**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Dr. Hafsa Iqbal **Date:** 26/02/2024

**Semester:** 8th **Group:** GP-1

**Lab Engineer:** Munadi Ahmad Sial

**EE-381 Robotics**

Lab 4: Publishing and Subscribing in ROS

**Group Members**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **PLO5 – CLO4** | | **PLO9 –**  **CLO4** |  |
| **Name** | **Reg. No** | **Analysis of Data in Lab Report** | **Modern Tool Usage** | **Individual and Team Work** | **Total Marks** |
|  |  | **10 Marks** | **5 Marks** | **5 Marks** | **20 Marks** |
| Muhammad Abdullah Sohail | 343642 |  |  |  |  |
| Muhammad Ahmed Mohsin | 333060 |  |  |  |  |
| Muhammad Umer | 345834 |  |  |  |  |
| Hassan Rizwan | 335753 |  |  |  |  |
| Muhammad Saad | 333414 |  |  |  |  |

**Table of Contents**

[2 Publishing and Subscribing in ROS 3](#_Toc160053133)

[2.1 Introduction 3](#_Toc160053134)

[2.2 Objectives 3](#_Toc160053135)

[2.3 Theory 3](#_Toc160053136)

[3 Lab Tasks 4](#_Toc160053137)

[3.1 Lab Task 1 – Simple Publisher 4](#_Toc160053138)

[3.2 Lab Task 2 – Simple Subscriber 6](#_Toc160053139)

[3.3 Lab Task 3 – Publish and Subscribe 7](#_Toc160053140)

[3.4 Lab Task 4 – Multiple Subscribers 9](#_Toc160053141)

[4 Conclusion 11](#_Toc160053142)

# Publishing and Subscribing in ROS

## Introduction

This laboratory exercise will extend the previously introduced concept of nodes in ROS. The nodes are only useful when they start communicating with other nodes in the graph-based architecture. The communication involves transfer of data over message streams known as topics. The topic, which is the subject of this lab, is the most commonly used method for node communication.

## Objectives

The following are the main objectives of this lab:

* Creating and Executing nodes scripted in python
* Publishing a message over a topic
* Subscribe a message from a topic
* Passing messages between nodes via publishing and subscribing

## Theory

ROS 2 is a framework for programming robotic platforms in which nodes communicate with other nodes. The most common method of communicating is via topics which use a publish/subscribe mechanism. A node can publish a message to a topic. The message is then sent to the nodes that have subscribed to that topic. The message types are mainly taken from the std\_msgs package.

The terminal commands are given as:

**cd <directory>**  change directory

**cd..**  go back to previous directory

**pwd**  print the current directory

**ls** list the contents of the current directory

**python <script.py>** execute python script

A brief summary of the terminal commands needed for working in ROS are provided below:

**colcon build**

build the workspace whenever a node is created or modified

**. install/setup.bash**

makes the terminal “aware” of the workspace (notice the dot and the space)

**ros2 pkg create --build-type ament\_python <package>**

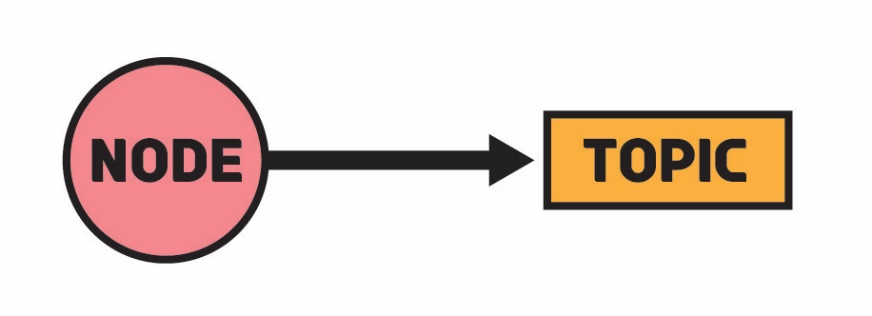
create a new package (must be done in src directory)

**ros2 run <package> <node>**

execute a node

# Lab Tasks

## Lab Task 1 – Simple Publisher



In this task, you will create a simple publisher node that will send messages to a topic. Before starting, you will have to create a new workspace with a unique name. You will create a package called *topic\_basics* in that workspace. The node for the first task is to be called simple\_pub.py

The following syntax shows how to create a workspace myWorkspace with a package myPackage and node myNode in that package.

