BÁO CÁO THIẾT KẾ VÀ MÔ PHÔNG ROBOT

Họ và tên: Nguyễn Tuấn Cảnh

MSV: 22027502

Mô hình xe: mecanum 4 banhs

Tay máy: Rotation

Cảm biến : IMU,Encoder,Lidar

1. Giới Thiệu

Báo cáo này trình bày quá trình thiết kế và mô phông robot trong ROS, bao gồm thiết kế URDF/XACRO, cấu hình Gazebo, RViz, và các cơ chế điều khiển.

2. Dạng Robot, Động Học, Kích Thước

- Robot được thiết kế theo dạng 4 bánh Mecanum, cho phép di chuyển linh hoạt.
- Động học:

The inverse kinematic equations allow us to compute the indiviual wheel velocities when we want to achieve an overall base velocity.

- ω_{fl}, ω_{fr}, ω_{rl} and ω_{rr} represent the *angular velocities* for the front left, front right, rear left and rear right wheel respectively.
- v_x and v_y represent the robot's base linear velocity in the x and y direction respectively. The x direction is in front of the robot.
- ω_z is angular velocity of the robot's base around the z-axis.
- l_x and l_y represent the distance from the robot's center to the wheels projected on the x and y axis respectively.

$$\begin{cases} \omega_{fl} &= \frac{1}{r}[v_x - v_y - (l_x + l_y)\omega_z] \\ \omega_{fr} &= \frac{1}{r}[v_x + v_y + (l_x + l_y)\omega_z] \\ \omega_{rl} &= \frac{1}{r}[v_x + v_y - (l_x + l_y)\omega_z] \\ \omega_{rr} &= \frac{1}{r}[v_x - v_y + (l_x + l_y)\omega_z] \end{cases}$$

+ Khi đi theo chiều x, 4 bánh xe cung vận tốc

+ Khi đi theo chiều y thì bánh trước trái, bánh sau phải cùng vận tốc và bánh trước phải và bánh sau trái cùng vận tốc.

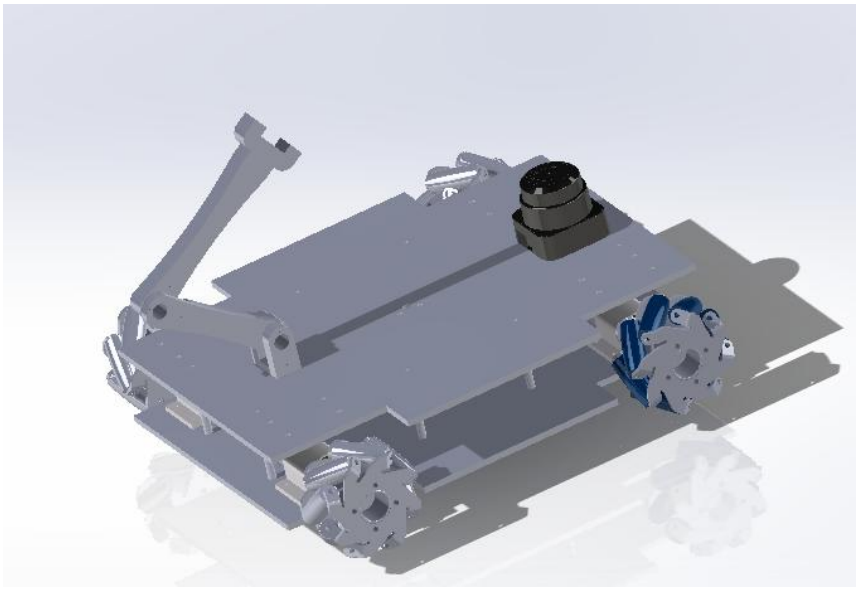
- Kích thước tổng thể:
  - Chiều ngang : 300 mm
  - Chiều dài : 350mm
  - Chiều cao đế dưới so với mặt đất: 13 mm
  - Bán kính bánh: 42 mm

- Chiều dài arm1: 100.8mm
- Chiều dài arm2: 146 mm

- Robot bao gồm các khớp tay máy (đểm bảo các bài toán điều khiển linh hoạt).

