TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐẠI

HỌC QUỐC GIA HÀ NỘI

…….oOo…….

BÁO CÁO

Đề tài : Xe robot 4 bánh vi sai kết hợp tay máy 2 bậc tự do

Môn học : Lập trình ROS

Sinh viên thực hiện: Nguyễn Thành Trung – 22027516

I. THIẾT KẾ

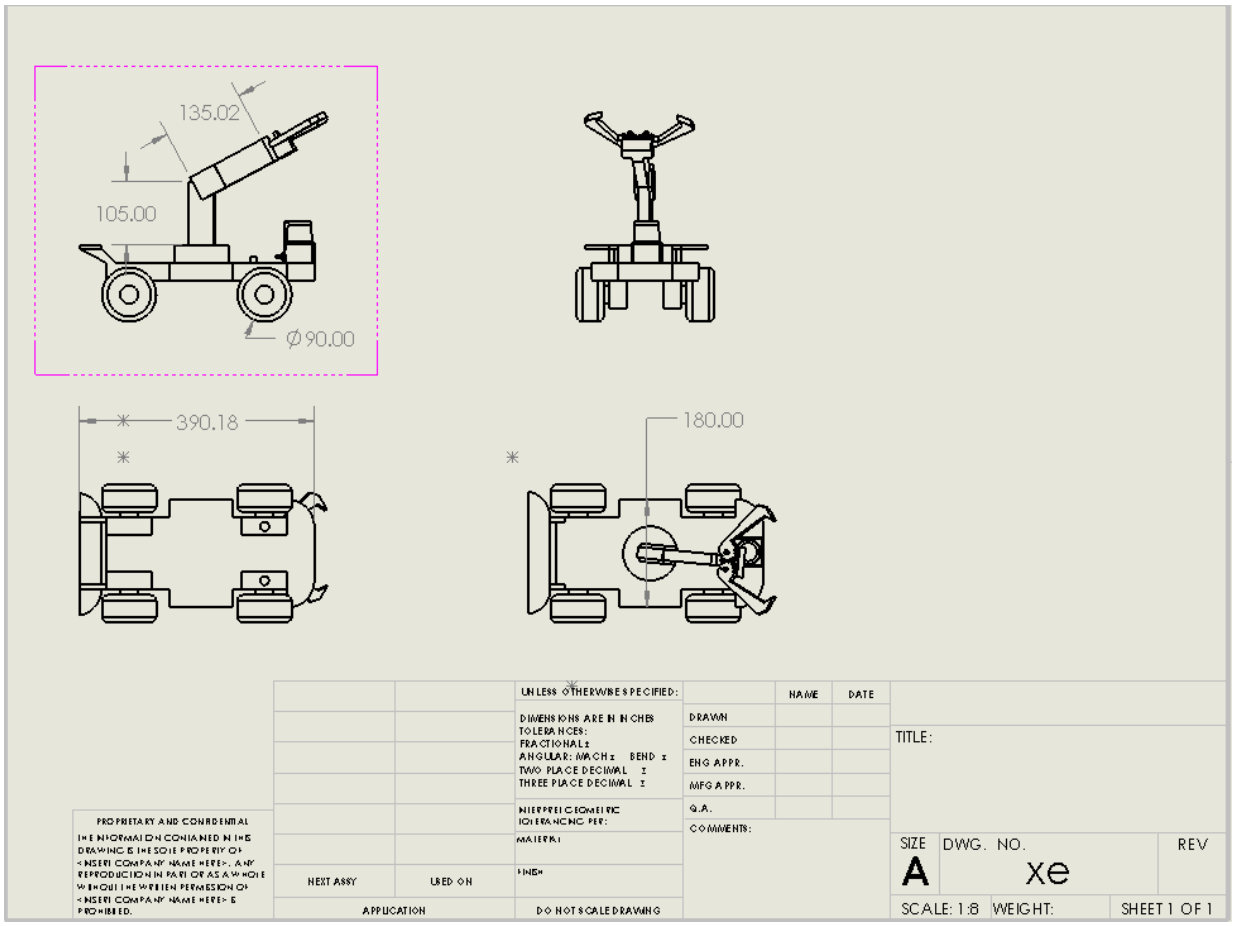
1. Mục tiêu thiết kế

- Xây dựng mô hình robot di chuyển linh hoạt trên địa hình phẳng.
- Kết hợp tay máy để thực hiện thao tác như gắp, nhặt hoặc di chuyển vật. Phục vụ mô phỏng trong môi trường Gazebo tích hợp với ROS1.

2. Tổng quan thiết kế

Robot bao gồm ba bộ phận chính:

- Khung gầm di động gắn 4 bánh xe.
- Tay máy gắn trên thân robot, có khả năng quay gắp vật ththể
- Các cảm biến LiDAR, IMU, Encoder được tích hợp để thu thập thông tin môi trường và trạng thái robot.



Kích thước:

Bộ phận	Kích thước
Thân robot	390 x 180 mm
Tay gắp arm1	105 mm
Tay gắp arm2	135 mm
Đường kính bánh xe	30 mm

3. Hệ thống di chuyển

Cấu trúc: Robot sử dụng hệ thống 4 bánh xe kiểu skid-steering (vi sai), mỗi cặp bánh hai bên hoạt động độc lập.

