**Model and simulate a differential drive mobile robot in ROS2 Jazzy Jalisco and Gazebo starting from URDF and Xacro XML files**

**Author: Aleksandar Haber**

In this ROS2 tutorial, we explain how to model and simulate a two-wheel differential drive mobile robot in ROS2 Jazzy Jalisco and Gazebo.

**Copyright notice: this document and the lesson video should not be copied, redistributed, or publicly posted on public or private websites or social media platforms. The provided code files are the ownership of Aleksandar Haber. You have the right to use the provided code files for personal educational use and they should not be posted on public or private code repositories. The code should not be used for commercial purposes without paying an appropriate fee to the Author. The code files and the manual should not be used as a lecture material on online learning platforms and in university courses. Without the permission of the author and without paying an appropriate fee, the provided code files and this manual should not be used to produce academic papers, student reports, or other documents. This lesson and the accompanying material should not be used to train an AI algorithm or a large language model.**

**STEP 1: Verify that you have the correct version of Linux and ROS2**

Prerequisites: You have to make sure that you have the correct versions of ROS2 and Ubuntu installed on your system. Otherwise, the implementation presented in this tutorial might not work. This tutorial is based on ROS2 Jazzy Jalisco distribution and Ubuntu 24.04.

First of all, you need to have the following versions on Linux and ROS2

- Ubuntu 24.04

- ROS2 Jazzy Jalisco

To verify the Linux distribution, open a terminal and type:

**cat /etc/os-release**

You should see this:

**PRETTY\_NAME="Ubuntu 24.04 LTS"**

**NAME="Ubuntu"**

**VERSION\_ID="24.04"**

**VERSION="24.04 LTS (Noble Numbat)"**

**VERSION\_CODENAME=noble**

**ID=ubuntu**

**ID\_LIKE=debian**

**HOME\_URL="https://www.ubuntu.com/"**

**SUPPORT\_URL="https://help.ubuntu.com/"**

**BUG\_REPORT\_URL="https://bugs.launchpad.net/ubuntu/"**

**PRIVACY\_POLICY\_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"**

**UBUNTU\_CODENAME=noble**

**LOGO=ubuntu-logo**

It is important to see the version 24.04.

Let us verify the ROS2 distribution. Type

**source /opt/ros/jazzy/setup.bash**

**printenv ROS\_DISTRO**

***The output should be “jazzy”. If you see such an output, this means that ROS2 Jazzy Jalisco is installed on the computer.***

**STEP 2: Install the necessary packages**

Close the previous terminal and open a new terminal. Every time you want to do something in ROS2, you need to source the base environment. We do that by typing:

**source /opt/ros/jazzy/setup.bash**

Then, we need to install the necessary ROS2 and Gazebo packages. Some of these packages you might have, however, it is still an excellent idea to try to install them. If you already have them, the installation attempt will not change anything in your system.

**sudo apt-get update**

**sudo apt-get upgrade**

To install Gazebo Harmonic (recommended version for ROS2 Jazzy):

**sudo apt-get install ros-${ROS\_DISTRO}-ros-gz**

Then, you have to source the environment again in order to be able to run Gazebo. That is, you need to refresh the packages.

**source /opt/ros/jazzy/setup.bash**

To make sure that Gazebo is installed inside of ROS2 Jazzy, type this

**which gz**

You should get something like this as an output:

**/opt/ros/jazzy/opt/gz\_tools\_vendor/bin/gz**

Then, let us try to run Gazebo by typing

**gz sim**

Gazebo GUI simulation window should open. Try to run one of the build-in simulations to make sure that Gazebo is working.

Then, let us make sure that the following packages are installed

**sudo apt-get install gedit**

**sudo apt install ros-jazzy-joint-state-publisher**

**sudo apt install ros-jazzy-joint-state-publisher-gui**

**sudo apt install ros-jazzy-xacro  
sudo apt-get install ros-jazzy-teleop-twist-keyboard**

Let print the list of all Gazebo related packages:

**ros2 pkg list | grep gz**

It is important to see **ros\_gz** and **ros\_gz\_bridge**. These two packages are very important for displaying the model in Gazebo and for using ROS2 Jazzy to control a model in Gazebo.

## **STEP 3: Create the workspace and package**

Close the previously opened terminals, and open a new terminal, and source the environment:

**source /opt/ros/jazzy/setup.bash**

Then, create the workspace folders

**cd ~**

**mkdir -p ~/ws\_mobile/src**

Here, ws is an abbreviation for Workspace, and “mobile” is the name. Here, we are actually creating two folders. The first folder is “ws\_mobile” and inside of this folder, we created another folder called “src”. Next, we create and build the workspace:

**cd ~/ws\_mobile/**

**colcon build**

Next, we need to create the package:

**cd ~/ws\_mobile/src**

**ros2 pkg create --build-type ament\_cmake mobile\_robot**

The name of the package is “mobile\_robot”. The package source files and folders will be in the folder ~/ws\_mobile/src/mobile\_robot/ . Consequently, let us navigate to that folder and let us create several subfolders:

**cd ~/ws\_mobile/src/mobile\_robot/**

**mkdir launch model parameters**

Here, we created three folders: “launch”, “model”, and “parameters”.

