ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

---***---

BÀI BÁO CÁO

MÔN HỌC: LẬP TRÌNH ROBOT VỚI ROS

Họ và tên sinh viên: Nguyễn Tuấn Anh

MSV: 22027514

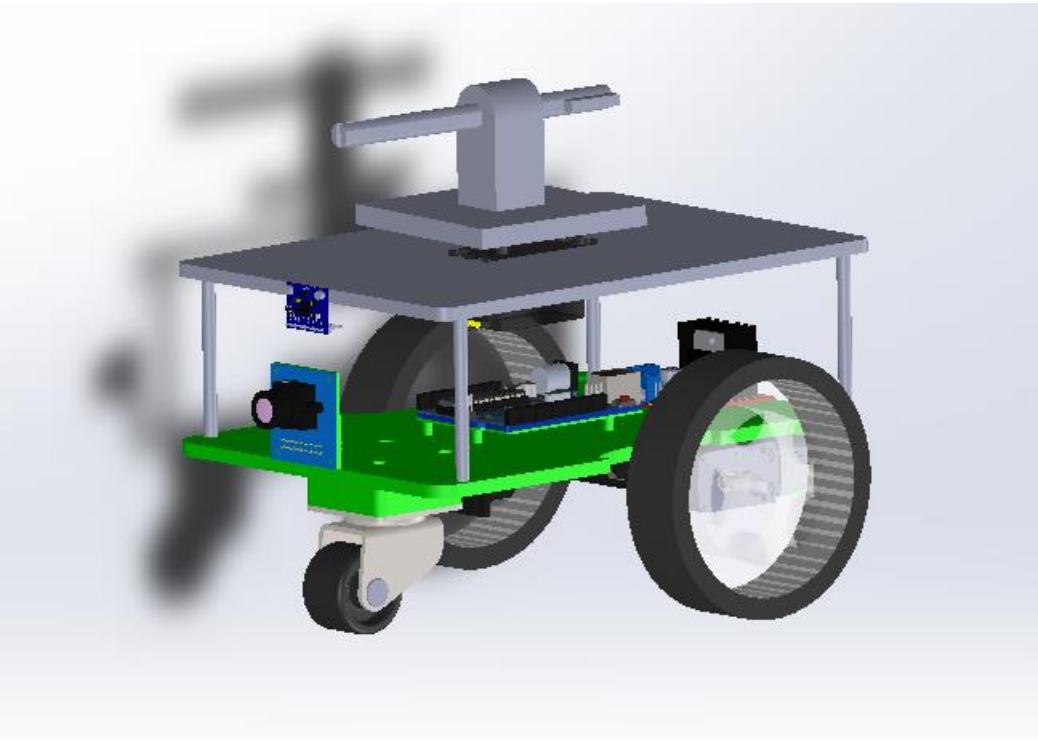
- Ngày 20 tháng 1 năm 2024 -

Mục lục trình bày báo cáo

TRÌNH BÀY BÁO CÁO

I. Dạng robot, động học và kích thước

1. Dạng robot



- Loại di chuyển: Robot sử dụng 2 bánh vi sai *(differential drive)*, thể hiện qua hai bánh lớn hai bên và một bánh tự do phía trước.

-Tay máy (khớp 1 và khớp 2):

+Khớp 1: khớp xoay *(rotation)*

+Khớp 2: khớp tịnh tiến *(prismatic)*

-Cảm biến:

+Robot có một **Camera** phía trước.

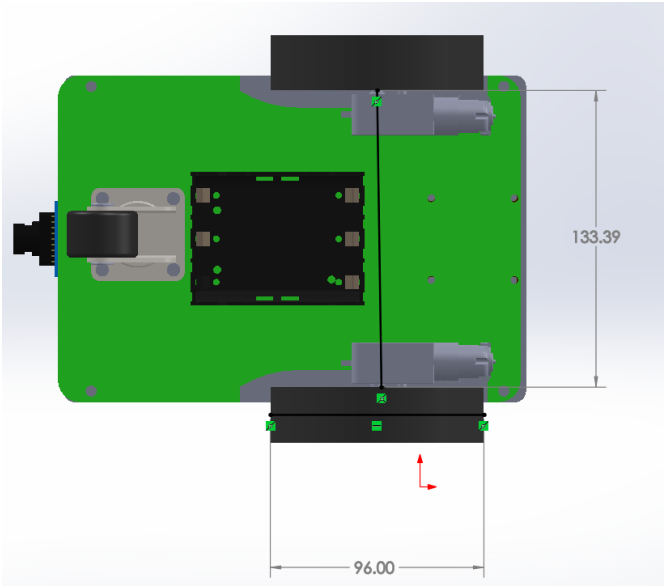
+Có **Encoder** trên bánh xe để đo vận tốc.

+Có **cảm biến LiDAR** để hỗ trợ điều hướng.

=>Dựa trên các yếu tố trên, robot được sử dụng cho bài tập kiểm tra giữa kỳ với chủ đề "Robot 2 bánh vi sai, tay máy có 1 khớp quay và 1 tịnh tiến, cảm biến **gồm LiDAR, Camera, Encoder".**

2. Kích thước

a. Kích thước ảnh hưởng đến sự di chuyển

b. 

