Robotics Operating System

1

In this chapter, we'll talk about Robotics in general and the Robotics Operating System (ROS). Also, we'll discuss the main concepts of ROS and its architecture, and the tools provided to it.

1.1 Mobile Robotics

The world grows come with future technology to help this grows comes more faster, one of those technologies is the robotics.
Robots can replace many people jobs and stealing, that's means the robots can have job opportunities in the future.
A robot has several definitions, therefore the question of what's the robot is very difficult.

- "A reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of task." Robot Institute of America, 1979
- "An automatic device that performs functions normally ascribed to humans or a machine in the form of a human." Webster's Dictionary
- "a reprogrammable manipulator device" British Department of Industry

There're multiple types of mobile robotics such as wheeled, tracked, legged, flying, and underwater robots.

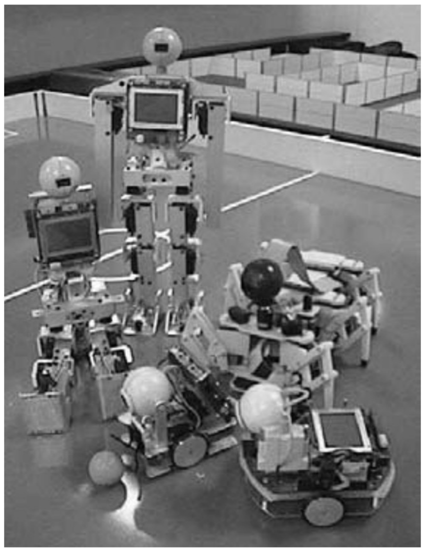


Figure 1.1 Mobile Robot Types

1.1. a Single Wheel Drive

The simplest concept of the wheeled robot is the single wheel that is driven and steered by the mobile robot, also has two passive caster wheels in the back because it must contain three points.

The linear velocity of the robot is to make sure the wheel is positioned in the middle and driven at the required speed.

The angular velocity of the robot is to position the wheel with an angle matching the required curve.

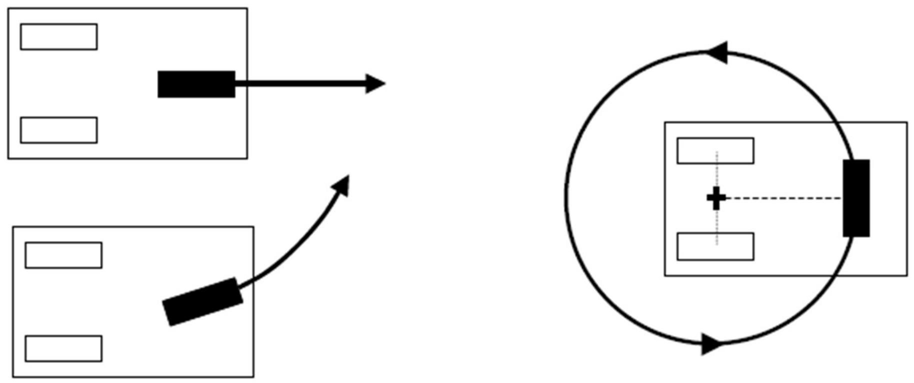


Figure 1.2 Driving and rotation of single wheel drive

1.1. b Differential Drive

The differential drive has two fixed wheels on the robot side (right and left) and one positive caster wheel, we can add an additional caster wheel depending on the location of the driven wheels.

The linear velocity of the robot is to make sure the two wheels are moving at the same speed required.

The angular velocity of the robot is to make one wheel faster, slower, or opposite direction than the other one to match the angle on the curve.

