TurtleBot3 ROS2 Project - Complete Running Guide

Prerequisites & Initial Setup

1. Install ROS2 Humble and Dependencies

```bash
# Update system
sudo apt update && sudo apt upgrade -y

# Install ROS2 Humble (if not already installed)
sudo apt install ros-humble-desktop

# Install TurtleBot3 packages
sudo apt install ros-humble-turtlebot3*
sudo apt install ros-humble-dynamixel-sdk
sudo apt install ros-humble-gazebo-*

# Install additional dependencies
sudo apt install ros-humble-navigation2
sudo apt install ros-humble-nav2-bringup
sudo apt install ros-humble-tf2-tools
sudo apt install python3-colcon-common-extensions
```

2. Environment Setup

```bash
# Add to ~/.bashrc for permanent setup
echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc
echo "export TURTLEBOT3_MODEL=burger" >> ~/.bashrc
echo "export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:/opt/ros/humble/share/turtlebot3_gazebo/models

# Source for current session
source ~/.bashrc

# Or manually set environment
export TURTLEBOT3_MODEL=burger
export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:/opt/ros/humble/share/turtlebot3_gazebo/models
```

Project Setup and Build

3. Create and Build the Project

```bash
# Create workspace
mkdir -p ~/turtlebot3_ws/src
cd ~/turtlebot3_ws/src

# Create the package (follow the guide structure)
ros2 pkg create --build-type ament_cmake turtlebot3_custom --dependencies rclcpp std_msgs geometry_msgs sens

# Copy all the source files from the guide into the package
# (action/, srv/, src/, launch/, config/ directories)

# Build the project
cd ~/turtlebot3_ws
colcon build --packages-select turtlebot3_custom

# Source the workspace
source install/setup.bash
```

4. Verify Installation

```bash
# Check if executables are built
ls install/turtlebot3_custom/lib/turtlebot3_custom/

# Check custom interfaces
ros2 interface list | grep turtlebot3_custom

# Expected output:
# turtlebot3_custom/action/NavigateToGoal
# turtlebot3_custom/srv/GetRobotStatus
```

Running Options

Option 1: Simulation (Recommended for Testing)

Method 1A: Complete Simulation Launch

```bash
```

```
# Terminal 1: Launch everything at once
cd ~/turtlebot3_ws
source install/setup.bash
ros2 launch turtlebot3_custom turtlebot3_simulation.launch.py
```

Method 1B: Step-by-Step Simulation

```
bash

# Terminal 1: Start Gazebo World
cd ~/turtlebot3_ws
source install/setup.bash
ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py

# Terminal 2: Start Custom Nodes
cd ~/turtlebot3_ws
source install/setup.bash
ros2 launch turtlebot3_custom turtlebot3_custom.launch.py use_sim_time:=true

# Terminal 3: (Optional) Start RViz for visualization
ros2 run rviz2 rviz2
```

Method 1C: Individual Node Testing

```
bash

# Terminal 1: Gazebo
ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py

# Terminal 2: Controller Node
cd ~/turtlebot3_ws && source install/setup.bash
ros2 run turtlebot3_custom turtlebot3_controller

# Terminal 3: Status Server
cd ~/turtlebot3_ws && source install/setup.bash
ros2 run turtlebot3_custom robot_status_server

# Terminal 4: Action Server
cd ~/turtlebot3_ws && source install/setup.bash
ros2 run turtlebot3_custom navigation_action_server
```

Option 2: Real Robot

Prerequisites for Real Robot

```bash
# On the TurtleBot3 (Raspberry Pi)
export TURTLEBOT3_MODEL=burger
export ROS_DOMAIN_ID=30
ros2 launch turtlebot3_bringup robot.launch.py

# On your PC
export TURTLEBOT3_MODEL=burger
export ROS_DOMAIN_ID=30
# Make sure PC and robot are on the same network
```

Running on Real Robot

```bash
# Terminal 1: Launch custom nodes
cd ~/turtlebot3_ws
source install/setup.bash
ros2 launch turtlebot3_custom turtlebot3_custom.launch.py

# Terminal 2: (Optional) Teleop for manual control
ros2 run turtlebot3_teleop teleop_keyboard
```

Testing the Functionality

5. Test the Service

```bash
```

```
# Terminal (new): Test status service
cd ~/turtlebot3_ws && source install/setup.bash

# Method 1: Using custom client
ros2 run turtlebot3_custom status_client

# Method 2: Using command line
ros2 service call /get_robot_status turtlebot3_custom/srv/GetRobotStatus "{query_type: 'position'}"

# Method 3: Test different queries
ros2 service call /get_robot_status turtlebot3_custom/srv/GetRobotStatus "{query_type: 'battery'}"
ros2 service call /get_robot_status turtlebot3_custom/srv/GetRobotStatus "{query_type: 'general'}"
```

