ROS2 for Beginners Level 2

- TF | URDF | RViz | Gazebo

*Udemy*

*By Edouard Renard*

Note: I have a Pack Book (“ROS 2 from scratch”) with basically the same info.

# Introduction

## 1. Welcome!

## 2. Course Overview - How to Follow the Course

## 3. Install and setup ROS2 [recap]

## 4. Install Quick Fix (Gazebo)

## 5. Programming tools for this course

# TF (TransForm) Overview

## 6. Intro

## 7. Visualise a Robot TFs in RViz 2

Inside RobotModel properties, “Alpha” refers to the transparence of the links.

## 8. Relationship Between TFs, TF tree

## 9. What problem are we trying to solve with TF?

# Create a URDF for a Robot

## 10. Intro - What is URDF

## 11. Your first URDF file: Create and Visualize a Link

## 12. Material - Add Some Colors

## 13. Combine 2 Links with a Joint

## 14. Another example of the process to write the URDF right the first time

## 15. Different Types of Joints in a URDF

## 16. Add a Wheel to the Robot

## 17. Activity 01 - Complete the URDF for the Robot

## 18. Activity 01 - Solution

# Broadcast TFs with the Robot State Publisher

## 19. Intro

## 20. How the Robot State Publisher and URDF Work Together

## 21. Run the Robot State Publisher with URDF in the Terminal (Command Line)

## 22. Create a Robot Description Package to Install the URDF

Let’s start by creating the folder:

$ mkdir ~/ros2\_ws

Then, inside this workspace, create a src directory:

$ mkdir ~/ros2\_ws/src

If we don't have colcon installed yet, we can install it with:

$ sudo pt install python3-colcon-common-extensions

Then run:

~/ros2\_ws$ colcon build

To source it:

~/ros2\_ws$ source install/setup.bash

I’s more pratical add this “source” line at the end of

source /opt/ros/jazzy/setup.bash

source ~/ros2\_ws/install/setup.bash

## 23. Write a Launch file to Start the Robot State Publisher with URDF (XML)

The CmakeLists.txt file would contain:

cmake\_minimum\_required(VERSION 3.8)

project(my\_robot\_description)

if(CMAKE\_COMPILER\_IS\_GNUCXX OR CMAKE\_CXX\_COMPILER\_ID MATCHES "Clang")

add\_compile\_options(-Wall -Wextra -Wpedantic)

endif()

# find dependencies

find\_package(ament\_cmake REQUIRED)

install(

DIRECTORY urdf launch rviz

DESTINATION share/${PROJECT\_NAME}/

)

ament\_package()

This file will be used by the colcon build to find out multiple things including which directories will be copied to the install directory (in the example above we have 3 directories to be copied).

The display.launch.xml file would contain:

<launch>

<let name="urdf\_path"

value="$(find-pkg-share my\_robot\_description)/urdf/my\_robot.urdf" />

<let name="rviz\_config\_path"

value="$(find-pkg-share my\_robot\_description)/rviz/urdf\_config.rviz" />

<node pkg="robot\_state\_publisher" exec="robot\_state\_publisher">

<param name="robot\_description"

value="$(command 'xacro $(var urdf\_path)')" />

</node>

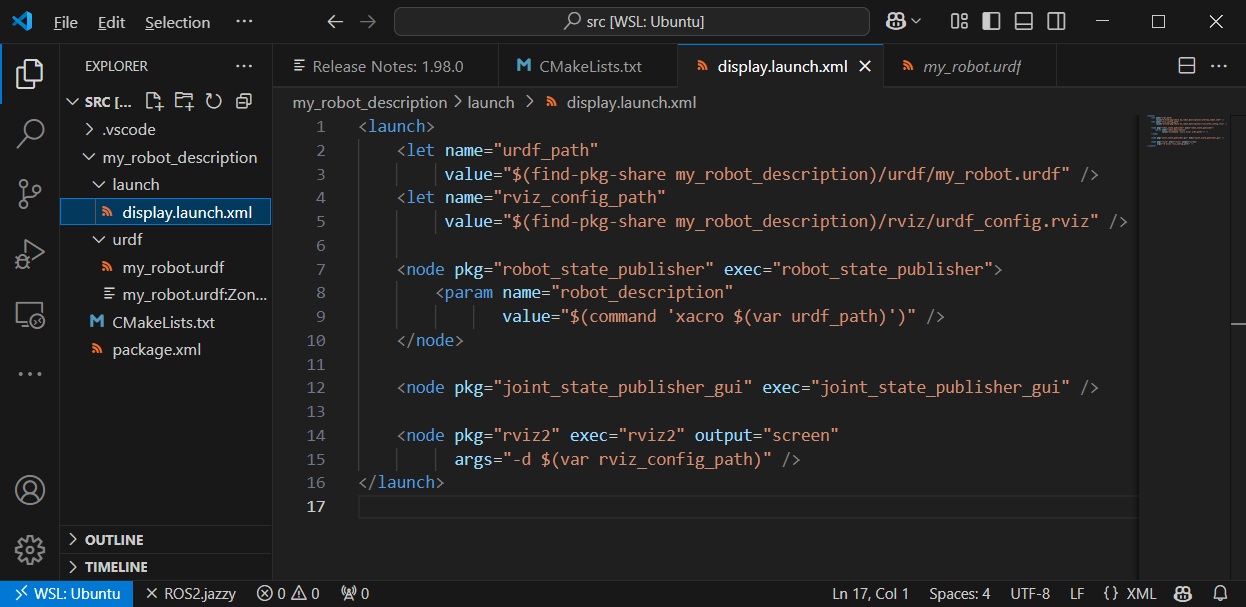
<node pkg="joint\_state\_publisher\_gui" exec="joint\_state\_publisher\_gui" />