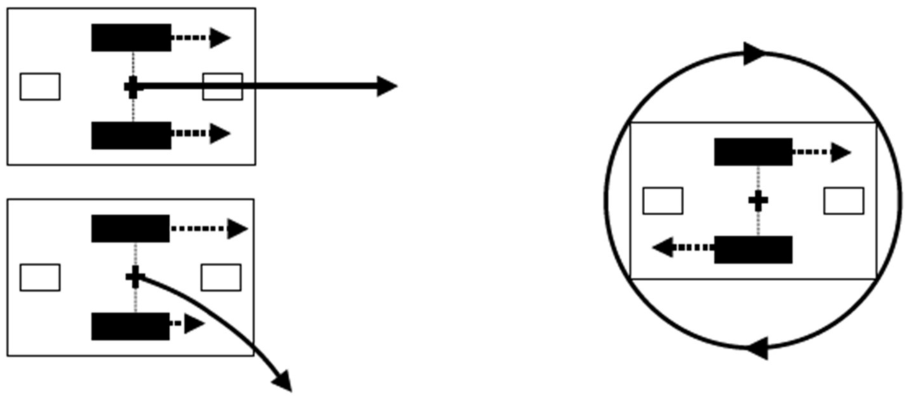


Figure 1.3 Driving and rotation of differential drive

1.1. c Synchro Drive

Synchro drive or omni directional robot is the extension to the single wheel driven robot with three wheels all driven and all being steered. The three wheels always pointed to the same driving direction, so they rotated together.
The robot stopped and realigned its wheels when driving from forward to sideways or opposite, it can’t drive and rotate at the same time.

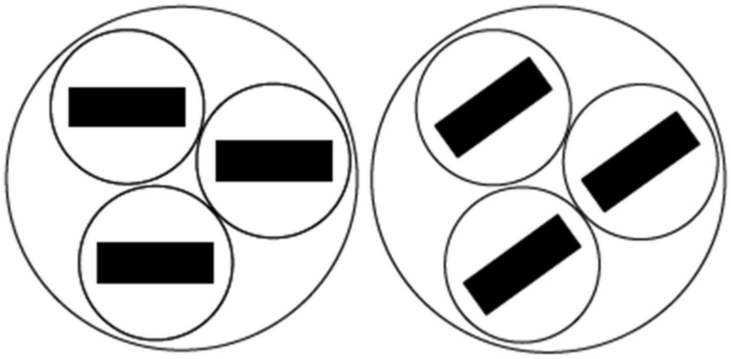


Figure 1.4 Driving and rotation of Synchro drive

1.2 Robotics with ROS

Robotic Operating System (ROS) is an open-source, flexible software framework for robotics programming, meta-operating system, hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management, and low-level control. Also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

ROS has many features can list as:

- Collaborative development: ROS is open source and free to use for industries and research. Developers can expand the functionalities of ROS by adding packages. Almost all the packages of ROS work on a hardware abstraction layer, so they can be reused easily for other robots.
- Programming language support: ROS communication framework can be implemented with modern programming languages like python, C++, and Lisp, and it has experimental libraries for java and Lua.
- Library integration: ROS supports interfacing with third-party libraries like Open-CV and PCL.
- Simulator integration: ROS provides integration with open-source simulators like Gazebo.
- Code testing: ROS provides an inbuilt testing framework to check the quality and code bugs called rostest.
- Scalability: ROS is appropriate for large runtime systems and for large development processes, it can perform heavy computation tasks.
- Customizability: ROS developers can customize this framework as per the robot's requirement. If we only want to work with the ROS messaging platform, we can remove all the other components and use only that.
- Community support

1.3 ROS Architecture

The ROS architecture has been designed and divided into three sections or levels of concepts:

- ROS Filesystem Level
- ROS Computation Graph Level
- ROS Community Level

1.3.a ROS Filesystem Level

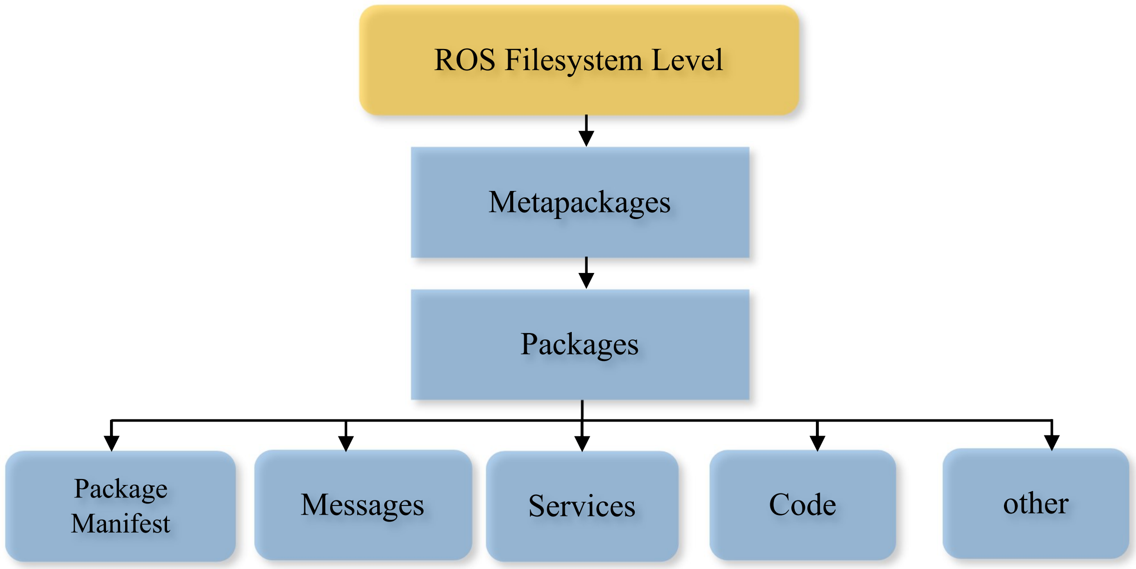


Figure 1.5 ROS Filesystem Level

- Metapackages: List of packages for a specific application such as navigation or SLAM.
- Packages: ROS software is organized as ROS packages; therefore, we can say the ROS packages are the basic build unit of ROS. Packages consist of nodes, processes, datasets, and configuration files organized in a single module.
- Package Manifest: Inside every package will be a manifest file called 'package.xml'. This file consists of information such as the name, version, author, license, and dependencies required for the package.
- Messages: ROS communicates by sending ROS messages. The type of message data can be defined inside a file with the '.msg' extension. These files are called message files.
- Services: One of the computation graph level concepts is services, can be defined inside a file with the '.srv' extension.

