ROS 2 from scratch – 2024

*Pack Publishing*



# Part 1: Getting Started with ROS 2

# Chapter 1 - Introduction to ROS 2 – What Is ROS 2?

## 1.1 Terminology

## 1.2 What is ROS, when should we use it, and why?

Why ROS?

What is ROS?

When to use ROS

## 1.3 ROS 1 versus ROS 2

A quick story of ROS, and how we got to ROS 2

Is ROS 1 dead already?

## 1.4 Prerequisites for starting with ROS 2

Knowledge prerequisites

Hardware and software

## 1.5 How to follow this book

# Chapter 2 - Installing and Setting Up ROS 2

## 2.1 Which ROS 2 distribution to choose

What is a ROS 2 distribution?

LTS and non-LTS distributions

How to choose a ROS distribution

## 2.2 Installing the OS (Ubuntu)

The relationship between ROS 2 and Ubuntu

Installing Ubuntu 24.04 natively with a dual boot

Installing Ubuntu 24.04 on a VM

## 2.3 Installing ROS 2 on Ubuntu

Setting the locale

Setting up the sources

Installing ROS 2 packages

## 2.4 Setting up the environment for ROS 2

Sourcing ROS 2 in the environment

Adding the source line to the .bashrc file

## 2.5 Extra tools for ROS development 29

Visual Studio Code

The Terminal and other tools

## 2.6 Summary

# Chapter 3 - Uncovering ROS 2 Core Concepts

## 3.1 Running your first node

Starting a node from the terminal with ros2 run

Introspecting the nodes with rqt\_graph

Running a 2D robot simulation

Recap – nodes

## 3.2 Topics

Running a topic publisher and subscriber

A name and an interface (data type)

More experimentation with topics

Recap – topics

## 3.3 Services

Running a service server and client

A name and an interface (data type)

Sending a request from the terminal

More experimentation with services

Recap – services

## 3.4 Actions

Running an action server

A name and an interface (data type)

Sending a goal from the terminal

Recap – actions

## 3.5 Parameters

Getting the parameters for a node

Setting up a parameter value for a node

Recap – parameters

## 3.6 Launch files

Starting a launch file

Recap – launch files

## 3.7 Summary

# Part 2: Developing with ROS 2 – Python and C++

# Chapter 4. Writing and Building a ROS 2 Node

## 4.1 Technical requirements

## 4.2 Creating and setting up a ROS 2 working

Creating a workspace

Building the workspace

Sourcing the workspace

## 4.3 Creating a package

What is a ROS 2 package?

Creating a Python package

Creating a C++ package

Building a package

How are nodes organized in a package?

## 4.4 Creating a Python node

Creating a file for the node

Writing a minimal ROS 2 Python node

Building the node

Running the node

Improving the node – timer and callback

## 4.5 Creating a C++ node

Writing a C++ node

Building and running the node

## 4.6 Node template for Python and C++ nodes

Template for a Python node

Template for a C++ node

## 4.7 Introspecting your nodes

ros2 node command line

Changing the node name at run time

# Chapter 5 - Topics – Sending and Receiving Messages between Nodes

## 5.1 What is a ROS 2 topic?

A publisher and a subscriber

Multiple publishers and subscribers

Multiple publishers and subscribers inside one node

Wrapping things up

## 5.2 Writing a topic publisher

Writing a Python publisher

Writing a C++ publisher

## 5.3 Writing a topic subscriber

Writing a Python subscriber

Writing a C++ subscriber

Running the Python and C++ nodes together

## 5.4 Additional tools to handle topics

Introspecting topics with rqt\_graph

The ros2 topic command line

Changing a topic name at runtime

Replaying topic data with bags

## 5.5 Creating a custom interface for a topic

Using existing interfaces

Creating a new topic interface

## 5.6 Topic challenge – closed-loop control

Challenge

Solution

## 5.7 Summary

# Chapter 6 - Services – Client/Server Interaction between Nodes

## 6.1 What is a ROS 2 service?

A server and a client

Multiple clients for one service

Another service example with robotics

Wrapping things up

## 6.2 Creating a custom service interface

Finding an existing interface for our service

Creating a new service interface

## 6.3 Writing a service server

Writing a Python service server

Writing a C++ service server

## 6.4 Writing a service client

Writing a Python service client

Running the client and server nodes together

Writing a C++ service client

## 6.5 Additional tools to handle services

Listing and introspecting services

Sending a service request

Changing a service name at runtime

## 6.6 Service challenge – client and server

Challenge

Solution

## 6.7 Summary

# 7. Actions – When Services Are Not Enough

## 7.1 What is a ROS 2 action?

Why actions?

How do actions work?

