TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

ĐẠI HỌC QUỐC GIA HÀ NỘI

BÁO CÁO GIỮA KỲ MÔN

LẬP TRÌNH ROBOT VỚI ROS

ĐỀ TÀI: Robot bánh xích, 1 khớp rotation, 1 khớp rotation và các cảm biến IMU, Camera, Encoder

Họ và tên: TRẦN THÁI BÌNH

Lớp: K67E-RE

Mã số sinh viên: 22027543

1. Thiết kế của Robot, dạng Robot, động học, kích thước

- Dạng bánh: Bánh xích

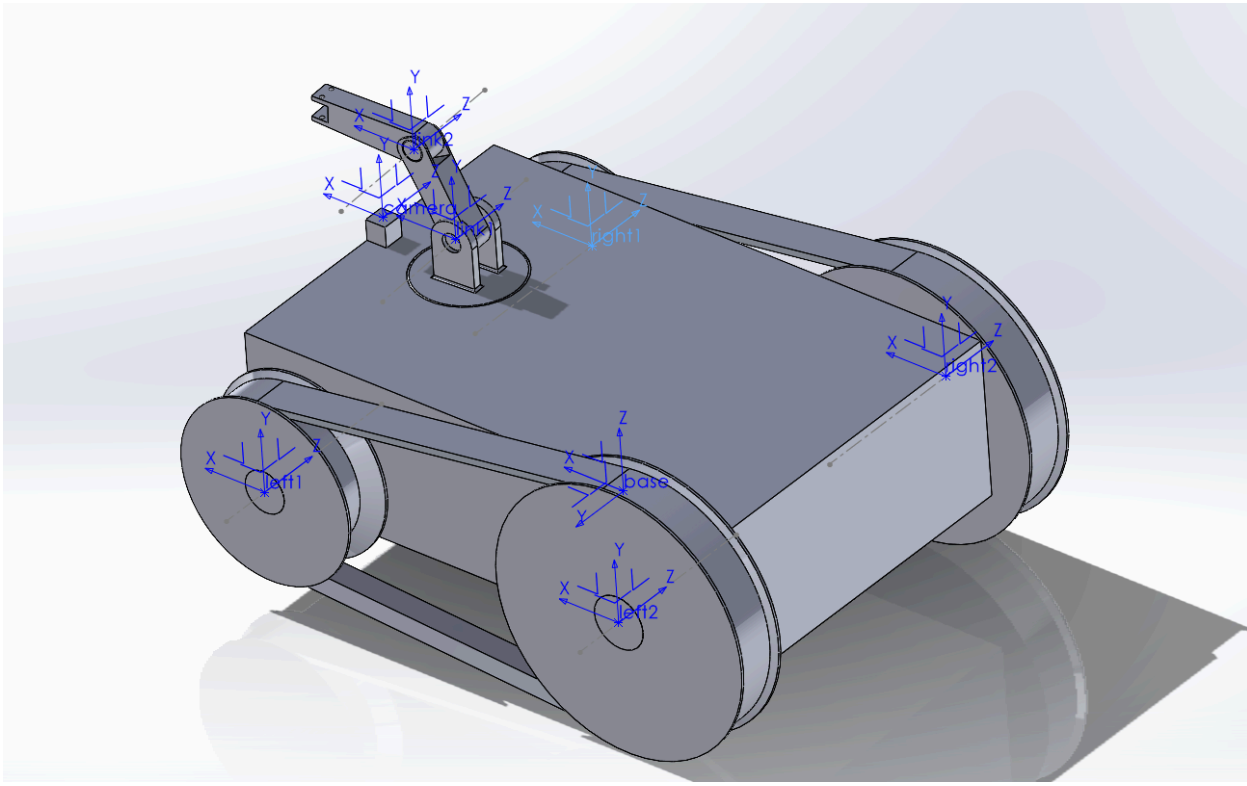
- Tay máy: 1 khớp xoay, 1 khớp trượt

- Cảm biến: IMU, Camera, Encoder

Thiết kế cơ bản của robot

- Thân robot:
    + Dạng bánh xích: Thân robot sẽ được thiết kế với một khối hình chữ nhật để mô phỏng thân chính của robot. Đặc điểm của bánh xích là có một cấu trúc chắc chắn, giúp robot di chuyển trên địa hình không bằng phẳng.
    + Kích thước thân: Bạn có thể đặt kích thước của thân robot phù hợp với các yêu cầu sử dụng. Ví dụ, kích thước thân có thể là 0.5m x 0.3m x 0.2m (dài x rộng x cao), tạo đủ không gian cho các thành phần khác như bánh xích và tay máy.
- Bánh xích:
    + Bánh xích của robot sẽ được kết nối với thân robot thông qua các joint. Bạn có thể mô phỏng bánh xích với các liên kết (links) có kích thước tương ứng và sử dụng các plugin trong Gazebo để mô phỏng chuyển động của bánh xích.
    + Mỗi bánh xích có thể được kết nối với thân robot bằng các joint loại fixed, có nghĩa là bánh xích không thay đổi vị trí mà chỉ chuyển động quanh trục của nó.
- Tay máy (Arm):
