNTU:38651/ time=0.882864ms
xmlrpc reply from http://UBUNTU:38651/ time=0.882626ms
xmlrpc reply from http://UBUNTU:38651/ time=0.885725ms
xmlrpc reply from http://UBUNTU:38651/ time=0.850439ms
xmlrpc reply from http://UBUNTU:38651/ time=0.775814ms
xmlrpc reply from http://UBUNTU:38651/ time=0.896454ms
xmlrpc reply from http://UBUNTU:38651/ time=0.903368ms
^Cping average: 0.820547ms
student@UBUNTU:~/catkin_ws$
```

Figure 8. Understanding ROS Nodes (Tutorial 5)



student@UBUNTU: ~/catkin_ws

roscore... studen... studen... student... student... student... x

student@UBUNTU: ~/catkin_ws 70x35

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

student@UBUNTU:~/catkin_ws\$ rostopic

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'

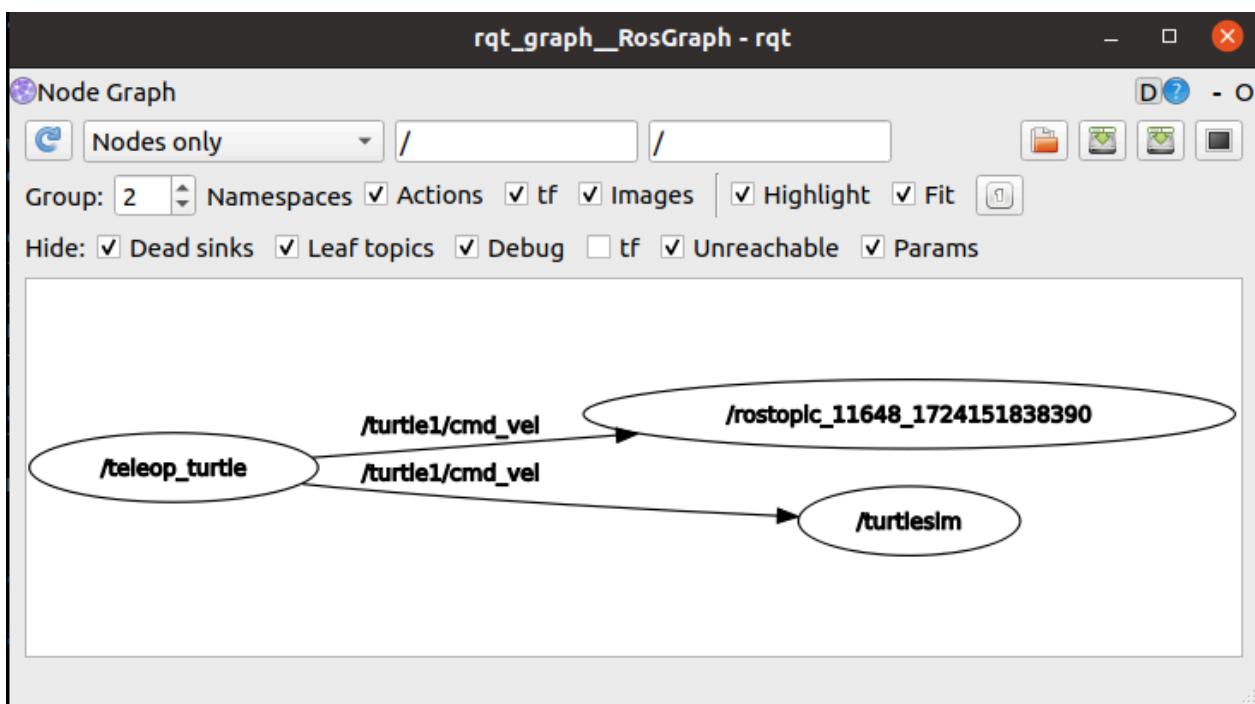
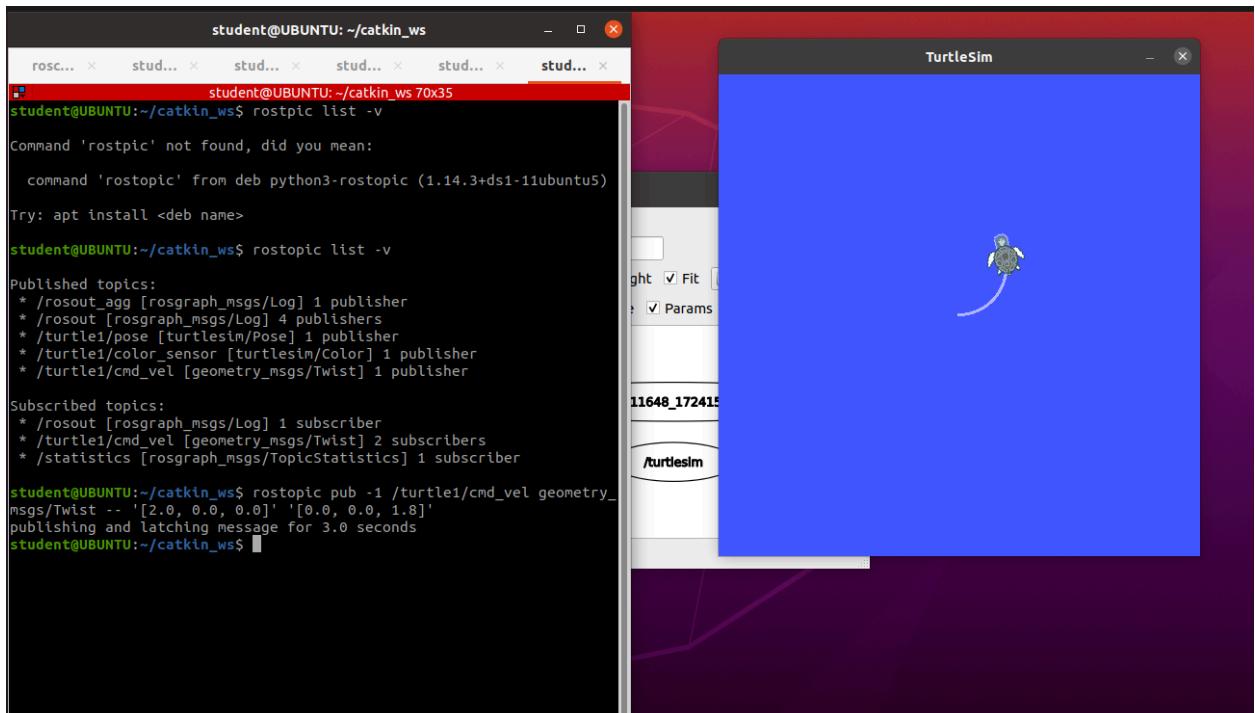
student@UBUNTU:~/catkin_ws\$ rostopic echo /turtle1/

/turtle1/cmd_vel /turtle1/color_sensor /turtle1/pose

student@UBUNTU:~/catkin_ws\$ rostopic echo /turtle1/cmd_vel

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

TurtleSim



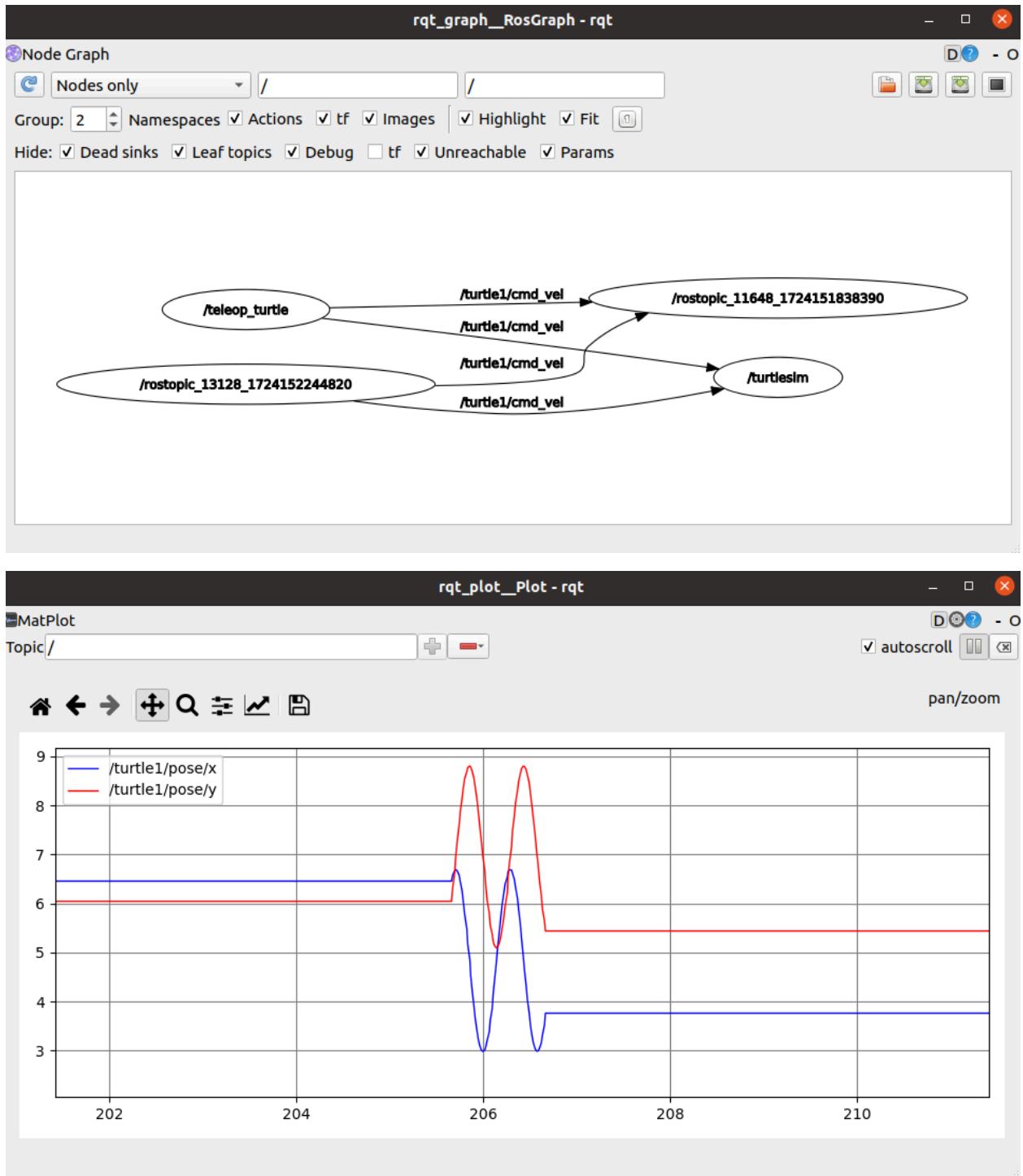
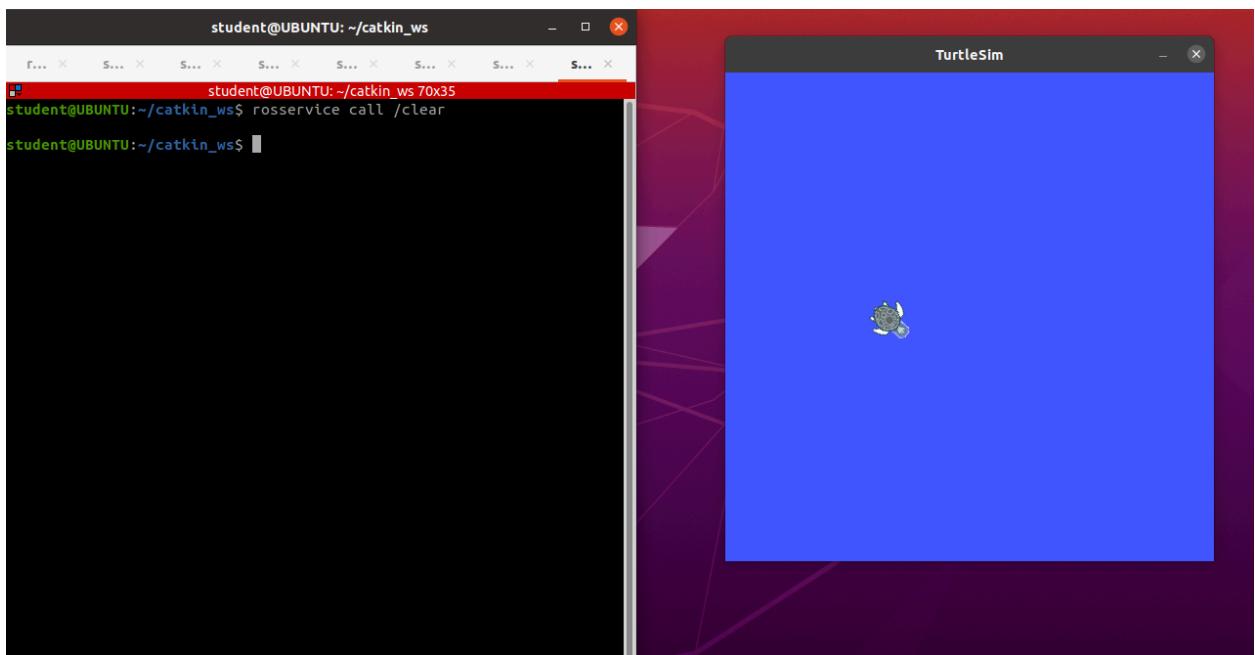
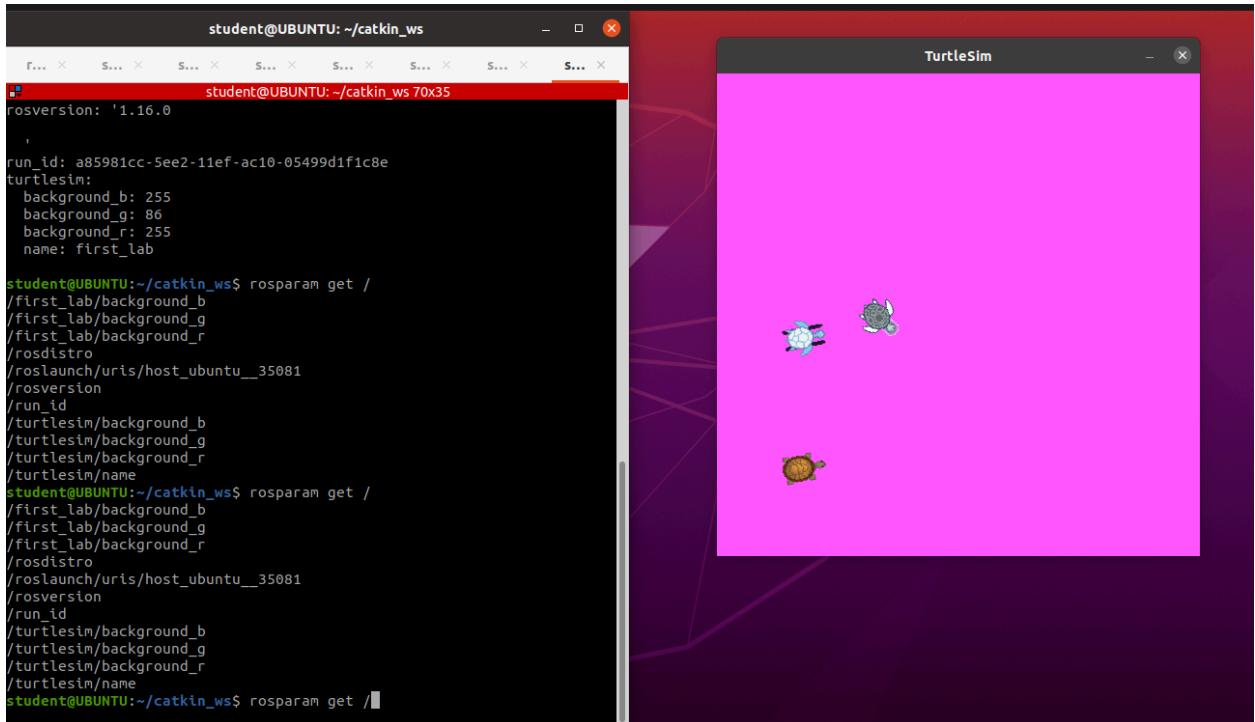


Figure 9. Understanding ROS Topics (Tutorial 6)



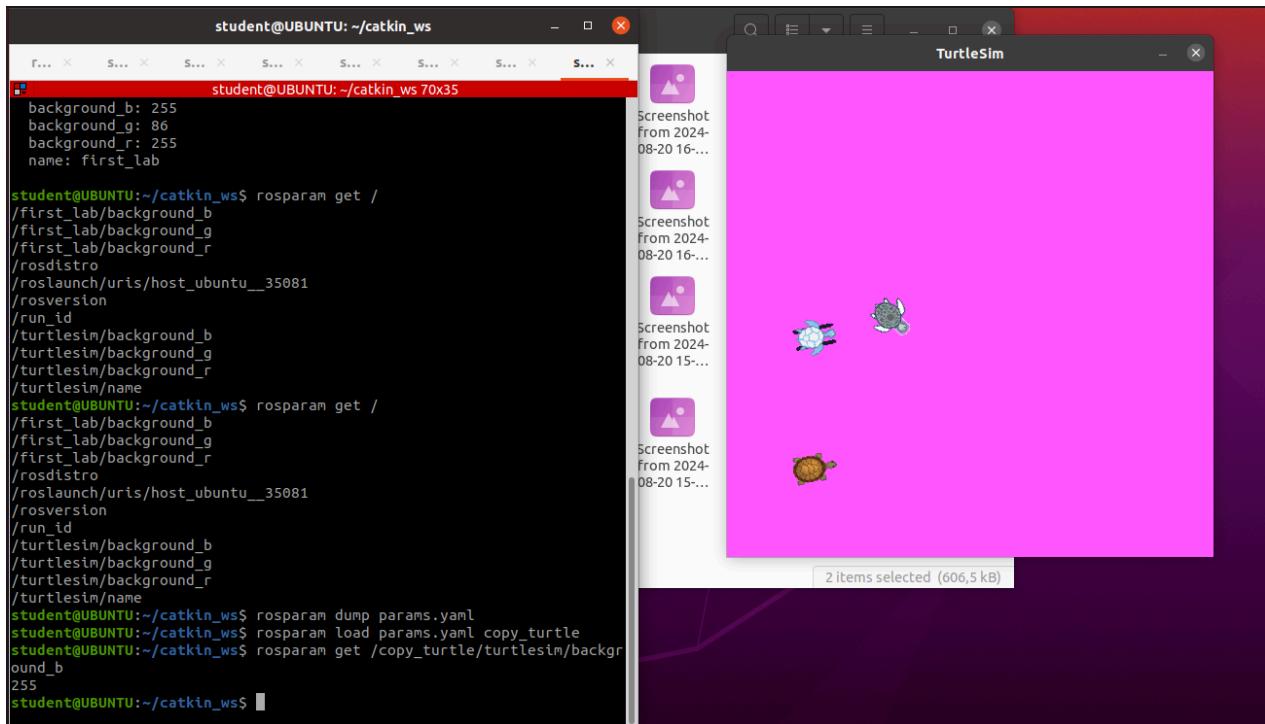
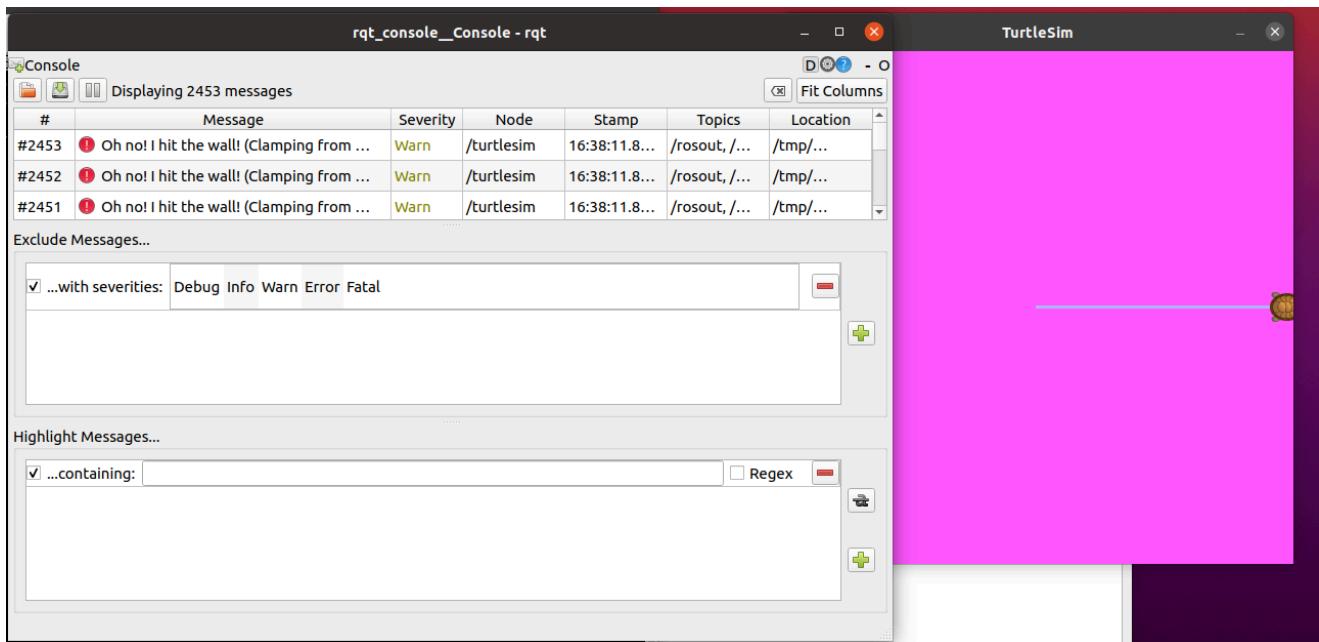
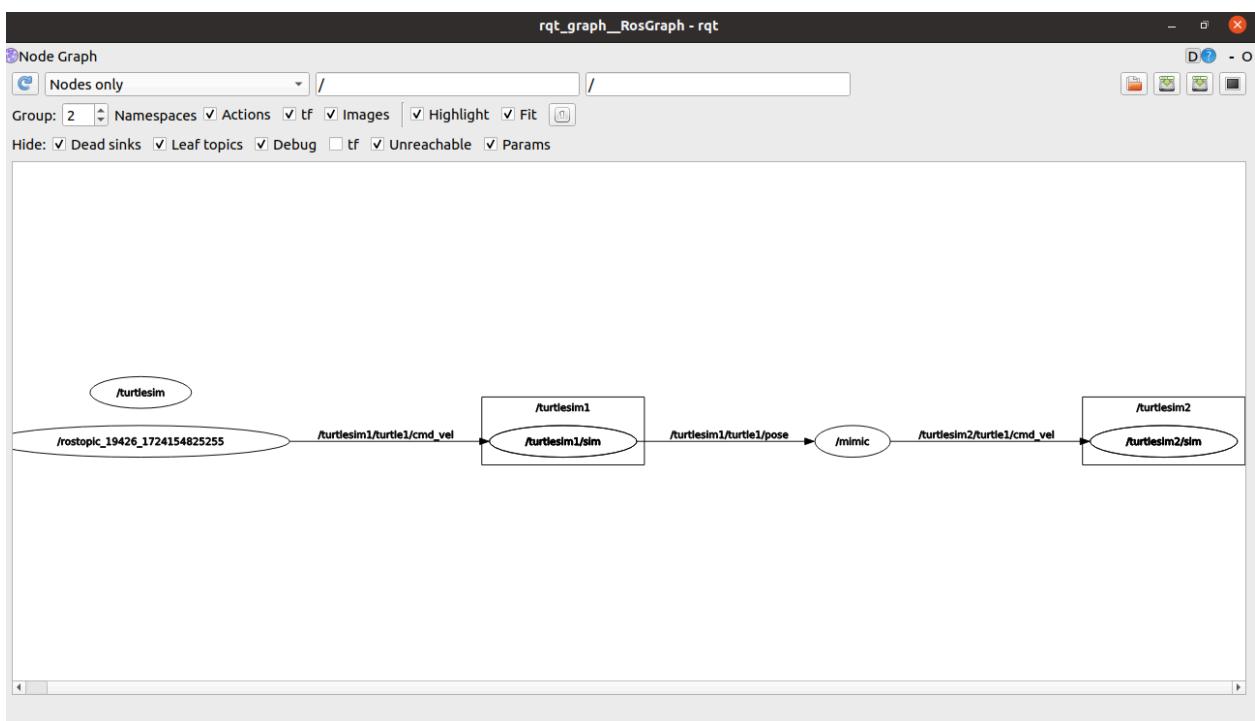
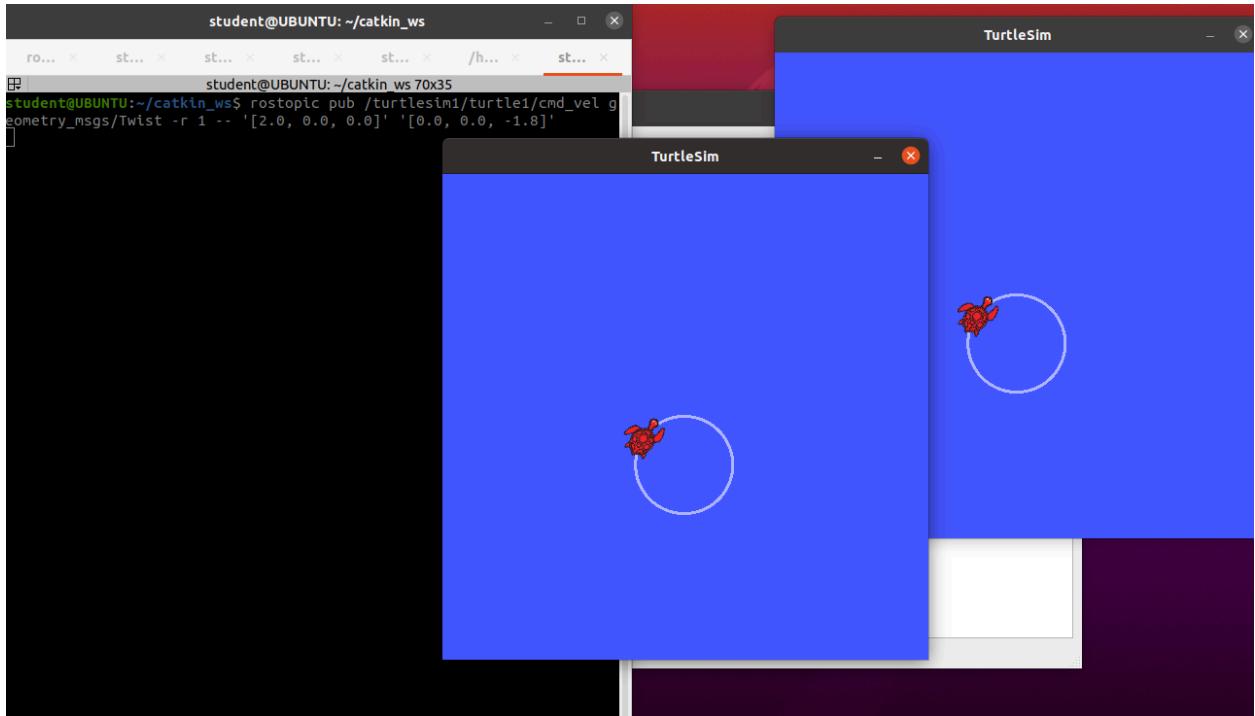


Figure 10. Understanding ROS Services and Parameters (Tutorial 7)