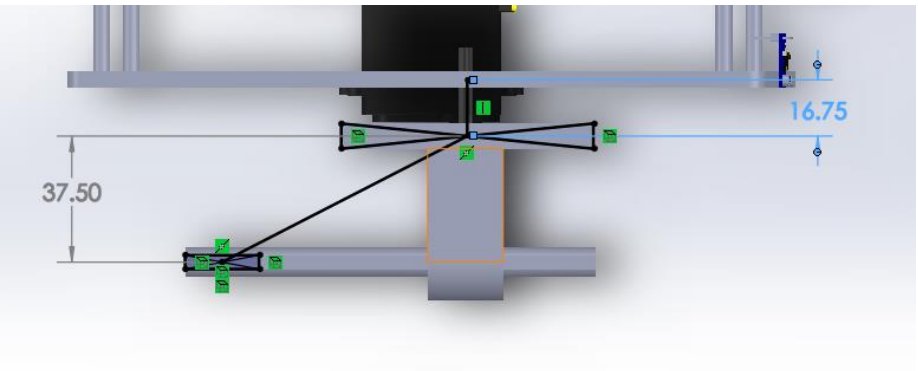
-Khoảng cách giữ 2 bánh: 133.39mm

-Đường kính bánh xe: 96mm

-Tốc độ động cơ: sử dụng động cơ DC giảm tốc vàng có tốc độ 208rpm (≈3,5 vòng / giây)

c. Kích thước liên qua đến tay máy: ta cần xác định các tham số sau

- ✓ $a_i = O_{i'}O_i$ Khoảng cách giữa 2 trục z
- ✓ $d_i = O_{i-1}O_{i'}$ dịch chuyển dọc theo trục z_{i-1}
- ✓ α_i: góc quay quanh trục x_i(góc giữa trục z_{i-1} và z_i)
- ✓ θ_i: góc quay quanh trục z_{i-1} (góc giữa trục x_{i-1} và x_i)



-Tham số từ động cơ đên khớp xoay

+ a1= 0

+ d1= 16.75

+ α1= 0

+ θ1= θ (phụ thuộc vào chuyển động của khớp xoay)

-Tham số từ khớp xoay đến khớp tịnh tiến

+ a2= a (phụ thuộc vào chuyển động khớp tịnh tiến)

+ d2= 16.75 (mm)

+ α2= 0

+ θ2= 0

d. Kích thước cần cho cảm biến

-Camera OV7670

+Khoảng cách quan sát: Tùy thuộc vào ống kính sử dụng, thường từ **10 cm** đến **vài mét**.

+Góc quét (Field of View - FOV): Thông thường khoảng **25° đến 60°** (tùy vào loại ống kính).

-Cảm biến BNO055 (IMU - đo góc quay)

+Góc quét (phạm vi đo góc quay): Yaw, Pitch, Roll: **±180°**

3. Động học

-Từ các tham số của cánh tay robot ta có:

Khớp xoay (Từ động cơ đến khớp xoay):

$$T_0^1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 16.75 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Khớp tịnh tiến (Từ khớp xoay đến khớp tịnh tiến):

$$T_1^2 = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 16.75 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

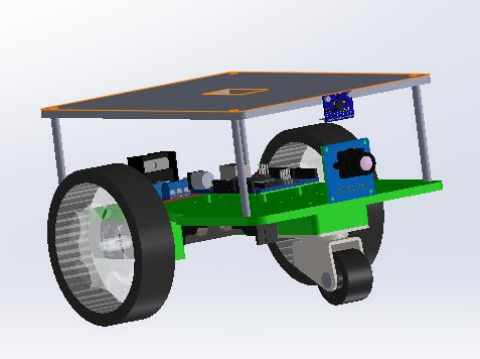
⇨ Ma trận tổng thể từ gốc đến cuối cánh tay:

$$T_0^2 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & a\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & a\sin\theta_1 \\ 0 & 0 & 1 & 33.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

II. Thiết kế solidworks, cách đặt hệ trục tọa độ

1. Thiết kế solidworks

-Phần xe

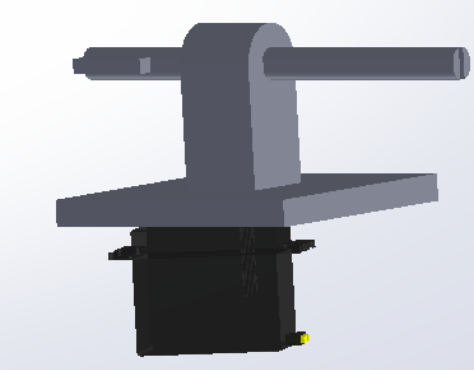


+Thiết kế khung xe

+Thiết kế bánh xe

+Lấy các linh kiện gồm: động cơ DC, arduino, L298, IMU, encoder, camera trên grabcad