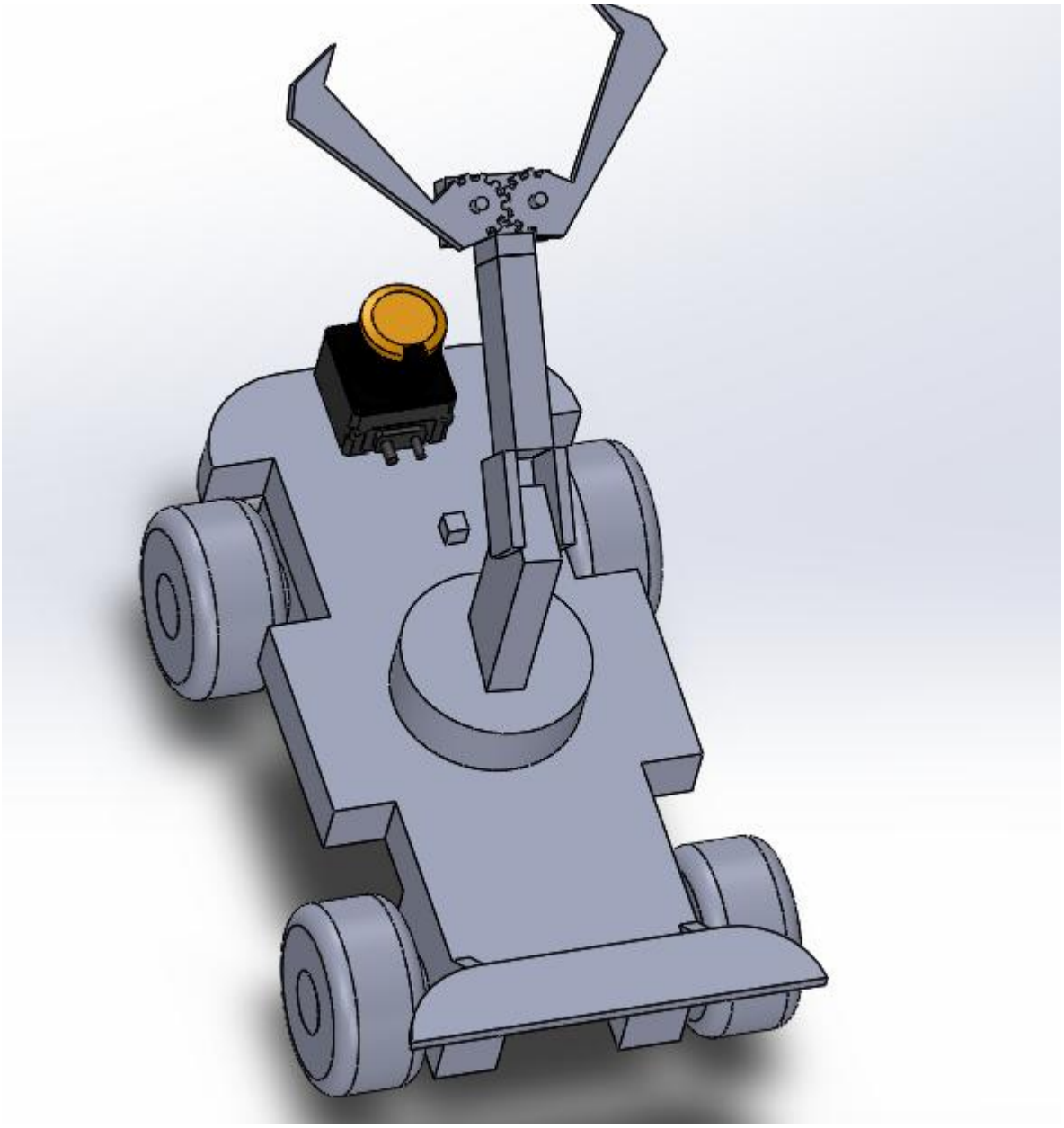
Kiểu khớp: URDF sử dụng joint kiểu "continuous" cho các bánh xe. Kích thước các bánh xe = 90mm

Khoảng cách bánh trước và sau: ~180 mm

Đặc điểm: Di chuyển đơn giản, dễ điều khiển trong giám sát môi trường phẳng.

Ưu điểm của 4 bánh vi sai

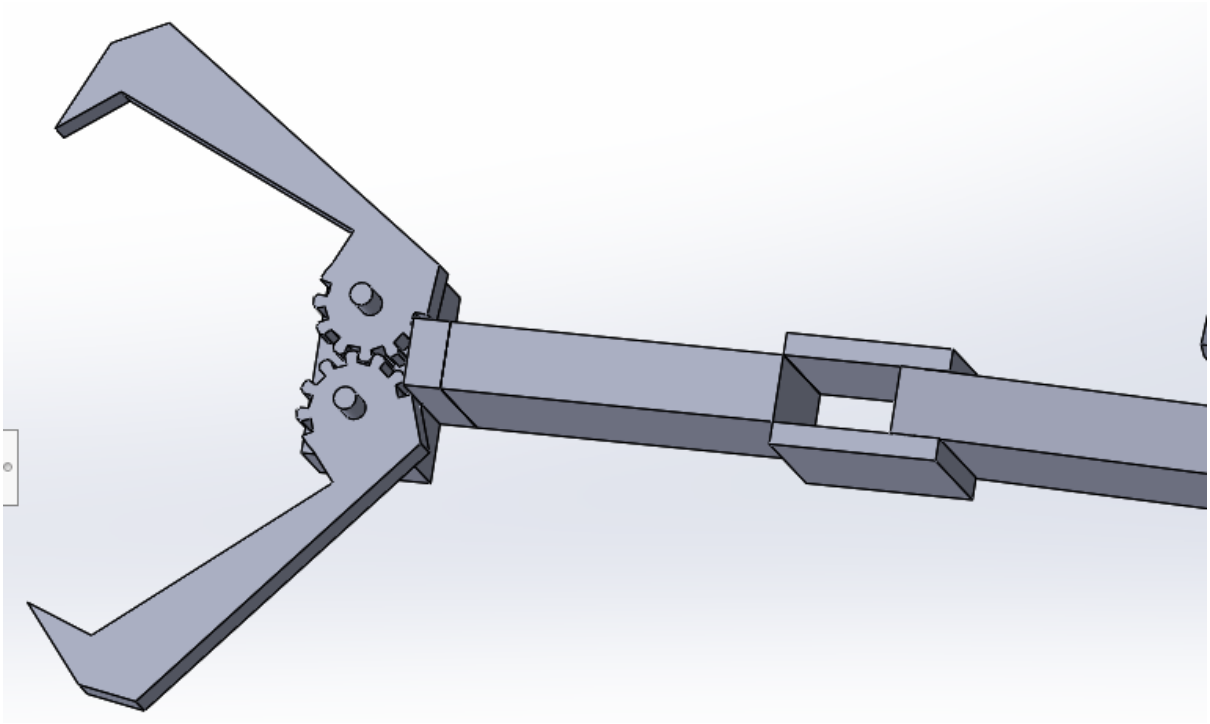
- Đơn giản, ổn định
- Dễ dàng mô phỏng và triển khai



bản vẽ 3d của mô hình xe robot 4 bánh kết hợp tay máy

4. Thiết kế tay máy

Hệ thống tay máy trên robot là một bộ phận quan trọng, đóng vai trò thực hiện các tác vụ thao tác như gắp, giữ, nâng, di chuyển hoặc đặt vật thể tại vị trí mong muốn. Trong thiết kế này, tay máy được tích hợp trực tiếp lên thân xe, hoạt động linh hoạt và phối hợp chặt chẽ với phần đế di chuyển.



Tay máy bao gồm 2 khớp quay (bậc tự do), cùng một cơ cấu chấp hành là càng gắp ở đầu cuối. Các bộ phận chính bao gồm:

- Cánh tay thứ nhất: gắn cố định với khung xe thông qua một khớp quay tại trục thẳng đứng (khớp cơ sở), cho phép tay máy quay ngang quanh trục z.

- Cánh tay thứ hai: nối với cánh tay thứ nhất thông qua khớp quay thứ hai, tạo khả năng nâng hạ cánh tay theo phương thẳng đứng.

- Càng gắp (Gripper): gồm hai ngón gắp đối xứng, được điều khiển bởi một cơ cấu truyền động độc lập để mở/đóng, phục vụ việc thao tác với vật thể.

II. ĐỘNG HỌC HỆ THỐNG

1. Động học hệ thống xe 4 bánh vi sai

Cấu trúc cơ bản:

Số bánh chủ động: 2 bánh (bên trái và phải).

Khoảng cách giữa hai bánh chủ động: L (track width).

Bánh xe giả định không bị trượt (no slip assumption).

Tâm robot: Đặt tại điểm giữa của hai bánh chủ động.

