Mastering ROS 2 for Robotics Programming, Simulation, and Robot Development

Course Description

This course covers ROS 2 programming, robot modeling, simulation, navigation, manipulation, perception, aerial robotics, mobile robot development, testing, continuous integration and deployment, large language model integration, deep reinforcement learning, and ROS 2 visualization and simulation plugins. Participants will work with ROS 2 Jazzy, DDS, C++, URDF, Xacro, Gazebo Sim, Webots, Isaac Sim, ros2_control, BehaviorTree.CPP, Nav2, MoveIt 2, OpenCV, NVIDIA Isaac ROS, PX4, Raspberry Pi, GTest, GitHub Actions, Isaac Lab, rqt, and RViz2.

Audience Profile

This course is suitable for:

- Robotics developers
- ROS and ROS 2 developers
- C++ developers working on robotics applications
- Robot simulation engineers
- Autonomous navigation and manipulation developers
- Computer vision and perception developers
- UAV and mobile robot developers
- Robotics researchers and engineers

Prerequisites

Participants should have the technical environment and supporting hardware or software required for the relevant modules, including ROS 2 Jazzy, Ubuntu 24.04 LTS, Windows 10/11 or macOS, Docker, C++, Gazebo Sim, Webots, NVIDIA Isaac Sim, Raspberry Pi, robotics sensors, and robot hardware where applicable.

Course Objectives

By the end of this course, participants will be able to:

- Understand ROS 2 architecture, DDS, middleware, client libraries, and application layers.
- Install and configure ROS 2 Jazzy across supported operating systems and containers.

- Develop ROS 2 nodes, topics, services, actions, parameters, interfaces, and launch files.
- Design and visualize robot models using URDF, Xacro, Fusion 360, and RViz.
- Simulate robotic systems using Gazebo Sim, Webots, and Isaac Sim.
- Control simulated and physical robots using the ros2_control framework.
- Develop applications using BehaviorTree.CPP, Nav2, MoveIt 2, and the ROS 2 perception stack.
- Implement aerial robotics and autonomous mobile robot applications.
- Test ROS 2 nodes and implement continuous integration and deployment.
- Integrate large language models and AI agents with ROS 2.
- Train and deploy robots using deep reinforcement learning.
- Develop plugins for rqt, Gazebo, and RViz2.

Table of Contents

Part 1: ROS 2 Programming and Simulation

Module 1: Introduction to ROS 2

- Understanding ROS as a Robotics Framework
  - What Is ROS?
  - Important Features of ROS
  - ROS Equation
  - ROS Development Timeline
- Comparing ROS 1 and ROS 2 Architecture
  - OS Layer
  - Middleware Layer
  - Application Layer
  - Feature Comparison Between ROS 1 and ROS 2
- Why You Should Migrate from ROS 1 to ROS 2
- Diving into DDS in ROS 2

- What Is DDS?
- Components of the DDS Standard
- Platform Layer
- Middleware Layer
- Application Layer
- How DDS Works
- Workflow for Sending and Receiving Data Using DDS
- DDS Vendors

- Dissecting Important ROS 2 Layers
  - RMW Layer—ROS Middleware Abstraction Interface
  - RCL Layer and Client Libraries
  - ROS 2 Application Layer

Module 2: Getting Started with ROS 2 Programming

- Mastering ROS 2 Installation
  - Installing Ubuntu 24.04 LTS and ROS 2 Jazzy
  - Installing Ubuntu 24.04 LTS
  - Installing ROS 2 Jazzy on Ubuntu 24.04 LTS
  - Installing ROS 2 Jazzy on Windows 11/10
  - Installing ROS 2 Jazzy on macOS
  - Installing ROS 2 Jazzy in Docker
  - Installing Docker and the NVIDIA Container Toolkit
  - Running Docker with ROS 2 Jazzy
  - Dockerfile for ROS 2 Jazzy
  - Enabling GUI in a ROS 2 Jazzy Container
  - Building the Docker Image

    - Docker Compose with ROS 2 Jazzy
    - Installing ROS 2 Jazzy in Robots
- Mastering ROS 2 Tools and Concepts
    - Running ROS 2 Nodes and Applications in a Robot
    - ROS 2 Packages
    - ROS Nodes
    - ROS Topics
    - ROS Computation Graph
    - ROS Messages
    - ROS Services and Turtlesim

Module 3: Implementing ROS 2 Concepts

- ROS 2 Actions
- ROS Parameters
- ROS 2 Launch Files
- Building a ROS 2 Package
    - Creating a ROS 2 Workspace
    - Building a Workspace
- Introduction to ROS 2 Client Libraries
- Diving into ROS 2 Nodes
    - Different Types of ROS 2 Nodes
    - Standard or Non-composable Nodes
    - Composable or Component Nodes
    - Lifecycle or Managed Nodes
    - Communication Between ROS 2 Nodes
    - QoS in ROS 2 Nodes

