BÁO CÁO ROS GIỮA KÌ - Xây dựng, mô phỏng và điều khiển robot

Họ và tên : Mai Quốc Hiếu - 22027521

- Dạng robot, động học, kích thước
- Thiết kế solidworks, cách đặt hệ trục tọa độ
- Mô tả file urdf, liên kết của các link, các cảm biến, mô tả gazebo
- Mô tả cơ chế điều khiển trên gazebo . ..
- Các thành phần chính của code, structure folder dự án

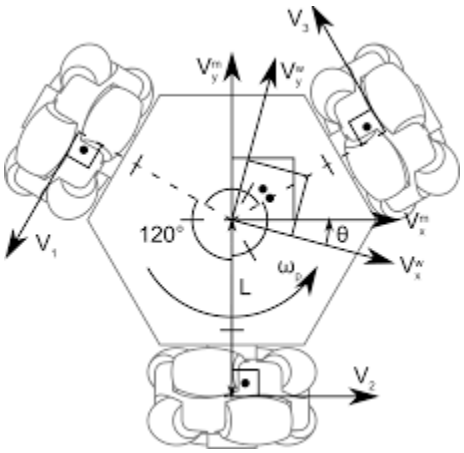
1. Dạng robot , động học , kích thước

1.1 Dạng robot :

- Dạng robot 3 bánh omni:

+Sử dụng 3 bánh omni độc lập được điều khiển riêng đặt cách đều nhau, góc giữa các bánh là 120◦

+ Ở trên cùng robot có 1 tay máy với 2 khớp : khớp 1 laf khớp tịnh tiến , khớp 2 là khớp xoay



- Động học:

+ Gốc tọa độ tại tâm robot

+ Mỗi bánh cách nhau 1 góc 120

+ Các bánh được ký hiệu lần lượt là: 1,2,3

+ Khoảng cách tâm đến trục quay của bánh: L

+ Vận tốc góc của bánh thứ i :ωi

2. Phương trình động học thuận

Dựa trên nguyên lý vận tốc, ta có ma trận động học thuận từ tốc độ góc của các bánh ($\omega_1, \omega_2, \omega_3$) sang vận tốc của xe ($V_x, V_y, \omega$):

$$\begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} = \frac{r}{3} \begin{bmatrix} -\sin(\theta_1) & -\sin(\theta_2) & -\sin(\theta_3) \\ \cos(\theta_1) & \cos(\theta_2) & \cos(\theta_3) \\ 1/L & 1/L & 1/L \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

Với:

$$\theta_1 = 0^\circ, \quad \theta_2 = 120^\circ, \quad \theta_3 = 240^\circ$$

Thay giá trị:

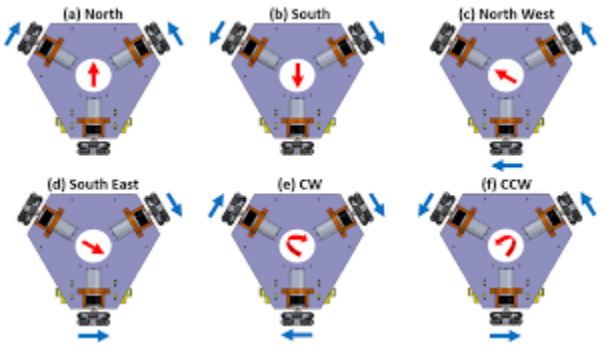
$$\begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} = \frac{r}{3} \begin{bmatrix} 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \\ 1/L & 1/L & 1/L \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

3. Phương trình động học nghịch

Ngược lại, nếu biết V_x, V_y, ω, ta có thể tính tốc độ góc của từng bánh:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 0 & 1 & L \\ -\frac{\sqrt{3}}{2} & -\frac{1}{2} & L \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} & L \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix}$$

Hình ảnh điều khiển các bánh để điều hướng cho robot



- Kích thước :