- Biến và thông số:

v: vận tốc tịnh tiến của robot (m/s)

ω: vận tốc góc của robot

v_L , v_R: vận tốc tuyến tính của bánh trái và bánh phải

r: bán kính bánh xe

- Công thức liên hệ:

vận tốc trái/phải:

$v_L = r * \omega_L$ $\quad\quad$ $v_R = r * \omega_R$

trong đó ω_L , ω_R là vận tốc góc bánh trái, bánh phải

Vận tốc và vận tốc góc của robot:

$v = \frac{v_L + v_R}{2}$ $\quad\quad$ $\omega = \frac{v_R - v_L}{L}$

Dạng ma trận:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix} \cdot \begin{bmatrix} v_R \\ v_L \end{bmatrix}$$

Nghịch đảo:

$$\begin{bmatrix} v_R \\ v_L \end{bmatrix} = \begin{bmatrix} 1 & \frac{L}{2} \\ 1 & -\frac{L}{2} \end{bmatrix} \cdot \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Từ đó, vận tốc robot trong hệ quy chiếu toàn cục:

$x' = v\cos(\theta)$ $y' = v\sin(\theta)$ $\theta' = \omega$

Ưu điểm:

Tính ổn định cao: do 4 điểm tiếp xúc với mặt đất.

Khả năng di chuyển chính xác: phù hợp cho các tác vụ trong không gian hẹp.

Dễ mô hình hóa và điều khiển: do chỉ cần 2 biến điều khiển (tốc độ trái và phải).

Khả năng chịu tải tốt do có 4 điểm chịu lực.

Hạn chế:

Không có khả năng quay tại chỗ như mecanum hoặc omni.

Khi tải trọng không đều, có thể gây ra sự sai lệch trong điều hướng.

Động học không thể hiện hiệu ứng trượt, cần mô hình động học mở rộng cho trường hợp thực tế.

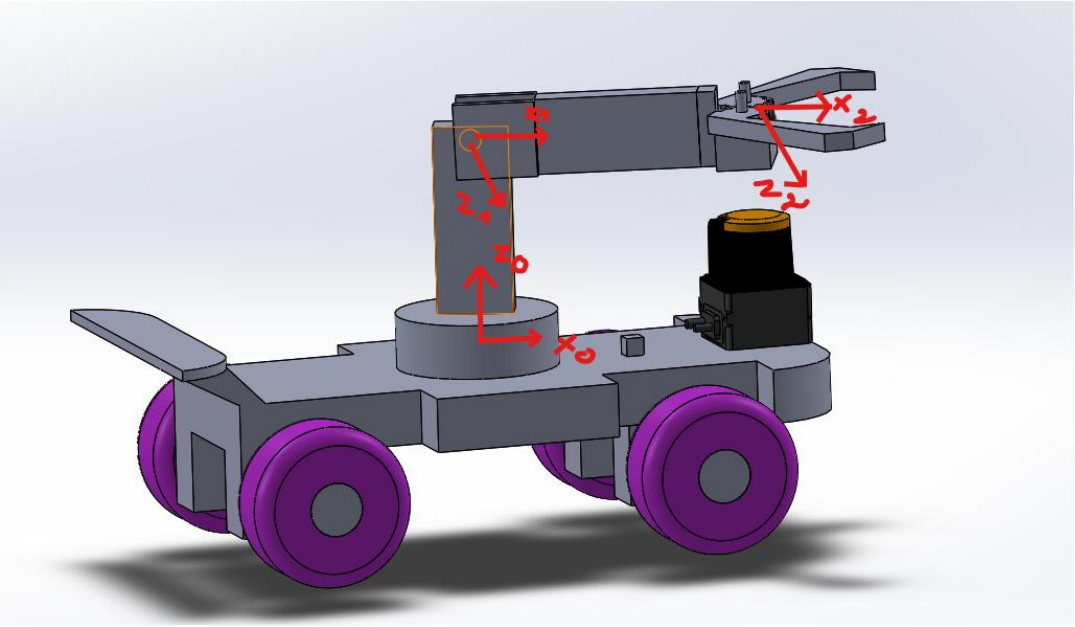
2. Động học tay máy

Tay máy được thiết kế gồm 2 khớp quay (revolute joints):

Khớp 1 (arm1_joint): Nối từ thân xe đến khâu thứ nhất (arm1_link) – quay quanh trục Z (thường là thẳng đứng).

Khớp 2 (arm2_joint): Nối từ arm1_link đến khâu thứ hai (arm2_link) – quay quanh trục Z (nằm ngang).

End effector: Gắn tại đầu arm2_link, là một bộ gắp hai ngón, hoạt động kẹp/nhả vật thể.



Bảng DH:

i	d_i	θ_i	a_i	$\propto_i$
1	L1	0	0	π/2
2	0	θ_2	L2	0

Ma trận biến đổi thuần nhất cho mỗi khớp

$$\mathbf{A_i}=\begin{bmatrix} \cos q & -\sin q\cos\alpha & \sin q\sin\alpha & a\cos q \\ \sin q & \cos q\cos\alpha & -\cos q\sin\alpha & a\sin q \\ 0 & \sin\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ma trận biến đổi thuần nhất cho EF: ${A_2}^0 = {A_1}^0 * {A_2}^1$

Vị trí tay gắp (end-effector) tính được theo:

$$x = l2cos(\theta 1)cos(\theta 2)$$

$$y = l2sin(\theta 1)\, cos(\theta 2)$$

$$z = l1 + l2sin(\theta 2)$$

Từ động học thuận, có thể tính toán vị trí tay gắp, qua đó điều khiển tay máy chính xác

III. ĐẶT TRỤC TỌA ĐỘ TRONG SOLIDWORKS

1. Mục tiêu

Việc đặt trục tọa độ hợp lý trong SolidWorks là bước quan trọng nhằm đảm bảo tính nhất quán khi xuất mô hình sang định dạng URDF để mô phỏng trong ROS và Gazebo. Hệ trục này quyết định cách robot sẽ được hiển thị, điều hướng và mô phỏng trong môi trường ảo.

2. Quy ước hệ trục trong robot

Theo chuẩn ROS, hệ trục tọa độ được quy ước như sau:

- Trục X: Hướng về phía trước robot (forward).
- Trục Y: Hướng về bên trái robot (left).
- Trục Z: Hướng lên trên (upward).

3. Đặt trục trong solidworks

Trong thiết kế mô hình robot 4 bánh có tay máy, mỗi bộ phận đều được gán một Coordinate System (CS) riêng trong SolidWorks để đảm bảo khi xuất URDF, việc xác định vị trí và hướng của các link và joint được chính xác, đồng bộ với quy ước trong ROS.