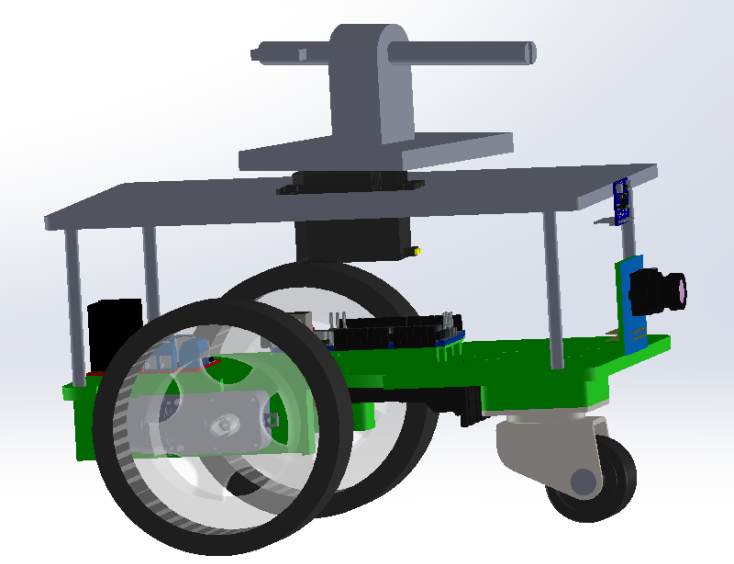
-Thiết kế tay máy:



+Khớp xoay

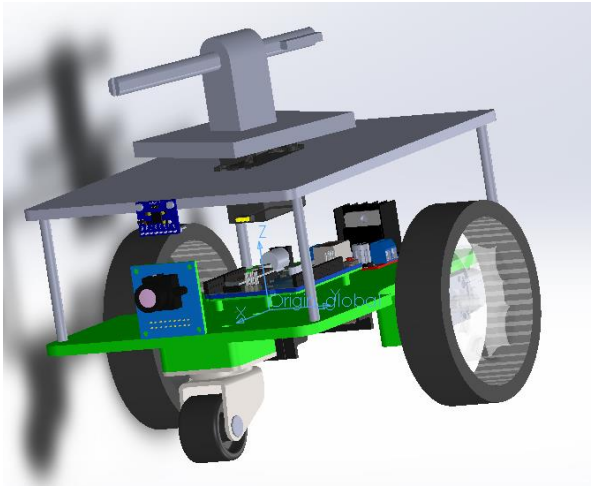
+Khớp tịnh tiến

-Ghép các bộ phận lại tạo nên robot hoàn chỉnh



2. Cách đặt hệ trục tọa độ

-Đặt trục với khung xe

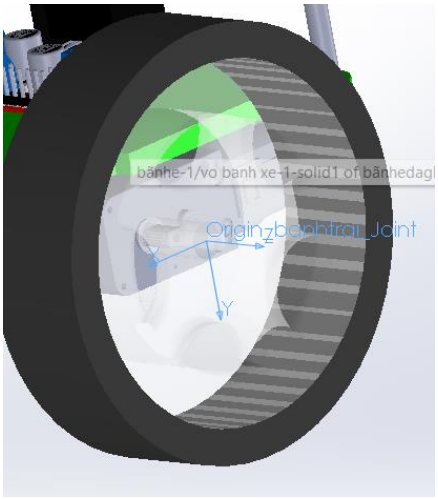


+Trục x trùng với phương tịnh tiến của xe

+Trục y trùng với chiều quay của xe

+Trục z tuân theo quy tắc tam diện thuận

-Đặt trục với bánh xe

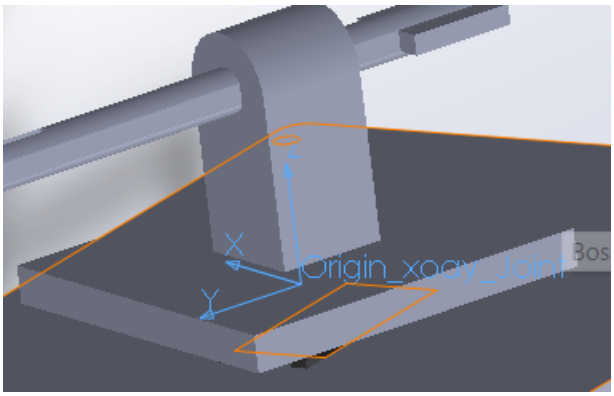


+Trục z đặt sao cho bánh quay quanh trục z

+Trục x đặt cùng phương tịnh tiến của xe

+Trục y tuân theo quy tắc tam diện thuận

-Đặt trục với khớp xoay

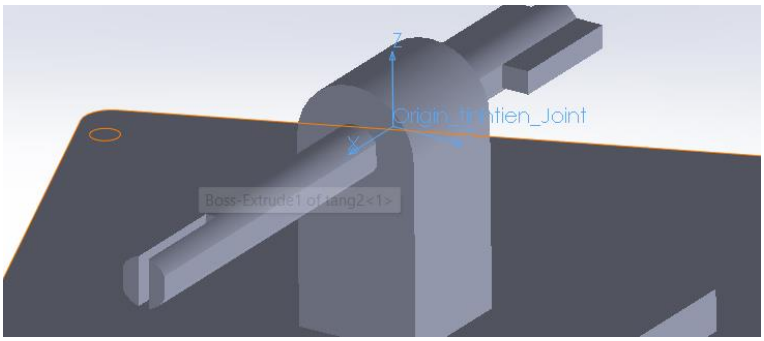


+Trục z đặt sao cho khớp xoay quanh trục z

+Trục x đặt cùng phương tịnh tiến của xe

+Trục y tuân theo quy tắc tam diện thuận

-Đặt trục với khớp tịnh tiến



+Trục x đặt cùng phương tịnh tiến của khớp

+Trục z đặt trùng với trục z của khung xe

+Trục y đặt theo quy tắc tam diện thuận

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

1. Mô tả file urdf

-Hệ thống gồm 3 phần chính:

+khung_link – khung chính, là bộ phận cố định.

