

Table of Contents

Preface

xxiii

Part 1: ROS 2 Programming and Simulation 1

Chapter 1: Introduction to ROS 2 3

Understanding ROS as a robotics framework 4

What is ROS? • 4

Important features of ROS • 5

ROS equation • 7

ROS development timeline • 8

Comparing ROS 1 and ROS 2 architecture 9

OS layer • 10

Middleware layer • 10

Application layer • 11

Feature comparison between ROS 1 and ROS 2 • 12

Why should you migrate from ROS 1 to ROS 2? 13

Diving into DDS in ROS 2 13

What is DDS? • 14

Components DDS standard • 15

Platform layer • 15

Middleware layer • 16

Application layer • 18

How DDS works? • 18	
Exploring a workflow for sending and receiving data using DDS • 20	
List of DDS vendors • 22	
Dissecting some important ROS 2 layers	24
RMW layer (ROS middleware abstraction interface) • 24	
RCL layer and client libraries • 26	
ROS 2 application layer • 26	
Summary	28
References	28
Chapter 2: Getting Started with ROS 2 Programming	31
Technical requirements	32
Mastering ROS 2 installation	32
Installing Ubuntu 24.04 LTS and ROS 2 Jazzy • 32	
<i>Installing Ubuntu 24.04 LTS • 33</i>	
<i>Installing ROS 2 Jazzy on Ubuntu 24.04 LTS • 35</i>	
Installing ROS 2 Jazzy on Windows 11/10 • 41	
Installing ROS 2 Jazzy on macOS • 41	
Installing ROS 2 Jazzy in Docker • 42	
<i>Installing Docker and the NVIDIA Container Toolkit • 43</i>	
<i>Running Docker with ROS 2 Jazzy • 44</i>	
<i>Dockerfile for ROS 2 Jazzy • 48</i>	
<i>Enabling GUI in a ROS 2 Jazzy container • 50</i>	
<i>Building the Docker image • 50</i>	
<i>Docker Compose with ROS 2 Jazzy • 54</i>	
<i>Installing ROS 2 Jazzy in robots • 56</i>	
Mastering ROS 2 tools and concepts	57
How to run ROS 2 nodes/apps in your robot • 57	
What is a ROS 2 package? • 59	
What is a ROS node? • 60	
What is a ROS topic? • 61	

What is a ROS computation graph? • 63	
What is a ROS message? • 65	
What is a ROS service and how does Turtlesim work? • 66	
Summary	71
References	72
Chapter 3: Implementing ROS 2 Concepts	75
Technical requirements	76
What is a ROS 2 action?	76
What is a ROS parameter?	78
What is a ROS 2 launch file?	80
Building a ROS 2 package	81
Creating a ROS 2 workspace • 82	
Building a workspace • 82	
Introduction to ROS 2 client libraries	84
Diving into ROS 2 nodes	85
Different types of ROS 2 nodes • 85	
<i>Standard (non-composable) nodes • 85</i>	
<i>Composable/component nodes • 86</i>	
<i>Lifecycle nodes (managed nodes) • 86</i>	
How do ROS 2 nodes communicate? • 88	
Diving into QoS in ROS 2 nodes • 89	
Implementing ROS 2 topics in C++	91
Creating a custom interface ROS 2 package • 92	
<i>Creating a ROS 2 package for a custom interface • 92</i>	
<i>Defining the custom message • 92</i>	
<i>Editing CMakeLists.txt to add message generation • 93</i>	
<i>Editing package.xml to add dependencies. • 93</i>	
<i>Building the interface package • 94</i>	

