



Teleoperation using ROS

Team Members:-

- Aditya
- Arpit Gandhi
- Divya Thakar
- Ram Rajavadha
- Rohit Tanwar
- Shubh Kawa
- Tanmay Jethwani
- Manishkumar Vanjara

Mentors:-

Tirth Jain
Saniya Kureshi

Contents

1	<u>Timeline</u>	3
2	<u>Problem Statement</u>	3
3	<u>Introduction</u>	4
4	<u>Software Used</u>	4
5	<u>Python</u>	5
6	<u>Object-oriented Programming</u>	5
6.1	<u>Structure of Object-Oriented Programming</u>	5
7	<u>Nodes in ROS2</u>	6
8	<u>Topics in ROS2</u>	6
9	<u>Publisher & Subscriber</u>	7
9.1	Writing a publisher/subscriber in ROS2	7
10	<u>Turtlesim</u>	7
11	<u>Gazebo</u>	9
12	<u>Errors Faced</u>	10
13	<u>Future aspects</u>	10

1 Timeline

- Python and OOPS
- Workspace and Packages
- Topics and Nodes
- Publisher and Subscriber
- Fibonacci Series
- Turtlesim
- Rqt
- Gazebo
- Code to teleoperate the car
- Solving the problem of jerks in car during teleoperation
- Latex

2 Problem Statement

To simulate a car model in Gazebo using ROS and control its motion using teleoperation.

3 Introduction

Teleoperation (or remote operation) indicates operation of a system or machine at a distance. It is similar in meaning to the phrase "remote control" but is usually encountered in research, academia and technology. It is most commonly associated with robotics and mobile robots but can be applied to a whole range of circumstances in which a device or machine is operated by a person from a distance. In this project, we will be using ROS2 Foxy software to operate robots using the joystick or keyboard externally.

- For this project we are using the Ubuntu Operating System, Ubuntu is a Linux based open source operating system. Unlike windows which is very user friendly, Ubuntu is very command oriented OS.
- In our project we are using Ubuntu to run ROS2. Even though ROS2 can run on windows and mac OS, Ubuntu just provides more freedom for ROS to function, while windows and mac OS may restrict some functionalities of ROS such as DDS(Data Distribution Services).

4 Software Used

- ROS (Robot Operating System) has two versions ROS and ROS2 here we use ROS2 which is a more optimised version of ROS the main reason to work on ROS2 is that in ROS it is not possible to create more than one node in a process but In ROS2 it is possible to create multiple nodes in a process.
- A ROS system is composed of nodes. ROS nodes primarily communicate with each other through the passing of messages in a publisher-subscriber manner.
- Teleoperation is done using ROS by publishing a message that can command a robot how to move. Basically, a message type is published which instructs a model to move in x, y, and z direction and as well as how to rotate in x, y, and z axis.

5 Python

- Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.
- It is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming. It is often described as a “batteries included” language due to its comprehensive standard library.
- It is commonly used in artificial intelligence projects and machine learning projects due to its modular architecture, simple syntax, and rich text processing tools. ROS works smoothly on Python.

6 Object-oriented Programming

- Object-oriented programming is a computer programming model that organises software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behaviour.
- OOPS focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. This approach to programming is well-suited for programs that are large, complex and actively updated or maintained.
- The organisation of an object-oriented program also makes the method beneficial to collaborative development, where projects are divided into groups. Additional benefits of OOPS include code reusability, scalability and efficiency.

6.1 Structure of Object-Oriented Programming

1. **Classes:-** are user-defined data types that act as the blueprint for individual objects, attributes and methods.
2. **Objects:-** are instances of a class created with specifically defined data. Objects can correspond to real-world objects or an abstract identity. When class is defined initially, the description is the only object that is defined.
3. **Methods:-** are functions that are defined inside a class that describe the behaviours of an object. Each method contained in class definitions starts with a reference to an instance object. Additionally, the subroutines contained in an object are called instance methods. Programmers use methods for reusability or keeping functionality encapsulated inside one object at a time.

4. **Attributes:-** are defined in the class template and represent the state of an object. Objects will have data stored in the attributes field. Class attributes belong to the class itself.