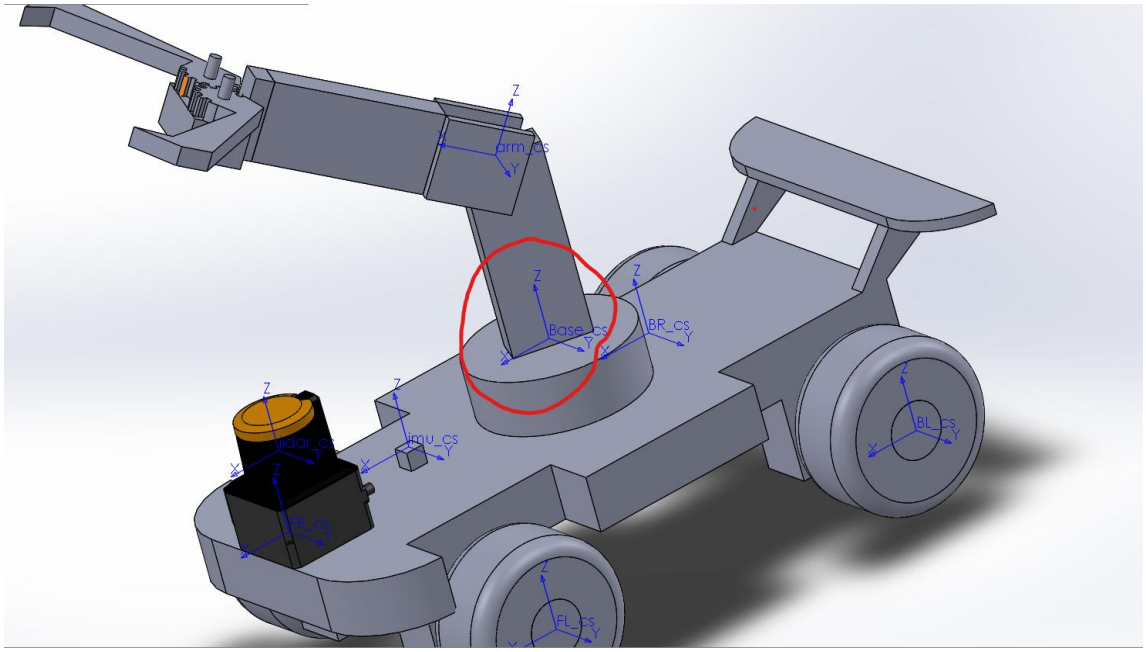
- Hệ tọa độ gốc của robot: base_cs

Vị trí: Đặt tại trung tâm của thân xe robot, nơi đặt khớp xoay tay máy.

Hướng trục:

X: Hướng về phía trước robot (hướng di chuyển chính).
Y: Hướng sang trái robot.
Z: Hướng lên trên.

Đây là gốc hệ quy chiếu chính của toàn bộ robot, phù hợp với chuẩn ROS. Tất cả các bộ phận khác sẽ có vị trí và chuyển động tương đối so với hệ trục này.



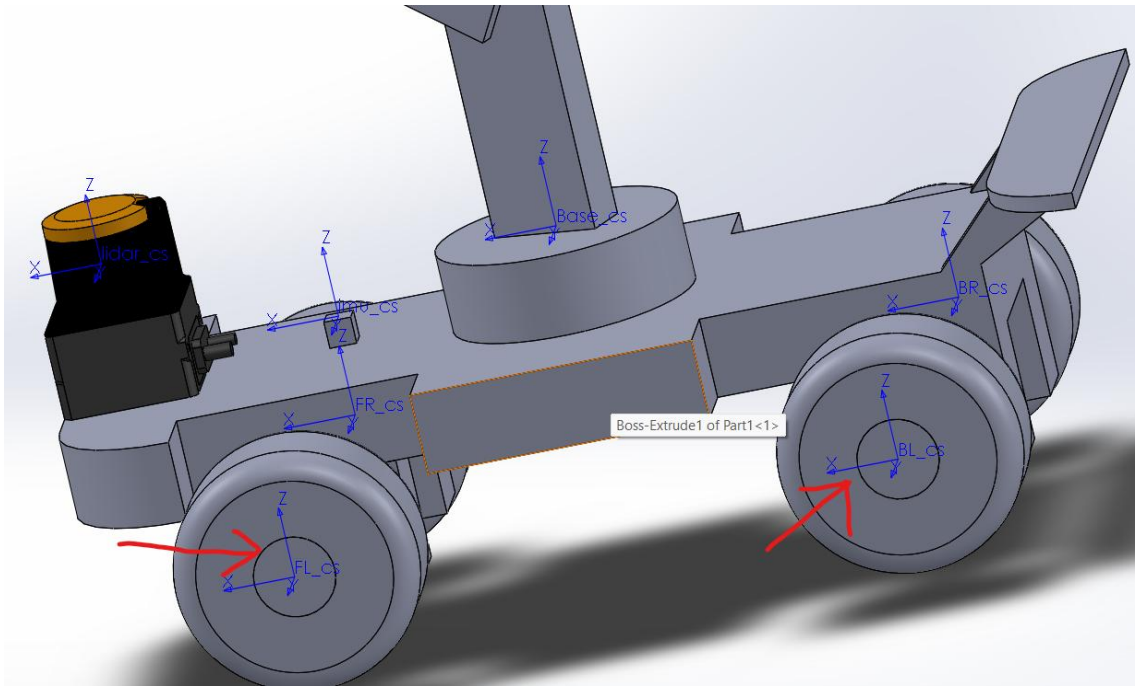
- Trục quay cho các bánh xe: FL_cs, BL_cs, FR_cs, BR_cs

Vị trí: Tâm các bánh xe.

Hướng trục:

X: Hướng phía trước, trùng chiều di chuyển của xe

Z: Hướng lên.



- Hệ trục cho khớp quay của tay máy (arm1) - base_cs

Vị trí: Gốc khớp nối giữa base_link và arm1_link.

Hướng trục:

Z: Trùng với trục quay của khớp đầu tiên (arm1_joint).
X, Y: Theo mặc định SolidWorks.

Để đơn giản, không phải đặt thêm trục tọa độ không cần thiết, em dùng tọa độ của thân robot làm tọa độ cho khớp xoay tay máy arm1, đảm bảo arm1 quay theo đúng trục mong muốn. Trục Z thường là trục xoay mặc định cho khớp revolute.

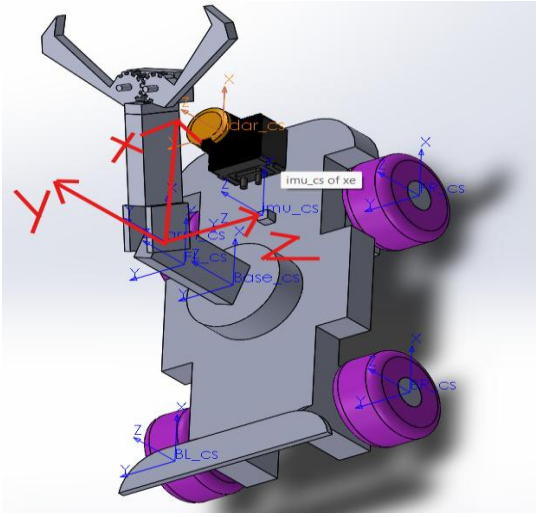
- Hệ trục cho khớp xoay của tay máy 2 - arm_cs

Vị trí: Khớp nối giữa arm1_link và arm2_link.

Hướng của các trục tọa độ:

Z: Được định hướng vuông góc với mặt phẳng xoay của khớp, trùng với trục quay của arm2_joint (dạng khớp revolute). Đây là trục chuyển động chính của khớp.
X: Được định hướng trùng với chiều dài của khâu arm2_link, tức là trục song song thân cánh tay phía sau khớp.
Y: Xác định theo quy tắc bàn tay phải để tạo thành một hệ trục orthonormal (vuông góc, thuận tay phải).

Trục Z trùng với trục hoạt động của actuator, thuận tiện cho việc mô phỏng, điều khiển và truyền động.
Trục X dọc theo khâu giúp đơn giản hóa mô hình động học thuận/nghịch.