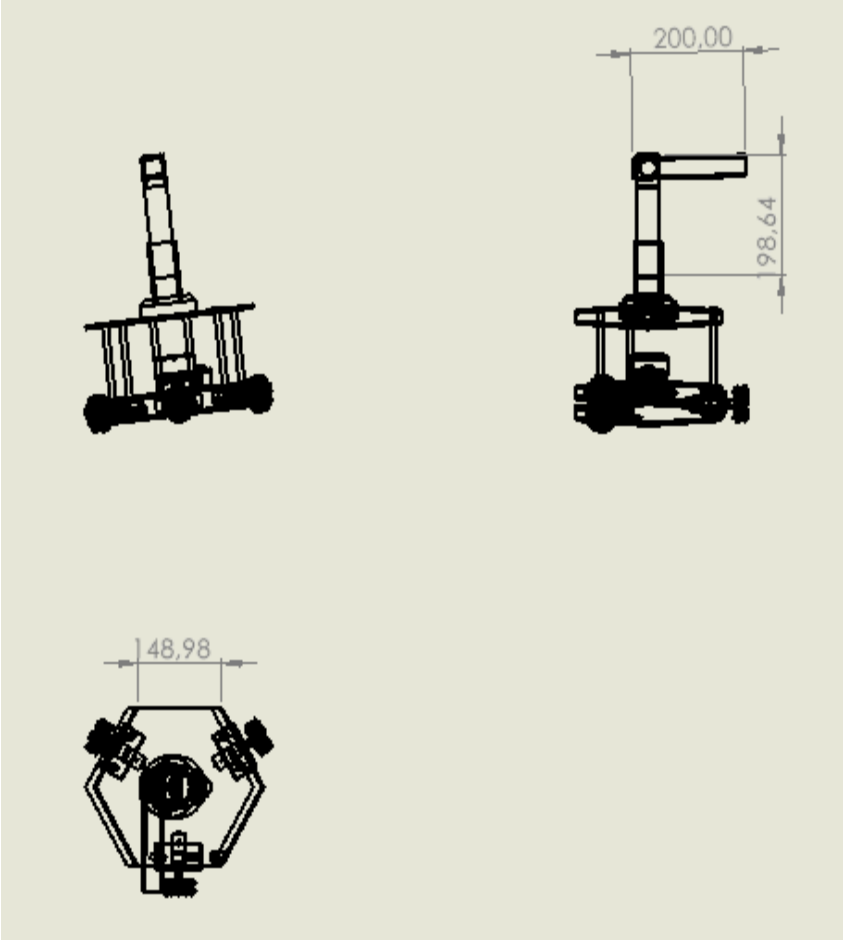
Chiều dài 2 link của tay là 20 cm

Khoảng cách từ thân đến bánh xe là 130 ccm

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

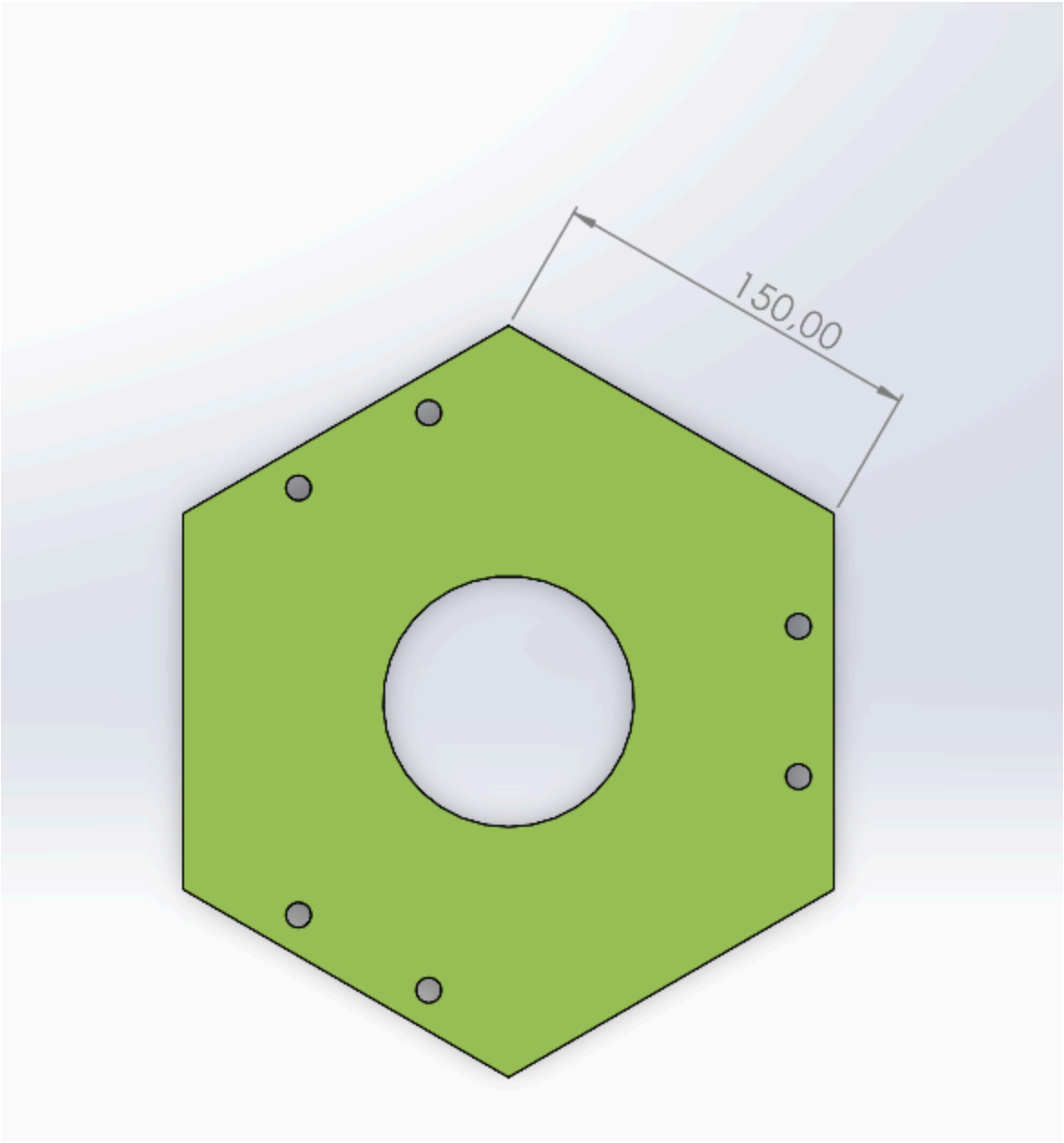
Thiết kế trên solidworks

Bản vẽ 2 d:

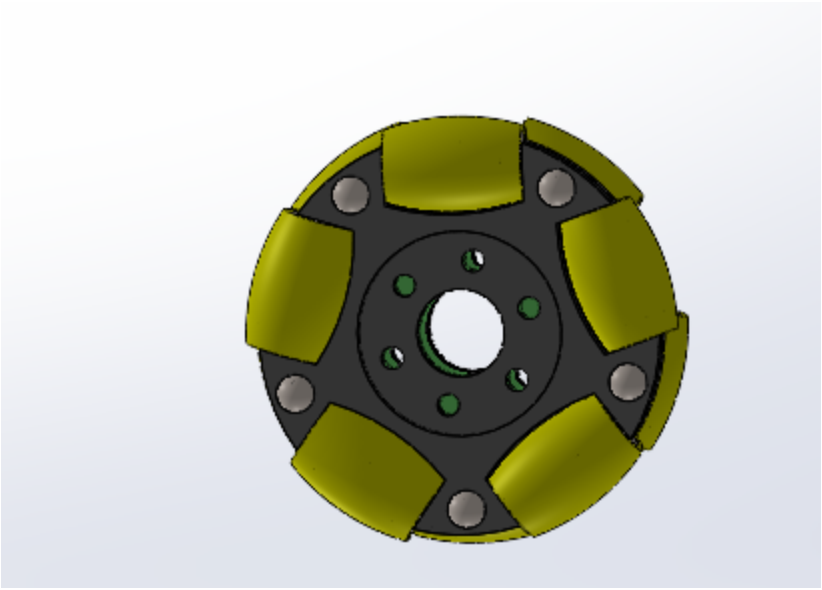


Robot gồm 3 phần chính

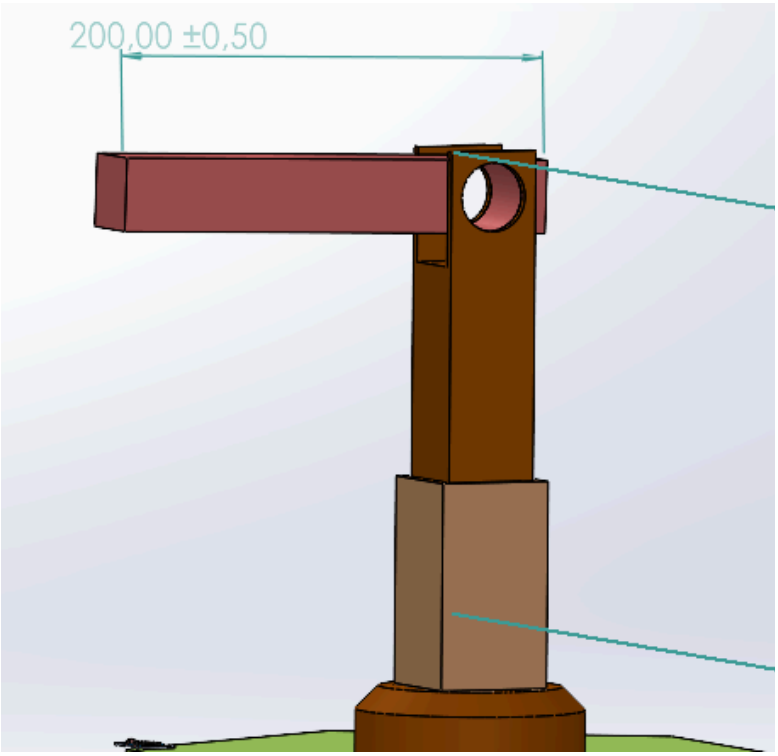
Khung xe

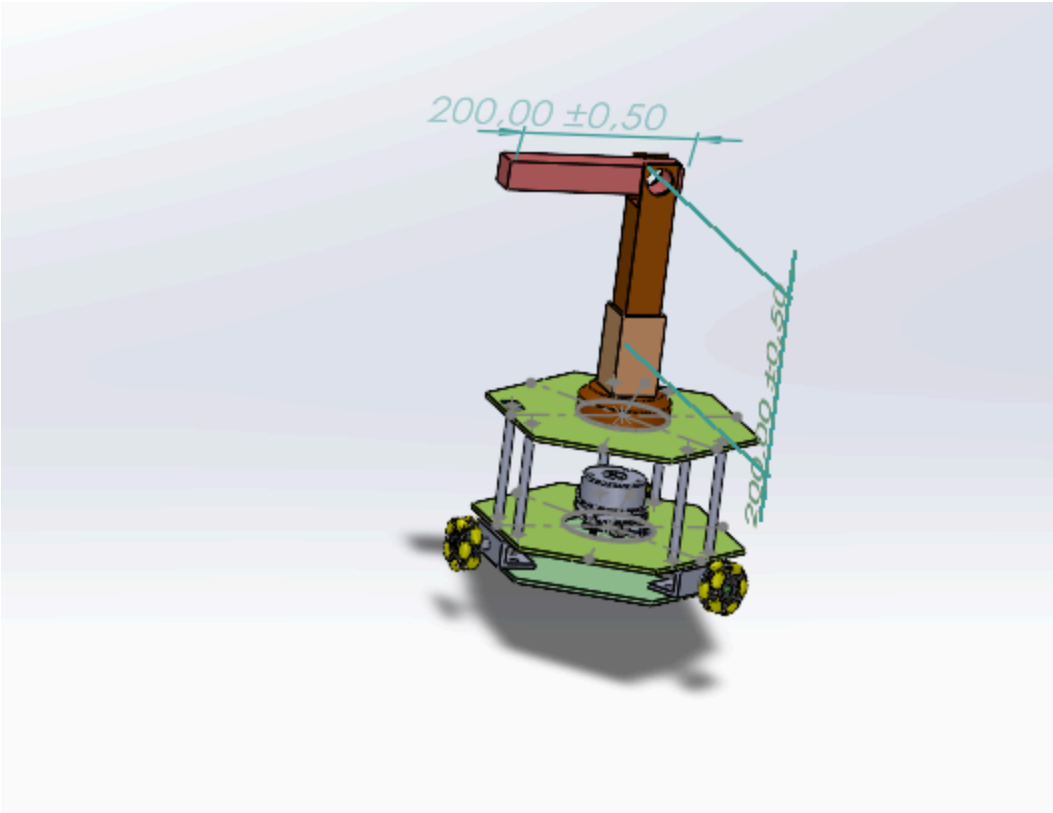


Bánh omni



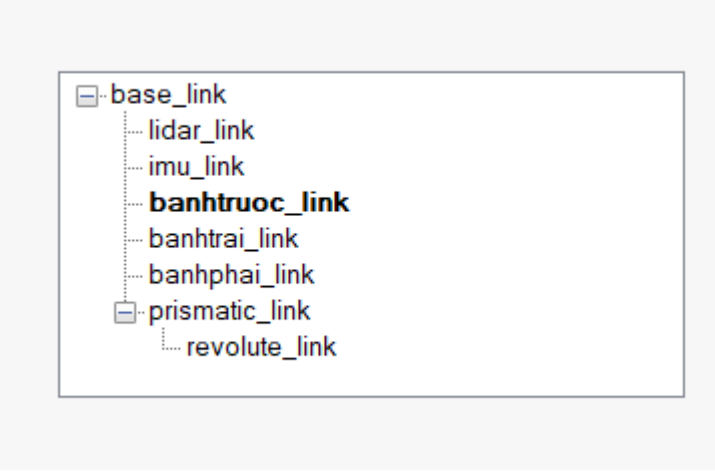
Cánh tay





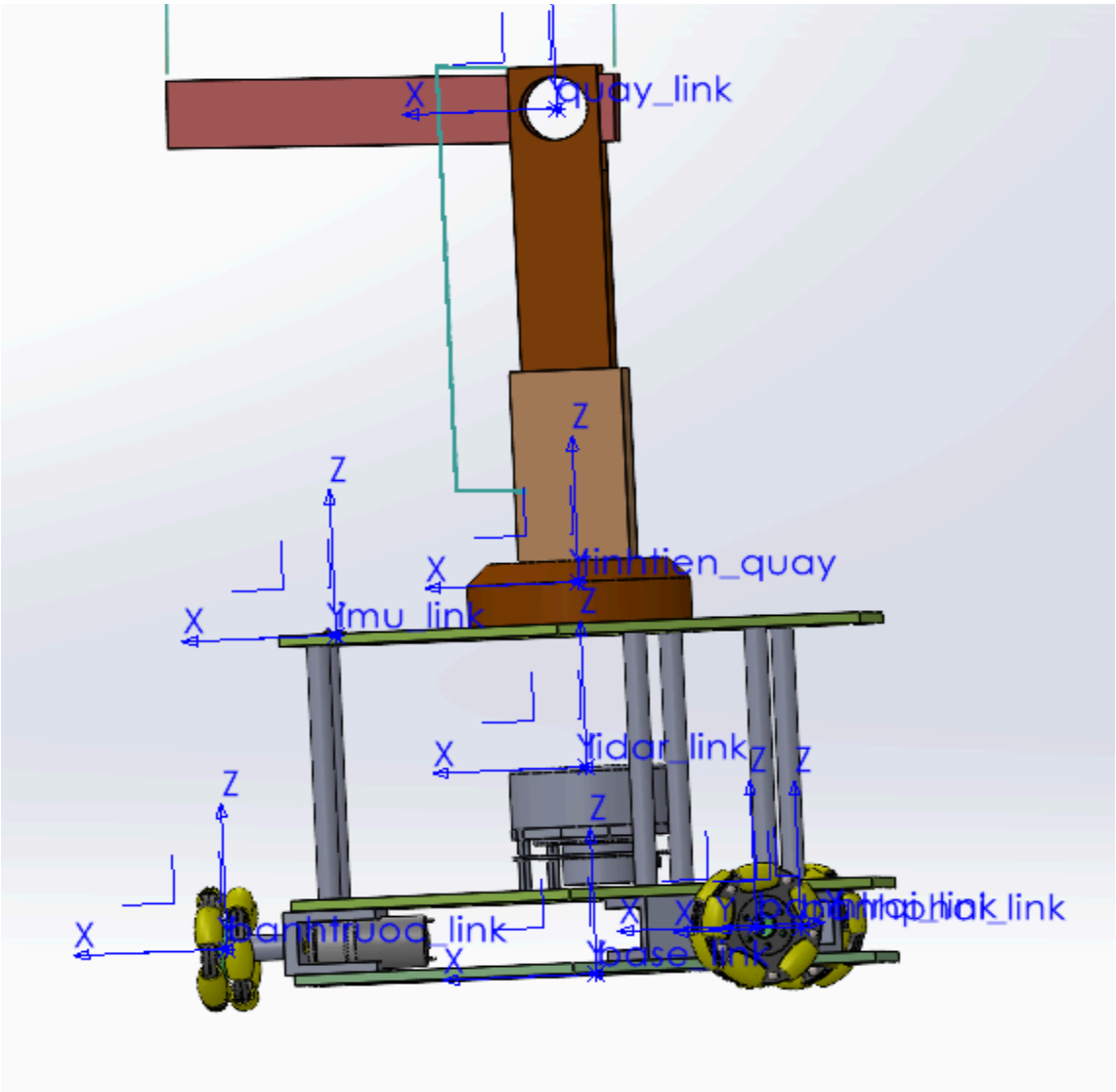
Mô hình hiển thị trên SolidWorks

- Cách đặt trục trên hệ tọa độ :



