Differential Drive TortoiseBot

Task Statement

The objective is to clearly separate robot description, simulation, and control, and to verify robot motion using velocity commands.

The task involves building two ROS 2 packages:

- **tortoisebot_description** – for robot modeling and RViz visualization
- **tortoisebot_gazebo** – for Gazebo simulation and controller integration

The robot must be controlled using the `/cmd_vel` velocity topic and tested using keyboard tele-operation.

Project Goal

Develop a clean, modular ROS 2 simulation where:

- The robot model loads correctly in RViz
- The robot spawns into an empty Gazebo world
- A differential drive controller converts velocity commands into wheel motion
- Robot movement is verified using tele-operation

Key Requirements

- Create a robot description package with URDF/Xacro files
- Configure wheel joints for differential drive motion
- Integrate ROS 2 Control with Gazebo
- Configure a differential drive controller
- Use `/cmd_vel` for velocity control
- Verify motion in Gazebo using teleop tools

Step-by-Step Approach

Phase 1: Package Setup

1.1 Workspace and Package Creation (Conceptual)

- Create a ROS 2 workspace
- Create two packages:
  - `tortoisebot_description`
  - `tortoisebot_gazebo`
- Install required ROS 2, Gazebo, and ROS 2 Control dependencies

Purpose: Keep robot description independent from simulation and control logic.

Phase 2: Robot Description Package (`tortoisebot_description`)

2.1 Robot Model Definition

- Define the robot structure using URDF/Xacro
- Declare a gazebo plugin to define ROS2 controls
- Visual geometry (for RViz)

2.2 Differential Drive Wheel Joints

- Define two continuous wheel joints:
  - Left wheel joint
  - Right wheel joint
- Ensure correct rotation axes and joint limits

2.3 ROS 2 Control Interface (Conceptual)

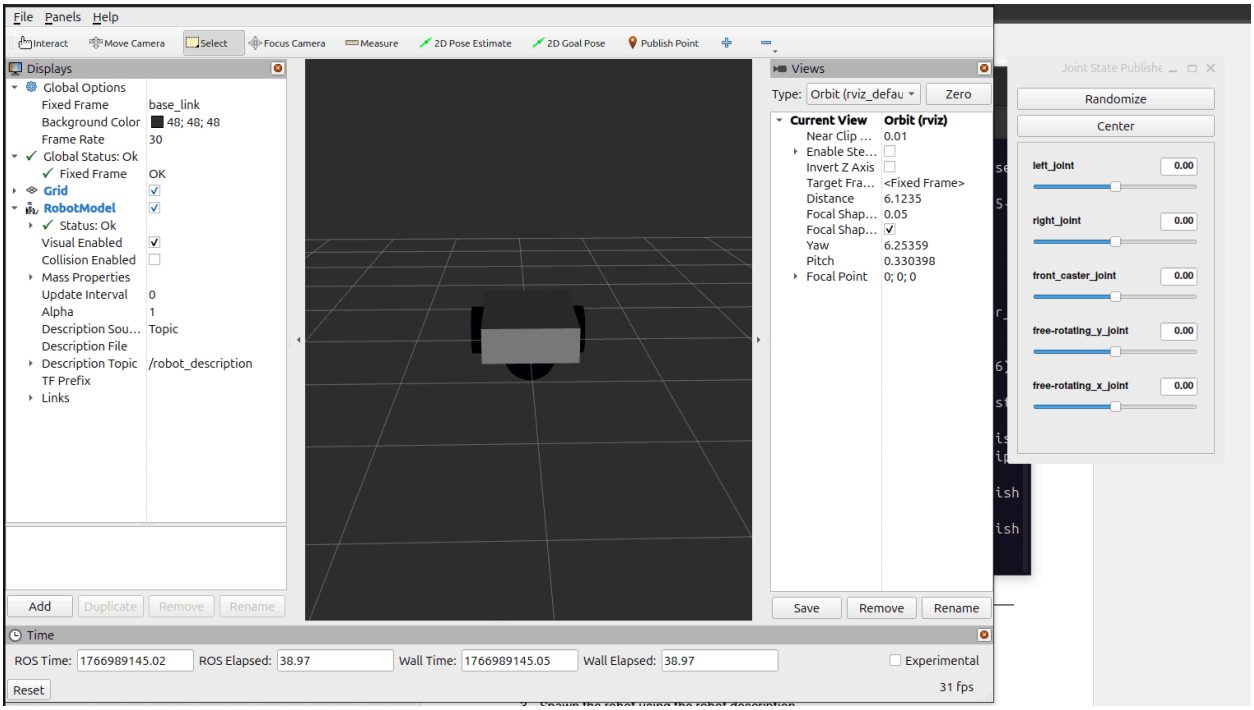
- Add a ROS 2 Control block to the robot description