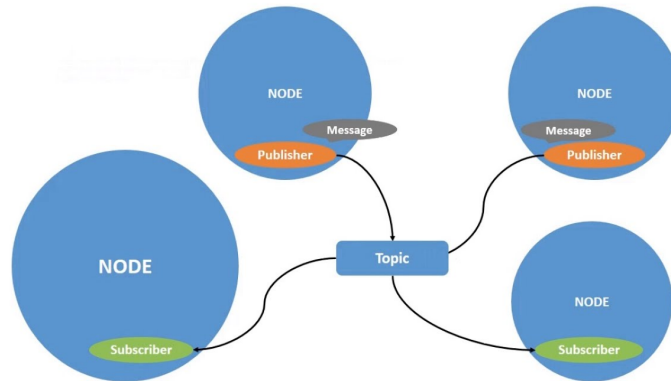
7 Nodes in ROS2

Each node in ROS should be responsible for a single module purpose (e.g. one node for controlling wheel motors, one node for controlling a laser range-finder, etc). Each node can send and receive data to other nodes via topics, services, actions, or parameters.

A full robotic system consists of many nodes working in concert. In ROS 2, a single executable (C++ program, Python program, etc.) can contain one or more nodes.

8 Topics in ROS2

- ROS 2 breaks complex systems down into many modular nodes. Topics are a vital element of the ROS graph that act as a bus for nodes to exchange messages.
 - ◇ Node may publish data to any number of topics and simultaneously have subscriptions to any number of topics.



- Topics are one of the main ways in which data is moved between nodes and therefore between different parts of the system.
- Nodes publish information over topics, which allows any number of other nodes to subscribe to and access that information.

9 Publisher & Subscriber

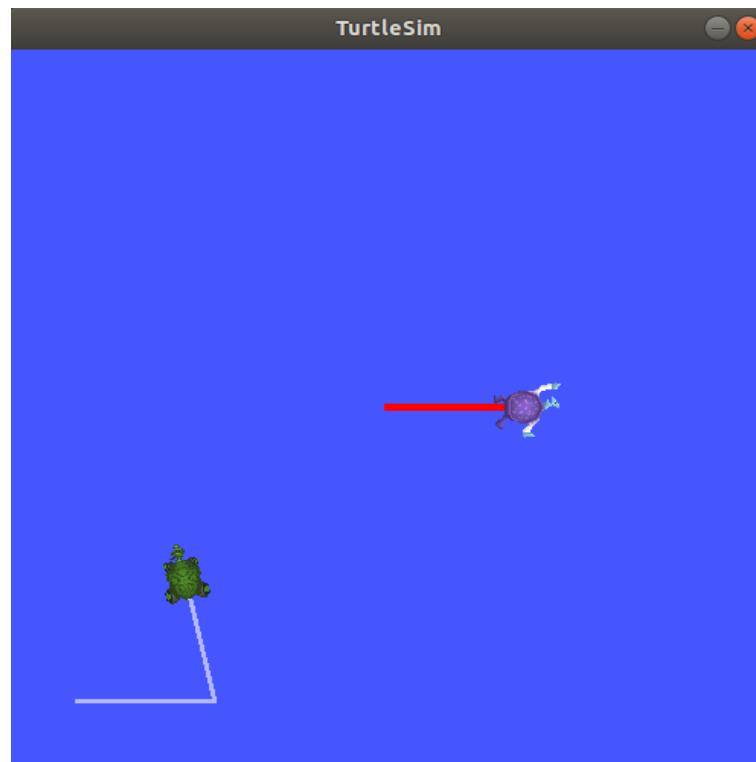
Publisher nodes publish data, subscriber nodes receive data, and a publishing subscriber node receives data and publishes data.

9.1 Writing a publisher/subscriber in ROS2

1. Create a package. Open a new terminal and source your ROS 2 installation so that ROS2 commands will work.
2. Write the publisher node. Navigate into `ros2_ws/src/file_name/topic_name`.
3. Write the subscriber node. Return to `ros2_ws/src/file_name/topic_name` to create the next node.
4. Build and run.

10 Turtlesim

- Turtlesim is a lightweight simulator for learning ROS 2. It illustrates what ROS 2 does at the most basic level, to give you an idea of what you will do with a real robot or robot simulation later on.
- RQT is a GUI(Graphics User Interface) tool for ROS 2. Everything done in RQT can be done on the command line, but it provides an easier, more user-friendly way to manipulate ROS 2 elements.