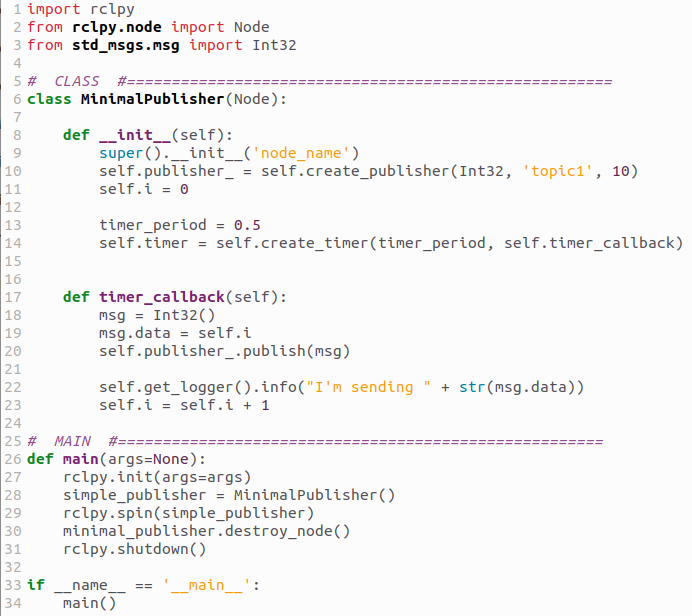
mkdir –p myWorkspace/src

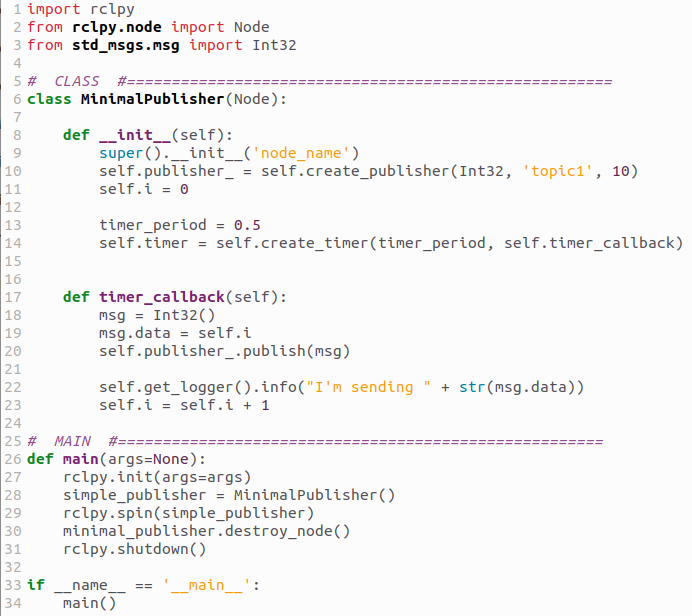
cd src

ros2 pkg create --build-type ament\_python myPackage –-dependencies rclpy std\_msgs --node-name myNode

The node you have created will be added to the setup.py in the entry points. For nodes in the later tasks, you will have to manually place their entries in the setup file otherwise they will not be executed.

Once you have created the node, place the following code in that node. You are required to manually type each line in order to properly understand the publishing syntax.





Once you have created the node, build the workspace (from the root of the workspace) and source the terminal. Then execute the node with ros2 run command. You will start sending the messages to a topic. Now open a new terminal, source that terminal and then execute the following command:

ros2 topic list

Find the topic where you are publishing (Look closely at the publisher code for this). Then use the following command to get the published messages:

ros2 topic echo <topic\_name>

You should start getting the messages in the second terminal. For this task submission, you will have to make slight adjustments to the node.

1. Change the node name in the constructor to ‘task1’
2. Change the topic name to ‘even\_num’
3. Modify the code so that it sends the numbers 2, 4, 6, 8, 10, 12, 14…

Once you have made the above changes, rebuild the workspace and execute the node. Use the echo command in the second terminal to get the values. You must provide the code showing your modifications. Only give the code of the modified class definition. You must also provide a single screenshot which shows the terminals (publisher node and echo) interacting. The same screenshot must show both the terminals.