- Define:
    - Command interfaces (velocity for wheels)
    - State interfaces (position and velocity feedback)

Phase 3: RViz Visualization

RViz Launch File (Pseudocode)

1. Locate the `tortoisebot_description` package
2. Load the robot URDF/Xacro file
3. Convert Xacro to a robot description string
4. Start `robot_state_publisher`
    - Publishes transforms based on joint states
5. Start RViz
    - Load a predefined RViz configuration
6. Launch all components together



Phase 4: Gazebo Simulation Package (`tortoisebot_gazebo`)

4.1 Empty World Setup

- Create an `empty_world.world` file

Purpose: Provide a simple environment for testing motion.

4.2 Gazebo Launch Process (Pseudocode)

1. Launch Gazebo using ROS-integrated Gazebo launcher
2. Load the empty world
3. Spawn the robot using the robot description
4. Start `robot_state_publisher` for transform updates
5. Enable Gazebo-ROS 2 Control plugin

Purpose: Bring together simulation, robot model, and control.

Phase 5: Differential Drive Controller Configuration

Controller Configuration (Conceptual)

- Use `diff_drive_controller`
- Configure:
    - Left and right wheel joint names
    - Wheel separation and radius
    - Velocity command topic: `/cmd_vel`
    - Message type: `TwistStamped`

Phase 6: Velocity Command Integration

`/cmd_vel` Communication Flow

- Tele-operation node publishes velocity commands to `/cmd_vel`

- Differential drive controller subscribes to `/cmd_vel`

Phase 7: Tele-operation Testing

Tele-op Workflow (Pseudo code)

1. Launch Gazebo simulation with the robot
2. Ensure the differential drive controller is active
3. Start `teleop_twist_keyboard` with stamped velocity enabled
4. Press movement keys
5. Observe robot movement in Gazebo

Alternative: Use a GUI-based teleop tool for button control.

Errors and Solutions

Error 1: `/cmd_vel` Message Type Mismatch

Problem Observed The robot did not respond to keyboard inputs even though `/cmd_vel` messages were being published.

Observed Output

$ ros2 topic info /cmd_vel -v

Type: ['geometry_msgs/msg/Twist', 'geometry_msgs/msg/TwistStamped']

Publisher:
teleop_twist_keyboard -> Twist

Subscriber:
diff_drive_controller -> TwistStamped

What This Means

- The teleop node was publishing `Twist`
- The differential drive controller was expecting `TwistStamped`

- Due to this mismatch, commands were ignored

Solution Applied

ros2 run teleop_twist_keyboard teleop_twist_keyboard \
 --ros-args -p stamped:=true

Result

- `/cmd_vel` publisher and subscriber used the same message type
- Robot responded correctly to velocity commands

Error 2: `/cmd_vel` Not Visible in Gazebo Topic List

Problem Observed At times, `/cmd_vel` did not appear when listing Gazebo topics:

$ gz topic -l

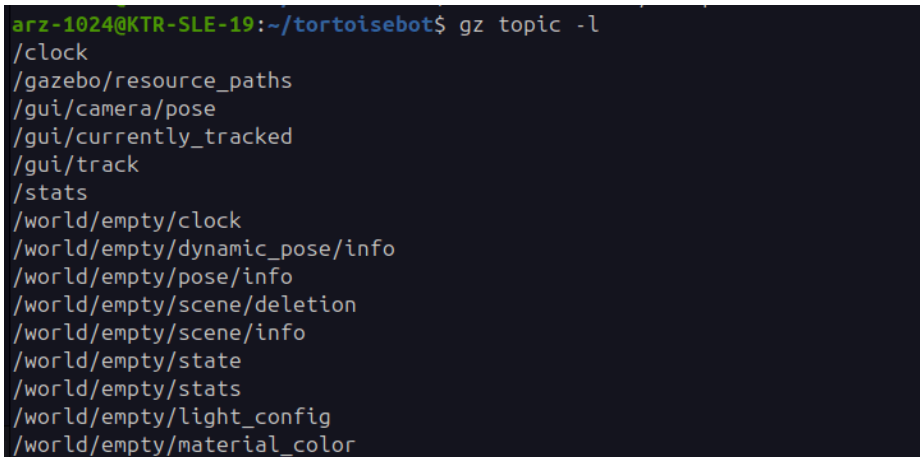
- `/cmd_vel` visible in ROS 2
- `/cmd_vel` missing from Gazebo
- Robot still moved correctly