+xoay_Link – liên kết có thể quay quanh trục **z** nhờ khớp xoay_Joint.

+tinhtien_Link – liên kết có thể trượt theo trục **x** nhờ khớp tinhtien_Joint.

-Các liên kết được mô tả với quán tính (khối lượng, mô men quán tính), hình học (mesh), màu sắc, và khả năng va chạm.

2. Liên kết của các link, các cảm biến

(1) *Khớp xoay (xoay_Joint)*

-Liên kết gốc (parent): khung_link (khung chính).

-Liên kết con (child): xoay_Link.

-Loại khớp: continuous (xoay tự do, không có giới hạn).

-Trục quay: (xyz=0, 0, 1), quay quanh trục z.

-Chức năng: Cho phép xoay_Link quay quanh trục z của khung_link, có thể dùng để thay đổi hướng của cánh tay robot hoặc cảm biến.

(2) *Khớp tịnh tiến (tinhtien_Joint)*

-Liên kết gốc (parent): xoay_Link.

-Liên kết con (child): tinhtien_Link.

-Loại khớp: prismatic (tịnh tiến).

-Trục di chuyển: (xyz=1, 0, 0), trượt theo trục x.

-Khoảng giới hạn: -0.05m đến 0.05m.

-Chức năng: Cho phép tinhtien_Link di chuyển tịnh tiến dọc theo trục x, giúp thay đổi

khoảng cách làm việc của cánh tay robot hoặc cảm biến gắn trên nó.

(3) *Cảm biến trong hệ thống*

1. *Cảm biến gắn trên tinhtien_Link (thường là cảm biến khoảng cách, camera, hoặc cảm biến tiếp xúc).*

-Chức năng: Theo dõi vị trí hoặc phát hiện vật cản khi hệ thống di chuyển.

-Mô tả:

+Đây là cảm biến khoảng cách sử dụng tia laser (ray sensor).

+Gắn trên tinhtien_Link, phát hiện vật cản phía trước với khoảng đo từ 1cm đến 2m.

+Quét theo phương ngang với góc ±0.5 radian.

2. *Cảm biến gắn trên xoay_Link (thường là IMU để đo góc quay hoặc cảm biến lực).*

-Chức năng: Theo dõi góc quay, đảm bảo cánh tay robot hoặc thiết bị không bị lệch ngoài phạm vi mong muốn.

-Mô tả:

+Cảm biến IMU (Inertial Measurement Unit), theo dõi góc quay và gia tốc.

+Giúp đo chính xác vị trí của xoay_Link khi quay.

3. Mô tả gazebo và rviz

1. Môi trường Gazebo

-Gazebo là một trình mô phỏng vật lý mạnh mẽ, thường được sử dụng trong hệ sinh thái Robot Operating System (ROS) để kiểm tra và phát triển robot trước khi triển khai vào thực tế.

-Cấu trúc và thành phần của mô phỏng Gazebo

- Mô hình robot:

+Robot có cấu trúc giống với một robot di động có bánh xe (có thể là một loại robot tự hành).

+Robot có thể được trang bị cảm biến như camera hoặc LiDAR để thu thập dữ liệu môi trường.

+Mô hình hiển thị trong Gazebo có đầy đủ các liên kết (links) và khớp động học (joints).

-Không gian mô phỏng:

+Robot được đặt trong một môi trường trống, không có chướng ngại vật hay vật thể xung quanh.

+Trục tọa độ (X - đỏ, Y - xanh lá, Z - xanh dương) giúp xác định vị trí và hướng của robot.

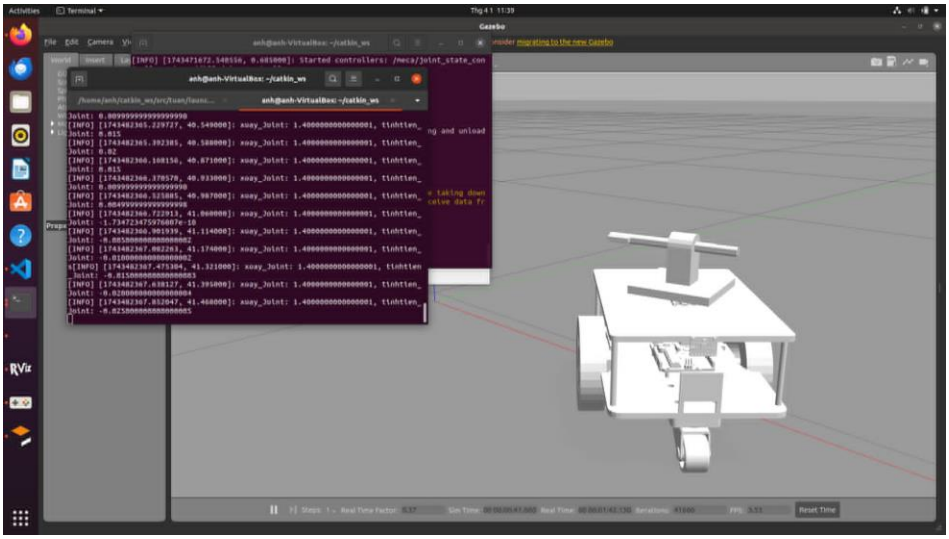
+Môi trường có lưới hiển thị, giúp quan sát chuyển động của robot dễ dàng hơn.