- Hệ trục cho lidar - lidar_cs

Vị trí: Tâm quay của LiDAR.

Hướng trục:

- Z: Hướng lên
- X: Hướng về phía trước robot. (hướng tia quét)

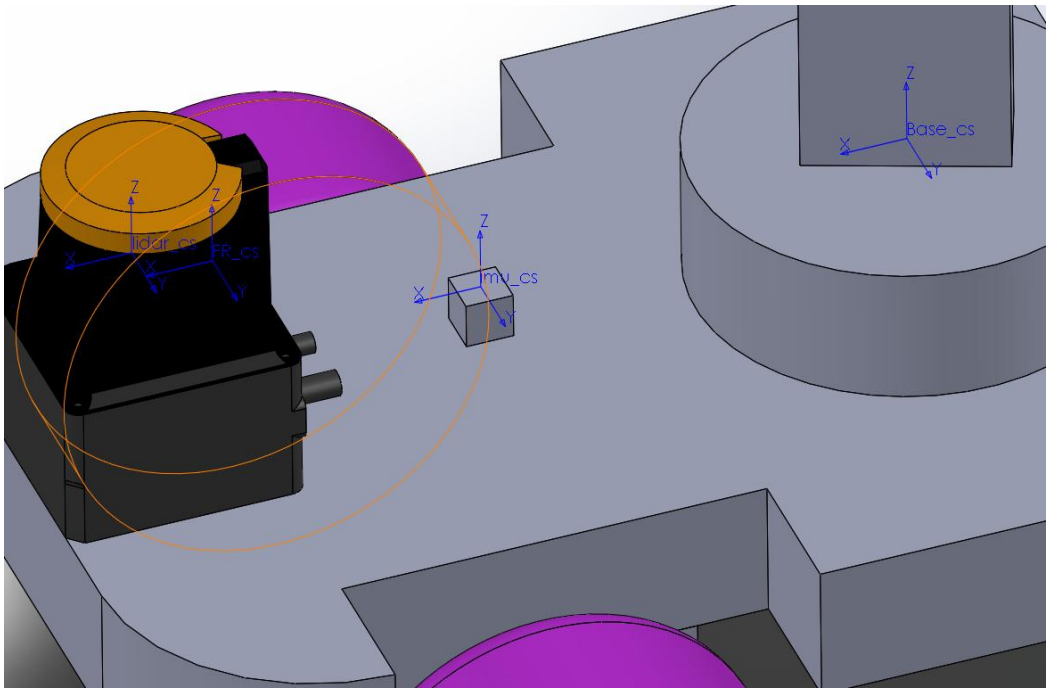
Phù hợp với chuẩn quét tia 2D của LiDAR (quét trong mặt phẳng X-Y, trục Z vuông góc với mặt phẳng quét).

- Hệ trục cảm biến IMU - imu_cs

Vị trí: Tâm IMU.

Hướng trục:

Z: Hướng lên (theo chiều trọng lực dương).
X: Hướng về trước robot.



IV. MÔ TẢ FILE URDF

1. Giới thiệu tổng quan

File URDF mô tả một robot di động có 4 bánh xe kết hợp với một tay máy 2 bậc tự do. Mô hình được xây dựng bằng công cụ SolidWorks và xuất qua plugin SW2URDF Exporter. Robot hoạt động trong môi trường mô phỏng Gazebo, tích hợp với hệ điều hành robot ROS1 (Noetic).

Robot có các thành phần chính:

- Thân chính (base_link)
- 4 bánh xe (FL, FR, BL, BR)
- Tay máy với 2 khớp quay (arm1_joint, arm2_joint)
- Cảm biến LiDAR và IMU
- Plugin điều khiển trong Gazebo

2. Cấu trúc các liên kết (links) và khớp nối (joints)

Thân chính (base_link)

- Là trung tâm kết nối các thành phần còn lại của robot.
- Có đầy đủ mô tả khối lượng, quán tính và mesh 3D.

Bánh xe

Robot sử dụng 4 bánh mô tả bởi 4 link và joint tương ứng:

Bánh xe	Link	Joint	Loại khớp	Trục quay
Trước trái (FL)	link_FL	joint_FL	continuous	xyz="0 1 0"
Sau trái (BL)	link_BL	joint_BL	continuous	xyz="0 1 0"
Sau phải (BR)	link_BR	joint_BR	continuous	xyz="0 1 0"
Trước phải (FR)	link_FR	joint_FR	continuous	xyz="0 1 0"

Tay máy (2 bậc tự do)

Khớp 1 (arm1_joint): kết nối từ base_link đến arm1_link.

- Loại khớp: revolute
- Trục quay: z (xyz="0 0 1")
- Giới hạn: từ -1.57 đến 1.57 rad

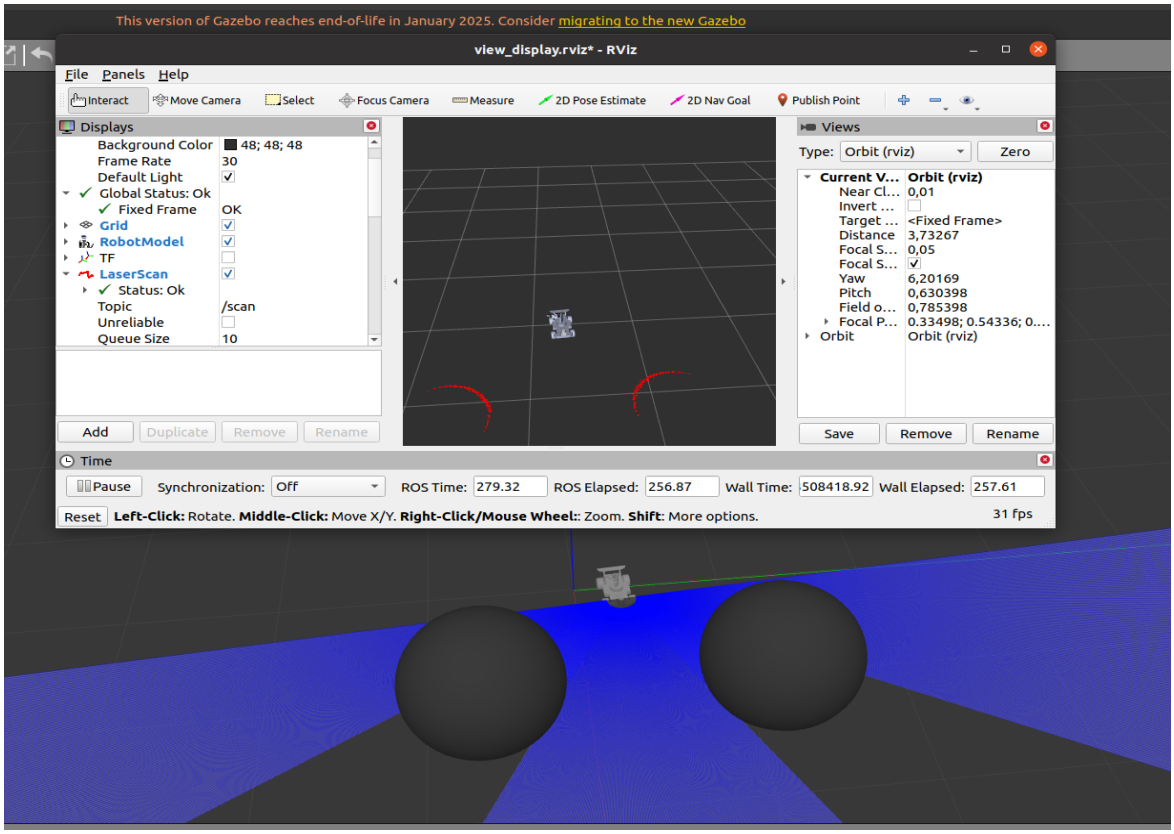
Khớp 2 (arm2_joint): kết nối từ arm1_link đến arm2_link