<node pkg="rviz2" exec="rviz2" output="screen"

args="-d $(var rviz\_config\_path)" />

</launch>

Ulisses’ Note: We can have a simplified version without the variable rviz\_config\_path, so we would also not have it in the rviz2 args at the end.



Let’s build the file

$ cd ~/ros2\_ws/

$ colcon build

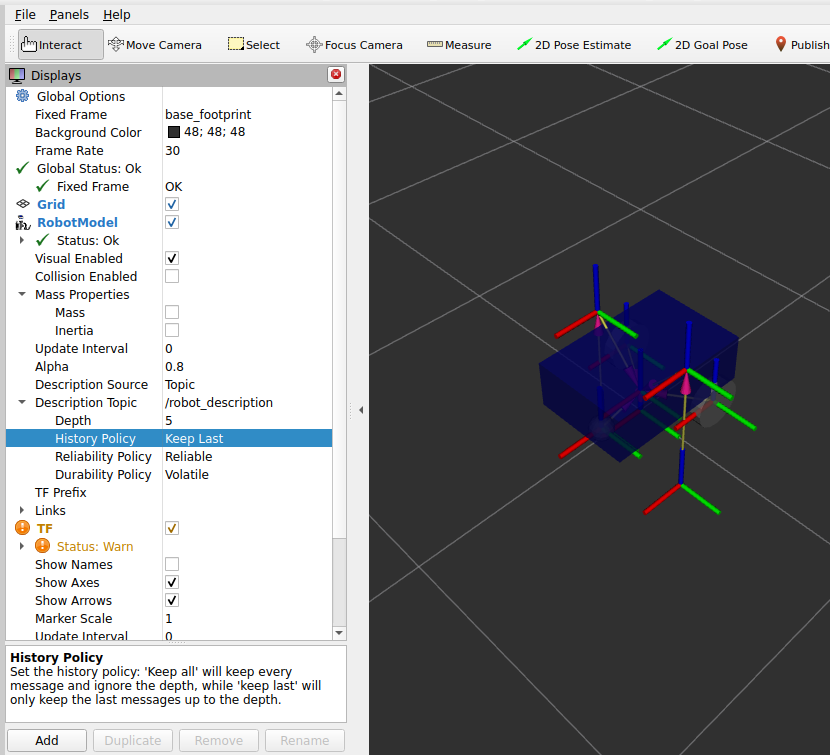
Then, don’t forget to source the workspace

source install/setup.bash

you can start your new launch file:

$ ros2 launch my\_robot\_description display.launch.xml

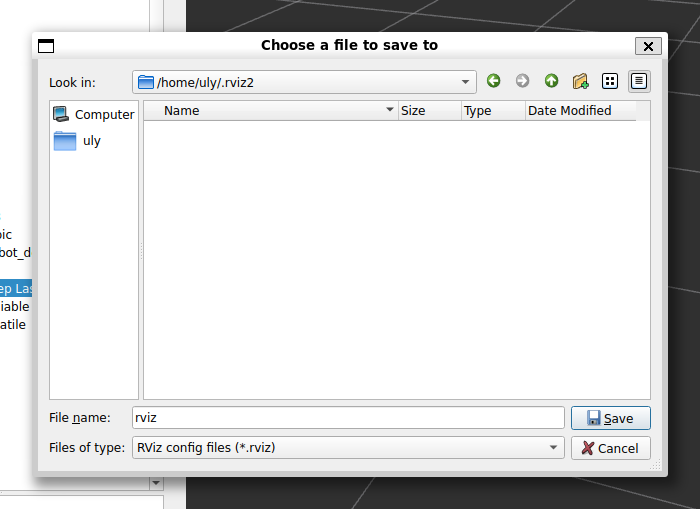
After a few fixes, we would have:



## 24. Python Launch File

## 25. Activity 02 - Add Rviz Config in the Launch File

After we manually fixed the rviz in the UI, we can save it as:



The we could use the “Open Config” menu option,or we could add it to the display.launch.xml file

## 26. Activity 02 - Solution

# Improve the URDF with Xacro

## 27. Intro

## 28. Make the URDF Compatible with Xacro

We can just rename my\_robot.urdf to my\_robot.urdf.xacro to make it compatible with Xacro.