-Giao diện điều khiển *(rqt_robot_steering)*:

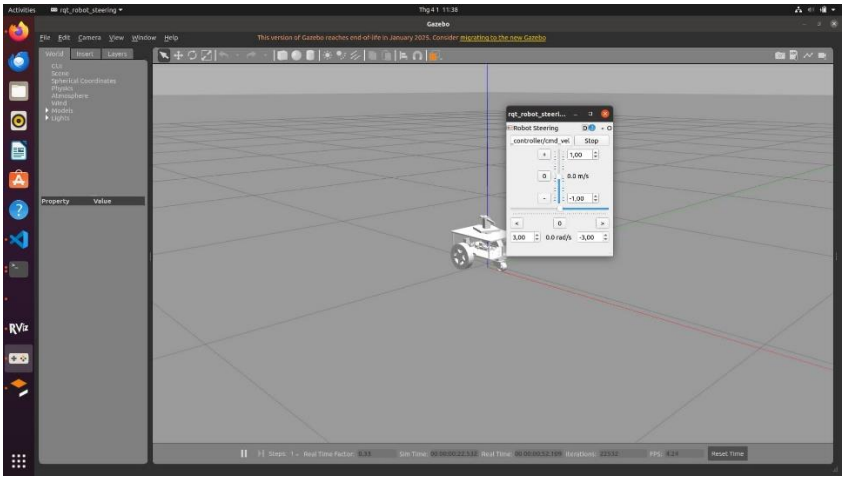
+Một cửa sổ nhỏ của *rqt_robot_steering* xuất hiện trong hình, cho phép điều khiển robot theo thời gian thực.

+Giao diện có các nút điều chỉnh vận tốc tuyến tính (m/s) và vận tốc góc (rad/s).

+Dải giá trị của tốc độ có thể từ -1.0 đến 1.0 m/s đối với tốc độ tuyến tính và từ -3.0 đến 3.0 rad/s đối với tốc độ góc.



(điều khiển tay máy)



(điều khiển vận tốc)

2. Môi trường Rviz: RViz là một công cụ hiển thị trực quan dữ liệu từ ROS, cho phép theo dõi trạng thái robot, cảm biến, và mô hình động học trong không gian 3D.

Các thành phần trong RViz

-Mô hình robot:

+Robot được hiển thị với các thành phần như các liên kết (links), khớp (joints) và cảm biến.

+Tên của các bộ phận của robot như "khung_link" (có thể là base_link hoặc một frame gốc khác).

+Cảm biến camera được gán trong robot để thu thập hình ảnh.

-Hệ thống hiển thị và trạng thái:

+Grid (Lưới): Lưới 3D giúp xác định mặt phẳng di chuyển của robot.

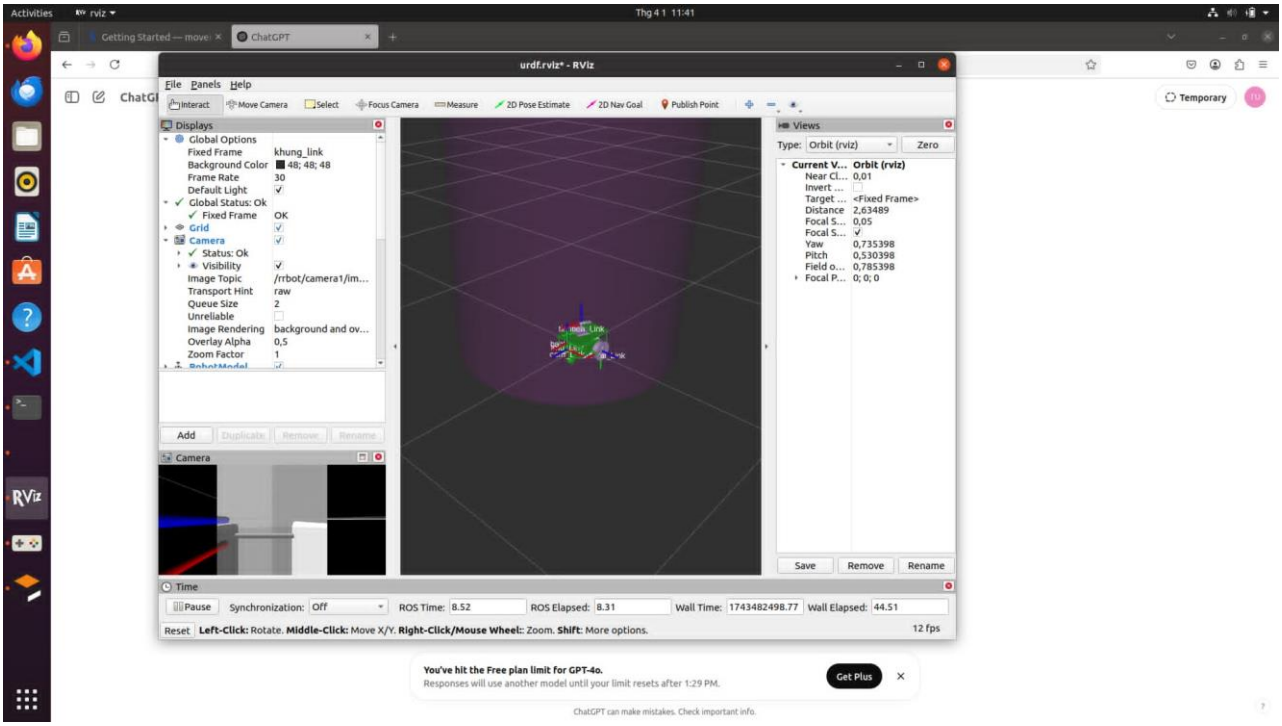
+Camera: Hiển thị dữ liệu từ camera của robot với hình ảnh nhận được.

