ROS2 Simulation Task - Student Challenge Guide

Objective

Build a ROS2 simulation (Gazebo or RViz) where a robot:

- ✅ Receives a target position (via topic)
- ✅ Moves toward that position automatically
- ✅ Publishes position feedback

You'll use:

- URDF (robot model)
- Launch files (for automation)
- Multiple topics (command & feedback)

Deliverables:

- Simulation video/screenshot
- GitHub repository (src + launch + URDF + README)

System Architecture

```text
             ┌────────────────────────────┐
             │     RViz / Gazebo              │
             │ (Simulation Environment)       │
             └──────────┬─────────────────┘
                        │
                 (Sensor feedback)
                        │
          ┌─────────────┴────────────────────┐
          │            Robot Node                 │
          │  – Publishes: /odom                   │   (nav_msgs/Odometry)
          │  – Subscribes: /target_pos            │   (geometry_msgs/PoseStamped)
          └─────────────┬────────────────────┘
                        │
                  (Control Command)
                        │
          ┌─────────────┴────────────────────┐
          │         Controller Node               │
          │  – Receives: /odom                    │
          │  – Publishes: /cmd_vel                │   (geometry_msgs/Twist)
          │  – Publishes: /feedback               │   (custom msg or String)
          └─────────────┬────────────────────┘
                         │
              (Target from User or UI)
                         │
          ┌─────────────┴────────────────────┐
          │        Target Sender Node             │
          │  – Publishes: /target_pos             │
          └──────────────────────────────────┘
```

Components to Include

1. URDF Robot Model

Simple differential-drive robot:

- 2 wheels + chassis + caster
- Gazebo plugin for differential drive
- Example: `urdf/simple_bot.urdf.xacro`

2. Node Descriptions

Node	Role	Topics
`target_sender`	Sends target position	Publishes `/target_pos`
`controller`	Calculates velocity to reach goal	Subscribes `/target_pos`, `/odom` → Publishes `/cmd_vel`
`robot_state_publisher`	Publishes TF + robot model	Publishes `/tf`, `/robot_description`
`gazebo`	Simulates physics & motion	Uses `/cmd_vel`, publishes `/odom`
`feedback_node` (optional)	Logs progress	Subscribes `/odom`, publishes `/feedback`

3. Topics & Message Types

Topic	Type	Description
`/target_pos`	`geometry_msgs/PoseStamped`	Target position for the robot
`/cmd_vel`	`geometry_msgs/Twist`	Velocity command to robot
`/odom`	`nav_msgs/Odometry`	Robot position feedback
`/feedback`	`std_msgs/String`	Optional status ("Moving to target...")

4. Launch File

Example: `launch/simulation.launch.py`

- Load robot description (URDF)
- Start Gazebo with world
- Launch controller + target_sender

Implementation Steps

Step 1: Create Package Structure

```
bash

ros2 pkg create navigation_bot --build-type ament_python --dependencies
rclpy geometry_msgs nav_msgs tf2_ros gazebo_ros_pkgs
```

Step 2: Build URDF Robot

Create `urdf/simple_robot.urdf.xacro`:

xml

```
<?xml version="1.0"?>
<robot name="simple_robot">
  <!-- Base Link -->
  <link name="base_link">
    <visual><geometry><box size="0.3 0.3 0.2"/></geometry></visual>
    <collision><geometry><box size="0.3 0.3 0.2"/></geometry></collision>
  </link>

  <!-- Wheels -->
  <link name="left_wheel">...</link>
  <link name="right_wheel">...</link>

  <!-- Gazebo Plugins -->
  <gazebo>
    <plugin name="diff_drive" filename="libgazebo_ros_diff_drive.so">
      <command_topic>cmd_vel</command_topic>
      <odometry_topic>odom</odometry_topic>
    </plugin>
  </gazebo>

</robot>
```

Step 3: Create Controller Node

`scripts/controller_node.py` - Implements:

- Subscribe to `/target_pos` and `/odom`
- Calculate required velocity (PID or simple proportional)
- Publish to `/cmd_vel`
- Publish feedback to `/feedback`

Step 4: Create Target Publisher

`scripts/target_publisher.py` - Publishes target positions:

- Can be manual input or predefined points
- Uses `geometry_msgs/PoseStamped`

Step 5: Build Launch File

`launch/simulation.launch.py` - Starts:

- Robot state publisher
- Gazebo with your robot
- Controller node
- Target publisher

Expected Behavior

1. Start simulation: Robot appears in Gazebo/RViz
2. Publish target: Send position via `/target_pos`
3. Robot moves: Automatically navigates to target
4. Feedback: Continuous position updates via `/odom`
5. Stop: Robot stops when close to target

📁 GitHub Repository Structure

text

```
navigation_bot/
├── package.xml
├── setup.py
├── launch/
│   └── simulation.launch.py
├── urdf/
│   └── simple_robot.urdf.xacro
├── scripts/
│   ├── controller_node.py
│   └── target_publisher.py
├── worlds/
│   └── empty.world

└── README.md
```

Testing Checklist

- Robot loads correctly in Gazebo
- `/target_pos` topic receives messages
- Robot moves when target is published
- `/cmd_vel` publishes velocity commands
- `/odom` provides position feedback
- Robot stops at target position
- All components start via launch file