Then we’ll need to make a change in the 2nd line of the file for it to be xacro compatible:

<robot name="my\_robot" xmlns:xacro="http://www.ros.org/wiki/xacro">

For colcon build to run automatically whenever there is some change in the file do:

colcon build –-symlink-install

## 29. Create Variables with Xacro Properties

## 30. Activity 03 - Xacro Properties

## 31. Activity 03 – Solution

## 32. Create Functions with Xacro Macros

## 33. Include a Xacro File in Another Xacro File

## 34. The Xacro Command to Generate the URDF

## 35. Real Meshes - Quick Overview

A mesh could be replacing the built-in <box> or <cylinder>. A mesh could come from Cad, *Blender* or *SolidWorks*.

Note: **STL** is a file format native to the stereolithography CAD software created by 3D Systems.

Let's see an example of using a mesh in the turtlebot3 robot:

<https://github.com/ROBOTIS-GIT/turtlebot3>

Inside the file:

<https://github.com/ROBOTIS-GIT/turtlebot3/blob/main/turtlebot3_description/urdf/turtlebot3_waffle.urdf>

We have:

<link name="base\_link">

<visual>

<origin xyz="-0.064 0 0.0" rpy="0 0 0"/>

<geometry>

<mesh filename="package://turtlebot3\_description/meshes/bases/waffle\_base.stl" scale="0.001 0.001 0.001"/>

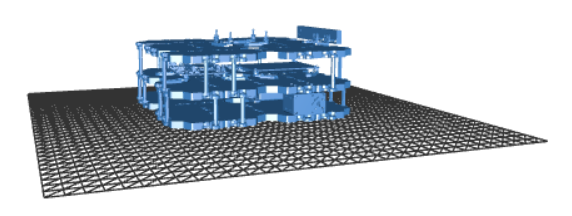
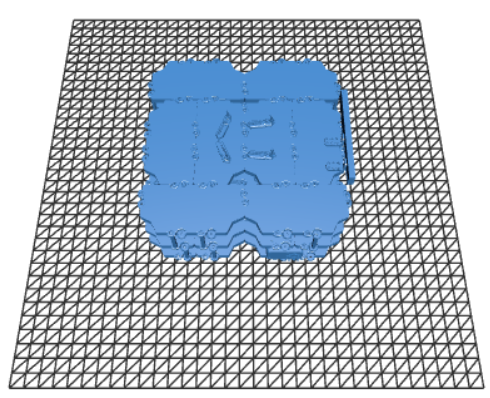
</geometry>

<material name="light\_black"/>

</visual>

In githyb, we can actually visualize how the stl looks like:

<https://github.com/ROBOTIS-GIT/turtlebot3/blob/main/turtlebot3_description/meshes/bases/waffle_base.stl>



We normally put the meshes inside the “meshes” folder.

We used “scale” above because the measurements in the stl file could be using different from the ones at the urdf. Eg. The stl was using meters and the urdf was using cm.

# Simulate Your Robot with Gazebo

## 36. [New Gazebo] UPDATE - PLEASE READ

## 37. Intro

## 38. [New Gazebo] Run Gazebo and how it works with ROS

You can start Gazebo either independently from ROS, or using a ROS launch file.

To run Gazebo without ROS:

$ gz sim

This will take you to a quick-start menu. Click on “Empty” to load an empty world, and you can create basic shapes just like in the following videos (menu bars look differently but they are not difficult to use). Make sure you click on the “play” button to start the simulation time.

To stop Gazebo, press Ctrl + C in the terminal where you started it.

To run Gazebo from a ROS launch file (will be useful later when we want to create our own launch file):

$ ros2 launch ros\_gz\_sim gz\_sim.launch.py

## 39. Run Gazebo

## 40. How Gazebo Works with ROS

## 41. [New Gazebo] Inertia and collision tags

## 42. Add Inertia Tags in the URDF

## 43. Activity 04 - Inertia Macros

## 44. Activity 04 – Solution

## 45. Add Collision Tags in the URDF

## 46. [New Gazebo] Spawn the robot in Gazebo

## 47. Spawn the Robot in Gazebo

## 48. [New Gazebo] Launch file to start the robot in Gazebo

## 49. Activity 05 - Launch File to Start Robot in Gazebo

## 50. Activity 05 – Solution

## 51. Fixing the Inertia Values

## 52. Fixing the Colors with Gazebo Material

## 53. [New Gazebo] Add a Gazebo plugin to control the robot

## 54. Add a Gazebo Plugin to Control the Robot

## 55. [New Gazebo] Create a world in your environment

## 56. Create a World in Gazebo

## 57. Launch the Robot in the World

# Add a Sensor in Gazebo

## 58. Intro

## 59. Add a Camera to the URDF

## 60. Add a Gazebo Plugin for the Camera

## 61. (Optional) Quick Fix For the Camera to Work with ROS

# Final Project

## 62. Intro - Final Project Overview

## 63. Step 1 - URDF Links and Joints

## 64. Step 2 - Adapt the Robot for Gazebo

## 65. Step 3 - Add Gazebo Plugins

## 66. Step 4 - Combine the 2 Robots

# Conclusion

## 67. What you’ve learned - Recap

## 68. What to do next

## 69. Bonus Lecture