- Loại khớp: revolute
- Trục quay: y (xyz="0 1 0")
- Giới hạn: từ -0.3 đến 1.3 rad
- Có xoay lệch ban đầu (rpy="-0.61041 -0.52197")
-

Cảm biến

a. LiDAR (link_lidar)

- Gắn cố định (fixed) trên base_link tại vị trí phía trước đỉnh thân xe.
- Gồm sensor loại ray (mô phỏng cảm biến LiDAR xoay).
- Góc quét: từ -90° đến +90°, 720 mẫu (1 mẫu = 0.25°)
- Topic: /scan



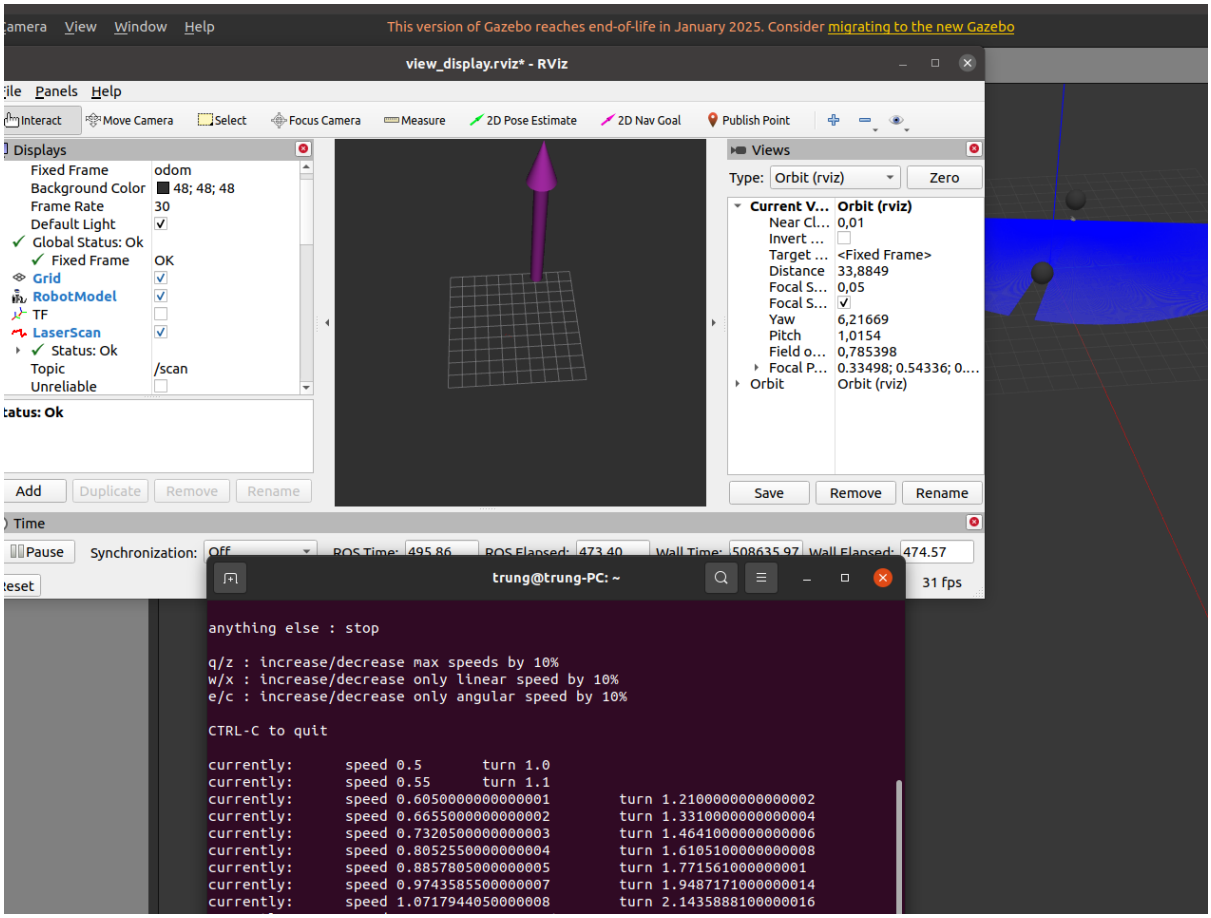
RViz hiển thị dữ liệu LaserScan:

- Dữ liệu từ topic /scan đã được hiển thị rõ ràng trong RViz dưới dạng hình vòng cung màu đỏ.
- Điều này xác nhận rằng plugin LiDAR đã cấu hình đúng trong mô phỏng và dữ liệu được publish chính xác.

b. IMU (imu_link)

- Gắn cố định trên thân robot.
- Mô phỏng IMU 3 trục
- Cập nhật 100 Hz.
- Topic: /imu/data

Kết quả:



c. Encoder (thông qua /odom)

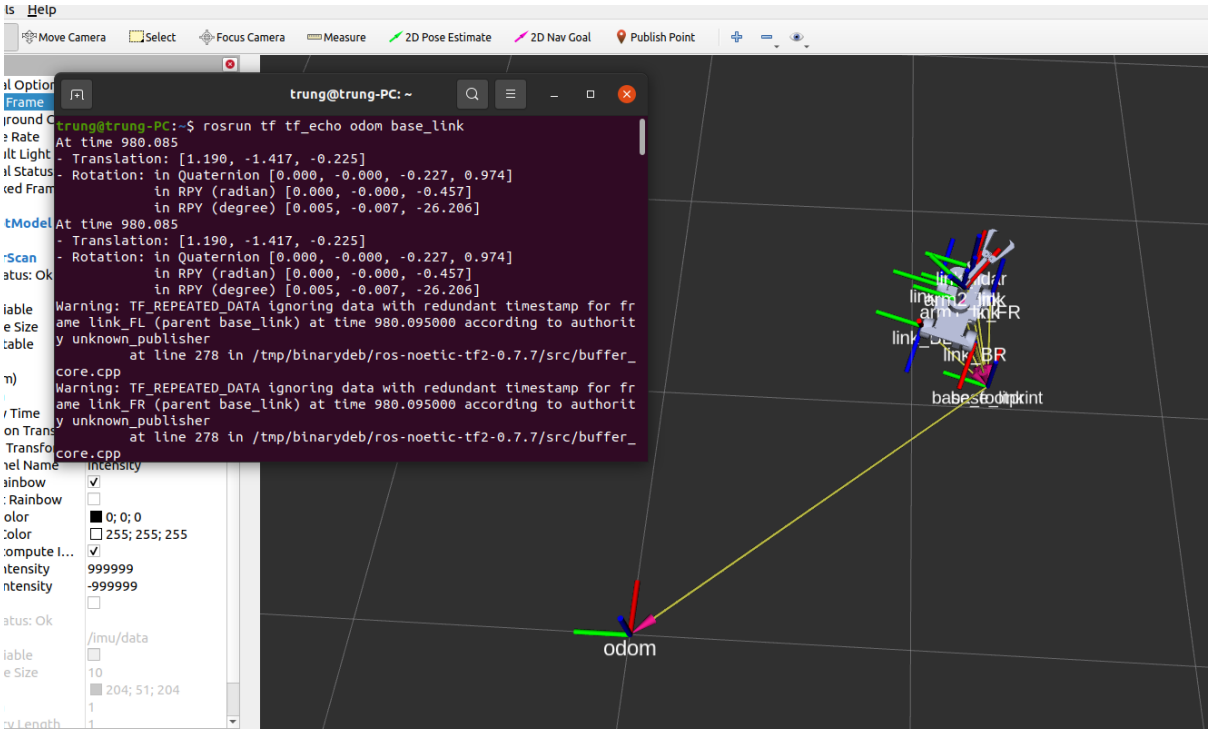
- **Nguồn dữ liệu:** Thông qua các bộ điều khiển bánh xe (diff_drive_controller) và cảm biến encoder trên các khớp (joint).