Creating the main ROS 2 package • 94	
<i>Creating the package • 94</i>	
<i>Creating the src directory for a C++ node • 94</i>	
<i>What does a publisher node do? • 95</i>	
<i>What does the subscriber node do? • 98</i>	
<i>Editing CMakeLists.txt to build the node • 99</i>	
<i>Building the package • 100</i>	
<i>Running the nodes • 101</i>	
Implementing ROS 2 parameters in C++ 102	
Declaring ROS 2 parameters • 102	
Reading values of ROS 2 parameters • 102	
Setting ROS 2 parameters • 103	
Editing CMakeLists.txt • 103	
Starting a ROS 2 node with parameters • 103	
Implementing ROS 2 services in C++ 105	
Adding a custom service interface • 105	
Editing CMakeLists.txt • 105	
ROS 2 service server • 106	
ROS 2 service client • 106	
Building server and client nodes • 107	
Running nodes • 108	
Implementing a ROS 2 action in C++ 108	
Creating a custom action interface for the ROS 2 action and server • 109	
Editing CMakeLists.txt and package.xml • 109	
Writing a ROS 2 action server • 109	
Writing the ROS 2 action client • 111	
Building nodes • 112	
Running nodes • 112	
Writing launch files for nodes • 113	

Configuring ROS 2 DDS/Zenoh and ROS_DOMAIN_ID	113
Configuring ROS 2 DDS • 113	
Setting ROS_DOMAIN_ID • 114	
Setting up ROS 2 Jazzy development in Docker • 115	
Setting up ROS 2 Development in Docker Compose • 116	
ROS 2 Development using VS Code Dev Container • 117	
Summary	117
References	117

Chapter 4: Working with Robot 3D Modeling in ROS 2 119

Technical requirements	120
Introduction to robot modeling in ROS 2	120
What is robot modeling? • 120	
Why is robot modeling important? • 122	
Robot modeling tools • 122	
Introduction to URDF and Xacro	124
Components of URDF/Xacro • 125	
<robot> tag • 126	
<link> tag • 127	
<joint> tag • 129	
Building our first URDF model • 131	
Verifying URDF files • 132	
Visualizing URDF links and joints • 132	
Visualizing URDF in RViz • 134	
Parsing a URDF file • 135	
Working with the Xacro model • 136	
Defining and calling a macro in a Xacro file • 138	
Constants and variables in Xacro • 139	
Math equations in Xacro • 139	
Writing the ROS 2 launch file for visualizing a URDF/Xacro file • 140	

Exporting a 3D CAD robot model to URDF	140
Installing Autodesk Fusion 360	142
Installing the Fusion 360 to URDF plugin	142
Important design practices for the URDF Export plugin	143
Correct the Fusion 360 coordinate system • 144	
Defining links of the robot • 145	
Defining joints of the robot • 145	
Defining a component of the robot • 145	
Modeling a robot in Fusion 360	146
Setting the design plane for export • 146	
Sketching the robot base • 147	
<i>Adding wheels to the robot base • 149</i>	
<i>Moving the robot to ground level • 152</i>	
<i>Adding caster wheels to the base of the robot • 153</i>	
<i>Adding a LiDAR base and lidar to the robot • 156</i>	
<i>Adding a material type and color • 158</i>	
<i>Converting bodies to components • 158</i>	
<i>Adding joints to the robot • 159</i>	
Converting a Fusion 360 model to URDF for ROS 2 • 164	
Building a ROS 2 description package • 165	
Visualizing the robot in RViz • 165	
Visualizing URDF models of existing robots	166
Visualizing the Universal Robot model • 167	
Visualizing the TurtleBot 4 model • 167	
Visualizing the Leo Rover model • 168	
Official ROS 2 URDF tutorials • 169	
Setting up ROS 2 Jazzy URDF development in Docker • 170	
Setting up ROS 2 URDF Development in Docker Compose • 171	
Best practices of URDF/Xacro modeling	173
Summary	174
References	174

Technical requirements	178
Introduction to Gazebo Sim	178
Major milestones of the Gazebo Project • 180	
Features of Gazebo simulators • 180	
The architecture of Gazebo Sim • 182	
<i>Gazebo Sim server/backend</i> • 183	
<i>Gazebo client</i> • 184	
Installing and configuring Gazebo Sim • 185	
Simulating a robot using Gazebo Sim	189
How does Gazebo simulation work? • 191	
Simulating a robot using ROS 2 and Gazebo	192
Integration of ROS 2 and Gazebo Sim • 192	
Starting Gazebo from ROS 2 • 193	
Spawning a URDF model to Gazebo Sim • 194	
<i>Updating URDF tags for simulation</i> • 195	
<i>Creating the world file for simulation</i> • 195	
<i>Creating the configuration files</i> • 196	
<i>Creating a simulation launch file</i> • 196	
Launching the simulation in Gazebo • 196	
<i>Moving the robotic arm</i> • 197	
Simulation of differential drive robots • 198	
Modifying the Fusion 360-generated URDF package for simulation • 200	
<i>Correction of the Xacro file</i> • 201	
<i>Adding Gazebo Sim plugins/ROS 2 controllers</i> • 202	
<i>Updating Gazebo-ROS 2 bridge topics</i> • 202	
Gazebo simulation of existing robots • 203	
Simulation using Docker • 204	
<i>Simulation using Docker Compose</i> • 206	
<i>Simulation using VS Code Dev Containers</i> • 207	