    + Tay máy có thể có 1 khớp xoay và 1 khớp trượt. Khớp xoay cho phép tay máy quay quanh trục, còn khớp trượt giúp di chuyển phần tay máy theo chiều dọc.

+ Tay máy sẽ được gắn vào thân robot qua một joint cố định, tạo sự ổn định khi robot di chuyển.



(cách đặt các trục cho file urdf trên solidworks)

2. Các cảm biến

- IMU (Inertial Measurement Unit): Được sử dụng để đo các thông số chuyển động của robot như gia tốc, tốc độ góc và định hướng. IMU sẽ giúp bạn theo dõi trạng thái chuyển động của robot và hỗ trợ điều hướng trong môi trường 3D.
- Camera: Cảm biến camera sẽ cung cấp dữ liệu hình ảnh cho robot, có thể dùng để phát hiện vật thể, điều hướng, hoặc hỗ trợ các thuật toán SLAM. Camera sẽ được gắn vào thân robot, có thể có góc nhìn (FOV) rộng để thu thập hình ảnh môi trường xung quanh.
- Encoder: Dùng để đo lường sự quay của bánh xích, giúp robot xác định vị trí và tốc độ di chuyển của nó trong không gian.

3. Các bước mô phỏng trên Gazebo

- Mô tả robot trong URDF/Xacro:

- Bạn sẽ viết các file URDF hoặc Xacro để mô tả cấu trúc của robot, các liên kết (links), và các khớp nối (joints). Ví dụ, bạn có thể có một mô tả riêng cho bánh xích, tay máy và cảm biến.
- Kết nối các liên kết và khớp nối:

→ Trong URDF, các thành phần như thân robot, bánh xích và tay máy sẽ được kết nối với nhau thông qua các khớp nối (joints). Mỗi khớp nối có thể có các thuộc tính như loại khớp (fixed, revolute, prismatic,...) và vị trí tương đối của các liên kết.

