my_diffbot ROS2 Package — Gazebo (ros_gz_sim) Edition

This document contains a regenerated ROS 2 package scaffold tuned for **Gazebo (modern) via ros_gz_sim** (Gazebo Fortress/Harmonic style) and the launch style you provided. It implements a differential-drive robot (DiffBot) with a 2D LiDAR that tracks a human actor in Gazebo and enforces a configurable safety buffer.

> Reference: your assignment PDF at `/mnt/data/CET3004-Assignment-2.pdf`.

Package file tree

```
my_diffbot/
├─ package.xml
├─ setup.py
├─ setup.cfg
├─ CMakeLists.txt
├─ resource/my_diffbot
├─ my_diffbot/
│  ├─ __init__.py
│  ├─ nodes/
│  │  ├─ perception_node.py
│  │  ├─ follow_stop_node.py
│  │  ├─ cmd_vel_relay.py
│  └─ launch/
│     ├─ spawn_gazebo.launch.py
│     └─ follow_bringup.launch.py
├─ urdf/
│  ├─ diffbot.urdf.xacro
│  └─ diffbot.urdf
├─ config/
│  └─ diff_controllers.yaml
├─ worlds/
│  └─ follow_world.sdf
└─ README.md
```

Highlights (Gazebo / ros_gz_sim specific)

- Uses `ros_gz_sim` to run `gz sim` and the ros<->gz bridge. Launch file follows the pattern you provided (processing xacro via `Command(['xacro ', xacro_file])`, `robot_state_publisher`, `ros_gz_sim create` to spawn from `/robot_description`, and `controller_manager` + spawners via `TimerAction`).

- URDF includes Gazebo `<sensor>` plugin compatible with `ros_gz_sim` (a ray sensor entry under `<gazebo>` that will be bridged to `/scan`).
- `cmd_vel_relay.py` node relays `/cmd_vel` (from controller) to `/cmd_vel` topic expected by `ros_gz_sim` bridge if needed (keeps compatibility with different topic names). This prevents missing commands when controllers and simulation expect different topics.
- World `follow_world.sdf` contains a simple moving actor represented by a Gazebo model (a box or simple actor) that moves along a short scripted path — useful as the human target for following.
- Default safety buffer is **0.8 m** and can be overridden via parameters in the launch file.

Provided / Important files (selected excerpts)

Launch file (spawn_gazebo.launch.py)

This launch is adapted from the snippet you gave. It starts `ros_gz_sim` (via its provided launch in ROS install), publishes `robot_description`, spawns the robot from that topic, loads ros2_control and the controllers, and starts a `cmd_vel_relay` node.

```
from launch import LaunchDescription
from launch_ros.actions import Node
from launch.actions import IncludeLaunchDescription, TimerAction
from launch.launch_description_sources import PythonLaunchDescriptionSource
from launch.substitutions import Command, PathJoinSubstitution
from ament_index_python.packages import get_package_share_directory
import os


def generate_launch_description():
    pkg_share = get_package_share_directory('my_diffbot')
    xacro_file = os.path.join(pkg_share, 'urdf', 'diffbot.urdf.xacro')

    # Process xacro to URDF
    robot_description = Command(['xacro ', xacro_file])

    gazebo = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(
            [os.path.join('/opt/ros/jazzy/share/ros_gz_sim/launch',
'gz_sim.launch.py')]
        ),
        launch_arguments={'world': os.path.join(pkg_share, 'worlds',
'follow_world.sdf')}.items()
    )

    # Robot State Publisher - publishes the robot description
    robot_state_publisher = Node(
        package='robot_state_publisher',
        executable='robot_state_publisher',
        name='robot_state_publisher',
        output='screen',
```

```
        parameters=[{'robot_description': robot_description, 'use_sim_time':
True}]
    )

    # Spawn entity - uses topic instead of file
    spawn_entity = Node(
        package='ros_gz_sim',
        executable='create',
        arguments=['-topic', '/robot_description', '-name', 'diffbot'],
        output='screen'
    )

    controller_config = os.path.join(pkg_share, 'config',
'diff_controllers.yaml')

    controller_manager = Node(
        package='controller_manager',
        executable='ros2_control_node',
        parameters=[{'robot_description': robot_description},
controller_config],
        output='screen'
    )

    load_joint_state_broadcaster = TimerAction(
        period=3.0,
        actions=[
            Node(
                package='controller_manager',
                executable='spawner',
                arguments=['joint_state_broadcaster', '--controller-manager',
'/controller_manager'],
                output='screen'
            )
        ]
    )

    load_diff_drive_controller = TimerAction(
        period=4.0,
        actions=[
            Node(
                package='controller_manager',
                executable='spawner',
                arguments=['diff_drive_controller', '--controller-manager',
'/controller_manager'],
                output='screen'
            )
        ]
    )

    cmd_vel_relay = TimerAction(
        period=5.0,
```

```
        actions=[
            Node(
                package='my_diffbot',
                executable='cmd_vel_relay',
                name='cmd_vel_relay',
                output='screen'
            )
        ]
    )

    return LaunchDescription([
        gazebo,
        robot_state_publisher,
        spawn_entity,
        controller_manager,
        load_joint_state_broadcaster,
        load_diff_drive_controller,
        cmd_vel_relay,
    ])
```