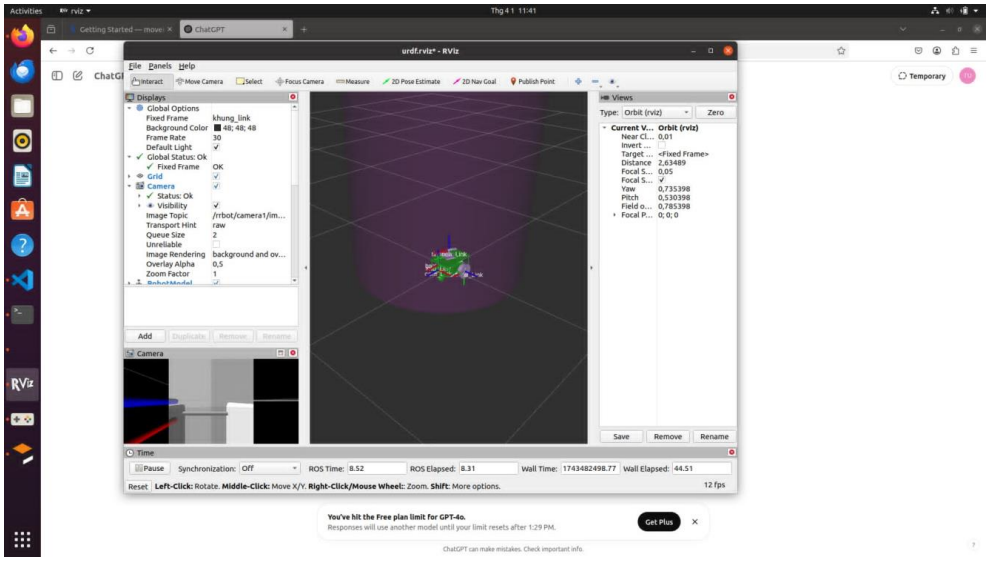
+Trạng thái cảm biến: Hiển thị trạng thái của camera (Status: OK), nghĩa là dữ liệu được nhận và xử lý đúng cách.

-Góc nhìn (View Controller):

+Chế độ xem được đặt thành Orbit, giúp xoay quanh robot để quan sát từ nhiều góc độ khác nhau.

+Yaw, Pitch, Field of View (FOV) được hiển thị để tùy chỉnh góc nhìn.





(cảm biến IMU và camera hoạt động bình thường)

IV. Mô tả cơ chế điều khiển trên gazebo

*Điều khiển 2 bánh

1. Phương pháp điều khiển:

-Xe có hai bánh, một bên trái và một bên phải, và mỗi bánh có một động cơ điều khiển riêng biệt.

-Các tín hiệu điều khiển mô tả tốc độ và hướng của từng bánh xe.

-Tốc độ của xe sẽ được xác định bởi sự kết hợp giữa tốc độ quay của hai bánh. Nếu hai bánh quay cùng tốc độ và cùng hướng, xe sẽ đi thẳng.

-Nếu tốc độ của hai bánh khác nhau, xe sẽ quay, với góc quay phụ thuộc vào sự chênh lệch tốc độ.

2. Các tham số điều khiển:

-Tốc độ tiến/tiến lùi (Vx): Điều khiển tốc độ di chuyển của xe theo trục x.

-Tốc độ quay (Vtheta): Điều khiển góc quay của xe, giúp xe quay tại chỗ.

3. Phương trình điều khiển:

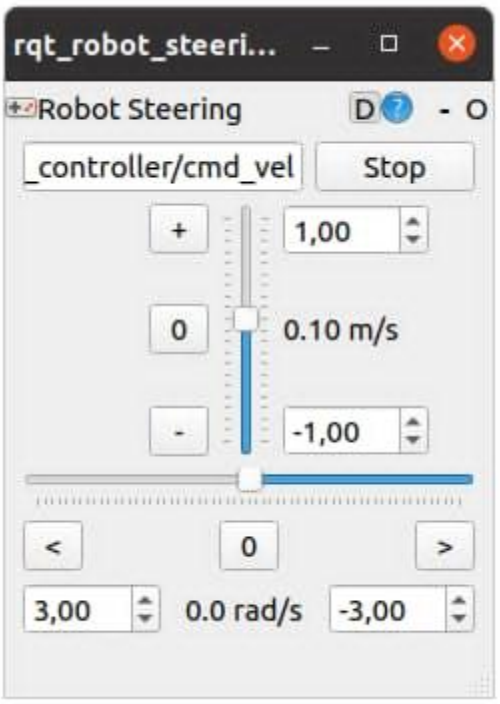
-Giả sử bạn có hai bánh xe với bán kính bánh xe là rrr và khoảng cách giữa hai bánh là LLL:

-Tốc độ của mỗi bánh được tính dựa trên các tham số sau:

$$v_{\text{left}} = \frac{Vx - \frac{L}{2}Vtheta}{r}$$

$$v_{\text{right}} = \frac{Vx + \frac{L}{2}Vtheta}{r}$$

-Tốc độ bánh trái và phải điều khiển việc di chuyển thẳng hoặc quay xe.