### TASK 1 CLASS DEFINTION CODE STARTS HERE ###

*class* MinimalPublisher(*Node*):

*def* \_\_init\_\_(*self*):

*super*().\_\_init\_\_("task1")

*self*.publisher\_ = *self*.create\_publisher(Int32, "even\_num", 10)

*self*.i = 2

        timer\_period = 0.5

*self*.timer = *self*.create\_timer(timer\_period, *self*.timer\_callback)

*def* timer\_callback(*self*):

        msg = Int32()

        msg.data = *self*.i

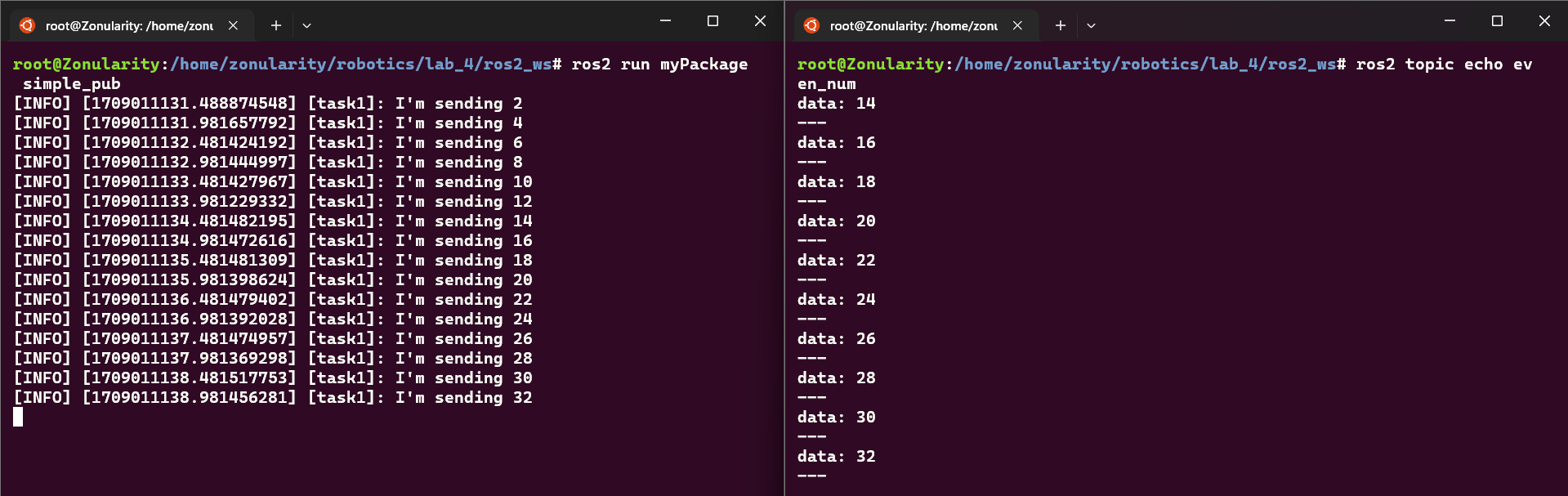
*self*.publisher\_.publish(msg)

*self*.get\_logger().info("I'm sending " + *str*(msg.data))

*self*.i = *self*.i + 2

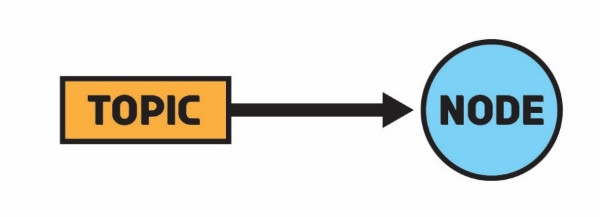
### TASK 1 CLASS DEFINTION CODE ENDS HERE ###