1.3.b ROS Computation Graph Level

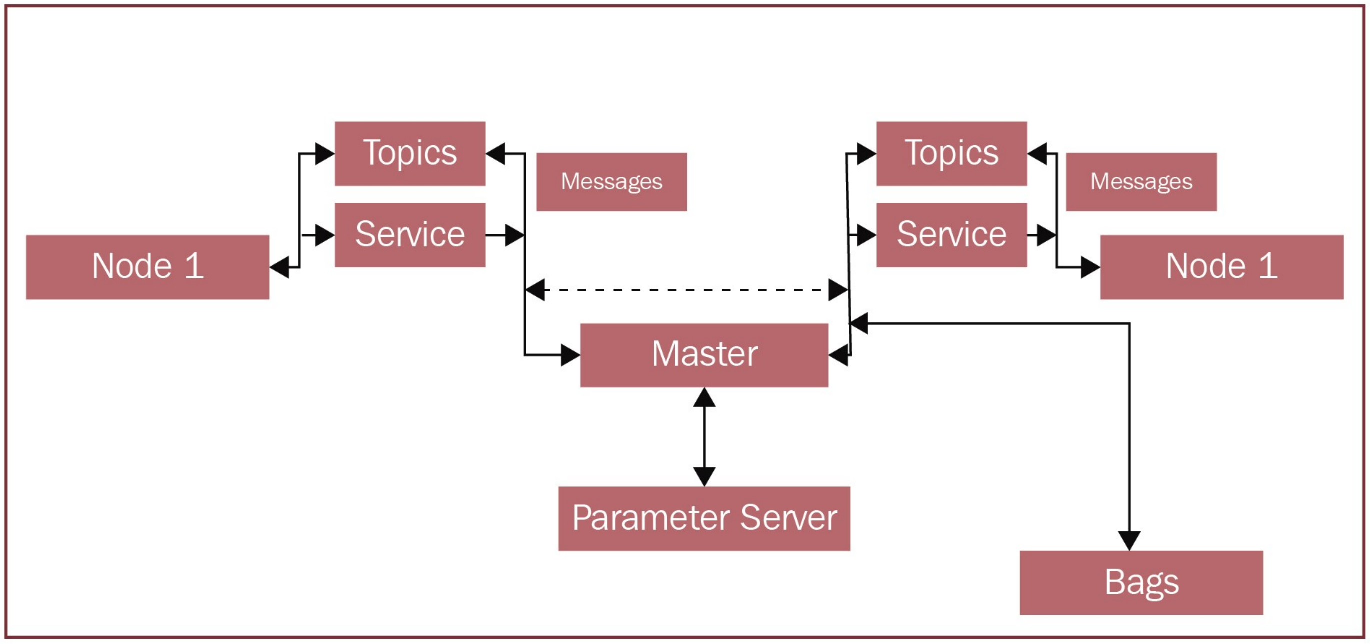


Figure 1.6 ROS Computation Graph Level

- Master: The master has all the details about all nodes running in the ROS environment. It will exchange details of one node with another to establish a connection between them. After exchanging the information, communication will start between the two ROS nodes.
- Parameter Server: The parameter server is a powerful thing in ROS. A node can store a variable in the parameter server and set its privacy too. If the parameter has a global scope, it can be accessed by all other nodes. The ROS parameter runs along with the ROS master.
- Nodes: ROS nodes are simply a process that is using ROS APIs to communicate with each other, ROS nodes are created using ROS client libraries such as roscpp and rospy.
- Message: ROS nodes can communicate with each other in many ways. In all methods, nodes send and receive data in the form of ROS messages. The ROS message is a data structure used by ROS nodes to exchange data.
- Topics: One of the methods to communicate and exchange ROS messages between two ROS nodes is called ROS topics. Topics are named buses, in which data is exchanged using ROS messages. Each topic will have a specific name, and one node will publish data to a topic and another node can read from the topic by subscribing to it.

- Services: Services are another kind of communication method, like topics. One node will act as the service provider, which has a service routine running, and a client node requests a service from the server. The server will execute the service routine and send the result to the client. The client node should wait until the server responds with the results.
- Bags: Bags are a useful utility in ROS for the recording and playback of ROS topics. While working on robots, there may be some situations where we need to work without actual hardware. Using rosbag, we can record sensor data and can copy the bag file to other computers to inspect data by playing it back.