(giao diện điều khiển hướng, vận tốc)

***Cơ chế điều khiển tay máy (robot arm):**

1. **Khớp xoay và tịnh tiến:**

-Với tay máy, bạn có thể điều khiển hai khớp, một khớp tịnh tiến và một khớp xoay. Các khớp này được điều khiển qua các tín hiệu "position controller."

-Khớp xoay sẽ thực hiện chuyển động quay quanh một trục cố định.

-Khớp tịnh tiến sẽ thực hiện chuyển động di chuyển thẳng trên một trục.

2. **Mối quan hệ giữa khớp:**

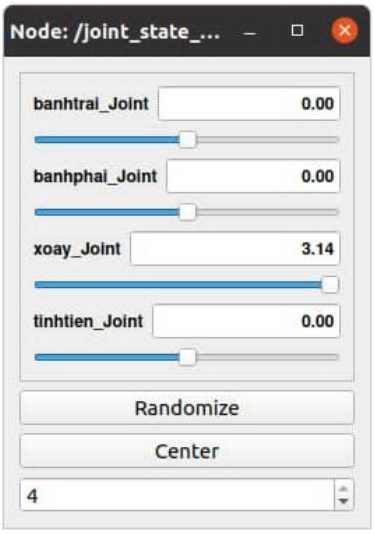
-Để điều khiển vị trí cuối cùng của tay máy, bạn cần phải sử dụng một mô hình động học ngược (inverse kinematics), cho phép tính toán các góc và khoảng cách của các khớp để đạt được vị trí mong muốn.

-Trong Gazebo:

+Gazebo sẽ mô phỏng chuyển động của cả xe và tay máy dựa trên các thông số đã định nghĩa trong URDF/XACRO.

+Các tín hiệu điều khiển từ ROS sẽ được gửi đến các mô-đun điều khiển của xe và tay máy, và chúng sẽ được mô phỏng trong Gazebo.

+Xe 2 bánh vi sai và tay máy sẽ có các bộ điều khiển riêng biệt nhưng đều có thể tương tác với nhau thông qua các tín hiệu từ ROS.



(giao diện điều khiển khớp)

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

1. Cấu trúc dự án

a. URDF (Unified Robot Description Format)

-**Mục đích**: Mô tả mô hình 3D của robot, bao gồm các bộ phận cơ khí, khớp (joints), cảm biến, động cơ, và kết nối giữa các phần tử.

-**Thành phần**:

+Links: Các phần tử cơ khí của robot, như bánh xe, thân, khớp.

+Joints: Các khớp kết nối các phần tử với nhau, có thể là khớp quay hoặc khớp tịnh tiến.

+Sensors: Các cảm biến như IMU, camera, lidar, v.v.

+Plugins: Cấu hình các plugin cho Gazebo để mô phỏng robot trong môi trường 3D.

b. Controllers (Bộ điều khiển)

-**Mục đích**: Điều khiển các khớp của robot, động cơ, hoặc các cảm biến.

-**Thành phần**:

+JointPositionController: Bộ điều khiển vị trí cho các khớp robot (như xoay_Joint và tinhtien_Joint).

+Plugin Gazebo: Được sử dụng để điều khiển các tác vụ mô phỏng trong Gazebo.

c. ROS Node

-**Mục đích**: Các nút (node) trong ROS là các đơn vị xử lý thông tin, thực hiện các tác vụ như điều khiển robot, nhận tín hiệu từ cảm biến, v.v.

-**Thành phần**:

+Publisher: Gửi thông tin về trạng thái robot, vị trí các khớp, hoặc cảm biến.

+Subscriber: Nhận tín hiệu từ các nguồn khác (ví dụ: tín hiệu từ các cảm biến).

+Services & Actions: Xử lý các yêu cầu và hành động dài hạn từ người dùng hoặc hệ thống.

d. Gazebo

-**Mục đích**: Mô phỏng môi trường vật lý và các tương tác của robot.

-**Thành phần**:

+Gazebo Plugins: Các plugin được cấu hình để mô phỏng các yếu tố vật lý như lực mô men, trọng lực, và các tương tác giữa các bộ phận của robot.

2. Thành phần chính

-Khớp (Joints)

+*xoay_Joint*: Khớp quay, có thể điều khiển góc quay của các bộ phận robot (ví dụ: bánh xe, cánh tay).

+*tinhtien_Joint*: Khớp tịnh tiến, có thể điều khiển vị trí tuyến tính của các bộ phận robot (ví dụ: phần thân robot dịch chuyển lên xuống).

-Điều khiển Khớp

+*JointPositionController*: Đây là bộ điều khiển giúp robot di chuyển các khớp đến vị trí mục tiêu. Đoạn mã của bạn cho thấy rằng hai khớp *xoay_Joint* và

tinhtien_Joint sẽ được điều khiển bởi loại bộ điều khiển này, giúp di chuyển các phần tử của robot một cách chính xác.

-Cảm biến và Kết nối

+Các cảm biến (ví dụ: IMU, Camera) có thể được định nghĩa trong URDF để cung cấp thông tin về trạng thái của robot.

+Các plugin của Gazebo sẽ giúp mô phỏng và tạo ra các phản hồi từ cảm biến trong môi trường 3D.