Introduction to the Webots simulator	207
Key features of Webots • 208	
Installing Webots and the ROS 2 interface • 208	
Introduction to Isaac Sim	210
Installing NVIDIA Isaac Sim in Ubuntu 24.04 LTS with ROS 2 Jazzy • 210	
Summary	211
References	212

Part 2: ROS 2 Applications: Navigation, Manipulation, and Control 215

Chapter 6: Controlling Robots Using the ros2_control Package	217
Technical requirements	217
Understanding the ros2_control framework	218
Understanding the ros2_control framework architecture	220
The ros2_control framework documentation	221
Interfacing with simulated controllers using Gazebo	222
Step 1 – Creating the ROS 2 package • 222	
Step 2 – Creating a robot model • 223	
Step 3 – Adding controllers to the robot • 223	
Step 4 – Creating launch and configuration files • 225	
Step 5 – Installing the necessary files • 228	
Interacting with the ros2_control framework using the CLI	228
Implementing a new controller	230
Step 1 – Creating the ros2 package • 230	
Step 2 – Writing the source code of the controller • 231	
Step 3 – Compiling the controller plugin • 238	
Step 4 – Configuring the controller • 239	
Step 5 – Starting the controller • 239	
ros2_control on real robots	240

Summary	241
References	242

Chapter 7: Implementing ROS 2 Applications Using BehaviorTree.CPP 243

Technical requirements	244
Understanding BTs	244
Basic concepts of BTs • 246	
BT nodes • 247	
Introducing the BehaviorTree.CPP library	248
Implementing your first BT	250
Step 1 – Defining the tree model • 250	
Step 2 – Creating the tree executor • 251	
Step 3 – Compiling and installing the necessary files • 254	
Step 4 – Creating a convenient launch file • 255	
Integrating ROS 2 and BTs	256
Creating a tree using Groot 2 • 256	
Using the BT blackboard • 259	
Integrating BTs with ROS 2 actions	262
Summary	267
References	267

Chapter 8: ROS 2 Navigation Stack: Nav2 269

Technical requirements	270
Getting started with Nav2	270
What is the Nav2 project? • 270	
Milestones of the Nav2 project • 271	
Features of Nav2 • 271	
Nav2 architecture • 273	
Lifecycle Manager • 274	
BT Navigator Server • 274	
Planner Server • 275	

Controller Server • 275	
Behavior Server • 275	
Smoother Server • 276	
Route Server • 276	
Velocity Smoother and Collision Monitor • 276	
Waypoint Follower • 277	
How does Nav2 work? • 277	
Installing Nav2 on ROS 2 Jazzy • 279	
Nav2 demo using TurtleBot3 • 280	
Setting up Nav2 in Docker • 282	
Setting up Nav2 in Docker Compose • 283	
Nav2 using VS Code Dev Containers • 285	
Mapping using Slam Toolbox 285	
What is Slam Toolbox? • 285	
Installing Slam Toolbox in ROS 2 Jazzy • 287	
Mapping demo using Slam Toolbox • 287	
Configuring Nav2 and Slam Toolbox for your robot 290	
Robot prerequisites for Nav2 • 292	
ROS 2 navigation and mapping launch files • 293	
Launching navigation • 293	
Launching mapping • 294	
Launching navigation and mapping • 294	
ROS 2 navigation parameter files • 294	
Launching rosbot simulation and navigation • 294	
Case study: Nav2-based robots during COVID-19 296	
Developing Nav2-based applications using ROS 2 and C++ 297	
Running the Nav2 C++ application • 299	
Developing Nav2-based applications using BT and C++ • 300	
BT Navigator Server in Nav2 • 300	
BT in the Nav2 action client • 301	
Running a BT-based Nav2 action client • 302	