- Implementing ROS 2 Topics in C++
    - Creating a Custom Interface ROS 2 Package
    - Creating a ROS 2 Package for a Custom Interface
    - Defining the Custom Message
    - Editing CMakeLists.txt to Add Message Generation
    - Editing package.xml to Add Dependencies
    - Building the Interface Package
- Creating the Main ROS 2 Package
    - Creating the Package
    - Creating the src Directory for a C++ Node
    - Publisher Node
    - Subscriber Node
    - Editing CMakeLists.txt to Build the Node
    - Building the Package
    - Running the Nodes
- Implementing ROS 2 Parameters in C++
    - Declaring ROS 2 Parameters
    - Reading ROS 2 Parameter Values
    - Setting ROS 2 Parameters
    - Editing CMakeLists.txt
    - Starting a ROS 2 Node with Parameters
- Implementing ROS 2 Services in C++
    - Adding a Custom Service Interface
    - Editing CMakeLists.txt
    - ROS 2 Service Server

    - ROS 2 Service Client
    - Building Server and Client Nodes
    - Running Nodes
- Implementing a ROS 2 Action in C++
    - Creating a Custom Action Interface for the ROS 2 Action and Server
    - Editing CMakeLists.txt and package.xml
    - Writing a ROS 2 Action Server
    - Writing a ROS 2 Action Client
    - Building Nodes
    - Running Nodes
    - Writing Launch Files for Nodes
- Configuring ROS 2 DDS, Zenoh, and ROS_DOMAIN_ID
    - Configuring ROS 2 DDS
    - Setting ROS_DOMAIN_ID
    - Setting Up ROS 2 Jazzy Development in Docker
    - Setting Up ROS 2 Development in Docker Compose
    - ROS 2 Development Using VS Code Dev Containers

Module 4: Working with Robot 3D Modeling in ROS 2

- Introduction to Robot Modeling in ROS 2
    - Robot Modeling
    - Importance of Robot Modeling
    - Robot Modeling Tools
- Introduction to URDF and Xacro
    - Components of URDF and Xacro
    - <robot> Tag

  - <link> Tag
  - <joint> Tag
  - Building the First URDF Model
  - Verifying URDF Files
  - Visualizing URDF Links and Joints
  - Visualizing URDF in RViz
  - Parsing a URDF File
  - Working with the Xacro Model
  - Defining and Calling a Macro in a Xacro File
  - Constants and Variables in Xacro
  - Mathematical Equations in Xacro
  - Writing the ROS 2 Launch File for Visualizing a URDF or Xacro File
- Exporting a 3D CAD Robot Model to URDF
- Installing Autodesk Fusion 360
- Installing the Fusion 360 to URDF Plugin
- Important Design Practices for the URDF Export Plugin
  - Correcting the Fusion 360 Coordinate System
  - Defining Robot Links
  - Defining Robot Joints
  - Defining a Robot Component
- Modeling a Robot in Fusion 360
  - Setting the Design Plane for Export
  - Sketching the Robot Base
  - Adding Wheels to the Robot Base
  - Moving the Robot to Ground Level

    - Adding Caster Wheels to the Base
    - Adding a LiDAR Base and LiDAR
    - Adding a Material Type and Color
    - Converting Bodies to Components
    - Adding Joints to the Robot
    - Converting a Fusion 360 Model to URDF for ROS 2
    - Building a ROS 2 Description Package
    - Visualizing the Robot in RViz
- Visualizing URDF Models of Existing Robots
    - Universal Robot Model
    - TurtleBot 4 Model
    - Leo Rover Model
    - Official ROS 2 URDF Tutorials
    - Setting Up ROS 2 Jazzy URDF Development in Docker
    - Setting Up ROS 2 URDF Development in Docker Compose
- Best Practices for URDF and Xacro Modeling

Module 5: Simulating Robots in a Realistic Environment

- Introduction to Gazebo Sim
    - Major Milestones of the Gazebo Project
    - Features of Gazebo Simulators
    - Architecture of Gazebo Sim
    - Gazebo Sim Server and Backend
    - Gazebo Client
    - Installing and Configuring Gazebo Sim
- Simulating a Robot Using Gazebo Sim