### TASK 1 SCREENSHOT STARTS HERE ###

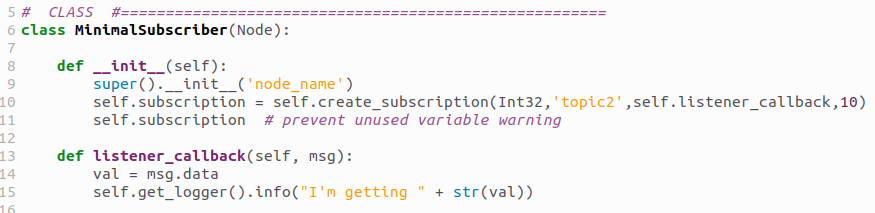


### TASK 1 SCREENSHOT ENDS HERE ###

## Lab Task 2 – Simple Subscriber



For this task, create a copy of the publisher node and rename it to simple\_sub.py. Add its entry points in the setup file. In the new node, replace the publisher class with the subscriber class given below. You will also have to make some adjustments in the main function so that the subscriber object is initialized. Change the node name in the constructor to ‘task2’ and the topic name to ‘numbers’.

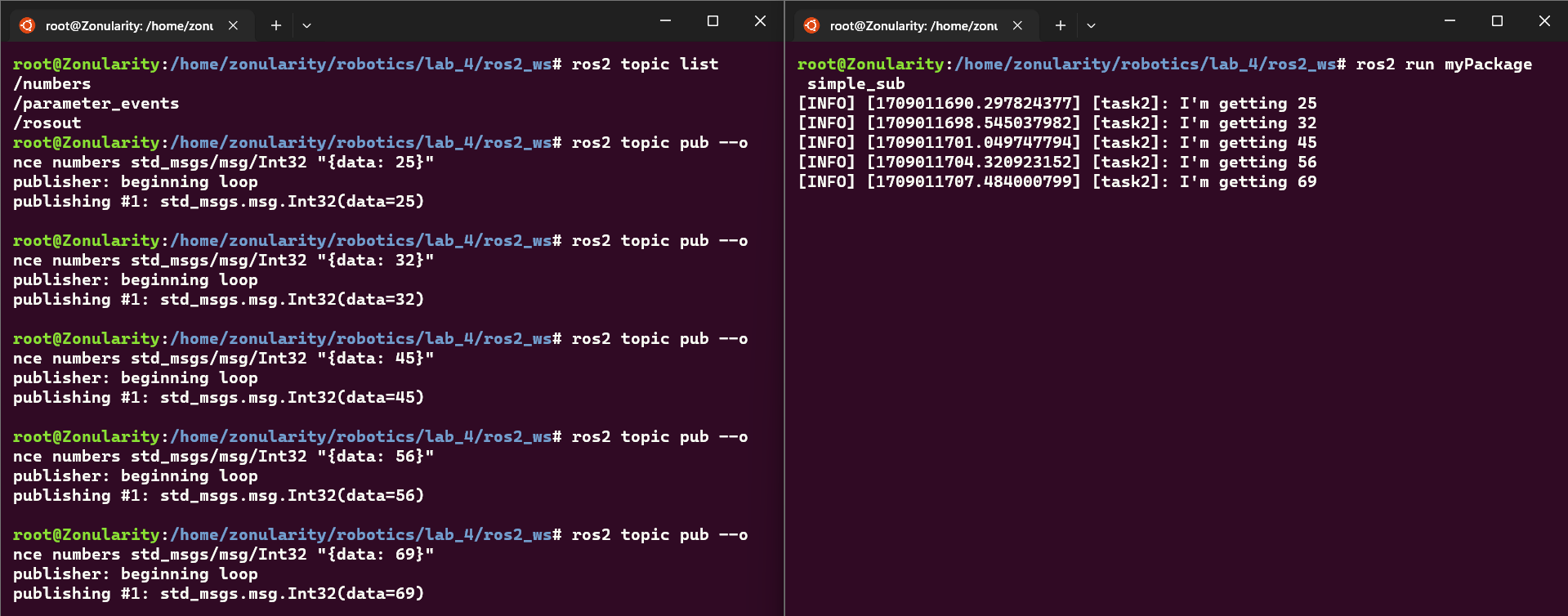


Build the workspace and execute the subscriber node. Open a new terminal (source it if needed) and use the following command to manually publish a value to the topic (you will have to specify the proper topic name):

ros2 topic pub --once <topic> std\_msgs/msg/Int32 "{data: 25}"