Summary	303
---------------	-----

References	304
------------------	-----

Chapter 9: Robot Manipulation Using MoveIt 2 **307**

Technical requirements	308
------------------------------	-----

Introduction to MoveIt 2	308
--------------------------------	-----

MoveIt 2 system architecture •	308
--------------------------------	-----

MoveIt 2 planning pipeline •	309
------------------------------	-----

Preparing the robot model to use MoveIt 2	310
---	-----

Step 1 – Adding robot controllers •	311
-------------------------------------	-----

Step 2 – Enabling the controllers from the launch file •	312
--	-----

Configuring new Robots for MoveIt 2	313
---	-----

Testing MoveIt 2 using Rviz2	318
------------------------------------	-----

Motion planning using the move_group ROS 2 wrapper	320
--	-----

Step 1 – Implementing the ROS 2 node •	321
--	-----

Step 2 – Creating the launch file •	322
-------------------------------------	-----

Step 3 – Filling the CMakeFile.txt •	322
--------------------------------------	-----

Step 4 – Compiling and executing the planning node •	323
--	-----

Cartesian space planning •	323
----------------------------	-----

Planning with obstacles	328
-------------------------------	-----

Detecting obstacles using a depth sensor	331
--	-----

Summary	335
---------------	-----

Chapter 10: Working with ROS 2 and Perception Stack **337**

Technical requirements	338
------------------------------	-----

Integrating robotics and computer vision	338
--	-----

Integrating the camera with ROS 2 •	339
-------------------------------------	-----

Getting started with OpenCV – ROS 2 integration •	342
---	-----

Getting started with camera calibration •	347
---	-----

Getting started with depth sensors	350
--	-----

Using NVIDIA ISAAC ROS to speed up image processing	356
ISAAC structure • 357	
ROS 2 integration and NITROS • 357	
Requirements • 358	
Starting the isaac_ros_apriltag node • 359	
Performing object classification using YOLOv8 and NVIDIA ISAAC • 361	
Summary	364
References	364

Part 3: Advanced Applications and Machine Learning 367

Chapter 11: Aerial Robotics and ROS 2 369

Technical requirements	369
Introducing aerial robotics	370
Leading companies in aerial robotics and their objectives • 370	
Comparison of ground and aerial systems • 371	
Understanding the hardware and software architecture of a UAV	372
Pixhawk autopilot and PX4 control stack • 375	
Simulating an aerial robot using Gazebo and the PX4 control stack • 379	
Interfacing ROS 2 and PX4 • 382	
Controlling a UAV using ROS 2	384
Engaging the OFFBOARD mode • 384	
Exploring the px4_ctrl.cpp file • 386	
Working with the node • 392	
Summary	395
References	395

Chapter 12: Designing and Programming a DIY Mobile Robot from Scratch 397

Technical requirements	398
------------------------------	-----

Understanding the DIY mobile robot	398
Installing Linux on the Raspberry Pi • 399	
Configuring the Raspberry Pi • 402	
Using the Raspberry Pi remotely • 402	
Advanced configuration of the Raspberry Pi • 404	
Choosing the robot frame • 404	
Mobile robot electronics and sensors • 405	
Understanding the electronic connection	406
Programming and configuring the DIY robot	408
Interfacing the robot's motors • 409	
Configuring the LiDAR sensor • 412	
Defining the robot model • 413	
Odometry calculation • 414	
Enabling autonomous navigation • 414	
Future improvements	416
Summary	417
References	417
 Chapter 13: Testing, Continuous Integration, and Continuous Deployment with ROS 2	 419
Technical requirements	420
Testing ROS 2 nodes using GTest	420
Exploring the GTest features and programming pipeline • 421	
Integrating GTest with ROS 2 for robust testing • 422	
Integrating ROS 2 APIs in the GTest framework	426
Exploring the content of the source files • 426	
Implementing automatic testing using GTest • 428	
Implementing CI/CD for ROS 2 packages	432
Using GitHub Actions for code compilation • 433	
Adding a status badge • 439	
Summary	440