Pointers

- `/cmd_vel` is consumed by the ROS 2 controller
- Gazebo receives **wheel joint commands**, not `/cmd_vel`
- Gazebo does not need to subscribe to `/cmd_vel`

Solution

- No bridge was added for `/cmd_vel`
- Velocity control was left entirely within ROS 2 Control



```
arz-1024@KTR-SLE-19:~/tortoisebot$ ros2 node list
/controller_manager
/diff_drive_controller
/gz_ros_control
/joint_state_broadcaster
/robot_state_publisher_bot
/ros_gz_bridge
/rqt_gui_py_node_23124
/teleop_twist_keyboard
arz-1024@KTR-SLE-19:~/tortoisebot$ ros2 topic list
/clock
/cmd_vel
/controller_manager/activity
/controller_manager/introspection_data/full
/controller_manager/introspection_data/names
/controller_manager/introspection_data/values
/controller_manager/statistics/full
/controller_manager/statistics/names
/controller_manager/statistics/values
/diagnostics
/diff_drive_controller/odom
/diff_drive_controller/transition_event
/dynamic_joint_states
/joint_state_broadcaster/transition_event
/joint_states
/parameter_events
/robot_description
/rosout
/tf
/tf_static
arz-1024@KTR-SLE-19:~/tortoisebot$ ros2 control list_controllers
[INFO] [1766033772.744681762] [_ros2cli_23295]: waiting for service /controller_manager
/list_controllers to become available...
diff_drive_controller    diff_drive_controller/DiffDriveController      active
joint_state_broadcaster  joint_state_broadcaster/JointStateBroadcaster  active
```

Error 3: Delayed Robot Response to Teleop Commands

Problem Observed

- Robot did not move immediately after pressing keys
- Movement started only after a short delay

-Controllers were not fully active when teleop started

-Gazebo and ROS 2 Control initialization required synchronization

Solution

- use_sim_time is added as bridge to link the simulation tools

Using rqt_robot_steering

Purpose

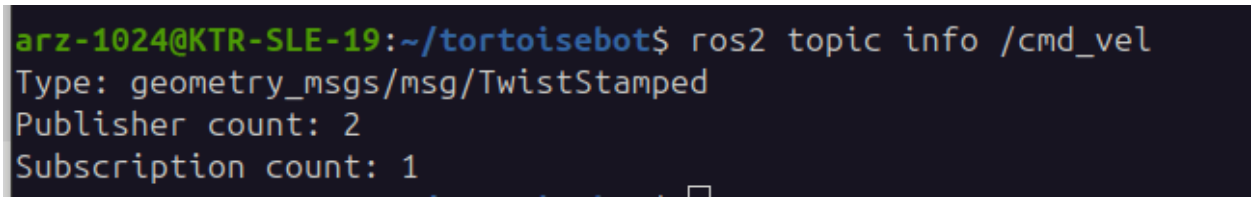
`rqt_robot_steering` was used only to **visualize and demonstrate an alternative way** of sending velocity commands to the differential drive controller. It was not used as the primary control method.

The goal was to confirm that:

- The differential drive controller accepts velocity commands from multiple ROS 2 tools
- Controller behavior remains consistent regardless of the command source

Observed Behavior

- `rqt_robot_steering` publishes velocity commands on `/cmd_vel`
- Commands are received by the same differential drive controller
- Robot motion in Gazebo matches keyboard tele-operation behavior



Simulation Video

https://meeting.zoho.in/meeting/public/videoprv?recordingId=de3fdfb0521b97c63be27cba662996c245b01fb79a551514a9a1474a776f926e&x-meeting-org=60014996923