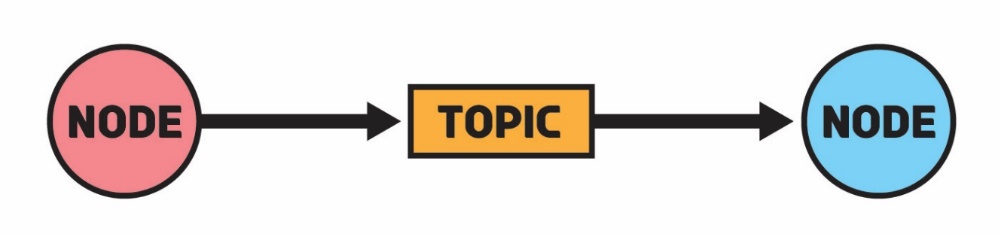
You should see the published message in the subscriber. For this task, you need to publish 5 different values and show them displayed in the subscriber. Provide a single screenshot showing both of the terminals.

### TASK 2 SCREENSHOT STARTS HERE ###



### TASK 2 SCREENSHOT ENDS HERE ###

## Lab Task 3 – Publish and Subscribe



In the previous tasks, you have a simple publisher and simple subscriber separately. You must use your previous understanding to combine the publisher and subscriber nodes (If you’re having difficulty with this task, review the previous two tasks).

Create two nodes pub.py and sub.py. The publisher node must keep publishing random values between 1 and 9 to a ‘random’ topic. The subscriber node must display these published random values. You must provide the publisher and subscriber class codes as well as a single screenshot showing the two nodes (in two terminals) interacting.

### TASK 3 PUBLISHER CLASS DEFINTION CODE STARTS HERE ###

*class* Publisher(*Node*):

*def* \_\_init\_\_(*self*):

*super*().\_\_init\_\_("task3")

*self*.publisher\_ = *self*.create\_publisher(Int32, "random", 10)

        timer\_period = 1

*self*.timer = *self*.create\_timer(timer\_period, *self*.timer\_callback)

*def* timer\_callback(*self*):

        msg = Int32()

        rand\_num = random.randint(0, 10)

        msg.data = rand\_num

*self*.publisher\_.publish(msg)

*self*.get\_logger().info("I'm sending " + *str*(msg.data))

### TASK 3 PUBLISHER CLASS DEFINTION CODE ENDS HERE ###

### TASK 3 SUBSCRIBER CLASS DEFINTION CODE STARTS HERE ###

*class* Subscriber(*Node*):

*def* \_\_init\_\_(*self*):

*super*().\_\_init\_\_("task3")

*self*.subscription = *self*.create\_subscription(

            Int32, "random", *self*.listener\_callback, 10

        )

*self*.subscription

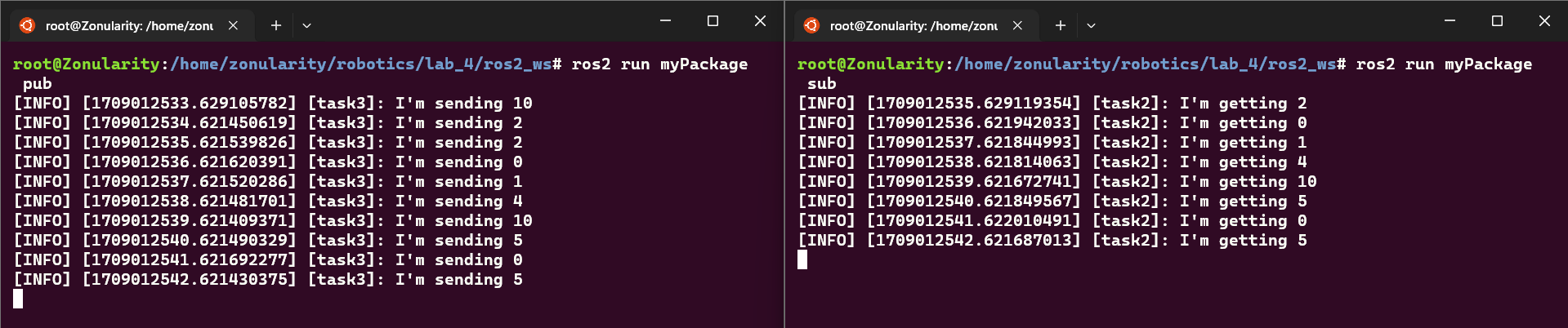
*def* listener\_callback(*self*, *msg*):

        val = msg.data

*self*.get\_logger().info(*f*"I'm getting " + *str*(val))