```

/home/abdra/catkin_ws/src/beginner_tutorials/launch/turtlemimic.launch http://localhost:11311
abdra@abdra-Lenovo-Legion-5-15ARH05H:~$ roslaunch
usage: command not found
abdra@abdra-Lenovo-Legion-5-15ARH05H:~$ rosrun beginner_tutorials/
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws/src/beginner_tutorials$ roscd beginner_tutorials/
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws/src/beginner_tutorials$ mkdir launch
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws/src/beginner_tutorials$ cd launch/
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws/src/beginner_tutorials/launch$ touch turtlemimic.launch
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws/src/beginner_tutorials/launch$ nano turtlemimic.launch
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws/src/beginner_tutorials/launch$ roslaunch beginner_tutorials turtlemimic.launch
[ INFO] [rosout]: Logging to /home/abdra/.ros/log/b993b6da-6944-11ef-8890-d75ee561a0de/roslaunch-abdra-Lenovo-Legion-5-15ARH05H-4230.log
[ INFO] [rosout]: Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://abdra-Lenovo-Legion-5-15ARH05H:45669/
SUMMARY
========
PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.16.0
NODES
 /
  mmlc (turtlesim/mmlc)
  /turtlesim1/
    sim (turtlesim/turtlesim_node)
  /turtlesim2/
    sim (turtlesim/turtlesim_node)

auto-starting new master
process[master]: started with pid [4240]
ROS_MASTER_URI=http://localhost:11311

setting /run_id b993b6da-6944-11ef-8890-d75ee561a0de
process[rosout-1]: started with pid [4250]
startInfo: service /rosout
process[turtle1sim1/turtle1]: started with pid [4253]
process[turtle1sim1/sim-1]: started with pid [4254]
process[turtle1sim2/sim-3]: started with pid [4255]
process[mmlc-4]: started with pid [4255]

```

The terminal window shows the execution of ROS commands to start a TurtleSim simulation. It includes the creation of a launch directory, the creation of a launch file named turtlemimic.launch, and the execution of roslaunch. The output shows the parameters set (rosdistro: noetic, rosversion: 1.16.0), the nodes being started (mmlc, /turtlesim1/sim, /turtlesim2/sim), and the starting of the master node and various process IDs.

Two separate windows titled "TurtleSim" are displayed side-by-side. Each window shows a green turtle icon inside a white circle, representing a robot in a simulated environment.

Figure 11. Using rqt_console and roslaunch (Tutorial 8)