Wrapping things up

## 7.2 Creating a custom action interface

Defining the application and the interface we need

Creating a new action interface

## 7.3 Writing an action server

Writing a Python action server

Writing a C++ action server

## 7.4 Writing an action client

Writing a Python action client

Creating an action client

Writing a C++ action client

## 7.5 Taking advantage of all the action mechanisms

Adding the feedback mechanism

Adding the cancel mechanism

## 7.6 Additional tools to handle actions

Listing and introspecting actions

Sending a goal from the Terminal

Topics and services inside actions

## 7.7 Summary

# Chapter 8 - Parameters – Making Nodes More Dynamic

## 8.1 What is a ROS 2 parameter?

Why parameters?

Example of a node with parameters

ROS 2 parameters – wrapping things up

## 8.2 Using parameters in your nodes

Declaring, getting, and using parameters with Python

Providing parameters at runtime

Parameters with C++

## 8.3 Storing parameters in YAML files

Loading parameters from a YAML file

Parameters for multiple nodes

Recapping all parameters’ data types

## 8.4 Additional tools for handling parameters

Getting parameters’ values from the terminal

Exporting parameters into YAML

Setting a parameter’s value from the terminal

Parameter services

## 8.5 Updating parameters with parameter callbacks

Python parameter callback

C++ parameter callback

## 8.6 Parameter challenge

Challenge

Solution

## 8.7 Summary

# Chapter 9 - Launch Files – Starting All Your Nodes at Once

## 9.1 What is a ROS 2 launch file?

Why launch files?

Example of a launch file with seven nodes

## 9.2 Creating and installing an XML launch file

Setting up a package for launch files

Writing an XML launch file

Installing and starting a launch file

## 9.3 Creating a Python launch file – XML or Python for launch files?

Writing a Python launch file

XML versus Python for launch files

## 9.4 Configuring nodes inside a launch file

Renaming nodes and communications

Parameters in a launch file

Namespaces

## 9.5 Launch file challenge

Challenge

Solution

## 9.6 Summary

# Part 3: Creating and Simulating a Custom Robot with ROS 2

# Chapter 10 - Discovering TFs with RViz

## 10.1 Technical requirements

## 10.2 Visualizing a robot model in RViz

Installation and setup

Starting RViz with a robot model

## 10.3 What are TFs?

Links

TFs

## 10.4 Relationship between TFs

Parent and child

The /tf topic

Visualizing the TF tree

## 10.5 What problem are we trying to solve with TFs?

What we want to achieve

How to compute TFs

## 10.6 Summary

# Chapter 11 - Creating a URDF for a Robot 253

## 11.1 Creating a URDF with a link

Setting up a URDF file

Creating a link

Customizing the link visual

## 11.2 The process of assembling links and joints

Step 1 – adding a second link

Step 2 – adding a joint

Step 3 – fixing the joint origin

Step 4 – setting up the joint type

Step 5 – fixing the visual origin

Recap – the process to follow every time

## 11.3 Writing a URDF for a mobile robot

What we want to achieve

Adding the wheels

Adding the caster wheel

Extra link – base footprint

## 11.4 Improving the URDF with Xacro

Making a URDF file compatible with Xacro

Xacro properties

Xacro macros

Including a Xacro file in another file

## 11.5 Summary

# Chapter 12 - Publishing TFs and Packaging the URDF

## 12.1 Understanding how to publish TFs with our URDF

The robot\_state\_publisher node

Inputs for the robot\_state\_publisher

Recap – how to publish TFs

## 12.2 Starting all nodes from the terminal

Publishing the TFs from the terminal

Visualizing the robot model in RViz

## 12.3 Creating a package to install the URDF

Adding a new workspace

Creating a \_description package

Installing the URDF and other files

## 12.4 Writing a launch file to publish TFs and visualize the robot 295

The XML launch file 295

The Python launch file 298

## 12.5 Summary

# Chapter 13 - Simulating a Robot in Gazebo

## 13.1 Technical requirements

## 13.2 How Gazebo works

Clarifying – Gazebo versus RViz

Starting Gazebo

How Gazebo works with ROS 2

## 13.3 Adapting the URDF for Gazebo

Inertial tags

Collision tags

## 13.4 Spawning the robot in Gazebo

Spawning the robot from the terminal

Spawning the robot from a launch file

## 13.5 Controlling the robot in Gazebo

What do we need to do?

Adding Gazebo systems

Bridging Gazebo and ROS 2 communications

Testing the robot

## 13.6 Summary

# Chapter 14 - Going Further – What To Do Next

## 14.1 ROS 2 roadmap – exploration phase

Common stacks and frameworks

More exploration topics

## 14.2 Learning for a specific goal

What to learn for a project?

What to learn to get a job?

# Summary

# Other Books You May Enjoy