---

cmd_vel_relay.py (small helper node)

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist

class CmdVelRelay(Node):
    def __init__(self):
        super().__init__('cmd_vel_relay')
        # subscribe to /cmd_vel from controllers and republish to /diffbot/
cmd_vel (or /cmd_vel)
        self.sub = self.create_subscription(Twist, '/cmd_vel', self.cb, 10)
        self.pub = self.create_publisher(Twist, '/diffbot/cmd_vel', 10)

    def cb(self, msg: Twist):
        # simply forward the message
        self.pub.publish(msg)


def main(args=None):
    rclpy.init(args=args)
    node = CmdVelRelay()
    rclpy.spin(node)
    node.destroy_node()
    rclpy.shutdown()
```

---

perception_node.py

Perception node remains the same architecture: it subscribes to `/scan` and publishes `/target_distance`, `/target_detected`, and `/safe_to_move`. It treats the forward-facing sector and computes the median range as target distance.

(Implementation placed in `my_diffbot/nodes/perception_node.py` — same behavior as earlier but with `use_sim_time` compatibility.)

---

follow_stop_node.py

Follow-and-stop FSM with states SEARCH, FOLLOW, STOP. Subscribes to `/target_distance`, `/target_detected`, and `/safe_to_move`. Publishes `/cmd_vel` that controllers/relay forward to the simulator. Safety buffer param and gains are exposed as ROS parameters and configurable from the launch.

URDF notes

- `urdf/diffbot.urdf.xacro` contains a `laser_link` with a Gazebo `<sensor type="ray">` block, and a `<plugin>` entry for the Gazebo ROS2 laser plugin; this is compatible with `ros_gz_sim` bridging to `/scan`.
- The robot uses `transmission` and `ros2_control` compatible joints so `ros2_control_node` can be used to attach controllers (diff_drive_controller).

---

worlds/follow_world.sdf

- Includes ground and sun.
- Adds a moving model (simple box or actor) that moves along a short path using a model plugin or a simple `script` element. This serves as the human to follow. The world is passed to the `gz_sim.launch.py` via launch arguments.

How to build & run

1. From a ROS 2 workspace root:

```
colcon build --packages-select my_diffbot
source install/setup.bash
```

1. Launch the package:

```
ros2 launch my_diffbot spawn_gazebo.launch.py
```

1. The robot should spawn into the Gazebo world, sensors bridged to ROS 2, controllers loaded, and the `perception_node` + `follow_stop_node` (and `cmd_vel_relay`) active.

Next steps / customization

- If you want the actor to be a humanoid model or to follow a prerecorded trajectory, I can replace the box actor with a Gazebo actor model and script its animation.
- If you have your Week-7 URDF, paste it and I will adapt joint names and controller config.
- I used `/mnt/data/CET3004-Assignment-2.pdf` as the assignment reference supplied by you; if you want a different file or updated spec, paste it or upload here.

End of regenerated Gazebo (ros_gz_sim) edition.