1.3.c ROS Community Level

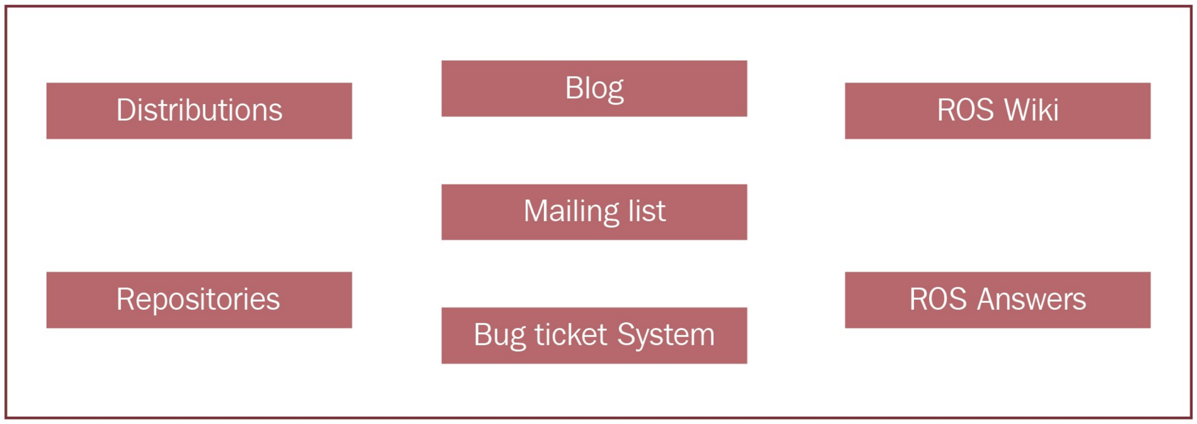


Figure 1.7 ROS Computation Graph Level

- Distributions: ROS distributions are versioned collections of ROS packages, like Linux distribution.
- Repositories: ROS-related packages and files depend on a version-control system (VCS) such as Git, SVN, and Mercurial, using which developers around the world can contribute to the packages.
- ROS Wiki: The ROS community wiki is the knowledge center of ROS, in which anyone can create documentation for their packages. You can find standard documentation and tutorials about ROS from the ROS wiki.
- ROS Answers: The ROS Answers website is the Stack Overflow of ROS. Users can ask questions regarding ROS and related areas.
- Mailing lists: Subscribing to the ROS mailing lists enables users to get new updates regarding ROS packages and gives them a place to ask questions about ROS.
- Blog: The ROS blog provides regular updates about the ROS community with photos and videos.

1.4 ROS Communication

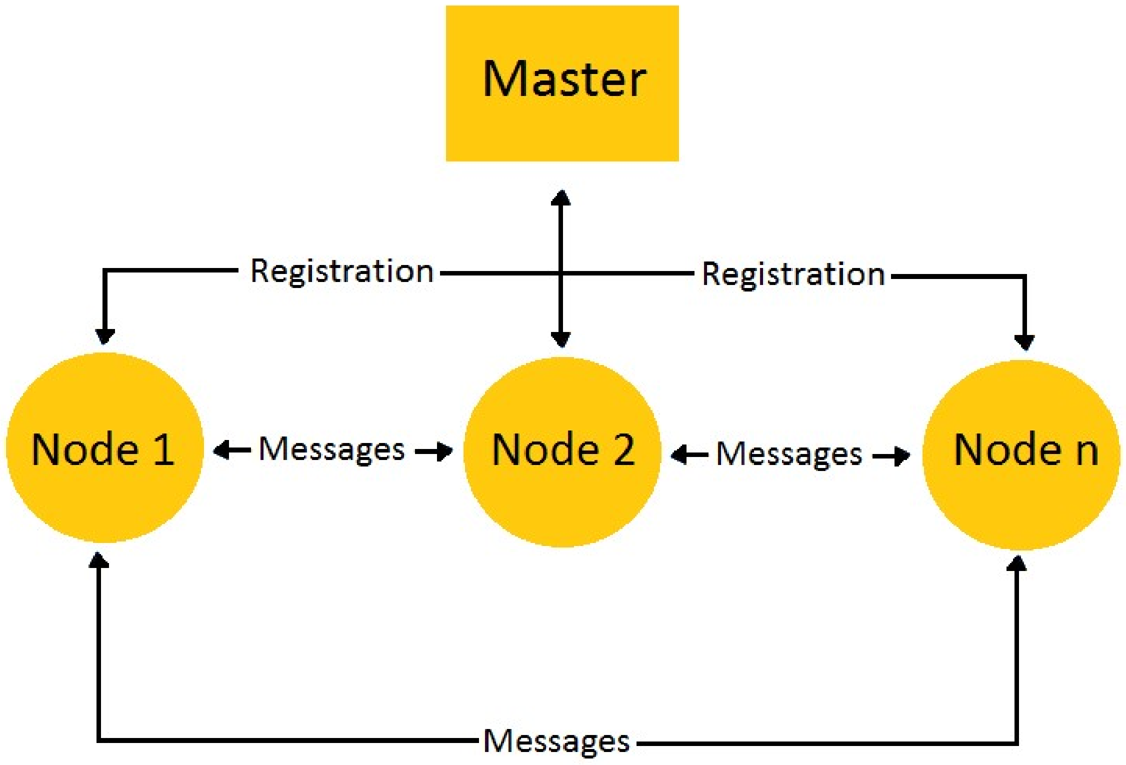


Figure 1.8 ROS Communication graph

- Before running any nodes in ROS, we should start the ROS Master. After it has been started, it will wait for nodes. When the publisher node starts running, it will first connect to the ROS Master and exchange the publishing topic details with the master. This includes topic name, message type, and publishing node URI. The URI of the master is a global value, and all nodes can connect to it. The master maintains tables of the publisher connected to it. Whenever a publisher's details change, the table updates automatically.
- When we start the subscriber node, it will connect to the master and exchange the details of the node, such as the topic going to be subscribed to, its message type, and the node URI. The master also maintains a table of subscribers as the publisher.
- Whenever there is a subscriber and publisher for the same topic, the master node will exchange the publisher URI with the subscriber. This will help both nodes connect with each other and exchange data. After they've connected with each other, there is no role for the master. The data is not flowing through the master; instead, the nodes are interconnected and exchange messages.