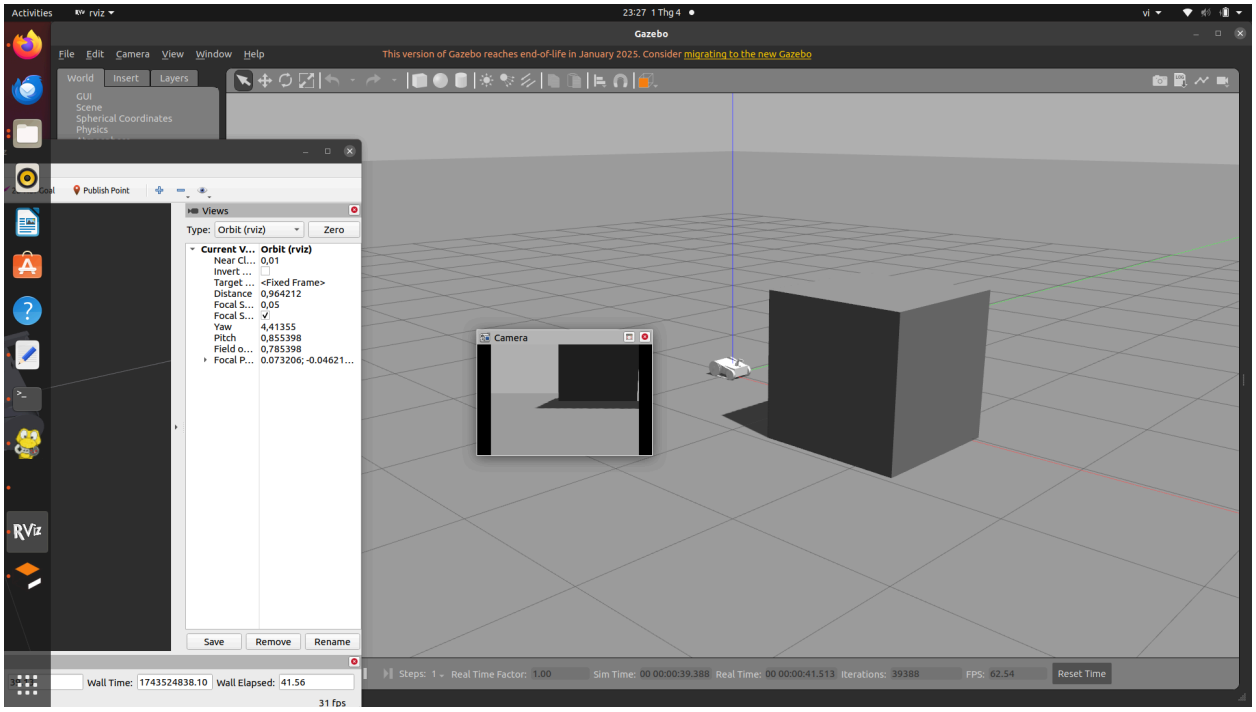
1) Cảm biến và plugin:
   + Các cảm biến như camera, IMU, và encoder sẽ được kết nối vào mô hình URDF và cần các plugin Gazebo để mô phỏng chức năng thực tế của chúng. Các plugin này giúp bạn lấy dữ liệu từ cảm biến và sử dụng trong các thuật toán điều khiển và xử lý dữ liệu.

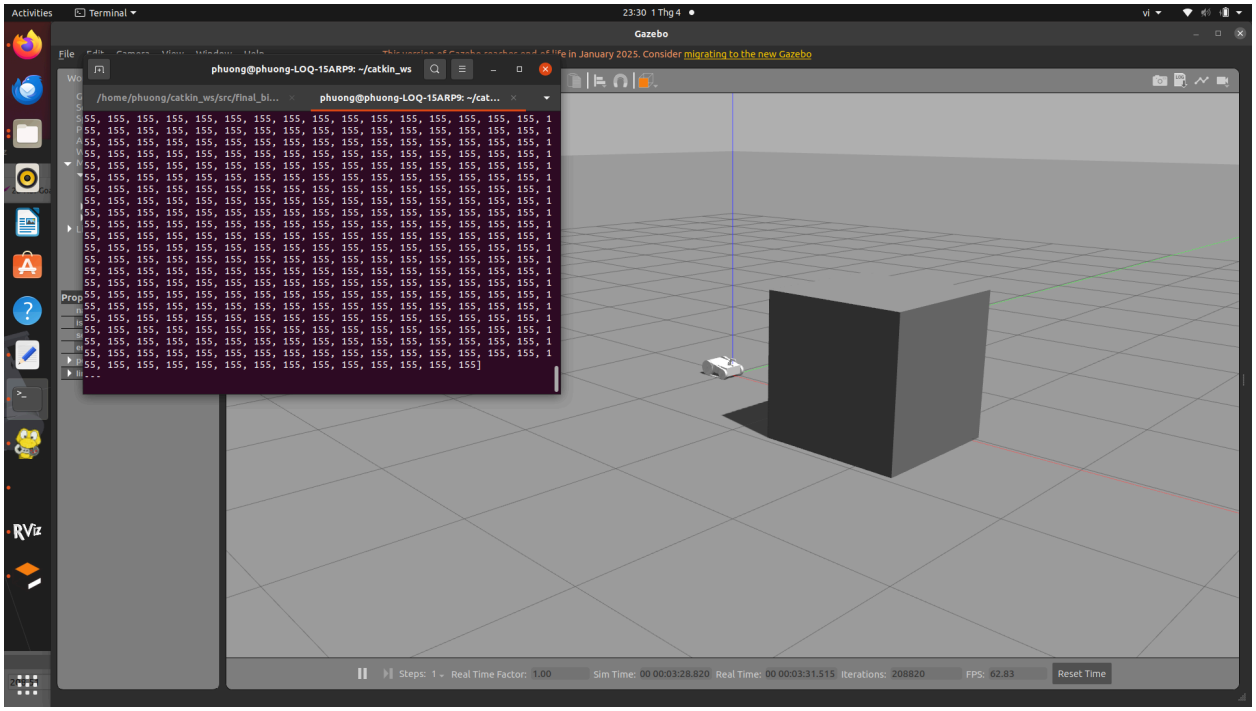
2) Hiển thị và kiểm tra mô phỏng trong Gazebo:
   + Sau khi mô tả robot xong, bạn sẽ sử dụng launch file để khởi động Gazebo và RViz. Gazebo sẽ mô phỏng hành vi vật lý của robot, bao gồm chuyển động của bánh xích và tay máy, trong khi RViz sẽ giúp bạn trực quan hóa và kiểm tra các cảm biến như camera và IMU.