- **Chức năng:** Encoder đo số vòng quay bánh xe, từ đó tính toán:
    - Vị trí tương đối của robot so với vị trí ban đầu
    - Tốc độ tuyến tính và góc quay (twist)
    -

- **Tích hợp với ROS:** Bộ điều khiển diff_drive_controller sử dụng thông tin encoder để xuất bản:

Topic: /odom chứa thông tin vị trí và vận tốc của robot

Kết quả:



Từ topic /odom, có thể thấy robot đang cách odom với giá trị

[x, y, z] = [1.19, -1.417, -0.225]

z khác 0 do trong quá trình thiết kế trên solidworks, em không vẽ theo tọa độ gốc mà bị thấp hơn so với tọa độ gốc 1 chút dẫn đến z khác 0

3. Plugin điều khiển trong Gazebo

Plugin gazebo_ros_control

- Cho phép điều khiển các khớp tay máy (EffortJointInterface)
- Là điều kiện bắt buộc để sử dụng controller từ ROS (ví dụ: effort_controllers/JointPositionController)

Plugin differential_drive_controller

2 plugin điều khiển vi sai

- Một cho cặp bánh trái (joint_BL) và bánh phải sau (joint_BR)
- Một cho cặp bánh trước trái (joint_FL) và phải (joint_FR)

Thông số:

- commandTopic: /cmd_vel
- odometryTopic: /odom
- wheelSeparation: 0.18 m
- wheelDiameter: 0.03 m

Truyền động (Transmission) của tay máy

Được khai báo để sử dụng với gazebo_ros_control:

Joint	Interface	Truyền động
arm1_joint	EffortJointInterface	SimpleTransmission
arm2_joint	EffortJointInterface	SimpleTransmission

Nhờ đó có thể điều khiển tay máy bằng effort controllers từ ROS như:

effort_controllers/JointPositionController

V. CƠ CHẾ ĐIỀU KHIỂN ROBOT 4 BÁNH + TAY MÁY TRONG GAZEBO

1. Tổng quan hệ thống

Robot được mô phỏng trong môi trường ROS1 Noetic + Gazebo, bao gồm:

- Hệ thống cơ khí: robot di động 4 bánh vi sai, tích hợp tay máy 2 bậc tự do.
- Cảm biến: LiDAR, IMU, Encoder
- Cơ chế điều khiển:
  - ✓ Bánh xe: điều khiển bằng plugin gazebo_ros_diff_drive thông qua lệnh /cmd_vel từ bàn phím (teleop).
  - ✓ Tay máy: điều khiển vị trí các khớp bằng effort_controllers/JointPositionController.

2. Điều khiển chuyển động xe 4 bánh

<u>Plugin điều khiển vi sai (gazebo_ros_diff_drive)</u>

Khai báo plugin differential_drive_controller:

```
<gazebo>
  <plugin name="differential_drive_controller"
filename="libgazebo_ros_diff_drive.so">
        <leftJoint>joint_BL</leftJoint>
        <rightJoint>joint_BR</rightJoint>
        <commandTopic>cmd_vel</commandTopic>
        <odometryTopic>odom</odometryTopic>
        ...
  </plugin>
</gazebo>
```

Chức năng:

- Nhận lệnh tốc độ từ topic /cmd_vel.
- Tính toán tốc độ góc cho bánh trái – phải dựa trên tốc độ tuyến tính và góc.
- Mô phỏng chuyển động bánh và publish TF từ odom → base_link.
- Xuất dữ liệu odometry trên /odom.

Điều khiển bằng bàn phím (teleop)

- Cài gói: *sudo apt install ros-noetic-teleop-twist-keyboard*

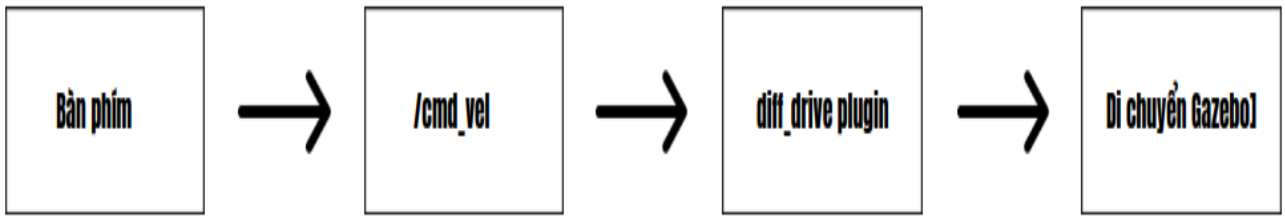
- Khởi chạy node teleop:

rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/cmd_vel

- Kết nối với plugin:

- Lệnh từ teleop_twist_keyboard sẽ gửi thông điệp geometry_msgs/Twist lên /cmd_vel.
- Plugin gazebo_ros_diff_drive nhận lệnh này và tính toán vận tốc bánh → điều khiển robot di chuyển trong Gazebo.

Tổng quan:



3. Cơ chế điều khiển tay máy

Plugin điều khiển cơ bản

gazebo_ros_control plugin được khai báo trong URDF

<gazebo>

<plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so"/>

</gazebo>

Chức năng:
Cho phép Gazebo mô phỏng tương tác phần cứng, từ đó ROS có thể sử dụng các controller như JointPositionController, JointEffortController, JointVelocityController để điều khiển các khớp của robot.

Yêu cầu: Mỗi khớp phải có phần **transmission** tương ứng để ánh xạ controller ROS với actuator mô phỏng trong Gazebo.

Mô tả điều khiển tay máy

- Truyền động (transmission) cho tay máy:

```
<transmission name="arm1_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="arm1_joint">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="arm1_motor">
      <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
```

Gồm 2 khớp:

- arm1_joint: quay quanh trục z (đế tay máy).
- arm2_joint: quay quanh trục y (liên kết cánh tay thứ hai).

Cả hai khớp đều sử dụng **EffortJointInterface**, tức là điều khiển bằng mô-men xoắn (torque).