```
student@UBUNTU:~/catkin_ws/beginner_tutorials/srv$ rossrv show AddTwoInts.srv
[rospy_tutorials/AddTwoInts]:
int64 a
int64 b
---
int64 sum

student@UBUNTU:~/catkin_ws/beginner_tutorials/srv$
```

```
student@UBUNTU:~/catkin_ws/beginner_tutorials 70x35
eginner_tutorials/msg']]>
student@UBUNTU:~/catkin_ws$ ls
beginner_tutorials build devel params.yaml src
student@UBUNTU:~/catkin_ws$ roscd beginner_tutorials/
student@UBUNTU:~/catkin_ws/src/beginner_tutorials$ ls
CMakeLists.txt include launch package.xml src
student@UBUNTU:~/catkin_ws/src/beginner_tutorials$ cd ..
student@UBUNTU:~/catkin_ws/src$ ls
beginner_tutorials CMakeLists.txt coolpack
student@UBUNTU:~/catkin_ws/src$ cd beginner_tutorials/
student@UBUNTU:~/catkin_ws/src/beginner_tutorials$ ls
CMakeLists.txt include launch package.xml src
student@UBUNTU:~/catkin_ws/src/beginner_tutorials$ mkdir msg
student@UBUNTU:~/catkin_ws/src/beginner_tutorials$ cd msg
student@UBUNTU:~/catkin_ws/src/beginner_tutorials/msg$ echo "int64 num
" > Num.msg
student@UBUNTU:~/catkin_ws/src/beginner_tutorials/msg$ cat Num.msg
int64 num
student@UBUNTU:~/catkin_ws/src/beginner_tutorials/msg$ ls
Num.msg
student@UBUNTU:~/catkin_ws/src/beginner_tutorials/msg$ cd ..
student@UBUNTU:~/catkin_ws/src/beginner_tutorials$ cd ..
student@UBUNTU:~/catkin_ws/src$ cd ..
student@UBUNTU:~/catkin_ws$ ls
beginner_tutorials build devel params.yaml src
student@UBUNTU:~/catkin_ws$ cd beg
bash: cd: beg: No such file or directory
student@UBUNTU:~/catkin_ws$ cd beginner_tutorials/
student@UBUNTU:~/catkin_ws/beginner_tutorials$ ls
CMakeLists.txt include msg package.xml src
student@UBUNTU:~/catkin_ws/beginner_tutorials$ rosmsg show beginner_tutorials/Num
int64 num

student@UBUNTU:~/catkin_ws/beginner_tutorials$
```

```

# Action2.action
# )

## Generate added messages and services with any dependencies
generate_messages(
    DEPENDENCIES
        std_msgs
)

#####
## Declare ROS dynamic reconfigure parameters ##

## DEPENDS: system dependencies of this project that catkin_package()
# INCLUDE_DIRS include
# LIBRARIES beginner_tutorials
CATKIN_DEPENDS roscpp rospy std_msgs message_runtime
# DEPENDS system_lib
)

## Generate messages in the 'msg' folder
add_message_files(
    FILES
        Num.msg
)

## if COMPONENTS list like find_package(catkin REQUIRED
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
    roscpp
    rospy
    std_msgs
    message_generation
)

```

Figure 12. Creating a ROS msg and srv (Tutorial 10)

```
student@UBUNTU: ~/catkin_ws/beginner_tutorials
student@UBUNTU: ~/catkin_ws/beginner_tutorials 70x35
GNU nano 4.8          package.xml
<!-- Examples: -->
<!-- Use depend as a shortcut for packages that are both build and >
<!--   <depend>roscpp</depend> -->
<!-- Note that this is equivalent to the following: -->
<!--   <build_depend>roscpp</build_depend> -->
<!--   <exec_depend>roscpp</exec_depend> -->
<!-- Use build_depend for packages you need at compile time: -->
<build_depend>message_generation</build_depend>
<!-- Use build_export_depend for packages you need in order to build -->
<!--   <build_export_depend>message_generation</build_export_depend>
<!-- Use buildtool_depend for build tool packages: -->
<!--   <buildtool_depend>catkin</buildtool_depend> -->
<!-- Use exec_depend for packages you need at runtime: -->
<exec_depend>message_runtime</exec_depend>
<!-- Use test_depend for packages you need only for testing: -->
<!--   <test_depend>gtest</test_depend> -->
<!-- Use doc_depend for packages you need only for building documentation -->
<!--   <doc_depend>doxygen</doc_depend> -->
<buildtool_depend>catkin</buildtool_depend>
<build_depend>roscpp</build_depend>
<build_depend>rospy</build_depend>
<build_depend>std_msgs</build_depend>
<build_export_depend>roscpp</build_export_depend>
<build_export_depend>rospy</build_export_depend>
<build_export_depend>std_msgs</build_export_depend>
<exec_depend>roscpp</exec_depend>
<exec_depend>rospy</exec_depend>
<exec_depend>std_msgs</exec_depend>

<!-- The export tag contains other unspecified tags -->
```

Figure 13. Writing a Simple Publisher and Subscriber (C++) (Tutorial 11)

```
student@UBUNTU:~/catkin_ws/beginner_tutorials/scripts$ ls
listener.py  talker.py
```

Figure 14. Writing a Simple Publisher and Subscriber (Python) (Tutorial 12)

```
-- BUILD_SHARED_LIBS is on
-- ~~ traversing 1 packages in topological order:
--   - beginner_tutorials
-- 
-- +++ processing catkin package: 'beginner_tutorials'
-- ==> add_subdirectory(beginner_tutorials)
-- Using these message generators: gencpp;geneus;genlisp;gennodejs;genpy
-- beginner_tutorials: 1 messages, 0 services
-- Configuring done
-- Generating done
-- Build files have been written to: /home/abdra/catkin_ws/build
#####
##### Running command: "make -j12 -l12" in "/home/abdra/catkin_ws/build"
#####
Scanning dependencies of target _beginner_tutorials_generate_messages_check_deps_Num
Scanning dependencies of target std_msgs_generate_messages_cpp
Scanning dependencies of target std_msgs_generate_messages_nodejs
Scanning dependencies of target std_msgs_generate_messages_eus
Scanning dependencies of target std_msgs_generate_messages_py
Scanning dependencies of target std_msgs_generate_messages_lisp
[ 0%] Built target std_msgs_generate_messages_eus
[ 0%] Built target std_msgs_generate_messages_nodejs
[ 0%] Built target std_msgs_generate_messages_lisp
[ 0%] Built target std_msgs_generate_messages_py
[ 0%] Built target std_msgs_generate_messages_cpp
[ 0%] Built target _beginner_tutorials_generate_messages_check_deps_Num
Scanning dependencies of target beginner_tutorials_generate_messages_cpp
Scanning dependencies of target beginner_tutorials_generate_messages_lisp
Scanning dependencies of target beginner_tutorials_generate_messages_py
Scanning dependencies of target beginner_tutorials_generate_messages_eus
Scanning dependencies of target beginner_tutorials_generate_messages_nodejs
[ 27%] Generating Lisp code from beginner_tutorials/Num.msg
[ 27%] Generating C++ code from beginner_tutorials/Num.msg
[ 27%] Generating Python from MSG beginner_tutorials/Num
[ 36%] Generating Javascript code from beginner_tutorials/Num.msg
[ 45%] Generating EusLisp manifest code for beginner_tutorials
[ 54%] Generating EusLisp code from beginner_tutorials/Num.msg
[ 54%] Built target beginner_tutorials_generate_messages_lisp
[ 54%] Built target beginner_tutorials_generate_messages_nodejs
[ 54%] Built target beginner_tutorials_generate_messages_cpp
Scanning dependencies of target talker
Scanning dependencies of target listener
[ 63%] Building CXX object beginner_tutorials/CMakeFiles/listener.dir/src/listener.cpp.o
[ 72%] Building CXX object beginner_tutorials/CMakeFiles/talker.dir/src/talker.cpp.o
[ 81%] Generating Python msg __init__.py for beginner_tutorials
[ 81%] Built target beginner_tutorials_generate_messages_eus
[ 81%] Built target beginner_tutorials_generate_messages_py
Scanning dependencies of target beginner_tutorials_generate_messages
[ 81%] Built target beginner_tutorials_generate_messages
[ 90%] Linking CXX executable /home/abdra/catkin_ws/devel/lib/beginner_tutorials/listener
[100%] Linking CXX executable /home/abdra/catkin_ws/devel/lib/beginner_tutorials/talker
[100%] Built target listener
[100%] Built target talker
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$
```

```
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$ rosrun beginner_tutorials listener
[ INFO] [1724749673.772840889]: I heard: [hello world 3]
[ INFO] [1724749673.872812628]: I heard: [hello world 4]
[ INFO] [1724749673.972785973]: I heard: [hello world 5]
[ INFO] [1724749674.073133815]: I heard: [hello world 6]
[ INFO] [1724749674.172642059]: I heard: [hello world 7]
[ INFO] [1724749674.272461807]: I heard: [hello world 8]
[ INFO] [1724749674.372821171]: I heard: [hello world 9]
[ INFO] [1724749674.472756930]: I heard: [hello world 10]
[ INFO] [1724749674.572568227]: I heard: [hello world 11]
[ INFO] [1724749674.672555951]: I heard: [hello world 12]
[ INFO] [1724749674.772737351]: I heard: [hello world 13]
[ INFO] [1724749674.872756295]: I heard: [hello world 14]
[ INFO] [1724749674.972540423]: I heard: [hello world 15]
[ INFO] [1724749675.072641496]: I heard: [hello world 16]
[ INFO] [1724749675.172576477]: I heard: [hello world 17]
[ INFO] [1724749675.272785038]: I heard: [hello world 18]
[ INFO] [1724749675.372776383]: I heard: [hello world 19]
[ INFO] [1724749675.472829750]: I heard: [hello world 20]
[ INFO] [1724749675.572852455]: I heard: [hello world 21]
[ INFO] [1724749675.672637342]: I heard: [hello world 22]
[ INFO] [1724749675.772833400]: I heard: [hello world 23]
[ INFO] [1724749675.872821113]: I heard: [hello world 24]
[ INFO] [1724749675.972829361]: I heard: [hello world 25]
[ INFO] [1724749676.072825169]: I heard: [hello world 26]
[ INFO] [1724749676.172533357]: I heard: [hello world 27]
[ INFO] [1724749676.272623033]: I heard: [hello world 28]
[ INFO] [1724749676.372610178]: I heard: [hello world 29]
```