→ Thiết kế Solidworks, cách đặt hệ trục tọa độ

→ Mô tả file urdf, liên kết của các link, các cảm biến, mô tả gazebo

3.1. camera





Camera trong Gazebo thường được khai báo trong file .xacro hoặc .sdf, bao gồm:

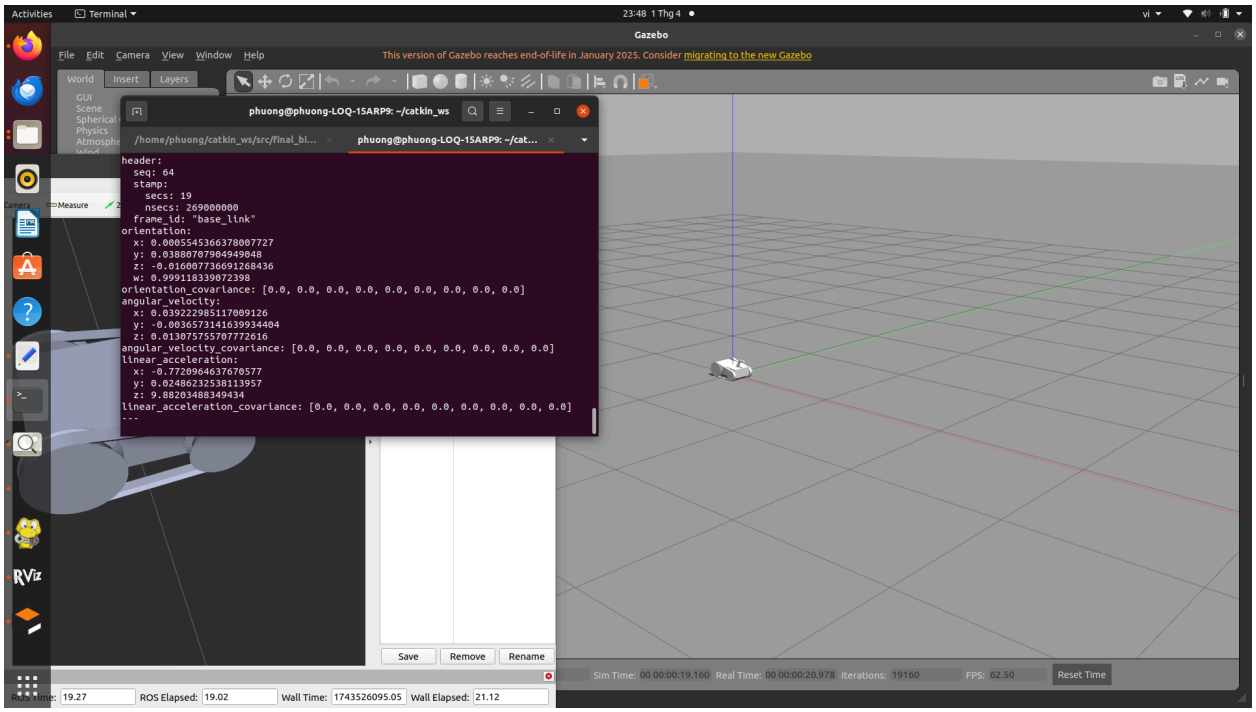
- Vị trí & hướng: Gắn trên robot với pose xác định.
- Thông số kỹ thuật: Độ phân giải, FOV (góc nhìn), tần số chụp.