  - How Gazebo Simulation Works
- Simulating a Robot Using ROS 2 and Gazebo
  - Integration of ROS 2 and Gazebo Sim
  - Starting Gazebo from ROS 2
  - Spawning a URDF Model in Gazebo Sim
  - Updating URDF Tags for Simulation
  - Creating the World File
  - Creating Configuration Files
  - Creating a Simulation Launch File
  - Launching the Simulation in Gazebo
  - Moving the Robotic Arm
  - Simulating Differential Drive Robots
  - Modifying the Fusion 360-Generated URDF Package for Simulation
  - Correcting the Xacro File
  - Adding Gazebo Sim Plugins and ROS 2 Controllers
  - Updating Gazebo–ROS 2 Bridge Topics
  - Gazebo Simulation of Existing Robots
  - Simulation Using Docker
  - Simulation Using Docker Compose
  - Simulation Using VS Code Dev Containers
- Introduction to the Webots Simulator
  - Key Features of Webots
  - Installing Webots and the ROS 2 Interface
- Introduction to Isaac Sim
  - Installing NVIDIA Isaac Sim on Ubuntu 24.04 LTS with ROS 2 Jazzy

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

Module 6: Controlling Robots Using the ros2_control Package

- Understanding the ros2_control Framework
- Understanding the ros2_control Framework Architecture
- ros2_control Framework Documentation
- Interfacing with Simulated Controllers Using Gazebo
    - Creating the ROS 2 Package
    - Creating a Robot Model
    - Adding Controllers to the Robot
    - Creating Launch and Configuration Files
    - Installing the Necessary Files
- Interacting with the ros2_control Framework Using the CLI
- Implementing a New Controller
    - Creating the ROS 2 Package
    - Writing the Controller Source Code
    - Compiling the Controller Plugin
    - Configuring the Controller
    - Starting the Controller
- ros2_control on Real Robots

Module 7: Implementing ROS 2 Applications Using BehaviorTree.CPP

- Understanding Behavior Trees
    - Basic Concepts of Behavior Trees
    - Behavior Tree Nodes
- Introducing the BehaviorTree.CPP Library
- Implementing the First Behavior Tree

    - Defining the Tree Model
    - Creating the Tree Executor
    - Compiling and Installing the Necessary Files
    - Creating a Launch File
- Integrating ROS 2 and Behavior Trees
    - Creating a Tree Using Groot 2
    - Using the Behavior Tree Blackboard
    - Integrating Behavior Trees with ROS 2 Actions

Module 8: ROS 2 Navigation Stack—Nav2

- Getting Started with Nav2
    - Nav2 Project
    - Milestones of the Nav2 Project
    - Features of Nav2
    - Nav2 Architecture
    - Lifecycle Manager
    - Behavior Tree Navigator Server
    - Planner Server
    - Controller Server
    - Behavior Server
    - Smoother Server
    - Route Server
    - Velocity Smoother and Collision Monitor
    - Waypoint Follower
    - How Nav2 Works
    - Installing Nav2 on ROS 2 Jazzy

    - Nav2 Demonstration Using TurtleBot3
    - Setting Up Nav2 in Docker
    - Setting Up Nav2 in Docker Compose
    - Nav2 Using VS Code Dev Containers
- Mapping Using Slam Toolbox
    - Slam Toolbox
    - Installing Slam Toolbox in ROS 2 Jazzy
    - Mapping Demonstration Using Slam Toolbox
- Configuring Nav2 and Slam Toolbox for a Robot
    - Robot Prerequisites for Nav2
    - ROS 2 Navigation and Mapping Launch Files
    - Launching Navigation
    - Launching Mapping
    - Launching Navigation and Mapping
    - ROS 2 Navigation Parameter Files
    - Launching ROSbot Simulation and Navigation
- Case Study: Nav2-Based Robots During COVID-19
- Developing Nav2-Based Applications Using ROS 2 and C++
    - Running the Nav2 C++ Application
- Developing Nav2-Based Applications Using Behavior Trees and C++
    - Behavior Tree Navigator Server in Nav2
    - Behavior Trees in the Nav2 Action Client
    - Running a Behavior Tree-Based Nav2 Action Client

Module 9: Robot Manipulation Using MoveIt 2

- Introduction to MoveIt 2

    - MoveIt 2 System Architecture
    - MoveIt 2 Planning Pipeline
- Preparing the Robot Model to Use MoveIt 2
    - Adding Robot Controllers
    - Enabling Controllers from the Launch File
- Configuring New Robots for MoveIt 2
- Testing MoveIt 2 Using RViz2
- Motion Planning Using the move_group ROS 2 Wrapper
    - Implementing the ROS 2 Node
    - Creating the Launch File
    - Completing CMakeLists.txt
    - Compiling and Executing the Planning Node
    - Cartesian Space Planning
- Planning with Obstacles
- Detecting Obstacles Using a Depth Sensor

Module 10: Working with ROS 2 and the Perception Stack

- Integrating Robotics and Computer Vision
    - Integrating a Camera with ROS 2
    - OpenCV and ROS 2 Integration
    - Camera Calibration
- Getting Started with Depth Sensors
- Using NVIDIA Isaac ROS to Accelerate Image Processing
    - Isaac Structure
    - ROS 2 Integration and NITROS
    - Requirements