Cấu trúc thể hiện cha con của các link trong model robot

Ta đặt trục cho tất cả các link trên



Mô hình sau khi hoàn thánh đặt trục

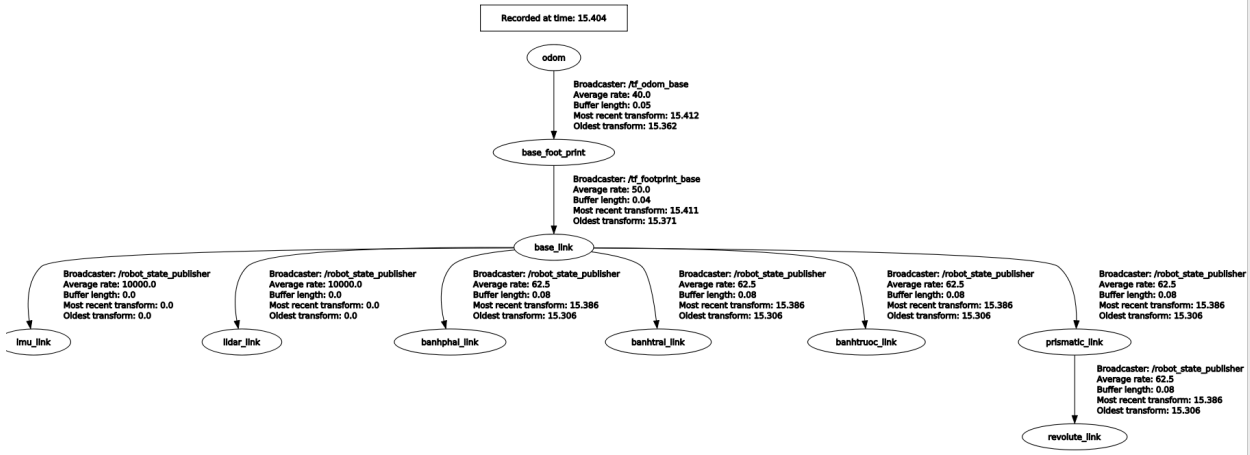
=> Từ đấy ta có thể sử dụng tool có sẵn của solidworks để xuất ra file urdf

3. Mô tả file URDF , các cảm biến ,mô tả gazebo

File URDF gồm các phần :

+ LInk: Đại diện thành phần của Robot
+ Joint: Phần giúp kết nối các link và cách chuyển động giữa các link
+ PLugin : mô-đun phần mềm được nhúng vào mô hình robot để mở rộng và điều khiển các tính năng của robot trong môi trường mô phỏng

+ Transmission:Giúp Định nghĩa cách thức truyền tín hiệu điều khiển từ controller tới actuator (động cơ hoặc thiết bị truyền động).



Tf của các farme cho model

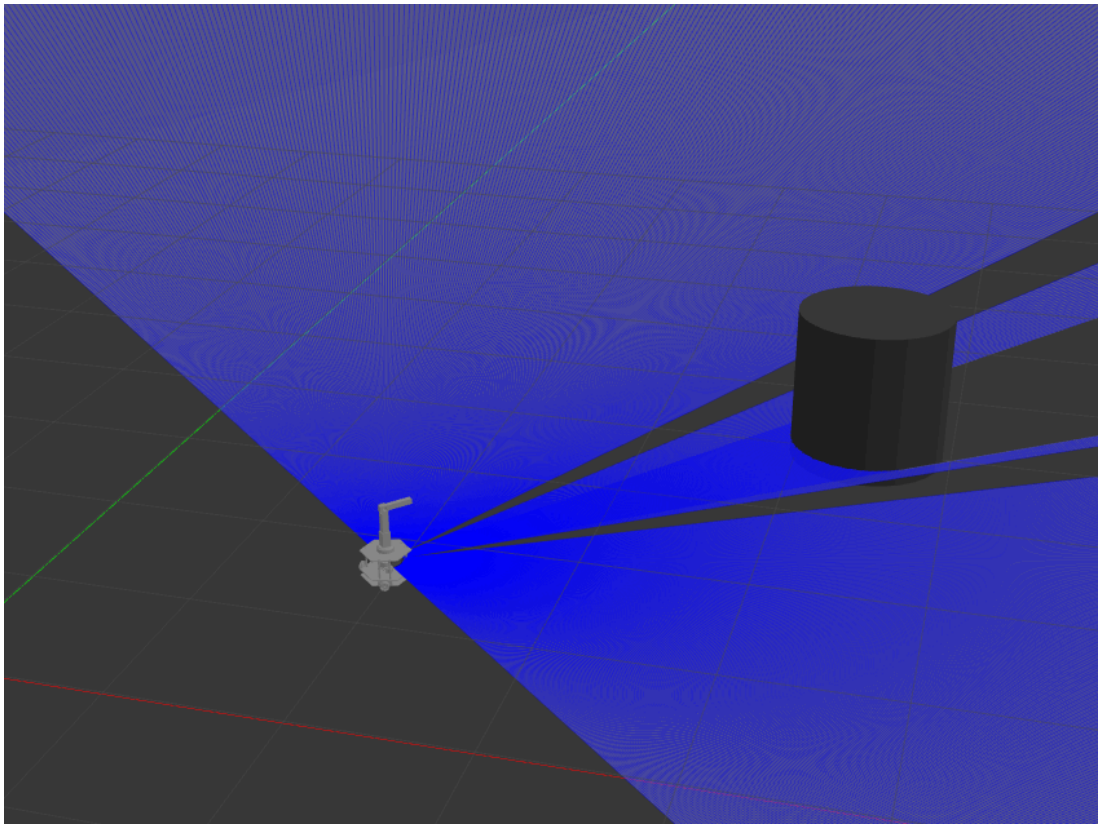
Các cảm biến : lidar , imu , encoder

+ Lidar: quét Laser để tạo bản đồ 2D,3D trong các môi trường
Sử dụng plugin gazebo_ros_laser :