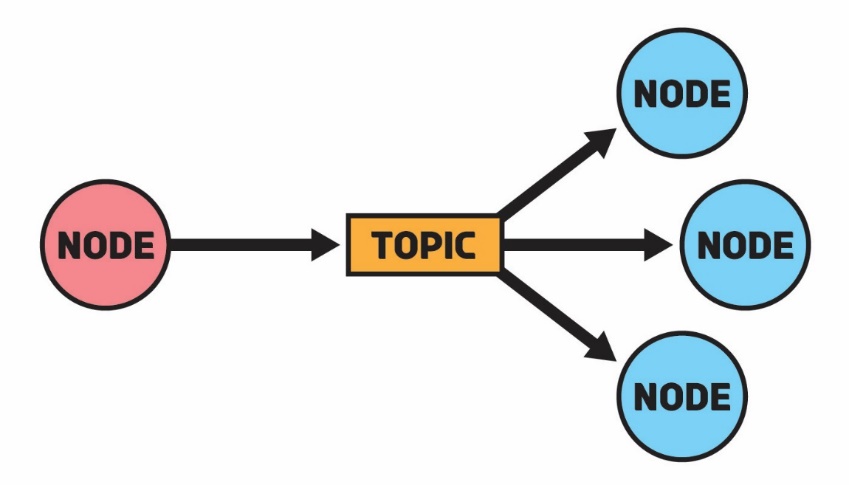
### TASK 3 SUBSCRIBER CLASS DEFINTION CODE ENDS HERE ###

### TASK 3 SCREENSHOT STARTS HERE ###



### TASK 3 SCREENSHOT ENDS HERE ###

## Lab Task 4 – Multiple Subscribers



In this task, you will use the same publisher node from the previous task to send random values to a ‘random’ topic. You will create 3 subscriber nodes sub1, sub2 and sub3 which will subscribe to the ‘random’ topic and use the received random values to perform additional tasks:

* **Sub1:** Doubles the received values and displays the result
* **Sub2:** Displays whether the received value is odd or even
* **Sub3:** Calculates the factorial of the received values and displays them

You will provide class definition code for all 3 subscribers and a single screenshot which shows four terminals interacting (1 publisher and 3 subscribers).

### TASK 4 SUB 1 CLASS DEFINTION CODE STARTS HERE ###

*class* Subscriber(*Node*):

*def* \_\_init\_\_(*self*):

*super*().\_\_init\_\_("task4")

*self*.subscription = *self*.create\_subscription(

            Int32, "random", *self*.listener\_callback, 10

        )

*self*.subscription

*def* listener\_callback(*self*, *msg*):

        val = msg.data

        val = val \* 2  *# double the value*

*self*.get\_logger().info(*f*"Doubled value: " + *str*(val))

### TASK 4 SUB 1 CLASS DEFINTION CODE ENDS HERE ###

### TASK 4 SUB 2 CLASS DEFINTION CODE STARTS HERE ###

*class* Subscriber(*Node*):

*def* \_\_init\_\_(*self*):

*super*().\_\_init\_\_("task4")

*self*.subscription = *self*.create\_subscription(

            Int32, "random", *self*.listener\_callback, 10

        )

*self*.subscription

*def* listener\_callback(*self*, *msg*):

        val = msg.data

        if val % 2 == 0:

*self*.get\_logger().info(*f*"{val} is even")

        else:

*self*.get\_logger().info(*f*"{val} is odd")

### TASK 4 SUB 2 CLASS DEFINTION CODE ENDS HERE ###

### TASK 4 SUB 3 CLASS DEFINTION CODE STARTS HERE ###

*class* Subscriber(*Node*):

*def* \_\_init\_\_(*self*):

*super*().\_\_init\_\_("task4")

*self*.subscription = *self*.create\_subscription(

            Int32, "random", *self*.listener\_callback, 10

        )

*self*.subscription

*def* factorial(*self*, *n*):

        if n == 0:

            return 1

        else:

            return n \* *self*.factorial(n - 1)

*def* listener\_callback(*self*, *msg*):