- Starting the isaac_ros_apriltag Node
- Performing Object Classification Using YOLOv8 and NVIDIA Isaac

Part 3: Advanced Applications and Machine Learning

Module 11: Aerial Robotics and ROS 2

- Introducing Aerial Robotics
  - Leading Companies in Aerial Robotics and Their Objectives
  - Comparison of Ground and Aerial Systems
- Understanding the Hardware and Software Architecture of a UAV
  - Pixhawk Autopilot and PX4 Control Stack
  - Simulating an Aerial Robot Using Gazebo and the PX4 Control Stack
  - Interfacing ROS 2 and PX4
- Controlling a UAV Using ROS 2
  - Engaging OFFBOARD Mode
  - Exploring the px4_ctrl.cpp File
  - Working with the Node

Module 12: Designing and Programming a DIY Mobile Robot from Scratch

- Understanding the DIY Mobile Robot
  - Installing Linux on Raspberry Pi
  - Configuring Raspberry Pi
  - Using Raspberry Pi Remotely
  - Advanced Raspberry Pi Configuration
  - Choosing the Robot Frame
  - Mobile Robot Electronics and Sensors
- Understanding the Electronic Connections
- Programming and Configuring the DIY Robot

    - Interfacing the Robot's Motors
    - Configuring the LiDAR Sensor
    - Defining the Robot Model
    - Odometry Calculation
    - Enabling Autonomous Navigation
- Future Improvements

Module 13: Testing, Continuous Integration, and Continuous Deployment with ROS 2

- Testing ROS 2 Nodes Using GTest
    - GTest Features and Programming Pipeline
    - Integrating GTest with ROS 2 for Robust Testing
- Integrating ROS 2 APIs into the GTest Framework
    - Exploring the Source Files
    - Implementing Automatic Testing Using GTest
- Implementing CI/CD for ROS 2 Packages
    - Using GitHub Actions for Code Compilation
    - Adding a Status Badge

Module 14: Interfacing Large Language Models with ROS 2

- Large Language Models for Robotics
- Integrating ROS 2 Robots with LLM Agents
    - Components of AI Agents with ROS 2
    - How an AI Agent Works with ROS 2
- Creating ROS 2 AI Agents
    - ROS 2 AI Agent Development Using VS Code Dev Containers
    - Building an AI Agent for ROS 2
    - Configuring OpenAI API Keys

- Architecture of a ROS 2 AI Agent Node
- Running the Basic ROS 2 AI Agent
- AI Agent with ROS 2 Tools
- AI Agent for ROS 2 Turtlesim
- AI Agent for Nav2
- AI Agent for MoveIt 2

- Popular ROS 2 AI Agent Projects
  - ROSA
  - RAI
  - ROS-LLM
  - ROS 2 NanoLLM

Module 15: ROS 2 and Deep Reinforcement Learning

- Introduction to Deep Reinforcement Learning
  - Basic Concepts of Reinforcement Learning
  - Transitioning from Classical Reinforcement Learning to Deep Reinforcement Learning
  - Applications of Deep Reinforcement Learning in Robotics
  - Popular Deep Reinforcement Learning Algorithms
  - Value-Based Methods
  - Policy-Based Methods
  - Actor-Critic Methods
- Setting Up Isaac Lab and Isaac Sim on Ubuntu 24.04
  - Isaac Sim and Isaac Lab
  - Installing Isaac Sim and Isaac Lab on Ubuntu 24.04
  - Setting Up Docker and the NVIDIA Container Toolkit
  - Installing Isaac Sim and Isaac Lab

- Training and Testing Robots Using Isaac Lab
    - Isaac Lab Architecture
    - Training and Testing an Existing Reinforcement Learning Environment
    - Training UR-10 Using Isaac Lab
    - Testing UR-10 Using Isaac Lab
    - Training and Testing ANYmal Robot Navigation
    - Training and Testing an H1 Robot for a Locomotion Task
- Deploying a Trained Reinforcement Learning Model to a Real Robot
    - Deploying a Reinforcement Learning Model on the Boston Dynamics Spot Robot

Module 16: Implementing ROS 2 Visualization and Simulation Plugins

- Introducing ROS 2 Plugins
    - Understanding pluginlib in ROS 2
    - Steps for Creating, Compiling, and Loading Plugins
- Creating a Plugin for rqt
    - Creating a Qt User Interface
    - Implementing the Plugin Source
    - Adding Compilation and Configuration Files
- Developing a Gazebo Plugin
    - Writing the Source Code
    - Compiling and Executing the Gazebo Plugin
- Developing Plugins for RViz2
    - Writing the Source Code
    - Adding and Configuring the RViz2 Plugin