1.5 ROS Tools

ROS has a variety of GUI and command-line tools to inspect and debug messages.

- 3D Visualization: RVIZ is a 3D visualization environment that lets you combine sensor data, robot model, and other 3D data into a combined view. Also, it can send 3D markers into RVIZ from written software.

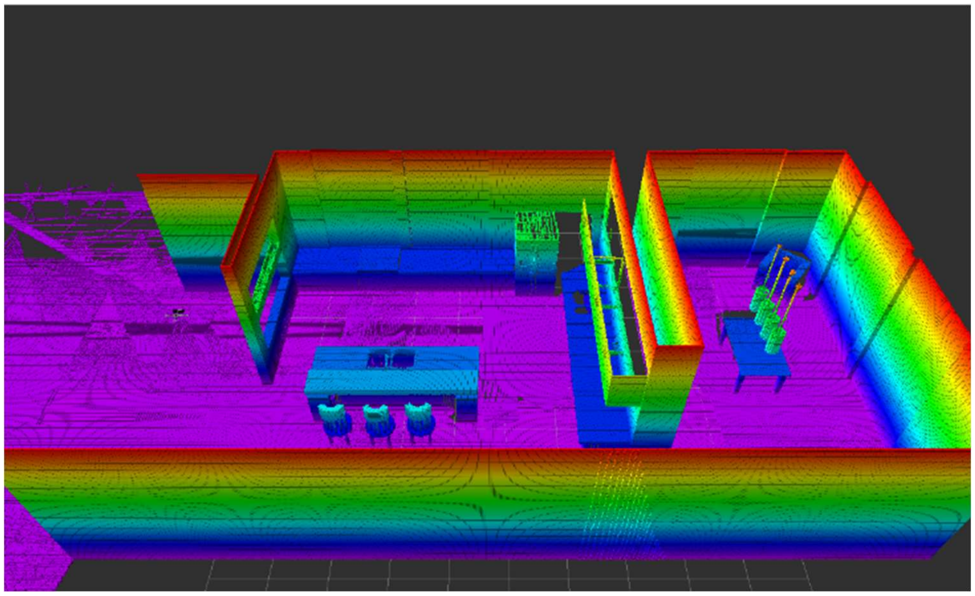


Figure 1.9 RVIZ 3D Visualization

- Foxglove Studio: Visualize and debug your ROS data in one integrated development environment, available as both a web and desktop app (macOS, Linux, Windows).

