

BAHRIA UNIVERSITY ISLAMABAD

ROBOTICS LAB

LAB 2: INTRODUCTION TO ROBOT OPERATING SYSTEM - 1

Name of Student: Muhammad Omar

Roll No.: 01-132182-024

Date of Experiment: 11/11/2021

Report submitted on: 18/11/2021

Marks obtained: ……………………………………

Remarks: ……………………………………………

Instructor’s Signature: ……………………………...

Fall 2021

**Introduction to ROS:**

ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

In ROS, all the major functionality is broken up into a number of chunks that communicate with each other using messages. Each chunk is called a node and is typically run as a separate process. Matchmaking between nodes is done by the ROS master.

# **ROS Master**

The ROS Master provides naming and registration services to the rest of the nodes in the ROS system. It tracks publishers and subscribers to topics as well as services. The role of the Master is to enable individual ROS nodes to locate one another. Once these nodes have located each other they communicate with each other peer-to-peer.

he Master is most commonly run using the roscore command, which loads the ROS Master along with other essential components.

# **Understanding ROS Nodes**

A node is simply an executable file that performs some tasks. Nodes exchange control messages, sensor readings, and other data by publishing or subscribing to topics or by sending requests to services offered by other nodes (these concepts will be discussed in detail later in the lab). Nodes can be written in a variety of languages, including Python and C++, and ROS transparently handles the details of converting between different datatypes, exchanging messages between nodes, etc.

ROS nodes use a ROS client library to communicate with other nodes. ROS client libraries allow nodes written in different programming languages to communicate:

* rospy = python client library
* roscpp = c++ client library

To run a rosnode you first have to run the ros master and then run the command for rosnode. Rosrun is used to run a node and you have write package name and node name that you want to run in this command.

* $ rosrun [package\_name] [node\_name]

Once the nodes start running, you can use command “rosnode list” to keep track on what nodes are currently running.

* $ rosnode list.

# ROS Topic

Topics are named buses over which nodes exchange messages. Topics have anonymous publisher/subscriber semantics. A node does not care which node published the data it receives or which one subscribes to the data it publishes. There can be multiple publishers and subscribers to a topic.

# ROS Messages

Nodes communicate with each other by publishing messages to topics. A message is a simple data structure, comprising typed fields. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays (much like C structs).

Some types of messages are as fellows:

* Std\_msgs/bool
* Std\_msgs/int32
* Std\_msgs/string
* Std\_msgs/float64

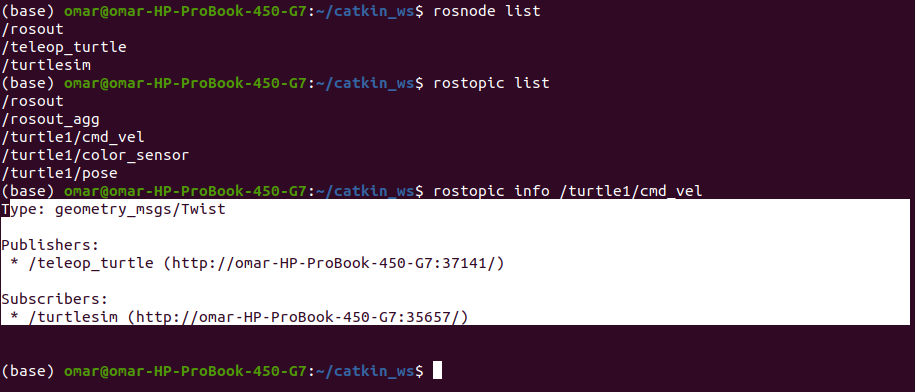
# **Goals /Objectives:**

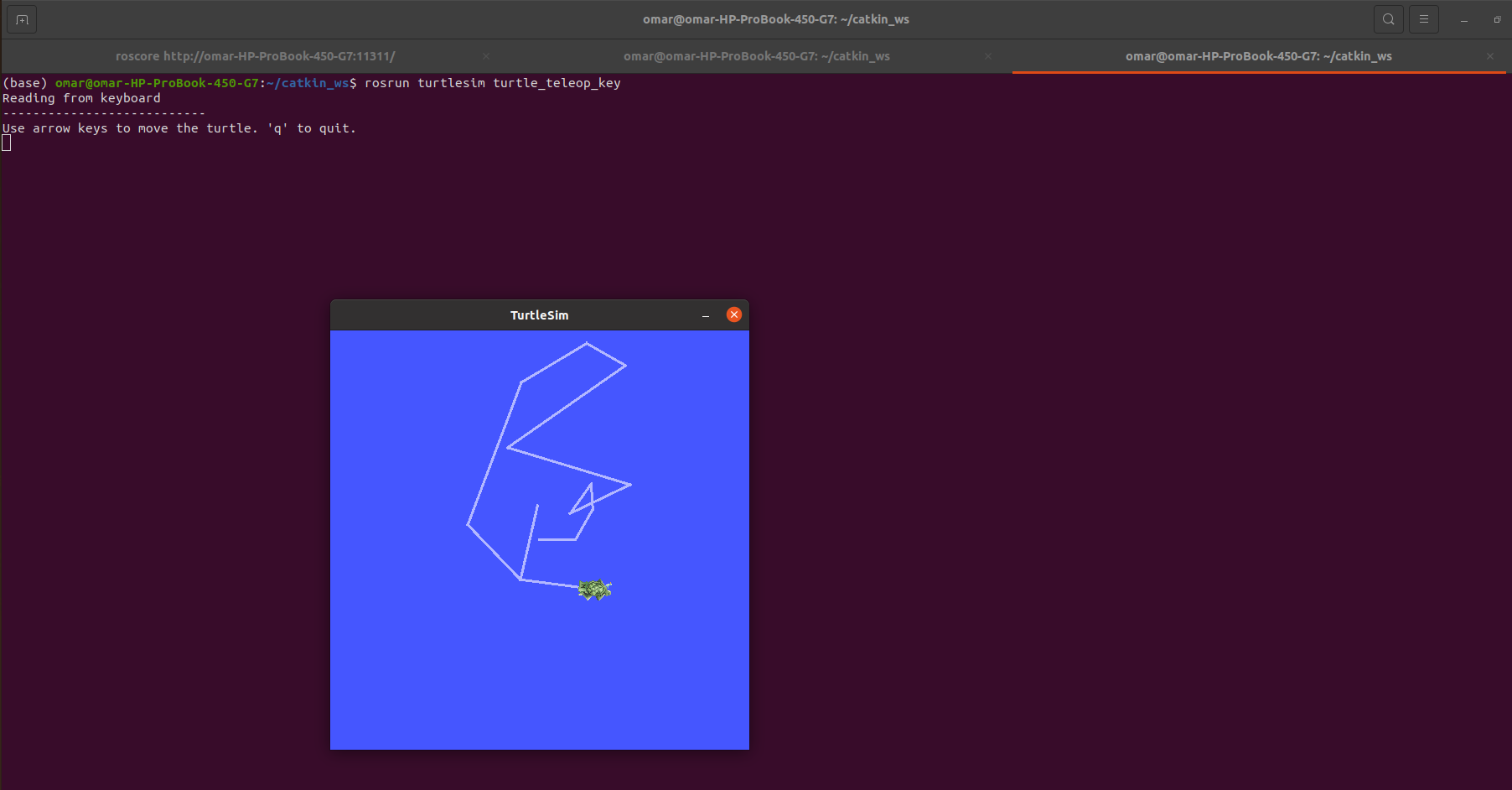
By the end of this lab, you should be able to:

* Set up a new ROS environment, including creating a new workspace and creating a package with the appropriate dependencies specified.
* Use the catkin tool to build the packages contained in a ROS workspace Run nodes using rosrun.
* Use ROS’s built-in tools to examine the topics and services used by a given node.