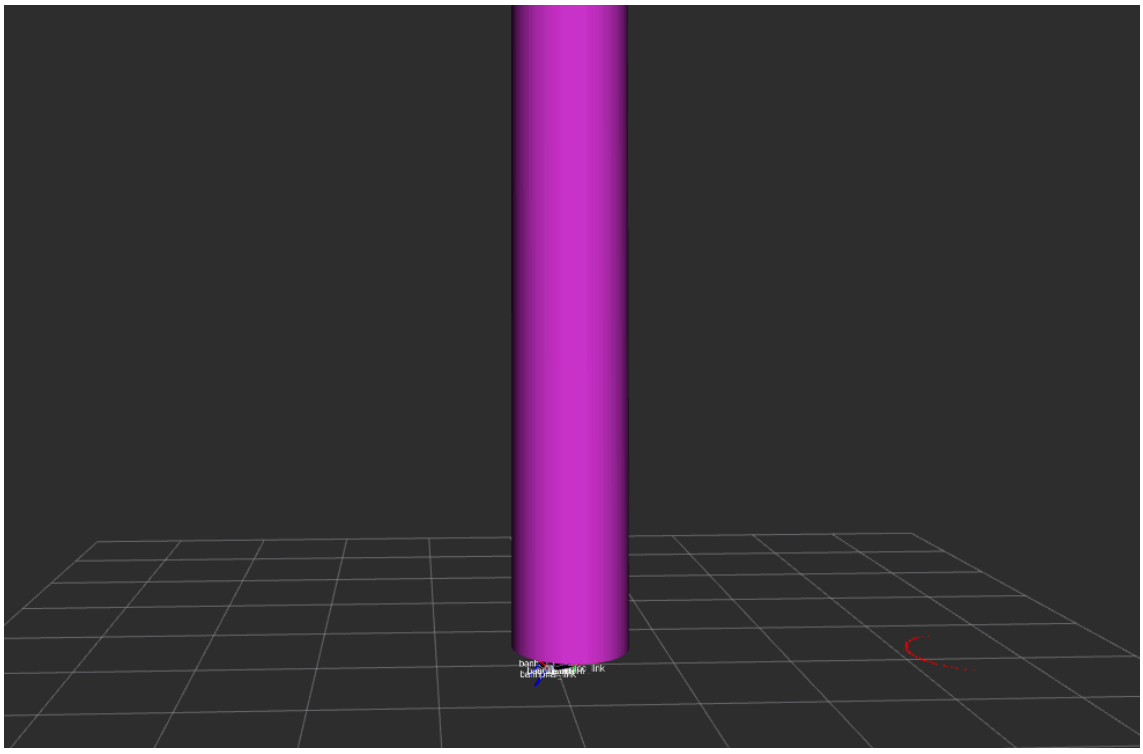
```
    <gazebo reference="lidar_link">
<sensor type="ray" name="lidar_sensor">
  <pose>0 0 0 0 0 0</pose>
  <visualize>true</visualize>
  <update_rate>30</update_rate>
  <ray>
 <scan>
   <horizontal>
     <samples>720</samples>
     <resolution>1</resolution>
     <min_angle>-1.5708</min_angle>
     <max_angle>1.5708</max_angle>
   </horizontal>
 </scan>
 <range>
   <min>0.1</min>
   <max>30.0</max>
   <resolution>0.01</resolution>
 </range>
 <noise>
   <type>gaussian</type>
   <mean>0.0</mean>
   <stddev>0.01</stddev>
 </noise>
  </ray>
  <plugin name="gazebo_ros_laser" filename="libgazebo_ros_laser.so">
 <topicName>/scan</topicName>
 <frameName>lidar_link</frameName>
  </plugin>
</sensor>
 </gazebo>
```

Kết quả:

Hiển thị trong gazebo :



Kết quả lidar trong rviz:



Phần đỏ cho thấy lidar phát hiện được vật cản:

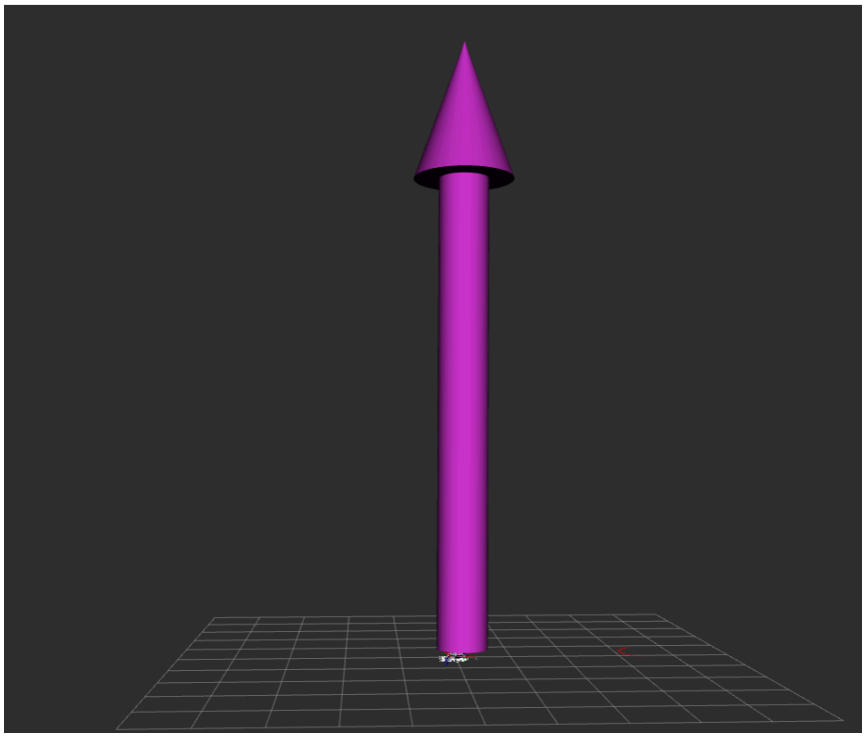
+ IMU : giúp xác định vận tốc gia tốc và vị trí của robot trong môi trường

Sử dụng plugin libgazebo_ros_imu_sensor.so

```
97
98      <gazebo reference="imu_link">
99       <gravity>true</gravity>
00       <sensor name="imu_sensor" type="imu">
01         <always_on>true</always_on>
02         <update_rate>100</update_rate>
03         <visualize>true</visualize>
04         <topic>__default_topic__</topic>
05         <plugin filename="libgazebo_ros_imu_sensor.so" name="imu_plugin">
06           <topicName>imu</topicName>
07           <bodyName>imu_link</bodyName>
08           <updateRateHZ>10.0</updateRateHZ>
09           <gaussianNoise>0.0</gaussianNoise>
10           <xyzOffset>0 0 0</xyzOffset>
11           <rpyOffset>0 0 0</rpyOffset>
12           <frameName>imu_link</frameName>
13           <initialOrientationAsReference>false</initialOrientationAsReference>
14         </plugin>
15         <pose>0 0 0 0 0 0</pose>
16       </sensor>
17     </gazebo>
18
19   <transmission name="left_wheel_trans">
```

Kết quả :

Hiển thị trong rvix:



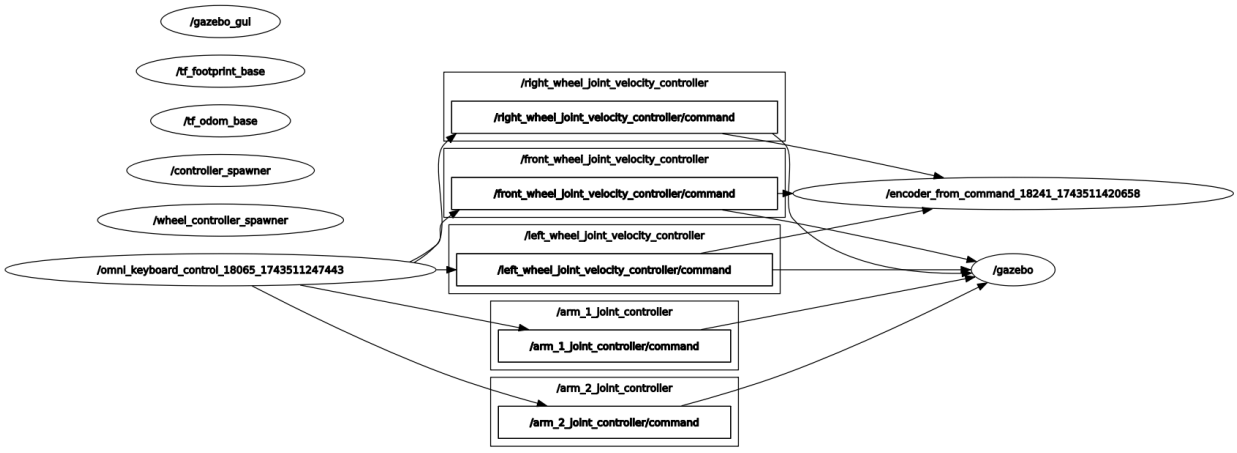
Dữ liệu được publish lên topic imu

```
---
header:
  seq: 400
  stamp:
    secs: 118
    nsecs: 286000000
  frame_id: "imu_link"
orientation:
  x: -0.0001412990682586537
  y: -3.835509547039943e-05
  z: 0.39334273606557396
  w: 0.9193919025899175
orientation_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
angular_velocity:
  x: 0.04209197716071851
  y: 0.020724971470798364
  z: 0.00856130048877917
angular_velocity_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
linear_acceleration:
  x: -0.05964784730803705
  y: -0.08234998834332531
  z: 9.73927833029276
linear_acceleration_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

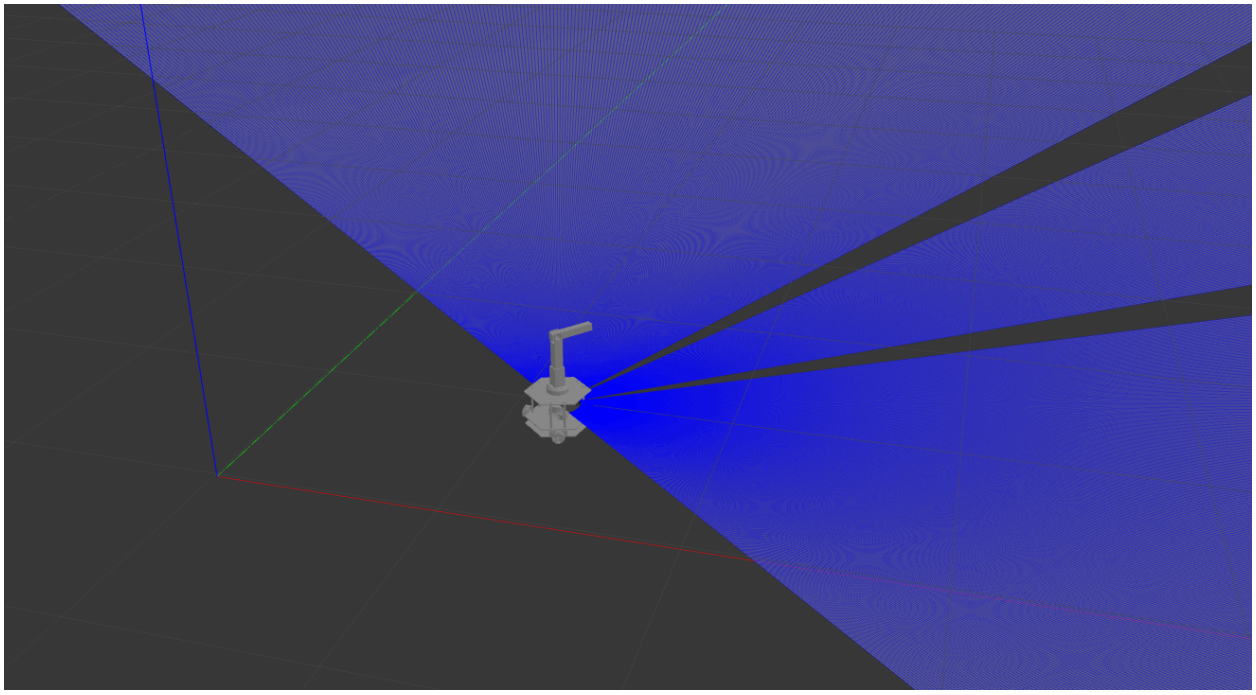
Imu sẽ cho biết hướng ,gia tóc ,vận tốc của robot

+ Encoder : giúp xác định và điều khiển tốc độ quay động cơ

viết node subcriber lấy thông tin vận tốc góc của các động cơ từ topic /right_wheel_joint_velocity_controller



Mô hình xuất hiện trong gazebo :



Hiện thị vận tốc của các động cơ