- Cấu hình controller từ file arm_controller.yaml

arm1_joint_position_controller:

type: effort_controllers/JointPositionController

joint: arm1_joint

pid: {p: 10.0, i: 0.01, d: 0.1}

arm2_joint_position_controller:

type: effort_controllers/JointPositionController

joint: arm2_joint

pid: {p: 10.0, i: 0.01, d: 0.1}

joint_state_controller:

type: joint_state_controller/JointStateController

publish_rate: 50

a. arm1_joint_position_controller

- **Loại:** JointPositionController
- **Giao diện:** Effort
- **Chức năng:** điều khiển vị trí góc của khớp arm1_joint bằng cách áp dụng mô-men xoắn.
- **PID**: bộ điều khiển với thông số P = 10.0, I = 0.01, D = 0.1.
  Việc đặt hệ số tỉ lệ (Proportional gain) P = 10.0 giúp khớp phản ứng nhanh hơn với sai số vị trí giữa góc hiện tại và góc mục tiêu.

b. arm2_joint_position_controller

Tương tự như arm1, điều khiển vị trí của arm2_joint.

c. joint_state_controller

Dùng để **xuất trạng thái của tất cả các khớp** (góc, vận tốc, effort) lên topic /joint_states, hỗ trợ hiển thị trong RViz, ghi log hoặc phản hồi điều khiển.

- Cơ chế hoạt động tổng thể

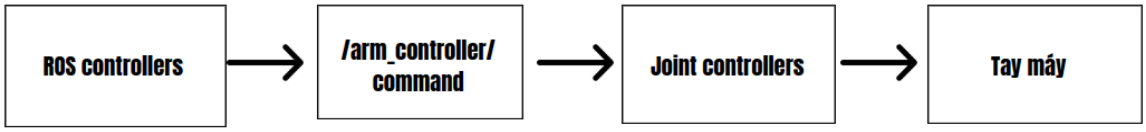
Khi khởi động Gazebo, plugin gazebo_ros_control sẽ kết nối mô phỏng với các controller ROS.

Các controller từ file arm_controller.yaml được load bằng launch file

Khi người dùng gửi lệnh điều khiển (ví dụ qua ROS topic /arm1_joint_position_controller/command), controller sẽ tính toán mô-men xoắn cần thiết để di chuyển khớp đến vị trí mong muốn.

Gazebo nhận giá trị effort, áp dụng vào khớp, mô phỏng động học vật lý thực tế.

joint_state_controller đảm bảo trạng thái khớp được cập nhật liên tục lên /joint_states → phục vụ hiển thị hoặc đóng vòng điều khiển.



Trong quá trình điều khiển tay máy, phải chú ý set limit hợp lí để tránh lỗi vật lý trong gazebo. Sau nhiều lần thử, tham số được đặt ra như sau

<limit lower="-1.57" upper="1.57" effort="5" velocity="1"/>

Để dễ quan sát tay máy chuyển động, tạo file .move_arm_loop.bash cho 2 tay máy chuyển động trong 1 vòng loop

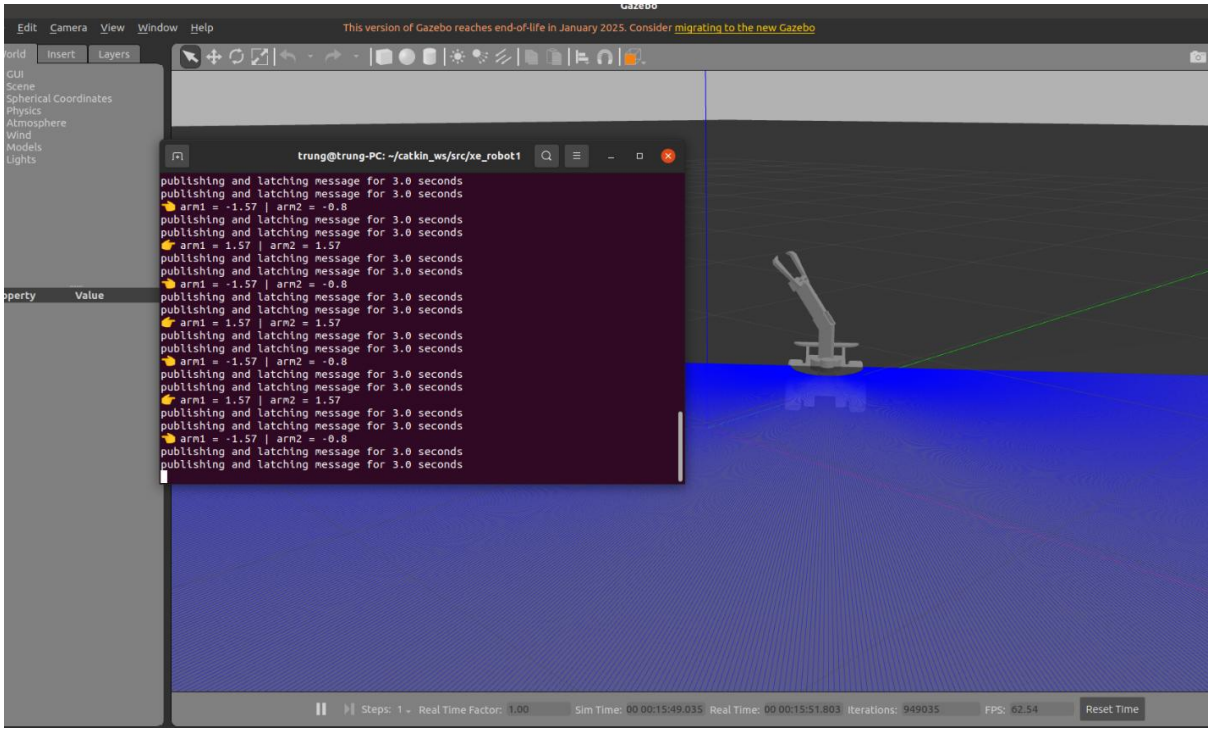
```
#!/bin/bash

echo "🔁 Đang điều khiển tay máy... (nhấn Ctrl + C để dừng)"

while true; do
  echo "👉 arm1 = 1.57 | arm2 = 1.57"
  rostopic pub -1 /arm1_joint_position_controller/command std_msgs/Float64 "data: 1.57"
  rostopic pub -1 /arm2_joint_position_controller/command std_msgs/Float64 "data: 1.57"
  sleep 3

  echo "👈 arm1 = -1.57 | arm2 = -0.8"
  rostopic pub -1 /arm1_joint_position_controller/command std_msgs/Float64 "data: -1.57"
  rostopic pub -1 /arm2_joint_position_controller/command std_msgs/Float64 "data: -0.8"
  sleep 3
done
```

Kết quả



TỔNG KẾT

Trong bài tập này, em đã thành công mô phỏng và điều khiển xe robot 4 bánh kết hợp tay máy. Mô hình được mô tả bằng URDF chi tiết, kết hợp với các plugin hỗ trợ điều khiển và cảm biến, giúp tái hiện sinh động chuyển động và phản hồi của robot trong không gian 3 chiều.

Trong giai đoạn tiếp theo, hệ thống có thể được mở rộng để giải quyết bài toán Navigation (điều hướng robot trong môi trường có chướng ngại vật)