# **Tasks**

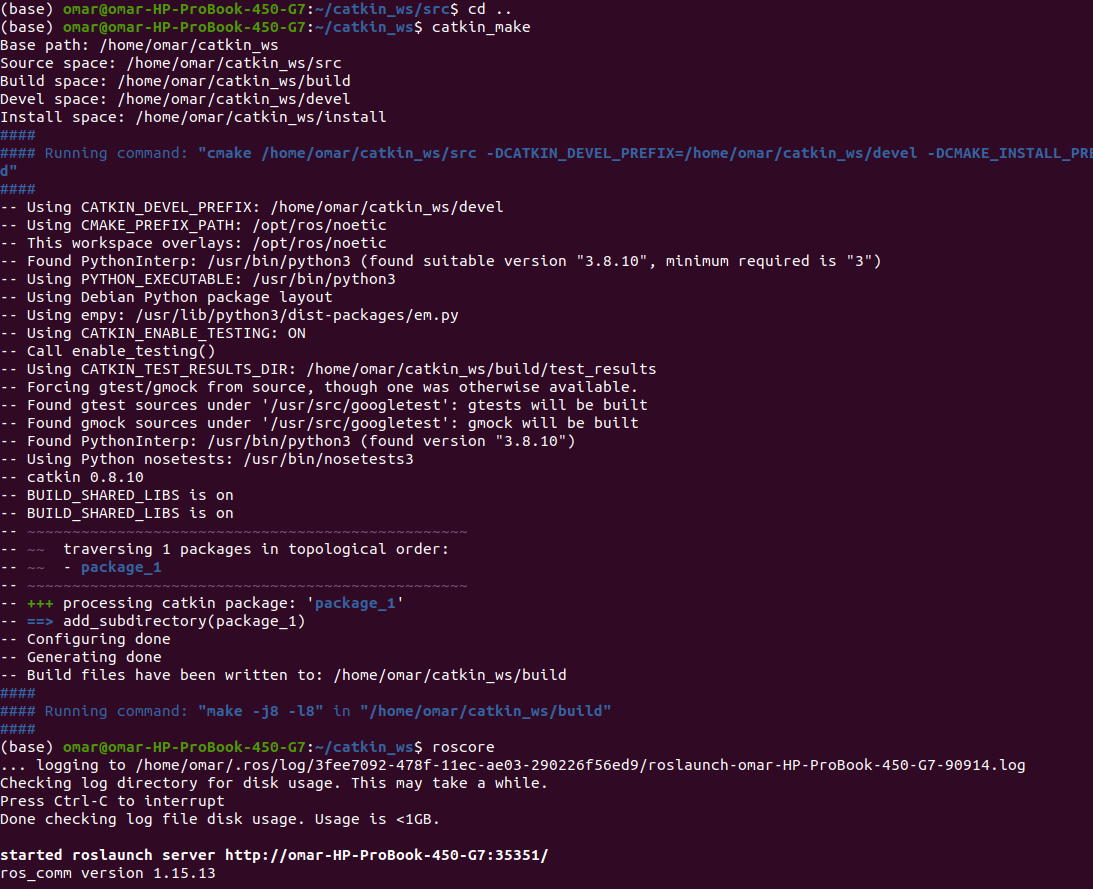
* **Task-1: Install the ROS and run command “roscore” to check if the ros is correctly installed**.