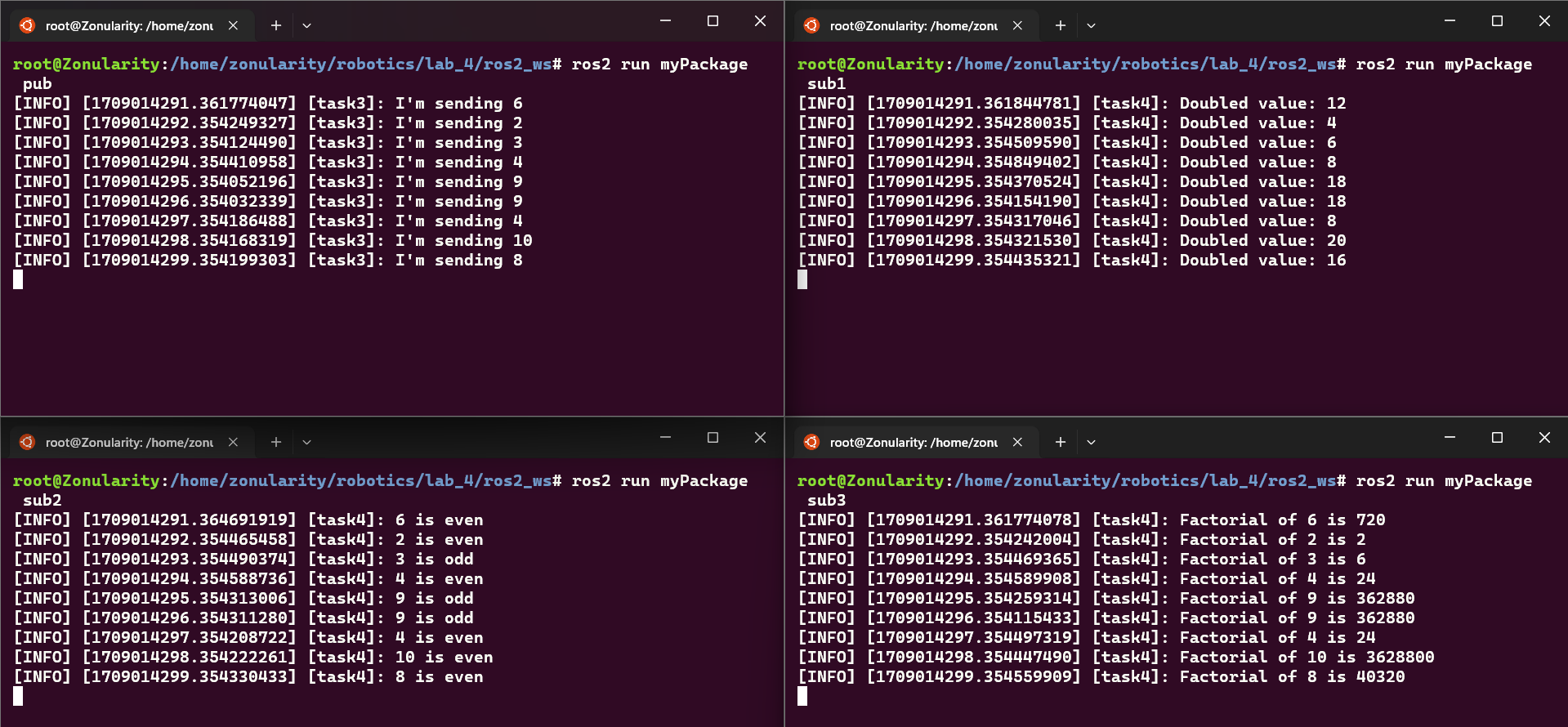
        val = msg.data

*self*.get\_logger().info(*f*"Factorial of {val} is " +

*str*(*self*.factorial(val)))

### TASK 4 SUB 3 CLASS DEFINTION CODE ENDS HERE ###

### TASK 4 SCREENSHOT STARTS HERE ###



### TASK 4 SCREENSHOT ENDS HERE ###

# Conclusion

In this lab, we explored how ROS nodes utilize topics to communicate and exchange data. We saw how these topics act as message streams, facilitating the flow of information between different nodes within the ROS network. This hands-on experience reinforced the importance of topics in enabling effective communication and collaboration between diverse components in a ROS system.