The “launch” folder will contain the Python launch file necessary to run the model in ROS2 and Gazebo.

The “model” folder will contain the Xacro and URDF source files of the model. Note that URDF stands for “Unified Robotics Description Format”. This is a file format that is used to specify the geometry of robots and some other properties. Xacro is an XML macro language used to parametrize and simplify URDF model development.

The “parameters” folder will be used to create a yaml file that will define the parameters necessary to bridge ROS2 and Gazebo topics. That is, it will define parameters that are necessary to redirect ROS2 to Gazebo topics and vice-versa. This is a new requirement introduced in Gazebo Harmonic.

To build the workspace and packages, you need to navigate to the source folder of the workspace:

**cd ~/ws\_mobile/**

**colcon build**

## Step 4: Create the URDF (Xacro), Gazebo, and Parameter Files

First, we will create the robot model. We create the robot model by creating two files: Xacro and Gazebo files. The first Xacro file contains the robot 3D model, and the second Gazebo file contains additional parameters and Gazebo controller plugins necessary to successfully represent and control the model in Gazebo. First, let us navigate to the model folder

**cd ~/ws\_mobile/src/mobile\_robot/model**

Next, we need to create the Xacro model file defining the geometry. The name of the file should be “**robot.xacro**”. The quick approach would be to copy the included “robot.xacro” file to this folder (the file is included with this lesson). However, try to avoid this and instead type the file by yourself. Only like that you will be able to correctly learn ROS2. To create this file, either use gedit

**gedit robot.xacro**

or use the VS code editor (preferable option). After creating the file, enter the content of the file.

Then, create an additional Gazebo file that should be included in the above Xacro file. The file is provided with this tutoril. The name of the file should be “robot.gazebo”. Again, you can either use VS Code to create this file

**gedit robot.gazebo**

and enter the content of the file (file is included). Next, we need to create a yaml parameter file that will bridge ROS2 Jazzy and Gazebo. To do that, first navigate to the “parameters” folder

**cd ~/ws\_mobile/src/mobile\_robot/parameters**

In this folder, create the file called “**bridge\_parameters.yaml**”. You can use VS Code or gedit to create this file. With gedit, type

**gedit** **bridge\_parameters.yaml**

and enter the content of the file.

## **STEP 5: Create the Python launch file**

We create the launch file in the “launch” folder. Let us navigate to the folder

**cd ~/ws\_mobile/src/mobile\_robot/launch**

Create and edit the file either by using VS Code (preferable option) or by using gedit

**gedit gazebo\_model.launch.py**

## **STEP 6: Final adjustments and run the model**

Next, we need to add the dependencies

**cd ~/ws\_mobile/src/mobile\_robot**

**gedit package.xml**

Add these lines to “package.xml”

**<exec\_depend>joint\_state\_publisher</exec\_depend>**

**<exec\_depend>robot\_state\_publisher</exec\_depend>**

**<exec\_depend>gazebo\_ros</exec\_depend>**

**<exec\_depend>xacro</exec\_depend>  
<exec\_depend>ros\_gz\_bridge</exec\_depend>**

Then, we need to adjust the CMakeLists.txt file. We need to tell to ROS2 that important files are in the two new folders “launch” and “model”

**cd ~/ws\_mobile/src/mobile\_robot**

**gedit CMakeLists.txt**

Then, above “if(BUILD\_TESTING)”, type the following

**install(**

**DIRECTORY launch model parameters**

**DESTINATION share/${PROJECT\_NAME}**

**)**

Next, we need to perform one final build of the workspace and the package

**cd ~/ws\_mobile**

**colcon build**

Then, we need to source the package. That is we need to create an overlay

**source /opt/ros/jazzy/setup.bash**

**source ~/ws\_mobile/install/setup.bash**

Finally, run the model in Gazebo:

**ros2 launch mobile\_robot gazebo\_model.launch.py**

The main Gazebo window with the model will start.

## **STEP 7: Control the robot by using keyboard keys**

Do not close the original terminal that is running Gazebo. Open a new terminal, and in the new terminal type:

**source /opt/ros/jazzy/setup.bash**

**ros2 run teleop\_twist\_keyboard teleop\_twist\_keyboard**

This will source the environment and start the **teleop\_twist\_keyboard** node that is used to control the robot.

## **STEP 8: Investigate topics, messages, etc.**

Here, we will learn how to obtain more information about the topics connecting the nodes and messages that are being sent through nodes.

To see the list of topics, open a new terminal and type:

**source /opt/ros/jazzy/setup.bash**

**ros2 topic list**

The output should look like this:

/clock

/cmd\_vel

/joint\_states

/odom

/parameter\_events

/robot\_description

/rosout

/tf

/tf\_static

To see the messages being sent through the /odom (odometry) topic, you should type:

**ros2 topic echo /odom**

To obtain information about the type of messages that are being sent through a particular topic, type

**ros2 topic info /cmd\_vel**