```
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$ rosrun beginner_tutorials talker
[ INFO] [1724749673.472046197]: hello world 0
[ INFO] [1724749673.572640525]: hello world 1
[ INFO] [1724749673.672140328]: hello world 2
[ INFO] [1724749673.772229125]: hello world 3
[ INFO] [1724749673.872338317]: hello world 4
[ INFO] [1724749673.972320812]: hello world 5
[ INFO] [1724749674.072528337]: hello world 6
[ INFO] [1724749674.172122140]: hello world 7
[ INFO] [1724749674.272078015]: hello world 8
[ INFO] [1724749674.372320738]: hello world 9
[ INFO] [1724749674.472310347]: hello world 10
[ INFO] [1724749674.572131493]: hello world 11
[ INFO] [1724749674.672124524]: hello world 12
[ INFO] [1724749674.772324014]: hello world 13
[ INFO] [1724749674.872326894]: hello world 14
[ INFO] [1724749674.972129950]: hello world 15
[ INFO] [1724749675.072162925]: hello world 16
[ INFO] [1724749675.172134365]: hello world 17
[ INFO] [1724749675.272356126]: hello world 18
[ INFO] [1724749675.372337763]: hello world 19
[ INFO] [1724749675.472342518]: hello world 20
[ INFO] [1724749675.572341476]: hello world 21
[ INFO] [1724749675.672146129]: hello world 22
[ INFO] [1724749675.772353711]: hello world 23
[ INFO] [1724749675.872343799]: hello world 24
[ INFO] [1724749675.972346879]: hello world 25
[ INFO] [1724749676.072352954]: hello world 26
[ INFO] [1724749676.172082445]: hello world 27
[ INFO] [1724749676.272139014]: hello world 28
[ INFO] [1724749676.372170022]: hello world 29
^C[ INFO] [1724749676.472151439]: hello world 30
```

Figure 15. Examining the Simple Publisher and Subscriber (Tutorial 13)

```
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws/src/beginner_tutorials$ cd srv
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws/src/beginner_tutorials/srv$ ls
AddTwoInts.srv
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws/src/beginner_tutorials/srv$ cd ..
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws/src/beginner_tutorials$ cd src
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws/src/beginner_tutorials/src$ ls
add_two_ints_client.cpp  add_two_ints_server.cpp  listener.cpp  talker.cpp
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws/src/beginner_tutorials/src$
```

Figure 16. Writing a Simple Service and Client (C++) (Tutorial 14)

```
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$ rosrun beginner_tutorials add_two_ints_client 1 3
[ INFO] [1724752628.711693895]: Sum: 4
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$
```

```
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$ rosrun beginner_tutorials add_two_ints_server
[ INFO] [1724752612.493347458]: Ready to add two ints.
[ INFO] [1724752628.711302299]: request: x=1, y=3
[ INFO] [1724752628.711384361]: sending back response: [4]
```

Figure 17. Examining the Simple Service and Client (Tutorial 16)

Finally, we have implemented a C++ code that sends the ID of Abdirakhman to the listener in 1 Hz and 100 Hz.

```
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$ rostopic list
/Abdirakhman
/rosout
/rosout_agg
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$
```

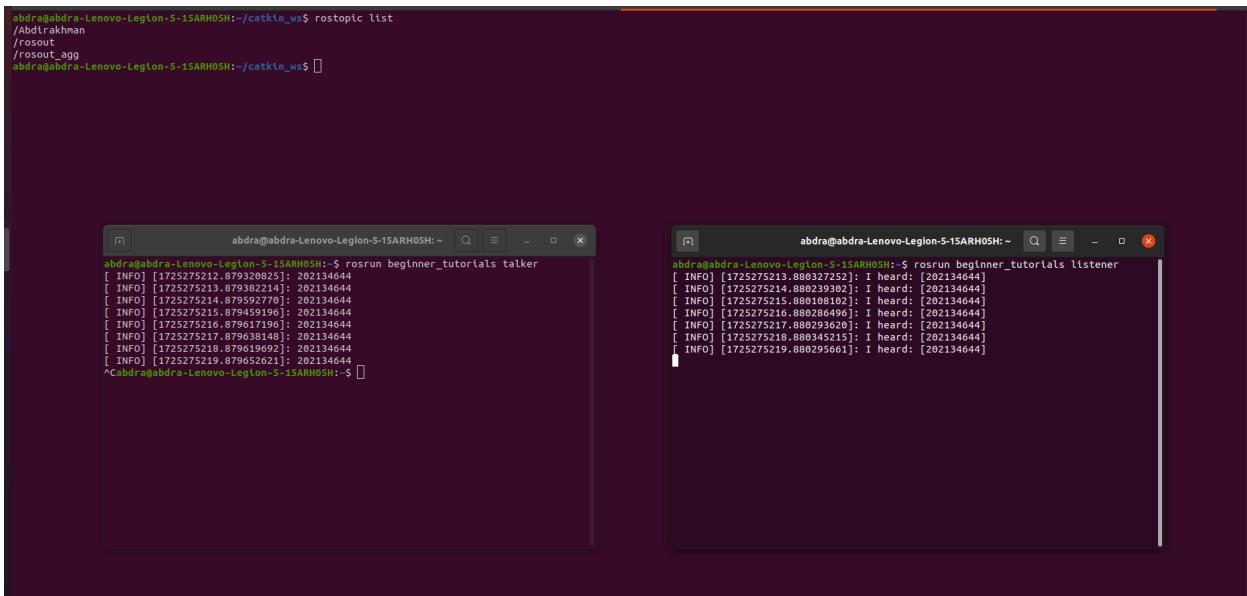


Figure 18. Sending ID in 1 Hz (Task 0)

The image shows two terminal windows side-by-side. Both windows have a dark background and white text. The left window has a title bar 'abdra@abdra-Lenovo-Legion-5-15ARH05H: ~'. The right window has a title bar 'abdra@abdra-Lenovo-Legion-5-15ARH05H: ~'. Both windows show command-line logs for ROS nodes.

```

abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$ rostopic list
/Abdrakhman
/rosout
/rosout_agg
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$ 

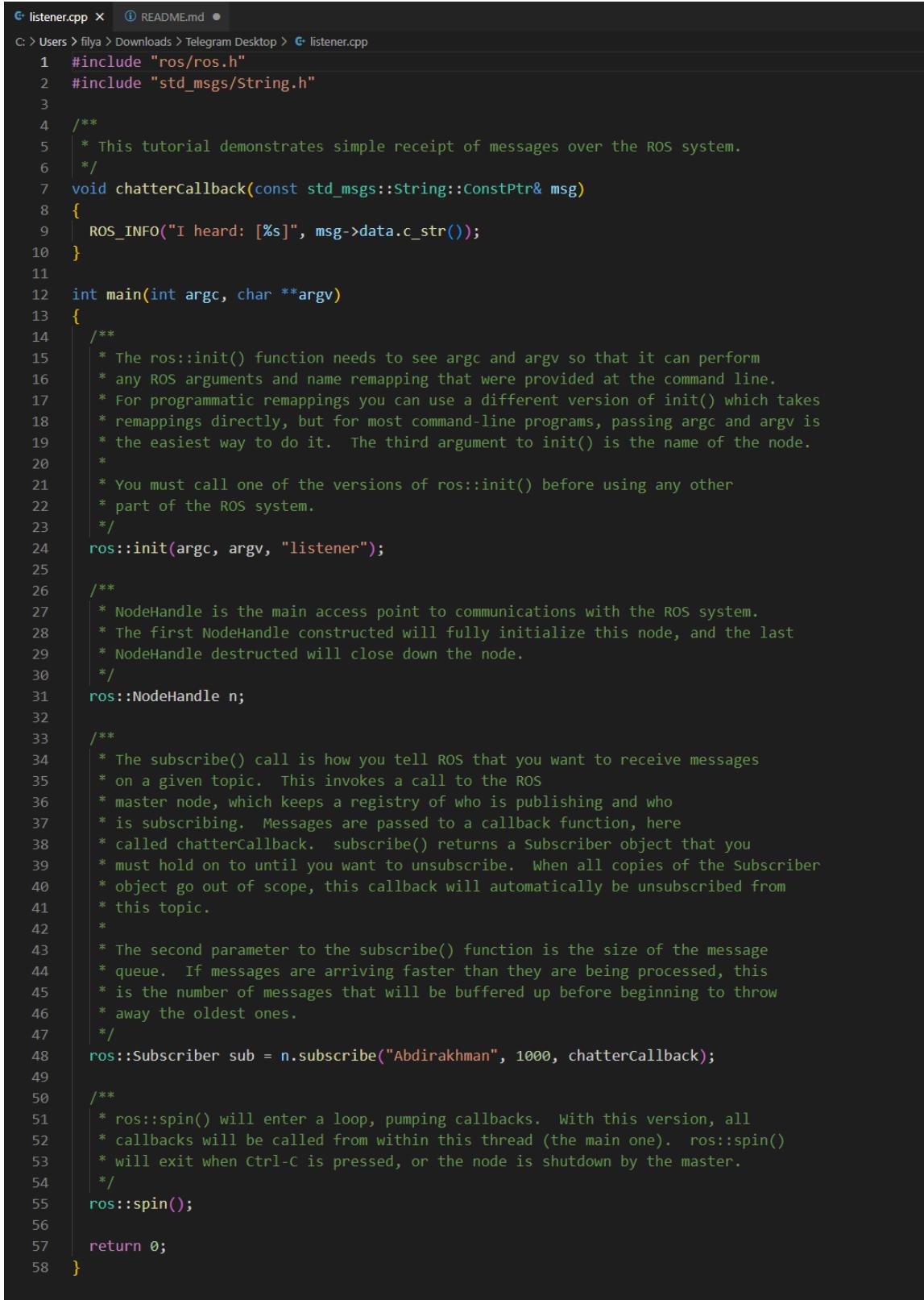
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$ rossrv beginner_tutorials talker
[ INFO] [1725275337.454659]: I heard: [202134644]
[ INFO] [1725275337.455209789]: I heard: [202134644]
[ INFO] [1725275337.464909695]: I heard: [202134644]
[ INFO] [1725275337.475146971]: I heard: [202134644]
[ INFO] [1725275337.484905056]: I heard: [202134644]
[ INFO] [1725275337.494902841]: I heard: [202134644]
[ INFO] [1725275337.504892594]: I heard: [202134644]
INFO] [1725275337.515149293]: I heard: [202134644]
[ INFO] [1725275337.525209431]: I heard: [202134644]
[ INFO] [1725275337.534924433]: I heard: [202134644]
[ INFO] [1725275337.544895049]: I heard: [202134644]
[ INFO] [1725275337.555209082]: I heard: [202134644]
[ INFO] [1725275337.565922482]: I heard: [202134644]
[ INFO] [1725275337.575189078]: I heard: [202134644]
[ INFO] [1725275337.585356474]: I heard: [202134644]
[ INFO] [1725275337.595176468]: I heard: [202134644]
[ INFO] [1725275337.604869550]: I heard: [202134644]
[ INFO] [1725275337.615201799]: I heard: [202134644]
[ INFO] [1725275337.624912321]: I heard: [202134644]
[ INFO] [1725275337.635158817]: I heard: [202134644]
[ INFO] [1725275337.644988630]: I heard: [202134644]
[ INFO] [1725275337.655195594]: I heard: [202134644]
[ INFO] [1725275337.665103631]: I heard: [202134644]

abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$ rossrv beginner_tutorials listener
[ INFO] [1725275337.686957564]: I heard: [202134644]
[ INFO] [1725275337.705857644]: I heard: [202134644]
[ INFO] [1725275337.705514527]: I heard: [202134644]
[ INFO] [1725275337.715867602]: I heard: [202134644]
[ INFO] [1725275337.725760693]: I heard: [202134644]
[ INFO] [1725275337.735722788]: I heard: [202134644]
[ INFO] [1725275337.745854672]: I heard: [202134644]
[ INFO] [1725275337.755954672]: I heard: [202134644]
[ INFO] [1725275337.765474932]: I heard: [202134644]
[ INFO] [1725275337.775920734]: I heard: [202134644]
[ INFO] [1725275337.785575730]: I heard: [202134644]
[ INFO] [1725275337.795620246]: I heard: [202134644]
[ INFO] [1725275337.805465756]: I heard: [202134644]
[ INFO] [1725275337.815869410]: I heard: [202134644]
[ INFO] [1725275337.825934823]: I heard: [202134644]
[ INFO] [1725275337.835934823]: I heard: [202134644]
[ INFO] [1725275337.845709809]: I heard: [202134644]
[ INFO] [1725275337.855457278]: I heard: [202134644]
[ INFO] [1725275337.865698395]: I heard: [202134644]
[ INFO] [1725275337.875963539]: I heard: [202134644]
[ INFO] [1725275337.885445395]: I heard: [202134644]
[ INFO] [1725275337.895750838]: I heard: [202134644]
[ INFO] [1725275337.905651017]: I heard: [202134644]

```

Figure 19. Sending ID in 100 Hz (Task 0)

Following the first 16 ROS beginner tutorials, we have learned several foundational skills. We start by installing and setting up the ROS environment, navigating the ROS filesystem, and creating and building ROS packages. We then delve into the core concepts of ROS, such as nodes, topics, services, and parameters, by writing simple publisher and subscriber nodes in both C++ and Python. Additionally, we create custom message types, work with ROS services and clients, and explore essential debugging tools to troubleshoot and optimize our ROS applications.



The screenshot shows a code editor window with the file "listener.cpp" open. The code is written in C++ and uses ROS message types from "std_msgs/String.h". The code implements a ROS node that listens to a topic named "Abdirakhman" and prints the received string to the console. The code includes comments explaining the purpose of various ROS functions like `ros::init`, `ros::NodeHandle`, and `ros::Subscriber`.

```
1 #include <ros/ros.h>
2 #include <std_msgs/String.h>
3
4 /**
5  * This tutorial demonstrates simple receipt of messages over the ROS system.
6  */
7 void chatterCallback(const std_msgs::String::ConstPtr& msg)
8 {
9     ROS_INFO("I heard: [%s]", msg->data.c_str());
10 }
11
12 int main(int argc, char **argv)
13 {
14     /**
15      * The ros::init() function needs to see argc and argv so that it can perform
16      * any ROS arguments and name remapping that were provided at the command line.
17      * For programmatic remappings you can use a different version of init() which takes
18      * remappings directly, but for most command-line programs, passing argc and argv is
19      * the easiest way to do it. The third argument to init() is the name of the node.
20      *
21      * You must call one of the versions of ros::init() before using any other
22      * part of the ROS system.
23      */
24     ros::init(argc, argv, "listener");
25
26     /**
27      * NodeHandle is the main access point to communications with the ROS system.
28      * The first NodeHandle constructed will fully initialize this node, and the last
29      * NodeHandle destructed will close down the node.
30      */
31     ros::NodeHandle n;
32
33     /**
34      * The subscribe() call is how you tell ROS that you want to receive messages
35      * on a given topic. This invokes a call to the ROS
36      * master node, which keeps a registry of who is publishing and who
37      * is subscribing. Messages are passed to a callback function, here
38      * called chatterCallback. subscribe() returns a Subscriber object that you
39      * must hold on to until you want to unsubscribe. When all copies of the Subscriber
40      * object go out of scope, this callback will automatically be unsubscribed from
41      * this topic.
42      *
43      * The second parameter to the subscribe() function is the size of the message
44      * queue. If messages are arriving faster than they are being processed, this
45      * is the number of messages that will be buffered up before beginning to throw
46      * away the oldest ones.
47      */
48     ros::Subscriber sub = n.subscribe("Abdirakhman", 1000, chatterCallback);
49
50     /**
51      * ros::spin() will enter a loop, pumping callbacks. With this version, all
52      * callbacks will be called from within this thread (the main one). ros::spin()
53      * will exit when Ctrl-C is pressed, or the node is shutdown by the master.
54      */
55     ros::spin();
56
57     return 0;
58 }
```

Figure 20. Code for sending ID in specified frequency (Task 0)

4. Task 1

In this task, we had to use two cpp files – publisher and subscriber and add them to the CMake list in order to be able to execute it. Figure below shows the result of execution of both files.

```
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$ rosrun my_package talker2
[ INFO] [1724753876.976206597]: hello world 0
[ INFO] [1724753877.076543686]: hello world 1
[ INFO] [1724753877.176293486]: hello world 2
[ INFO] [1724753877.276285772]: hello world 3
[ INFO] [1724753877.376269328]: hello world 4
[ INFO] [1724753877.476498652]: hello world 5
[ INFO] [1724753877.576348324]: hello world 6
[ INFO] [1724753877.676285436]: hello world 7
[ INFO] [1724753877.776621058]: hello world 8
[ INFO] [1724753877.876599516]: hello world 9
[ INFO] [1724753877.976295189]: hello world 10
[ INFO] [1724753878.076524723]: hello world 11
[ INFO] [1724753878.176527067]: hello world 12
[ INFO] [1724753878.276596107]: hello world 13
[ INFO] [1724753878.376328447]: hello world 14
[ INFO] [1724753878.476516147]: hello world 15
[ INFO] [1724753878.576256378]: hello world 16
[ INFO] [1724753878.676260747]: hello world 17
^C[ INFO] [1724753878.776405984]: hello world 18
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$
```

```
abdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$ rosrun my_package listener2
[ INFO] [1724753877.277160035]: I heard: [hello world 3]
[ INFO] [1724753877.376883854]: I heard: [hello world 4]
[ INFO] [1724753877.477453442]: I heard: [hello world 5]
[ INFO] [1724753877.577028152]: I heard: [hello world 6]
[ INFO] [1724753877.676842065]: I heard: [hello world 7]
[ INFO] [1724753877.777297045]: I heard: [hello world 8]
[ INFO] [1724753877.877255458]: I heard: [hello world 9]
[ INFO] [1724753877.976889323]: I heard: [hello world 10]
[ INFO] [1724753878.077161110]: I heard: [hello world 11]
[ INFO] [1724753878.177157726]: I heard: [hello world 12]
[ INFO] [1724753878.277301566]: I heard: [hello world 13]
[ INFO] [1724753878.377017005]: I heard: [hello world 14]
[ INFO] [1724753878.477453826]: I heard: [hello world 15]
[ INFO] [1724753878.576730526]: I heard: [hello world 16]
[ INFO] [1724753878.676863890]: I heard: [hello world 17]
```

Figure 21. Running publisher and subscriber (Task 1)

5. Task 2

After the success with task 1, it was time to use the listener from the turtlesim and get the position (x, y) of the turtle in the virtual environment. To make it possible, we had to run the roscore, turtlesim node, and listener. As could be seen below, we were successfully receiving turtlesim's positions.