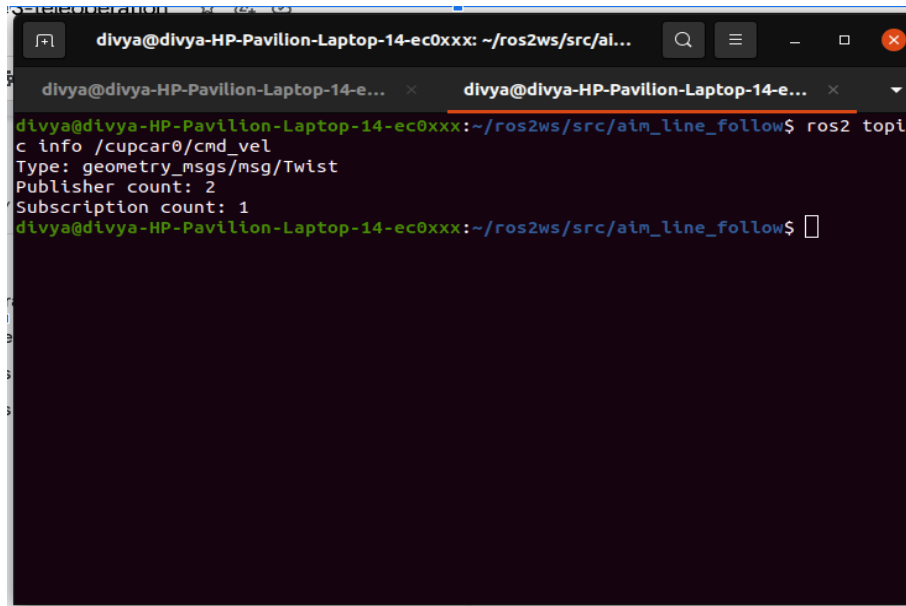


11 Gazebo

- **Gazebo** is one of the most popular multi-robot simulators, firstly developed in 2004, fully open-source and supporting a wide range of sensors and objects.
- It runs on Linux.
- It is designed to accurately reproduce the dynamics of the complex environments a robot could encounter.
- It is well integrated with ROS, flexible, provides accurate simulation with four different physical engines and benefits from a large and active community.
- Finally, robots, sensors and environment can be controlled through custom plugins.

```
ros2 topic info: error: the following arguments are required: topic_name
divya@divya-HP-Pavilion-Laptop-14-ec0xxx:~/ros2ws/src/aim_line_follow$ ros2 topic list
/clock
/cupcar0/PixyVector
/cupcar0/cmd_vel
/cupcar0/distance
/cupcar0/odom
/debugImage0
/parameter_events
/performance_metrics
/rosout
/tf
/trackImage0/camera_info
/trackImage0/image_raw
/trackImage0/image_raw/compressed
/trackImage0/image_raw/compressedDepth
/trackImage0/image_raw/theora
divya@divya-HP-Pavilion-Laptop-14-ec0xxx:~/ros2ws/src/aim_line_follow$
```

Anonymous Penguin

A terminal window with a dark background and light green text. The window title is 'divya@divya-HP-Pavilion-Laptop-14-ec0xxx: ~/ros2ws/src/ai...'. The prompt is 'divya@divya-HP-Pavilion-Laptop-14-ec0xxx:~/ros2ws/src/aim_line_follow\$'. The command 'ros2 topic info /cupcar0/cmd_vel' has been executed, resulting in the following output: 'Type: geometry_msgs/msg/Twist', 'Publisher count: 2', and 'Subscription count: 1'. The prompt is now 'divya@divya-HP-Pavilion-Laptop-14-ec0xxx:~/ros2ws/src/aim_line_follow\$' with a cursor.

```
divya@divya-HP-Pavilion-Laptop-14-ec0xxx:~/ros2ws/src/aim_line_follow$ ros2 topic info /cupcar0/cmd_vel
Type: geometry_msgs/msg/Twist
Publisher count: 2
Subscription count: 1
divya@divya-HP-Pavilion-Laptop-14-ec0xxx:~/ros2ws/src/aim_line_follow$
```

12 Errors Faced

1. **Dual Boot:-** Resolved by changing settings in BIOS environment.
2. **Gazebo Installation:-** Resolved by reinstalling the simulator strictly according to instructions.
3. **Removing jerks arising in car due to high velocities:-** Wrote a new code which changes the way the car is controlled. Rather than taking inputs multiple times, now only a single input is required and the car will maintain its motion until a different key is pressed.

13 Future aspects

- Can be used for military drones to check terrain in enemy area
- Can be used as unmanned rovers to cover surfaces on other planets
- Can be used as driverless car

References

- [1] Installation links :-
 - <https://releases.ubuntu.com/focal/>
 - <https://www.balena.io/etcher/>
- [2] Dual Boot Tutorial :-
 - <https://www.youtube.com/watch?v=u5QyjHIYwTQ>
- [3] OOPS Tutorial :-
 - <https://www.youtube.com/watch?v=qiSCMNBIP2g>
- [4] ROS2 Installation :-
 - <https://docs.ros.org/en/foxy/Installation/Alternatives/Ubuntu-Development-Setup.html>
- [5] Gazebo Installation :-
 - https://classic.gazebosim.org/tutorials?tut=ros2_installingcat_connect =
- [6] Teleoperation Codes :-
 - <https://github.com/grim010/myproject>