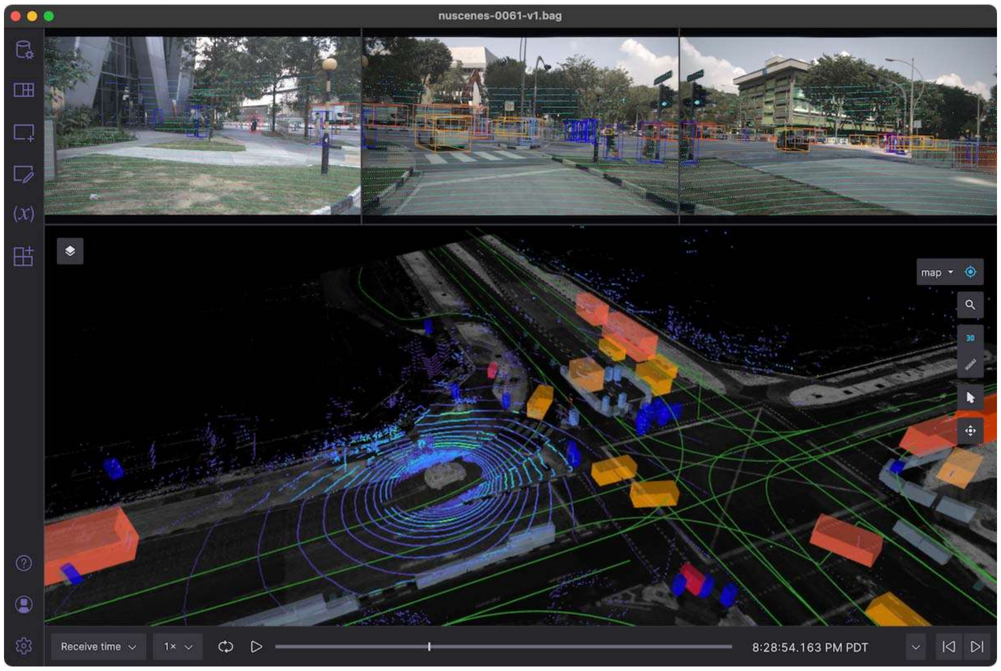


Figure 1.10 Foxglove Studio

- <u>Webviz:</u> A browser-based tool to look at data and ROS bag files, including mixed with live video.

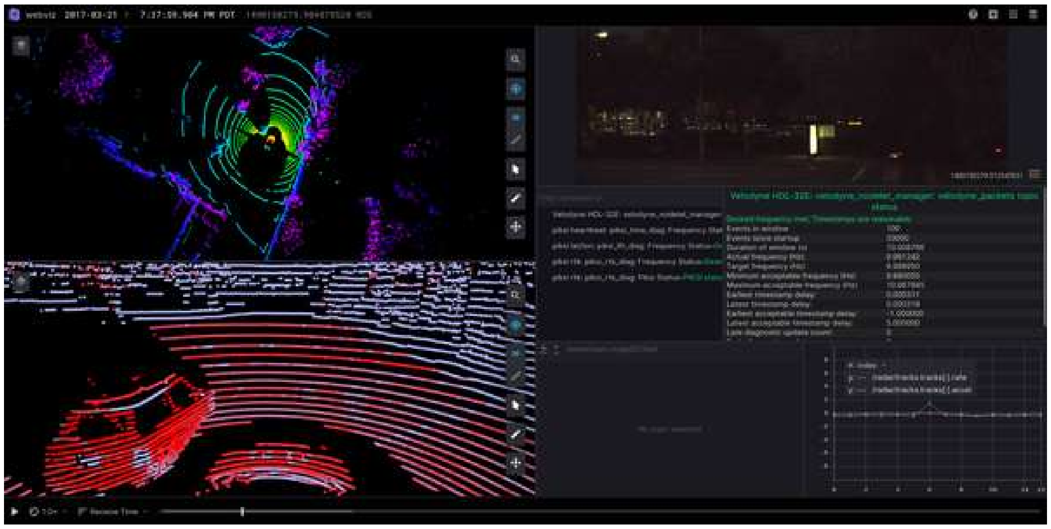


Figure 1.11 Webviz

- <u>Live plotting: rqt_plot</u> lets you visualize scalar data published to ROS topics.