```
^Cabdra@abdra-Lenovo-Legion-5-15ARH05H:~/catkin_ws$ rosrun turtlebot_controller turtle_listener
[ INFO] [1724755787.971361131]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755787.987667728]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.003833667]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.020599887]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.035716764]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.051993474]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.067480713]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.083687514]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.100003684]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.115596940]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.131964582]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.147554765]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.163857945]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.180131442]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.195335340]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.211759972]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.228239917]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.243800488]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.260326457]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.275605154]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.291993119]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.308079166]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.323809798]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.339886766]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.356578117]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.371788998]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.388482305]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
[ INFO] [1724755788.403454054]: Turtle subscriber@[5.544445, 5.544445, 0.000000]
```

Figure 22. Turtle Listener Output (Task 2)

6. Task 3

After the success of task 2, we felt a need to not just get the position of the turtle, but also to be able to control it from our own code. For this reason, we have written a turtle subscriber to send the desired velocities to the turtle. Result was good – the turtle was moving in a circle and the listener reported back its position.

```

1   #include "ros/ros.h"
2   #include "turtlesim/Pose.h"
3   #include "geometry_msgs/Twist.h"
4   ros::Publisher pub;
5   //pub = nh.advertise<geometry_msgs::Twist>("turtle1/cmd_vel", 1);
6
7   void turtleCallback(const turtlesim::Pose::ConstPtr& msg)
8   {
9       ROS_INFO("Turtle subscriber@[%f, %f, %f]", 
10      msg->x, msg->y, msg->theta);
11      geometry_msgs::Twist my_vel;
12      my_vel.linear.x = 1.0;
13      my_vel.angular.z = 1.0;
14      pub.publish(my_vel);
15  }
16
17
18
19  int main (int argc, char **argv)
20  {
21      // Initialize the node, setup the NodeHandle for handling the communication with the ROS
22      //system
23      ros::init(argc, argv, "turtlebot_subscriber");
24      ros::NodeHandle nh;
25      // Define the subscriber to turtle's position
26      ros::Subscriber sub = nh.subscribe("turtle1/pose", 1,turtleCallback);
27      pub = nh.advertise<geometry_msgs::Twist>("turtle1/cmd_vel", 1);
28      ros::spin();
29      return 0;
30  }

```

Figure 23. Turtle Subscriber modified code (Task 3)

The terminal window shows ROS log output from the node. The log entries are timestamped INFO messages from the turtlebot_subscriber node, detailing the turtle's pose (x, y, theta) over time. The turtle trajectory is visualized in a separate TurtleSim window, which shows a green turtle moving in a circular path on a blue background.

```

abdra@abdra-Lenovo-Legion-5-15ARH05H: ~ /catkin_ws
roscore http://abdra-Lenovo-Legion-5-15ARH05H: 11311 / catkin_ws
abdra@abdra-Lenovo-Legion-5-15ARH05H: ~ /catkin_ws/src/turtlebot_con... abdra@abdra-Lenovo-Legion-5-15ARH05H: ~ /catkin_ws
[ INFO] [1724756562.212036798]: Turtle subscriber@[4.554266, 6.732438, -1.767926]
[ INFO] [1724756562.228397259]: Turtle subscriber@[4.551383, 6.716691, -1.751927]
[ INFO] [1724756562.244165575]: Turtle subscriber@[4.548754, 6.708909, -1.735927]
[ INFO] [1724756562.259699346]: Turtle subscriber@[4.546376, 6.685087, -1.719926]
[ INFO] [1724756562.276145876]: Turtle subscriber@[4.544252, 6.669229, -1.703926]
[ INFO] [1724756562.292201384]: Turtle subscriber@[4.542383, 6.653338, -1.687927]
[ INFO] [1724756562.308301151]: Turtle subscriber@[4.540482, 6.642338, -1.671926]
[ INFO] [1724756562.324526692]: Turtle subscriber@[4.538487, 6.631478, -1.655926]
[ INFO] [1724756562.339761738]: Turtle subscriber@[4.538302, 6.605516, -1.639927]
[ INFO] [1724756562.355899663]: Turtle subscriber@[4.537452, 6.589539, -1.623927]
[ INFO] [1724756562.371608618]: Turtle subscriber@[4.536085, 6.573550, -1.607926]
[ INFO] [1724756562.388208229]: Turtle subscriber@[4.536522, 6.557553, -1.591926]
[ INFO] [1724756562.406995978]: Turtle subscriber@[4.536438, 6.541553, -1.575927]
[ INFO] [1724756562.425700048]: Turtle subscriber@[4.536348, 6.525553, -1.559927]
[ INFO] [1724756562.443509532]: Turtle subscriber@[4.537042, 6.508956, -1.543926]
[ INFO] [1724756562.451810878]: Turtle subscriber@[4.537727, 6.493575, -1.527926]
[ INFO] [1724756562.4674154029]: Turtle subscriber@[4.538666, 6.477662, -1.511927]
[ INFO] [1724756562.484099979]: Turtle subscriber@[4.539866, 6.461648, -1.495926]
[ INFO] [1724756562.499749556]: Turtle subscriber@[4.541317, 6.445714, -1.479926]
[ INFO] [1724756562.516596768]: Turtle subscriber@[4.543028, 6.429805, -1.403927]
[ INFO] [1724756562.531814646]: Turtle subscriber@[4.544986, 6.413925, -1.447927]
[ INFO] [1724756562.549518847]: Turtle subscriber@[4.546286, 6.402209, -1.431926]
[ INFO] [1724756562.564817998]: Turtle subscriber@[4.548089, 6.382214, -1.415926]
[ INFO] [1724756562.580295986]: Turtle subscriber@[4.552389, 6.366504, -1.399927]
[ INFO] [1724756562.598811799]: Turtle subscriber@[4.555361, 6.350782, -1.383927]
[ INFO] [1724756562.612325151]: Turtle subscriber@[4.558585, 6.335111, -1.367926]
[ INFO] [1724756562.628109878]: Turtle subscriber@[4.562059, 6.319492, -1.351926]
[ INFO] [1724756562.643367833]: Turtle subscriber@[4.565783, 6.303931, -1.335927]
[ INFO] [1724756562.657932819]: Turtle subscriber@[4.569413, 6.287470, -1.319927]
[ INFO] [1724756565.50865652]: Turtle subscriber@[5.221944, 5.591155, -0.279926]
[ INFO] [1724756565.523765617]: Turtle subscriber@[5.237172, 5.590244, -0.311927]
[ INFO] [1724756565.540125256]: Turtle subscriber@[5.252477, 5.585578, -0.295927]
[ INFO] [1724756565.556106109]: Turtle subscriber@[5.267855, 5.581158, -0.279927]
[ INFO] [1724756565.57150554]: Turtle subscriber@[5.283308, 5.576984, -0.263927]
[ INFO] [1724756565.587330691]: Turtle subscriber@[5.298811, 5.573058, -0.247927]
[ INFO] [1724756565.604911111]: Turtle subscriber@[5.313381, 5.569368, -0.231927]
[ INFO] [1724756565.622646593]: Turtle subscriber@[5.330414, 5.565352, -0.215926]
[ INFO] [1724756565.636487627]: Turtle subscriber@[5.345692, 5.562774, -0.199926]
[ INFO] [1724756565.651638790]: Turtle subscriber@[5.361423, 5.559848, -0.183926]
[ INFO] [1724756565.667923819]: Turtle subscriber@[5.377197, 5.557174, -0.167927]
[ INFO] [1724756565.684285747]: Turtle subscriber@[5.393013, 5.554752, -0.151927]
[ INFO] [1724756565.69792142]: Turtle subscriber@[5.408865, 5.552584, -0.135926]
[ INFO] [1724756565.71620669]: Turtle subscriber@[5.424750, 5.550670, -0.119926]
[ INFO] [1724756565.734846907]: Turtle subscriber@[5.440901, 5.548760, -0.103927]
[ INFO] [1724756565.747411179]: Turtle subscriber@[5.456682, 5.546045, -0.087926]
[ INFO] [1724756565.763089688]: Turtle subscriber@[5.472561, 5.546455, -0.071926]
[ INFO] [1724756565.780360426]: Turtle subscriber@[5.488530, 5.545561, -0.055927]
[ INFO] [1724756565.795882397]: Turtle subscriber@[5.504523, 5.544922, -0.039926]
[ INFO] [1724756565.811392284]: Turtle subscriber@[5.528519, 5.544539, -0.023927]
[ INFO] [1724756565.827768252]: Turtle subscriber@[5.536511, 5.544413, -0.007927]
[ INFO] [1724756565.844346484]: Turtle subscriber@[5.552517, 5.544542, 0.008074]
[ INFO] [1724756565.860309184]: Turtle subscriber@[5.568513, 5.544927, 0.024074]

```

Figure 24. Turtle Listener Output & Turtle trajectory (Task 3)

7. Task 4

While the Publish/Subscribe architecture is highly effective for decoupling components and managing data flow in distributed systems, it is not ideal for scenarios requiring Remote Procedure Call (RPC) or Request/Reply interactions. These interactions, which are common in many distributed systems, require a different communication model where a client sends a request and waits for a specific reply from a server. The nature of Publish/Subscribe, with its one-to-many communication pattern, doesn't align well with the direct, synchronous exchanges required by RPC. For this reason, we needed to add a service to the cpp code.

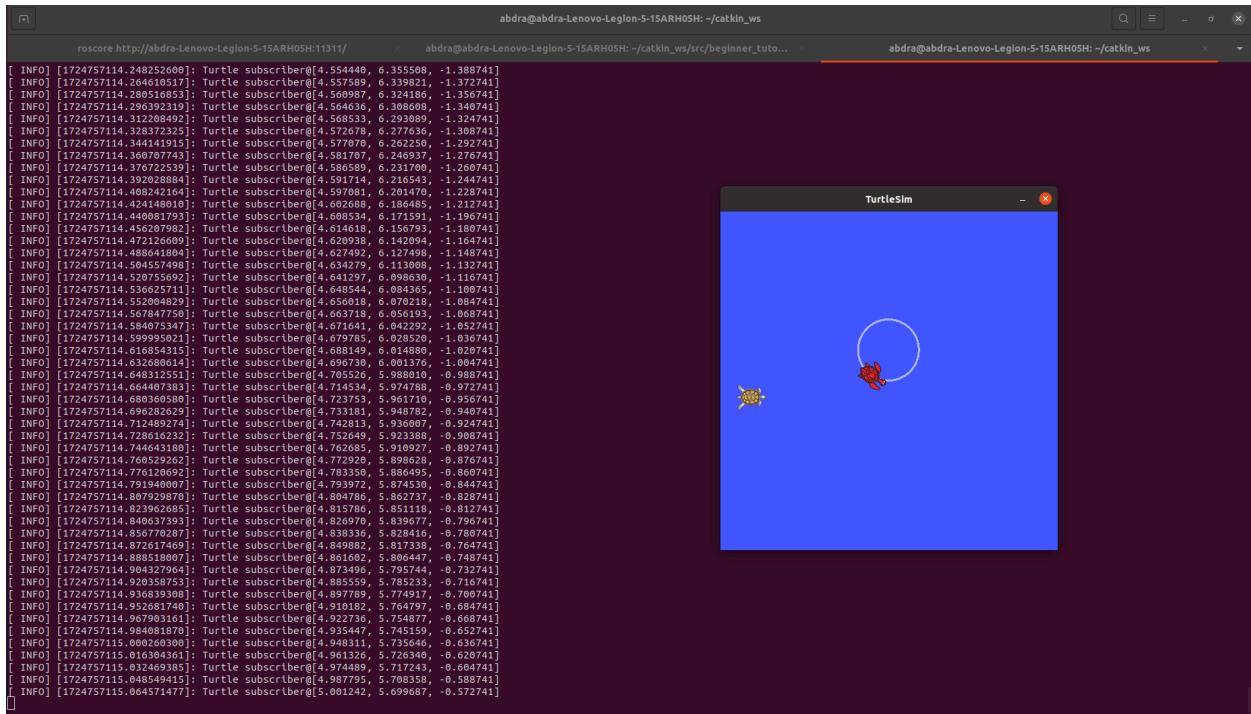


Figure 25. Turtle Listener Output, Turtle trajectory, Spawn of new turtle (Task 4)

FailTeamLab1 / src / turtlebot_controller / src / **spawn_client.cpp** ↗

 Menerallika This is the code that we produced for the Lab 1 tasks.

Code Blame 28 lines (24 loc) · 590 Bytes  **Code 55% faster with GitHub**