- Chế độ hoạt động: RGB, grayscale, depth.

Mỗi pixel có một giá trị từ 0 đến 255, trong đó:

- 0 → Màu đen (cường độ thấp nhất).
- 255 → Màu trắng (cường độ cao nhất).
- Các giá trị giữa 0 - 255 → Các mức xám khác nhau.

3.2. IMU



1. Phân tích từng phần của dữ liệu

→ Header (Thông tin chung về dữ liệu)

- seq: 88 → Số thứ tự của gói tin IMU, cho biết đây là lần đo thứ 88.
- stamp: secs: 21, nsecs: 669000000 → Dấu thời gian khi dữ liệu được ghi nhận (21.669 giây từ khi hệ thống bắt đầu).
- frame_id: "base_link" → Dữ liệu IMU được ghi nhận theo hệ quy chiếu của thân robot (base_link).

→ Orientation (Tư thế quay của robot - Quaternion)

- Quaternion thể hiện tư thế của robot mà không bị lỗi góc gimbal lock như Euler angles:
- x: 0.000428, y: 0.038697, z: -0.013740, w: 0.999156
- Đây là góc quay của robot trong không gian 3D.

→ Chuyển đổi sang góc Euler (Roll, Pitch, Yaw)

- Roll (Nghiêng trái/phải): Nhỏ (~ 0.04 rad) → Robot gần như cân bằng theo trục X.
- Pitch (Nghiêng trước/sau): Nhỏ (~ -0.01 rad) → Robot không quá nghiêng về phía trước hoặc sau.
- Yaw (Xoay quanh trục đứng Z): Dữ liệu từ z, w cho thấy robot gần như quay đúng hướng ban đầu.

→ Angular Velocity (Vận tốc góc - rad/s)

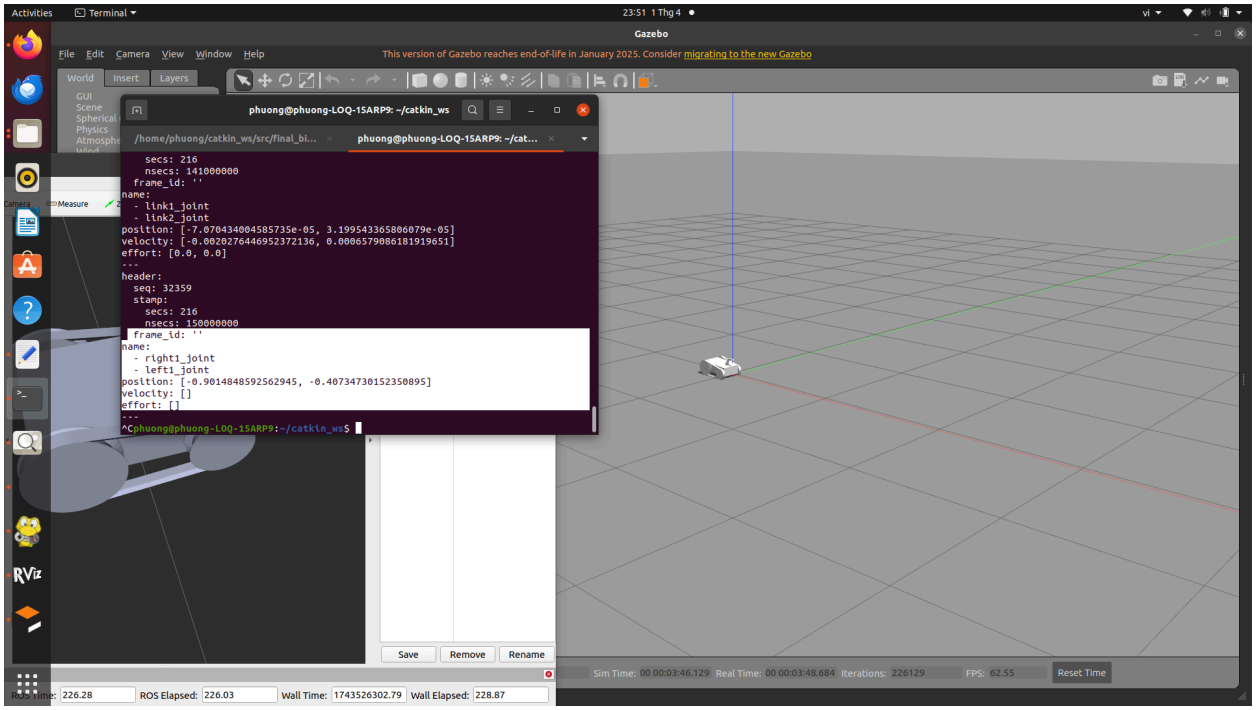
Đơn vị radian/giây, thể hiện tốc độ quay của robot quanh các trục X, Y, Z:

- x: -0.0008 rad/s → Robot gần như không quay quanh trục X.
- y: 0.0088 rad/s → Robot đang xoay rất nhẹ quanh trục Y.
- z: -0.0017 rad/s → Robot không có chuyển động xoay đáng kể quanh trục Z.

→ Linear Acceleration (Gia tốc tuyến tính - m/s^2)

- x: -0.768 → Robot có gia tốc âm theo trục X → có thể đang giảm tốc khi di chuyển về trước.
- y: 0.0061 → Gia tốc rất nhỏ theo trục Y → Không có nhiều di chuyển ngang.
- z: 9.874 → Gần bằng 9.81 m/s^2 (gia tốc trọng trường) → Robot đang trên mặt phẳng ngang.

3.3. encoder



Phân tích từng phần của dữ liệu

→ Header (Thông tin chung về dữ liệu)

- seq: 32359 → Đây là lần đo thứ 32,359 của encoder.
- stamp: secs: 216, nsecs: 150000000 → Dữ liệu được ghi nhận tại 216.15 giây kể từ khi hệ thống khởi động.
- frame_id: '' → Không có hệ quy chiếu cụ thể được chỉ định.

→ Name (Tên các joint được đo)

- right1_joint → Encoder đo góc quay của bánh xích bên phải.
- left1_joint → Encoder đo góc quay của bánh xích bên trái.

→ Position (Góc quay của bánh xích - radian)

- right1_joint: -0.9015 rad
- left1_joint: -0.4073 rad

→ Đây là góc quay tích lũy của các bánh xích, tức là tổng số radian mà bánh xích đã quay kể từ khi bắt đầu mô phỏng.

Ý nghĩa:

- Bánh xích phải đã quay nhiều hơn so với bánh xích trái.
- Giá trị âm có thể cho thấy robot đang di chuyển lùi hoặc sử dụng hệ tọa độ trong đó góc quay giảm khi quay theo một hướng nhất định.
- Sự chênh lệch giữa hai giá trị có thể cho thấy robot đang rẽ trái (bánh xích phải quay nhiều hơn).

→ Velocity (Không có dữ liệu về tốc độ)

- Trường velocity: [] rỗng → Encoder không ghi nhận dữ liệu tốc độ góc (rad/s).
- Nếu có dữ liệu, nó sẽ thể hiện tốc độ quay hiện tại của từng bánh xích.

→ Effort (Không có dữ liệu về lực mô-men xoắn)

- Trường effort: [] rỗng → Không có thông tin về lực tác động lên các joint.
- Nếu có, nó sẽ phản ánh mô-men xoắn (torque) tác động lên bánh xích.