```
hieu@hieu-Inspiron-15-3511:~$ rosrun gk_ros encoder.py
[INFO] [1743511420.832554, 46.906000]: Node encoder_from_command đang chạy...
[INFO] [1743511768.337492, 79.587000]: ENCODER | Left Wheel Speed: 0.000 rad/s
[INFO] [1743511768.338492, 79.588000]: ENCODER | Right Wheel Speed: 0.000 rad/s
[INFO] [1743511768.338974, 79.588000]: ENCODER | Left Wheel Speed: 0.000 rad/s
```

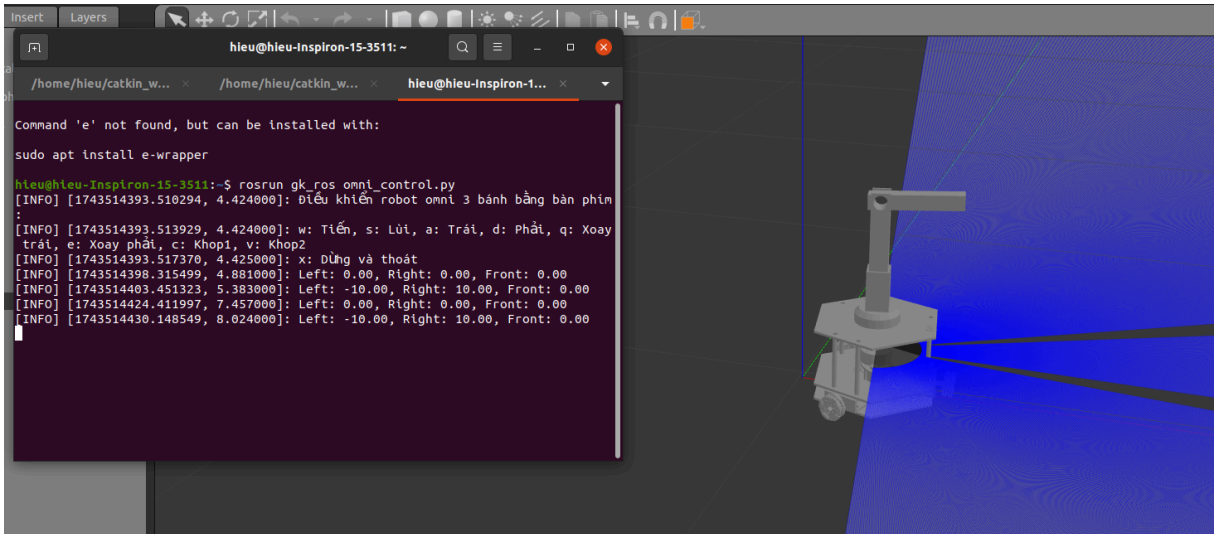
4. Mô tả cơ chế điều khiển trên gazebo

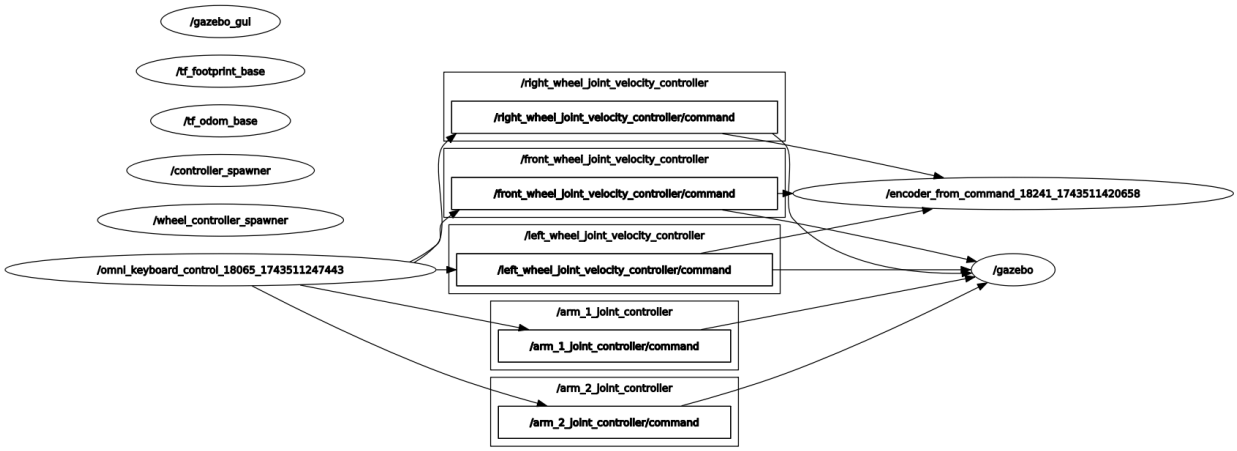
Cơ chế điều khiển trên gazebo:

- Điều khiển qua bàn phím :

node sẽ publish lên topic để điều khiển từng động cơ và cả cánh tay

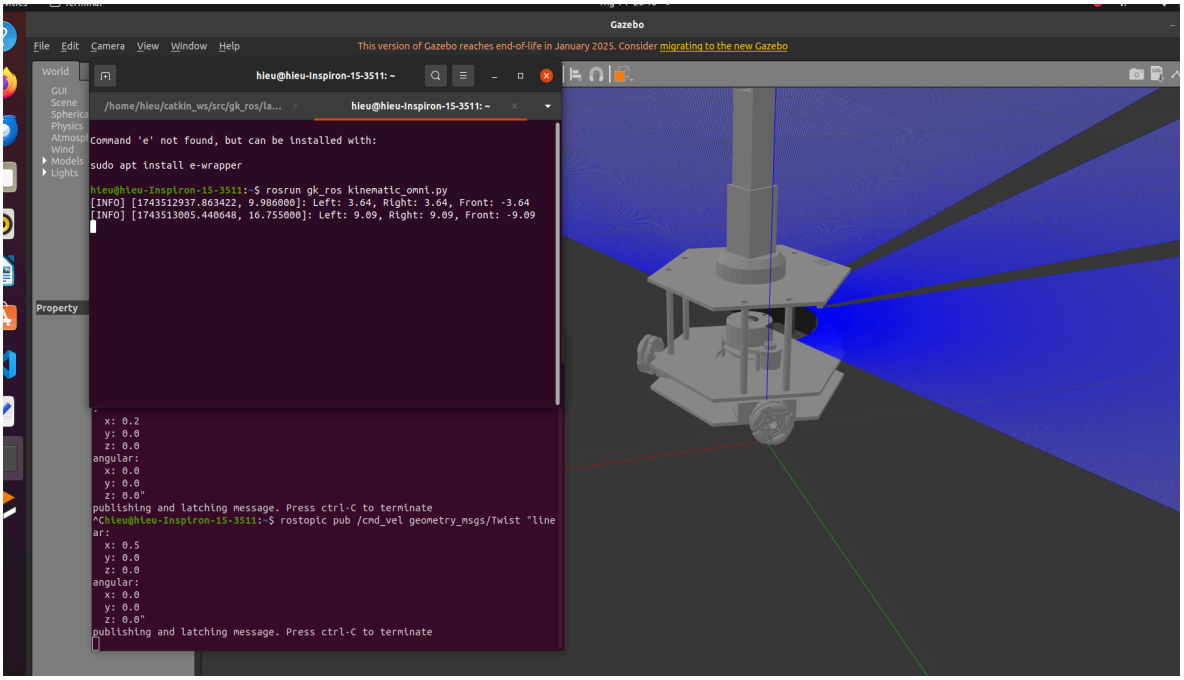
Cánh tay sẽ được điều khiển thông qua bàn phím

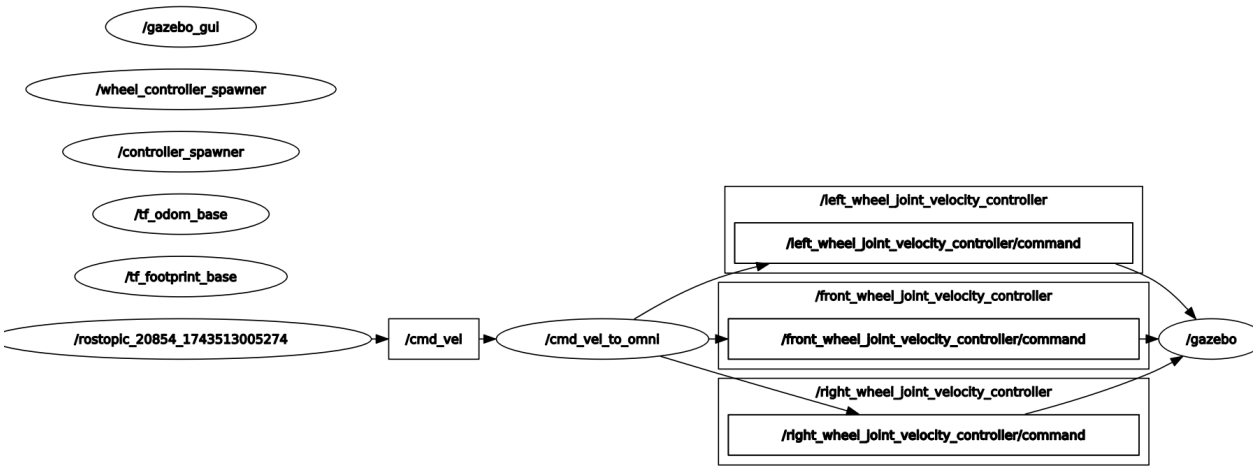




- Điều khiển vận tốc của xe thông qua tính toán động học ngược cho từng bánh

Node sẽ subcriber vận tốc mong muốn từ /cmd_vel và publish lên tốc độ cho từng động cơ





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

- Các thành phần chính của code :

Code điều khiển bằng bàn phím:

publish lên topic

```python
class OmniKeyboardControl:
    def __init__(self):
        # Khởi tạo node ROS
        rospy.init_node('omni_keyboard_control', anonymous=True)