-Plugin Gazebo

+Các plugin như *gazebo_ros_control* sẽ giúp kết nối các bộ điều khiển ROS với mô phỏng trong Gazebo, đảm bảo sự đồng bộ giữa điều khiển robot và mô phỏng vật lý.

-ROS Controllers: Cấu hình các bộ điều khiển ROS cho các khớp robot sẽ quyết định cách robot thực hiện các thao tác điều khiển. Trong trường hợp này, JointPositionController được sử dụng để điều khiển các khớp với thông số là *xoay_Joint* và *tinhtien_Joint.*

3. Phân tích code

3.1. Điều khiển 2 bánh vi sai

1. ***Nhận lệnh điều khiển từ ROS***

-Chủ đề nhận lệnh: /cmd_vel

-Thông số chính:

+linear_velocity: Tốc độ di chuyển theo trục X

+angular_velocity: Tốc độ quay quanh trục Z

2. ***Tính toán tốc độ từng bánh xe***

Công thức tính:

*left_wheel_speed = (linear_velocity - angular_velocity * wheel_base / 2)*

*right_wheel_speed = (linear_velocity + angular_velocity * wheel_base / 2)*

-wheel_base: Khoảng cách giữa hai bánh xe

-Điều chỉnh tốc độ từng bánh dựa trên hướng di chuyển và quay

3. ***Gửi tín hiệu điều khiển động cơ***

-Tốc độ tính toán được gửi đến động cơ bánh trái và bánh phải

-Giao tiếp với Gazebo qua ros_control và gazebo_ros_control

4. *Mô phỏng và kiểm tra*

-Các plugin của Gazebo đảm bảo mô phỏng chính xác

-Cảm biến phản hồi lại trạng thái di chuyển của robot

3.2. Mô phỏng và giao diện điều khiển

1. ***Cấu trúc Mô Phỏng trong Gazebo***

-Mô hình robot: Mô phỏng xe 2 bánh vi sai trong Gazebo.

-Plugin Gazebo: gazebo_ros_control kết nối ROS & Gazebo.

-Cảm biến: Tích hợp IMU, camera, LiDAR để nhận diện môi trường.

2. ***Cơ chế Điều Khiển (ROS)***

-Nhận lệnh điều khiển từ ROS (/cmd_vel)

```
def cmd_vel_callback(msg):
    linear_velocity = msg.linear.x
    angular_velocity = msg.angular.z
```

-Tính tốc độ từng bánh xe

```
left_wheel_speed = linear_velocity - angular_velocity * wheel_base / 2
right_wheel_speed = linear_velocity + angular_velocity * wheel_base / 2
```

-Gửi tốc độ bánh xe đến Gazebo qua ros_control

publish_wheel_speed(left_wheel_speed, right_wheel_speed)

3. ***Giao Diện Điều Khiển (GUI)***

-Nhận lệnh từ bàn phím (teleop_twist_keyboard)

-RViz & GUI hỗ trợ điều khiển và giám sát

4. ***Thành Phần Mã Quan Trọng***

-ROS Node: Nhận lệnh, tính toán tốc độ bánh xe.

-Plugin Gazebo: Nhận lệnh, mô phỏng động cơ và phản hồi trạng thái.

5. ***Mô phỏng & Điều Khiển***

-Mô phỏng: Robot di chuyển trong Gazebo.

-Giao diện: Quan sát & điều khiển xe qua GUI hoặc terminal.

3.3. Cấu trúc của robot

1. *Khung và Cấu trúc Cơ bản*

-Chassis (Khung xe): Gắn bánh xe, động cơ, cảm biến.

-Bánh xe (Wheels): 2 bánh vi sai điều khiển bằng tốc độ quay.

2. *Hệ thống truyền động*

-Động cơ DC/động cơ bước: Điều khiển tốc độ và hướng quay của bánh xe.

-Bộ điều khiển tốc độ: Nhận tín hiệu từ ROS để kiểm soát động cơ.

3. *Cảm biến & Nhận dạng*

-IMU: Đo gia tốc, tốc độ góc để theo dõi trạng thái.

-Camera/LiDAR: Thu thập dữ liệu môi trường xung quanh.

4. *Khớp và Trục (Joints & Axles)*

-Khớp giữa bánh xe và trục: Giúp bánh xe quay tự do.

-Yaw Joint: Điều chỉnh góc quay của robot.

5. *Hệ thống điều khiển (ROS & Gazebo)*

-ROS Nodes: Xử lý lệnh /cmd_vel để điều khiển robot.

-Plugin ros_control: Điều khiển động cơ và mô phỏng trong Gazebo.

-Tính toán tốc độ bánh xe:

*left_wheel_speed = linear_velocity - angular_velocity * wheel_base / 2*

*right_wheel_speed = linear_velocity + angular_velocity * wheel_base / 2*

-Gửi tín hiệu đến động cơ:

publish_wheel_speed(left_wheel_speed, right_wheel_speed)

6. *Cấu trúc phần mềm*

-Gazebo Plugins: Mô phỏng chuyển động và môi trường robot.

-GUI & Teleoperation: Điều khiển robot bằng bàn phím hoặc giao diện đồ họa (RViz).

7. *Hệ thống truyền động (Transmission)*

-Truyền động bánh xe: Điều chỉnh tốc độ & hướng quay của từng bánh xe.