```
1 #include "ros/ros.h"
2 #include "turtlesim/Spawn.h"
3 #include <cstdlib>
4
5 int main(int argc, char **argv)
6 {
7     ros::init(argc, argv, "spawn_turtle_client");
8     ros::NodeHandle nh;
9     ros::ServiceClient client1 = nh.serviceClient<turtlesim::Spawn>("/
10
11 turtlesim::Spawn srv1; // define a /spawn service message
12 srv1.request.x = 1.0;
13 srv1.request.y = 5.0;
14 srv1.request.theta = 0.0;
15 srv1.request.name = "Turtle_FAILED_TEAM";
16
17 if (client1.call(srv1))
18 {
19     ROS_INFO("Spawned");
20 }
21 else
22 {
23     ROS_ERROR("Failed to call service add_two_ints");
24     return 1;
25 }
26
27 return 0;
28 }
```

Figure 26. C++ code with added service to spawn a new turtle (Task 4)

8. Task 5

For the final task, we had to make the turtle move along a desired trajectory. For this task, we have used terminal commands to specify turtle's speed in each axis.

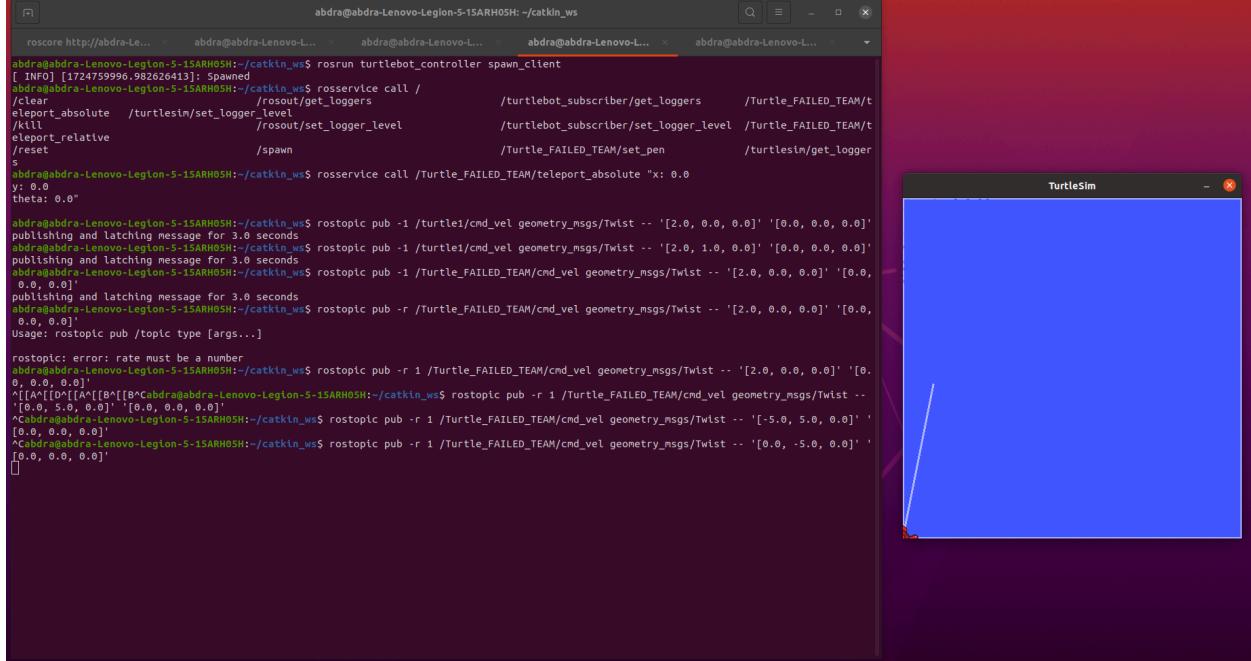


Figure 27. Turtle moving in rectangle trajectory (Task 5)

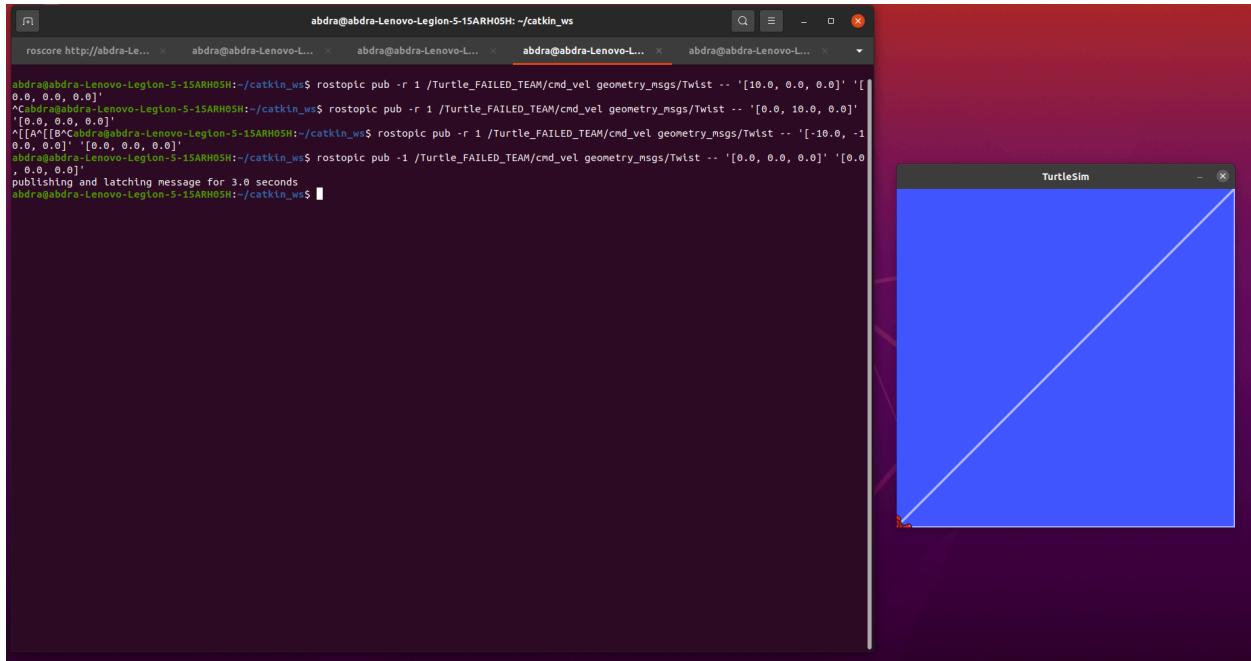


Figure 27. Turtle moving in triangle trajectory (Task 5)

9. Conclusion

In this laboratory report, we explored the practical applications of the Robot Operating System (ROS) through a series of structured tasks that progressively introduced us to the foundational concepts and tools within ROS. The exercises undertaken demonstrated the flexibility and power of ROS in managing robotic systems, from basic communication between nodes to more complex tasks involving real-time control and feedback.

Throughout the tasks, we successfully implemented and tested various components, including the creation of ROS packages, the implementation of publisher and subscriber nodes, and the manipulation of a virtual robot within the Turtlesim environment. By working through these tasks, we gained hands-on experience in setting up and configuring a ROS environment, developing custom message types, and debugging ROS applications.

One of the key takeaways from this lab is the importance of modularity and reusability in robotic software design, as exemplified by ROS's architecture. The ability to easily integrate different nodes and leverage a vast library of pre-existing packages significantly accelerates the development process, making ROS an indispensable tool in both academic research and industrial robotics.

The exercises also highlighted the strengths and limitations of different communication models within ROS, such as the Publish/Subscribe architecture and the use of services for request-reply interactions. Understanding when and how to use these models is crucial for developing efficient and effective robotic systems.

Overall, this lab has provided a solid foundation in ROS, equipping us with the knowledge and skills necessary to tackle more complex robotic projects in the future. The experience gained here will be invaluable as we continue to explore the capabilities of ROS in various applications and contribute to the ongoing development of robotic technologies.