Chapter 14: Interfacing Large Language Models with ROS 2 443

Technical requirements	443
LLMs for robotics	444
Integration of ROS 2 robots with LLM agents	445
Components of AI agents with ROS 2 •	446
How an AI agent works for ROS 2 •	448
Creating ROS 2 AI agents	448
ROS 2 AI-agent development using VS Code Dev Containers •	450
Building an AI agent for ROS 2 •	450
Configuring OpenAI API keys •	452
Architecture of ROS 2 AI agent node •	452
Running the basic ROS 2 AI agent •	454
<i>AI agent with ROS 2 tools •</i>	<i>458</i>
<i>AI agent for ROS 2 turtlesim •</i>	<i>459</i>
<i>AI agent for Nav2 •</i>	<i>461</i>
<i>AI agent for MoveIt2 •</i>	<i>464</i>
Popular ROS 2 AI agent projects	466
ROSA •	466
RAI •	466
ROS-LLM •	467
ROS 2 NanoLLM •	467
Summary	468
References	468

Chapter 15: ROS 2 and Deep Reinforcement Learning 471

Technical requirements	472
Dive into deep reinforcement learning	472
Basic concepts of reinforcement learning •	472
Transitioning from classic RL to DRL •	473
Application of DRL in robotics •	474

Popular DRL algorithms • 475	
<i>Value-based methods</i> • 475	
<i>Policy-based methods</i> • 475	
<i>Actor-critic methods</i> • 476	
Setting up Isaac Lab and Isaac Sim in Ubuntu 24.04	476
What is Isaac Sim and Isaac Lab? • 476	
Installing Isaac Sim and Isaac Lab in Ubuntu 24.04 • 479	
<i>Setting up Docker and the NVIDIA Container Toolkit</i> • 480	
<i>Installing Isaac Sim and Isaac Lab</i> • 480	
Training and testing robots using Isaac Lab	481
Isaac Lab architecture • 481	
Training and testing an existing RL environment in Isaac Lab • 483	
<i>Training UR-10 using Isaac Lab</i> • 484	
<i>Testing UR-10 using Isaac Lab</i> • 486	
<i>Training and testing ANYmal robot navigation</i> • 487	
<i>Training and testing of an H1 robot for a locomotion task</i> • 489	
Deploying a trained RL model to a real robot	490
Deploying an RL model in the Spot robot from Boston Dynamics	491
Summary	492
References	493
Chapter 16: Implementing ROS 2 Visualization and Simulation Plugins	495
Technical requirements	496
Introducing the ROS 2 plugin	496
Understanding pluginlib in ROS 2 • 497	
Glancing at steps for creating, compiling, and loading plugins • 498	
Creating a plugin for rqt	498
Creating a Qt user interface • 499	
Implementing the plugin source • 501	
Adding compilation and configuration files • 506	

Developing a Gazebo plugin	508
Writing the source code • 508	
Compiling and executing the Gazebo plugin • 511	
Developing plugins for RViz2	512
Writing the source code • 513	
Adding and configuring the RViz2 plugin • 518	
Summary	520
Other Books You May Enjoy	525
Index	529

Architecture of ROS 2 AI agent node • 452	
Running the basic ROS 2 AI agent • 454	
AI agent with ROS 2 tools • 458	
AI agent for ROS 2 turtlesim • 459	
AI agent for Nav2 • 461	
AI agent for MoveIt2 • 464	
Deploying a trained RL model to a real robot	465
Deploying an RL model in the Spot robot from Boston Dynamics	466
Summary	467
References	467
Chapter 16: Implementing ROS 2 Visualization and Simulation Plugins	468
Summary	468
References	468
Chapter 15: ROS 2 and Deep Reinforcement Learning	469
Technical requirements	470
Dive into deep reinforcement learning	471
Basic concepts of reinforcement learning • 472	
Transitioning from classic RL to DRL • 473	
Application of DRL in robotics • 474	