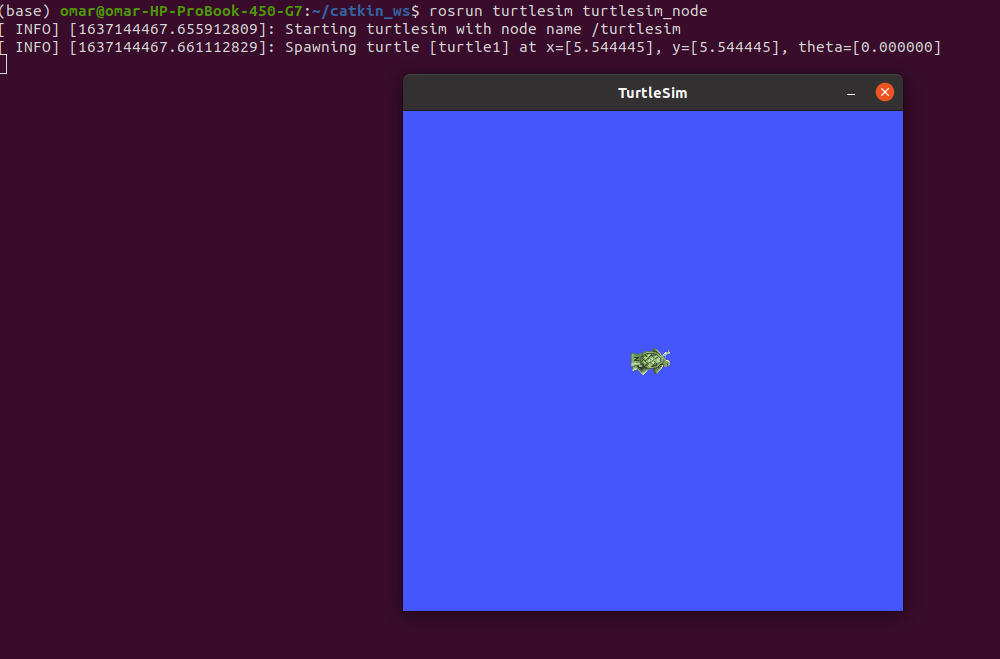


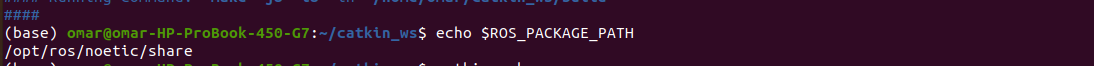




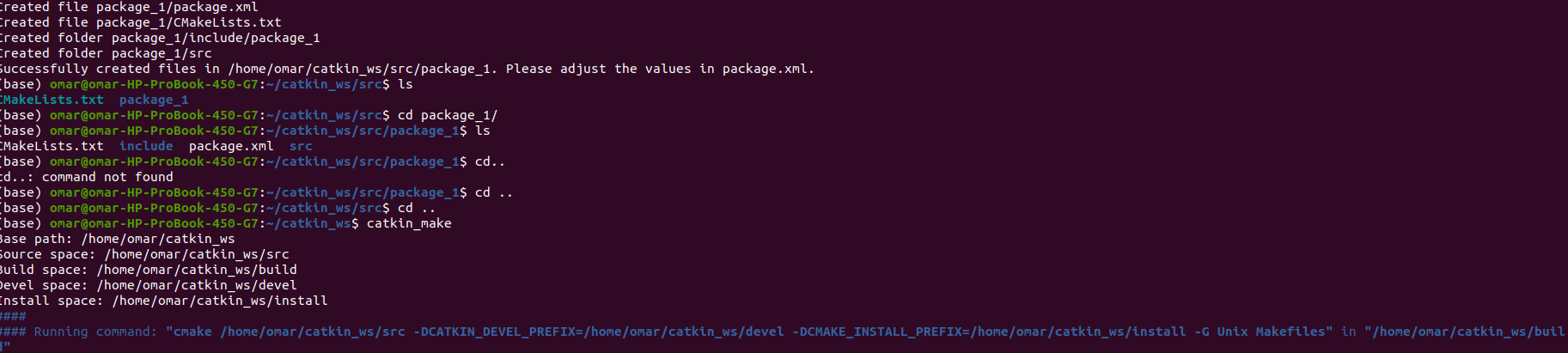
* **Task-2: Create a Ros workspace and run command “echo $ROS\_PACKAGE\_PATH” to check if workspace is properly overlayed by the setup script.**