6. Test the Action

```
bash

# Terminal (new): Test navigation action
cd ~/turtlebot3_ws && source install/setup.bash

# Method 1: Using custom client
ros2 run turtlebot3_custom navigation_client

# Method 2: Using command line
ros2 action send_goal /navigate_to_goal turtlebot3_custom/action/NavigateToGoal "{target_pose: {position: {x: 2.0,

# Method 3: Send goal with feedback
ros2 action send_goal --feedback /navigate_to_goal turtlebot3_custom/action/NavigateToGoal "{target_pose: {posi
```

7. Monitor the System

```
bash
```

```
# Monitor topics
ros2 topic list
ros2 topic echo /cmd_vel
ros2 topic echo /odom
ros2 topic echo /scan

# Monitor nodes
ros2 node list
ros2 node info /turtlebot3_controller

# Monitor services
ros2 service list
ros2 service type /get_robot_status

# Monitor actions
ros2 action list
ros2 action info /navigate_to_goal
```

Visualization and Debugging

8. Use RQT Tools

bash

```
# Graph visualization
ros2 run rqt_graph rqt_graph

# Plot data
ros2 run rqt_plot rqt_plot

# Console messages
ros2 run rqt_console rqt_console

# Parameter reconfigure
ros2 run rqt_reconfigure rqt_reconfigure
```

9. RViz Setup

bash

```
# Start RViz
ros2 run rviz2 rviz2

# In RViz, add these displays:
# - RobotModel (to see the robot)
# - LaserScan (topic: /scan)
# - Map (if using navigation)
# - Path (for navigation paths)
# - TF (to see coordinate frames)
```

Launch File Variations

10. Launch with Different Parameters

```bash
# Launch with RViz
ros2 launch turtlebot3_custom turtlebot3_custom.launch.py start_rviz:=true

# Launch with different robot model
export TURTLEBOT3_MODEL=waffle_pi
ros2 launch turtlebot3_custom turtlebot3_simulation.launch.py

# Launch with custom world
ros2 launch turtlebot3_custom turtlebot3_simulation.launch.py world:=/path/to/your/world.world
```

Troubleshooting

11. Common Issues and Solutions

Issue: "No executable found"

```bash
# Solution: Rebuild and source
cd ~/turtlebot3_ws
colcon build --packages-select turtlebot3_custom
source install/setup.bash
```

Issue: "Interface not found"

```bash
```

```
# Solution: Check interface generation
ros2 interface list | grep turtlebot3_custom
# If empty, rebuild with verbose output
colcon build --packages-select turtlebot3_custom --cmake-args -DCMAKE_VERBOSE_MAKEFILE=ON
```

Issue: "Gazebo doesn't start"

```bash
# Solution: Check environment variables
echo $GAZEBO_MODEL_PATH
echo $TURTLEBOT3_MODEL
# Reinstall gazebo if needed
sudo apt install --reinstall ros-humble-gazebo-*
```

Issue: "Robot doesn't move"

```bash
# Check if cmd_vel is being published
ros2 topic echo /cmd_vel
# Check if obstacles are detected
ros2 topic echo /scan
```

Issue: "Action/Service not available"

```bash
# Check if servers are running
ros2 node list
ros2 service list
ros2 action list
```

12. Performance Monitoring

```bash
```

```
# Check CPU usage
top -p $(pgrep -d',' -f ros)

# Check memory usage
ros2 run rqt_top rqt_top

# Check message frequencies
ros2 topic hz /cmd_vel
ros2 topic hz /odom
ros2 topic hz /scan
```

Complete Testing Workflow

13. Step-by-Step Testing

```
bash

# 1. Start simulation
ros2 launch turtlebot3_custom turtlebot3_simulation.launch.py

# 2. Wait for Gazebo to fully load, then test service (new terminal)
cd ~/turtlebot3_ws && source install/setup.bash
ros2 run turtlebot3_custom status_client

# 3. Test action client (new terminal)
cd ~/turtlebot3_ws && source install/setup.bash
ros2 run turtlebot3_custom navigation_client

# 4. Monitor behavior
ros2 topic echo /cmd_vel
ros2 run rqt_graph rqt_graph
```

This guide provides everything you need to successfully run the TurtleBot3 ROS2 project in both simulation and real robot environments!