. Mô tả cơ chế điều khiển trên gazebo

4. Điều khiển bánh xích để di chuyển robot

→ Robot bánh xích di chuyển dựa trên hai dải bánh xích trái và phải, điều khiển độc lập:

- Cả hai bánh xích quay cùng chiều, cùng tốc độ → Robot đi thẳng.
- Hai bánh xích quay ngược chiều nhau → Robot xoay tại chỗ.
- Bánh xích trái quay nhanh hơn bánh xích phải → Robot rẽ phải (và ngược lại).

→ Cách gửi lệnh điều khiển trong ROS1

- Robot nhận lệnh điều khiển từ topic /cmd_vel, sử dụng kiểu dữ liệu Twist (trong ROS1).
- Thông điệp Twist chứa thông tin vận tốc tuyến tính (đi thẳng, lùi) và vận tốc góc (quay trái, phải).

- Một node ROS sẽ đọc dữ liệu từ /cmd_vel và gửi điều khiển đến Gazebo để mô phỏng chuyển động.

Ví dụ:

- Di chuyển thẳng: Gửi lệnh với vận tốc tuyến tính dương (linear.x = 0.5).
- Rẽ phải: Giảm tốc độ bánh xích phải, tăng tốc độ bánh xích trái.
- Quay tại chỗ: Đặt linear.x = 0, angular.z = 1.0.

Cơ chế điều khiển tay máy (2 khớp xoay) trên Gazebo

→ Mô hình hóa hai khớp xoay trong URDF/Xacro

- Mỗi khớp xoay sẽ được mô tả bằng một joint kiểu "revolute", có giới hạn góc quay.
- Các khớp sẽ được kết nối tuần tự để tạo nên cánh tay robot.

→ Sử dụng ROS Controllers để điều khiển

- Sử dụng position_controllers/JointPositionController để đặt góc quay mong muốn.
- Hoặc dùng effort_controllers/JointEffortController nếu bạn muốn mô phỏng lực tác động lên khớp.

Gửi lệnh điều khiển từ ROS

- Bạn có thể sử dụng một node ROS để gửi lệnh đến /link1_joint/command và /link2_joint/command, đặt góc quay mong muốn cho từng khớp.

5. Các thành phần chính của code, structure folder dự án

1.1 Mô tả mô hình robot (my_robot_description)

- Thư mục này chứa các file định nghĩa cấu trúc vật lý của robot dưới dạng URDF/Xacro.
- Các thành phần quan trọng của robot được chia thành các file nhỏ như:
    + Mô tả thân robot và bánh xích: Bao gồm thông tin về kích thước, khớp nối và cách bánh xích di chuyển.
    + Mô tả tay máy (2 khớp xoay): Định nghĩa các khớp (revolute), giới hạn góc quay và cách tay máy kết nối với thân chính.

    + Mô tả cảm biến (IMU, Camera, Encoder): Xác định vị trí lắp đặt và thông số kỹ thuật của cảm biến.
      → Các file này được tổ chức theo kiểu module để dễ chỉnh sửa và mở rộng.

6. Điều khiển robot (my_robot_control)

- Chứa code và file cấu hình để điều khiển bánh xích và tay máy.
- Có hai phương thức điều khiển chính:
    + Điều khiển tự động (autonomous control): Sử dụng thuật toán để robot tự di chuyển theo cảm biến hoặc đường đi đã lập trình.
    + Điều khiển bằng bàn phím (teleop): Nhận lệnh từ người dùng để di chuyển robot.
    + Các bộ điều khiển trong ROS như Joint Position Controller hoặc Velocity Controller được sử dụng để điều khiển tay máy.
    + Có thể dùng ROS node để đọc lệnh từ topic /cmd_vel và gửi đến bánh xích.

→ Mô phỏng trên Gazebo (my_robot_gazebo)

- Chứa các file để thiết lập môi trường mô phỏng trên Gazebo.
- Bao gồm:
    + Cấu hình camera và cảm biến để thu nhận dữ liệu mô phỏng.
    + Mô hình thế giới (worlds/default.world): Xác định môi trường mô phỏng (địa hình, chướng ngại vật, ánh sáng, v.v.).
    + Plugin Gazebo: Cấu hình cách robot tương tác với môi trường như lực kéo, ma sát của bánh xích.