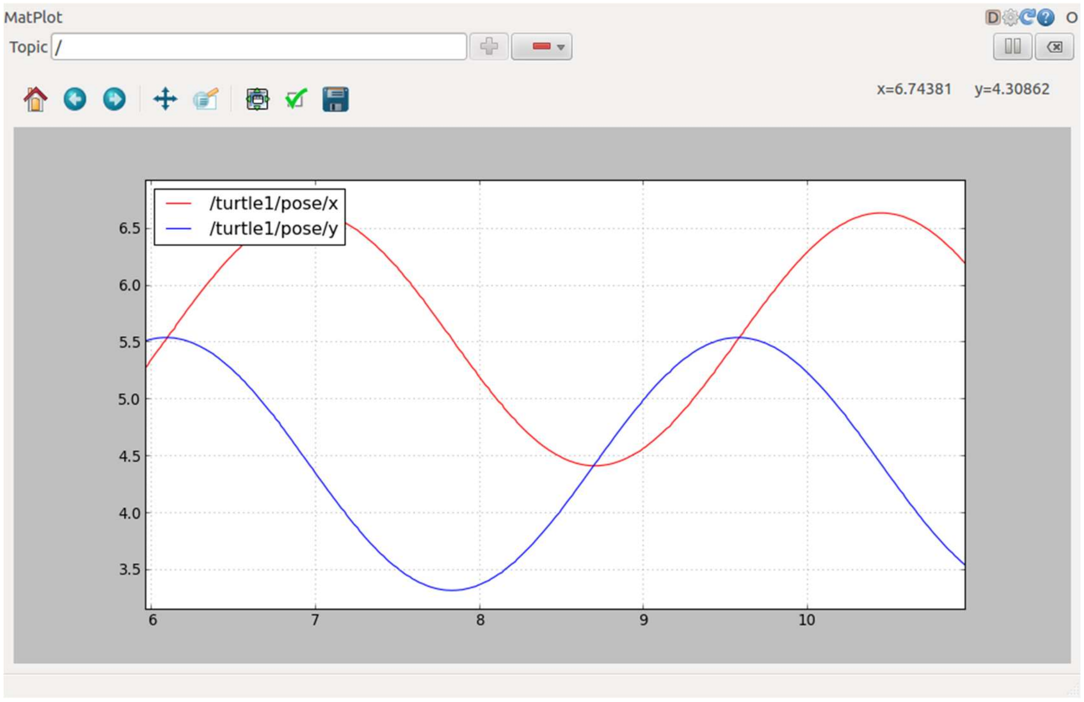
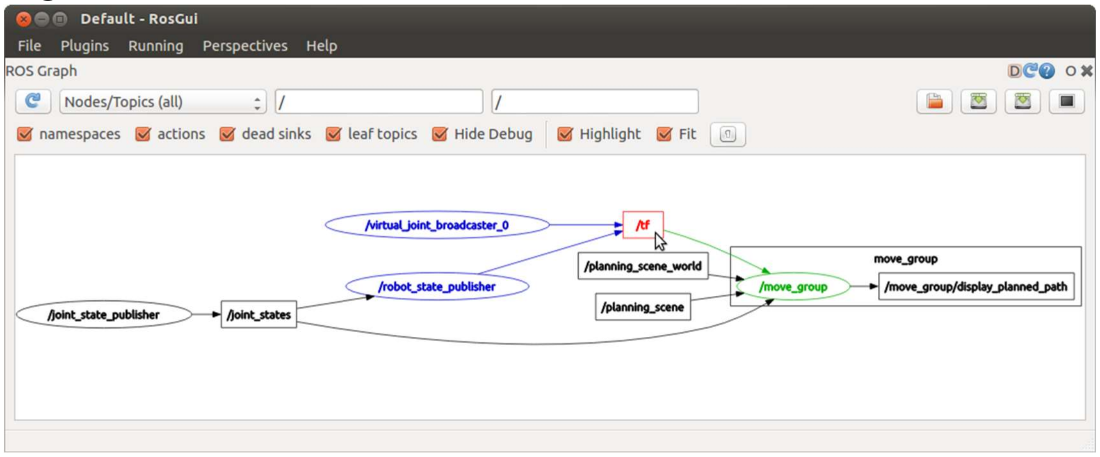


Figure 1.12 rqt_plot Live plotting

- <u>System visualization: rqt_graph</u> displays a visual graph of the processes running in ROS and their connections.



Nodes/Topics (all) | / | /

✓ namespaces ✓ actions ✓ dead sinks ✓ leaf topics ✓ Hide Debug ✓ Highlight ✓ Fit

Figure 1.13 rqt_graph System visualization

- <u>Command Line Tools</u>

Group	Command	Function
Running ROS Systems	roslaunch	Launching/configuring multiple programs
	rosrun	Run a single program
	roscore	Bring up the core system
Interacting with and debugging running system	rostopic	Topics debugging
	rosservice	Services debugging
	rosnode	Nodes debugging
	rosparam	Parameters debugging
	rosmasg	Messages debugging
	rossrv	Services debugging
	roswtf	General debugging
Install, build and filesystem tools	rosmake	Build
	rosinstall	Install from source
	roslocate	Searching for packages/stacks
	rosdep	Install third-party libraries
	rospack	Packages
	rosstack	Stacks

Table 1.11 ROS tools command line