****

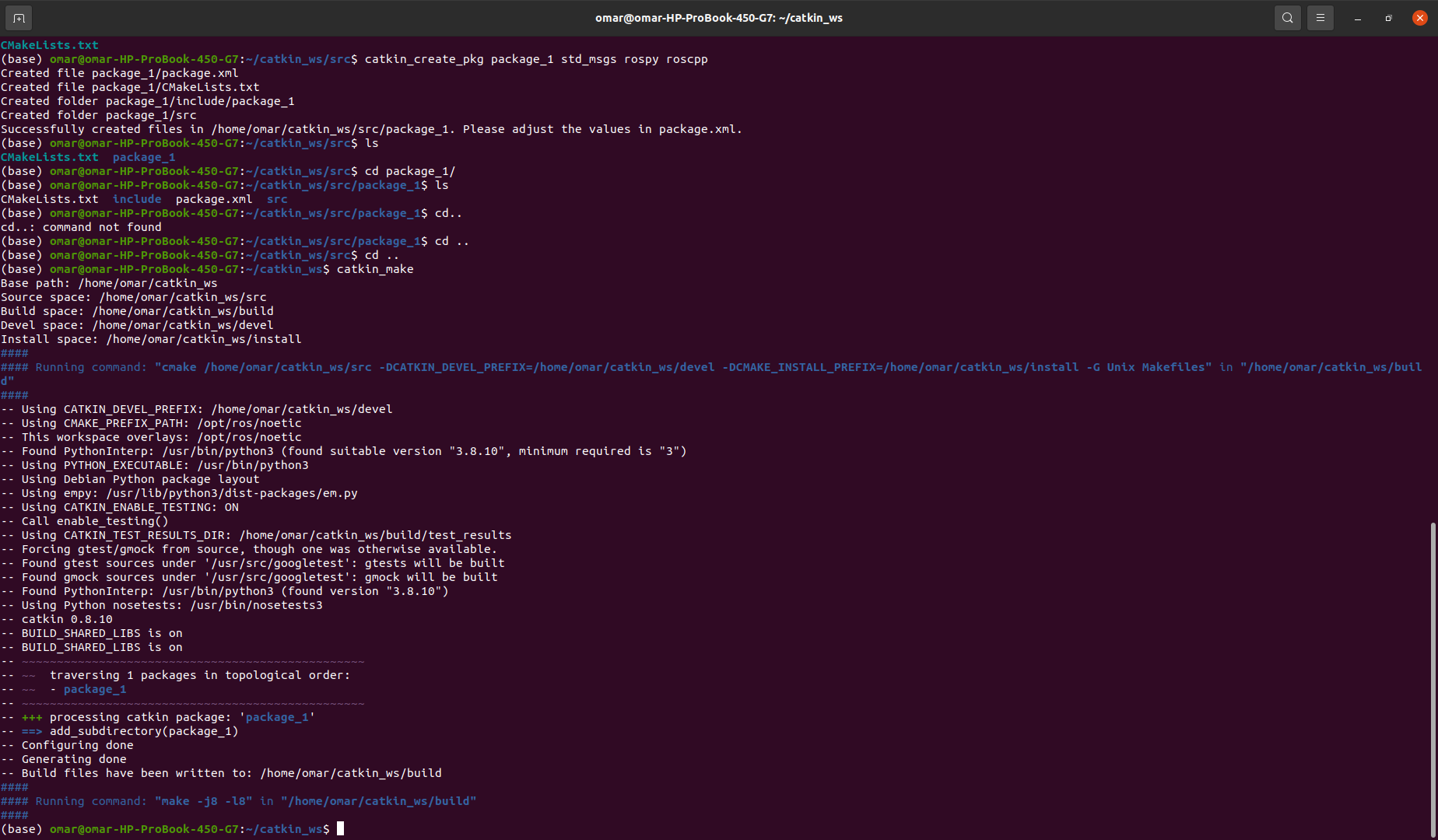
****

****

* **Task-3: Create a package by your group name and Explain the contents of your ~/ros\_workspaces directory.**

****

* **Task-4:Demonstrate the use of the catkin make command.**

****

* **Task-5**: Use ROS’s utility functions to get data about packages.



**Conclusion:**

* I have downloaded ROS in Linux Ubuntu and also created a workspace and a package inside that workspace.
* I have also implemented all other tasks as assigned to me in lab objectives.