        # Publisher cho các lệnh vận tốc bánh xe
        self.pub_left = rospy.Publisher('/left_wheel_joint_velocity_controller/command', Float64, queue_size=10)
        self.pub_right = rospy.Publisher('/right_wheel_joint_velocity_controller/command', Float64, queue_size=10)
        self.pub_front = rospy.Publisher('/front_wheel_joint_velocity_controller/command', Float64, queue_size=10)

        # Publisher cho cac lenh dieu khien khop tay
        self.pub_prismatic = rospy.Publisher('/arm_1_joint_controller/command', Float64, queue_size=10)
        self.pub_revolute = rospy.Publisher('/arm_2_joint_controller/command', Float64, queue_size=10)
```

```python
    def run(self):
        rospy.loginfo("Điều khiển robot omni 3 bánh bằng bàn phím:")
        rospy.loginfo("w: Tiến, s: Lùi, a: Trái, d: Phải, q: Xoay trái, e: Xoay phải, c: Khop1, v: Khop2")
        rospy.loginfo("x: Dừng và thoát")


        while not rospy.is_shutdown():
            key = self.get_key()


            # Đặt lại vận tốc về 0 trước khi tính toán
            self.left_speed = 0.0
            self.right_speed = 0.0
            self.front_speed = 0.0
            self.prismatic_speed = 0.0
            self.revolute_speed = 0.0

            # Điều khiển chuyển động
            if key == 's':  # lui
                self.left_speed = self.max_speed
                self.right_speed = -self.max_speed
                self.front_speed = 0.0
            elif key == 'w':  # Tien
                self.left_speed = -self.max_speed
                self.right_speed = self.max_speed
                self.front_speed = 0.0
            elif key == 'd':  # Sang phai
                self.left_speed = -self.max_speed
                self.right_speed = self.max_speed
                self.front_speed = -self.max_speed
            elif key == 'a':  # Sang trai
                self.left_speed = self.max_speed
                self.right_speed = -self.max_speed
                self.front_speed = self.max_speed
            elif key == 'q':  # Xoay trái
                self.left_speed = -self.max_speed
                self.right_speed  = self.max_speed
                self.front_speed = self.max_speed
            elif key == 'e':  # Xoay phải
                self.left_speed = self.max_speed
                self.right_speed = -self.max_speed
                self.front_speed = -self.max_speed
            elif key == 'z':  # Dừng và thoát
                self.left_speed = 0.0
                self.right_speed = 0.0
                self.front_speed = 0.0
            elif key =='c': #khop1
                 if self.prismatic_position + (self.prismatic_direction * 0.01) > self.max_prismatic:
```

phần logic điều khiển từng động cơ

Code điều khiển thống qua vận tôc mong muốn:

```python
#!/usr/bin/env python3
import rospy
from geometry_msgs.msg import Twist
from std_msgs.msg import Float64

# Thông số robot
WHEEL_RADIUS = 0.055  # Bán kính bánh xe (m)
L = 0.2  # Khoảng cách từ tâm robot đến bánh xe

class OmniWheelController:
    def __init__(self):
        rospy.init_node('cmd_vel_to_omni')

        # Publisher cho 3 bánh xe
        self.pub_left = rospy.Publisher('/left_wheel_joint_velocity_controller/command', Float64, queue_size=10)
        self.pub_right = rospy.Publisher('/right_wheel_joint_velocity_controller/command', Float64, queue_size=10)
        self.pub_front = rospy.Publisher('/front_wheel_joint_velocity_controller/command', Float64, queue_size=10)

        # Subscriber nhận lệnh /cmd_vel
        rospy.Subscriber('/cmd_vel', Twist, self.cmd_vel_callback)

    def cmd_vel_callback(self, msg):
        Vx = msg.linear.x
        Vy = msg.linear.y
        omega = msg.angular.z

        # Tính vận tốc cho từng bánh xe
        v_left = (Vx - Vy - omega * L) / WHEEL_RADIUS
        v_right = (Vx + Vy + omega * L) / WHEEL_RADIUS
        v_front = (-Vx + Vy - omega * L) / WHEEL_RADIUS

        # Xuất lệnh điều khiển
        self.pub_left.publish(-v_left)
        self.pub_right.publish(v_right)
        self.pub_front.publish(v_front)

        rospy.loginfo(f"Left: {v_left:.2f}, Right: {v_right:.2f}, Front: {v_front:.2f}")

if __name__ == '__main__':
    controller = OmniWheelController()
    rospy.spin()

```

từ vận tốc mong muốn ta sẽ tính được vận tốc cần của từng động cơ sau đó publish lên topic để điều khiển động cơ

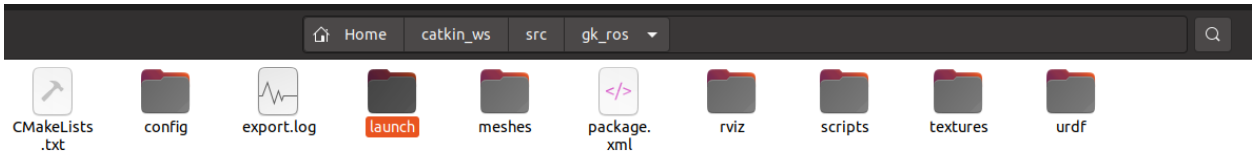
code viêt node encoder hiển thị tốc độ quay động cơ :

```python
#!/usr/bin/env python3
import rospy
from std_msgs.msg import Float64

class EncoderFromCommand:
    def __init__(self):
        rospy.init_node('encoder_from_command', anonymous=True)

        # Subscriber để nhận dữ liệu vận tốc từ các topic điều khiển
        rospy.Subscriber('/front_wheel_joint_velocity_controller/command', Float64, self.front_wheel_callback)
        rospy.Subscriber('/left_wheel_joint_velocity_controller/command', Float64, self.left_wheel_callback)
        rospy.Subscriber('/right_wheel_joint_velocity_controller/command', Float64, self.right_wheel_callback)

        self.left_wheel_speed = 0.0
        self.right_wheel_speed = 0.0
        self.front_wheel_speed=0.0

        rospy.loginfo("Node encoder_from_command đang chạy...")

        rospy.spin()

    def front_wheel_callback(self, msg):
        self.front_wheel_speed = msg.data
        rospy.loginfo(f"ENCODER | Left Wheel Speed: {self.front_wheel_speed:.3f} rad/s")

    def left_wheel_callback(self, msg):
        self.left_wheel_speed = msg.data
        rospy.loginfo(f"ENCODER | Left Wheel Speed: {self.left_wheel_speed:.3f} rad/s")

    def right_wheel_callback(self, msg):
        self.right_wheel_speed = msg.data
        rospy.loginfo(f"ENCODER | Right Wheel Speed: {self.right_wheel_speed:.3f} rad/s")

if __name__ == '__main__':
    try:
        EncoderFromCommand()
    except rospy.ROSInterruptException:
        pass
```

node sẽ lấy thoong tin từ topic của các joint để đưa ra vân tộc của bánh

- Structure folder dự án :

Folder tên : gk_ros



-phần config sẽ chứa các file .yaml

-launch sẽ chứa 3 file launch : gazebo.launch, display.launch,control.launch

-rviz sẽ lưu file config rviz

-scripts sẽ chứa file node encoder , điều khiển qua bàn phím và điều khiển qua động học nguoc

-file meshes sẽ chứa các file stl của model robot

- file urdf chứa file urdf