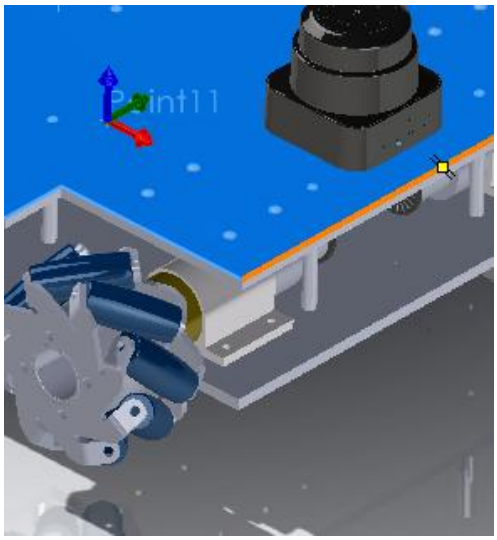
3. Thiết Kế SolidWorks, Trục Tọa Độ

- Mô hình được thiết kế bằng SolidWorks trước khi chuyển sang URDF

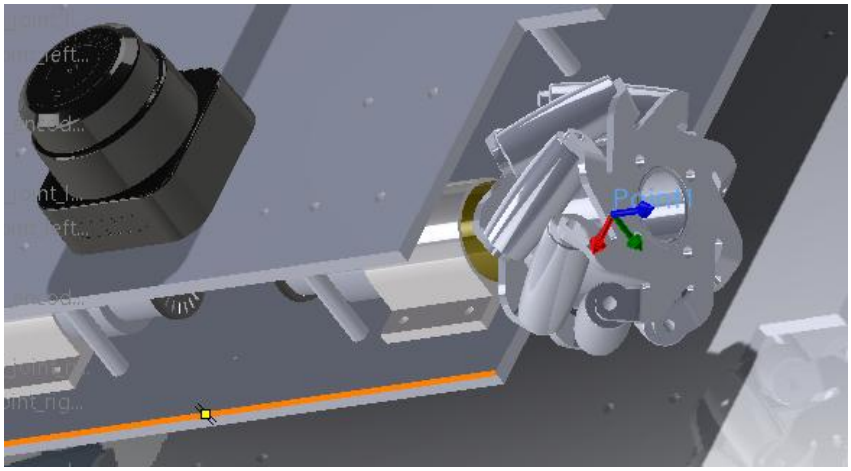


- Xác định trục toà độ toàn robot và từng bộ phận con.

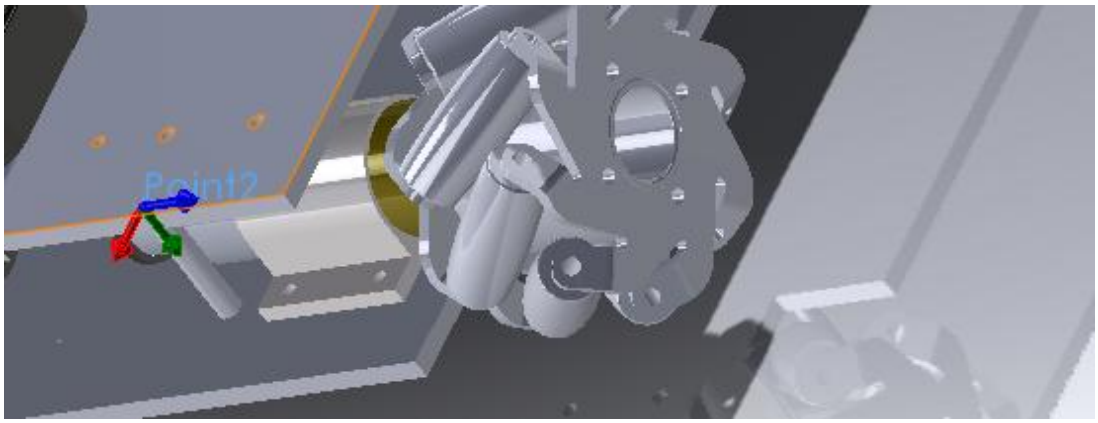
  - Trục tọa độ của base:



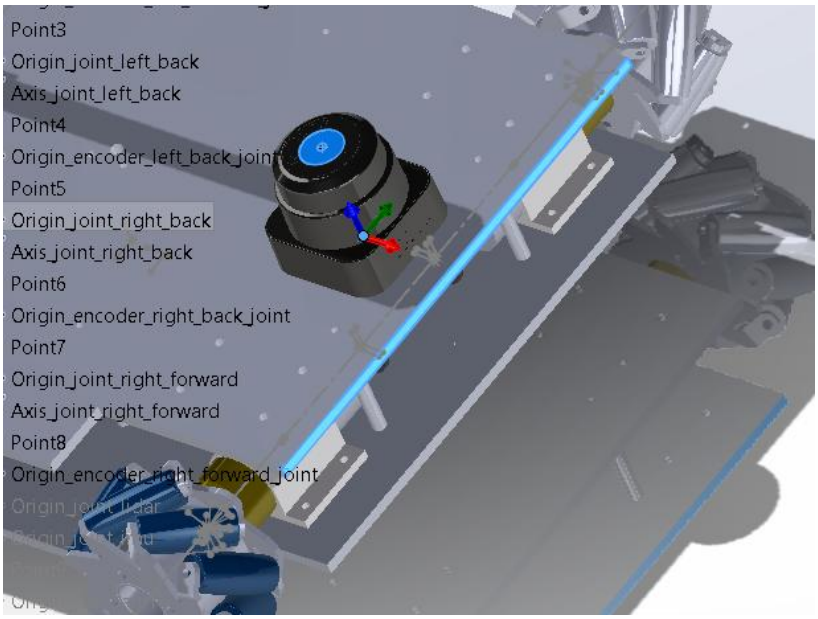
  -Trục tọa độ của bánh xe:



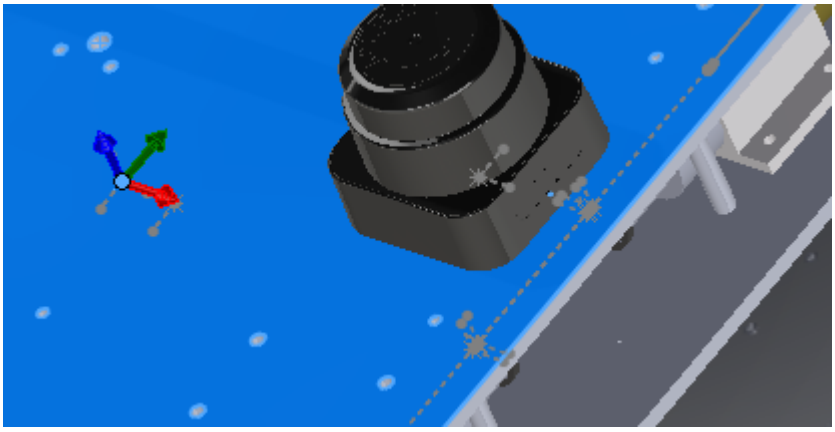
-Trục tọa độ của encoder:



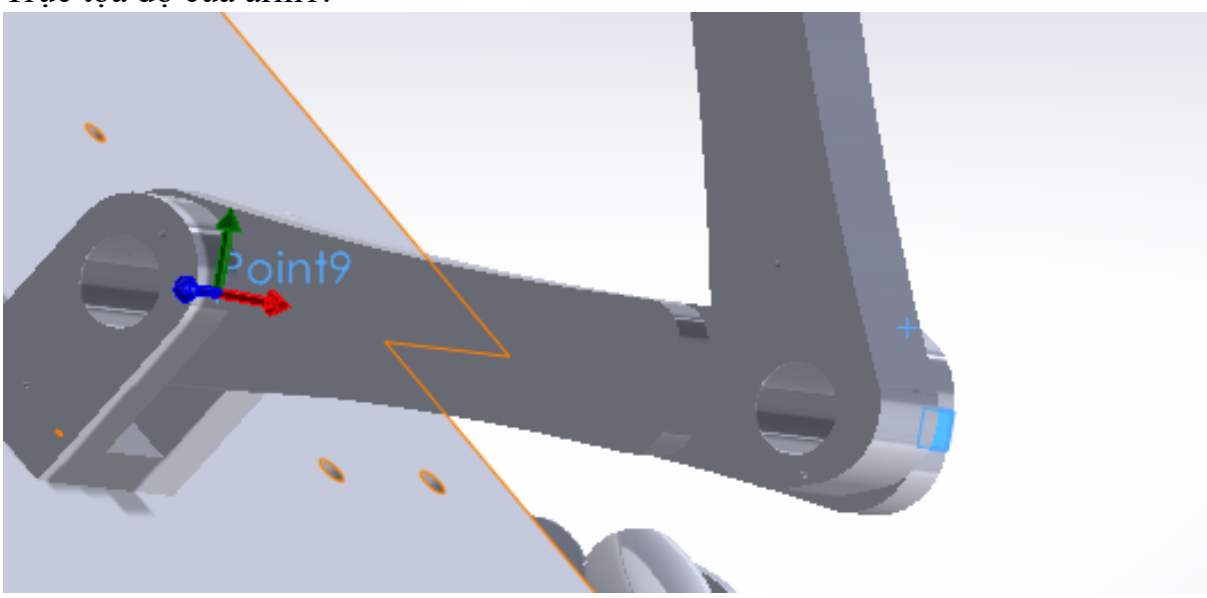
- Trục tọa độ cảu lidar:



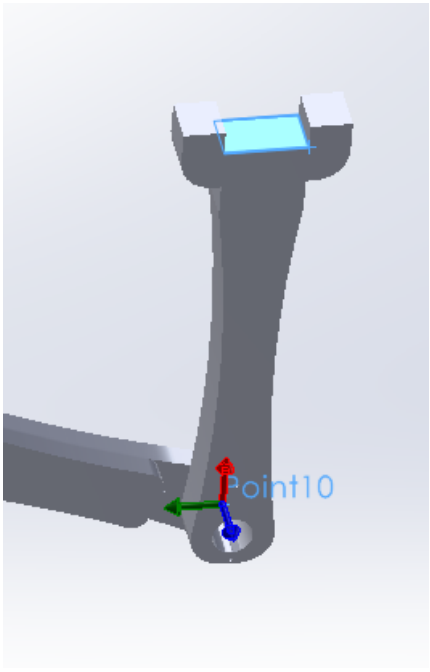
- Trục tọa độ của IMU:



- Trục tọa độ của arm1:



- Trục tọa độ của arm2:



4. Mô Tả File URDF/XACRO, Liên Kết, Cảm Biến, Gazebo

- Robot được mô hình hóa trong URDF/XACRO.
- File XACRO chính: robot.xacro, gồm các XACRO con:
    - base.xacro: Thân robot.
    - wheel.xacro: Bánh xe Mecanum.
    - arm.xacro: Tay máy
    - IMU.xacro
    - Lidar.xacro
    - Encoder.xacro
    - Plugin.xacro
    - Tranmission.xacro
- Mô tả liên kết:
    - Bánh xe liên kết với Base theo khớp kiểu continous
    - Encoder liên kết với Bánh theo khớp continous để đọc giá trị của /joint_states
    - Lidar liên kết với Base theo kiểu fixed
    - Imu liên kết với Base theo kiểu fixed
    - Arm1 liên kết với Base theo kiểu revolute
    - Arm2 liên kết với Arm1 theo kiểu revolute
- Gazebo plugin:
    - Sử dụng gazebo_ros_control để điều khiển các khớp.
    - Sử dụng libgazebo_ros_planar_move để điều khiển mô phỏng lại dáng đi của Mecanum.
    - Sử dụng Gazebo_Ros_ImuSensor để lấy cảm biến imu.
    - Sử dụng Gazebo_Ros_Laser để lấy cảm biến Lidar
- Gazebo Tranmission:
    - Sử dụng transmission với **transmission_interface//SimpleTransmission**

- Wheel và Encoder sử dụng **hardware_interface/VelocityJointInterface** để điều khiển vận tốc bánh xe
- Arm sử dụng **hardware_interface/PositionJointInterface**để điều khiển vị trí cánh tay

- Mô tả gazebo:

```
<world name="default">
  <physics type="ode">
    <real_time_update_rate>1000</real_time_update_rate>
    <max_step_size>0.001</max_step_size>
    <gravity>0 0 -9.81</gravity>
  </physics>
  <!-- A global light source -->
  <include>
    <uri>model://sun</uri>
  </include>
  <!-- A ground plane -->
  <include>
    <uri>model://ground_plane</uri>
  </include>
```

- Cập nhật 1000 lần/giây
- Bước thời gian 1ms
- Gravity = (0, 0, -9.81) : Trọng lực theo trục Z

5. Mô Tả Cơ Chế Điều Khiển Trong Gazebo

-Viết file YAML được sử dụng để cấu hình các tham số của bộ điều khiển bao gom workspace và type điều khiển:

mecanum_steering_control > config > ! diff_drive.yaml

```yaml
meca:
  mecanum_drive_controller:
    type: "velocity_controllers/JointGroupVelocityController"
    publish_rate: 50
    joints:
      - joint_left_forward1
      - joint_right_forward1
      - joint_left_back1
      - joint_right_back1
```

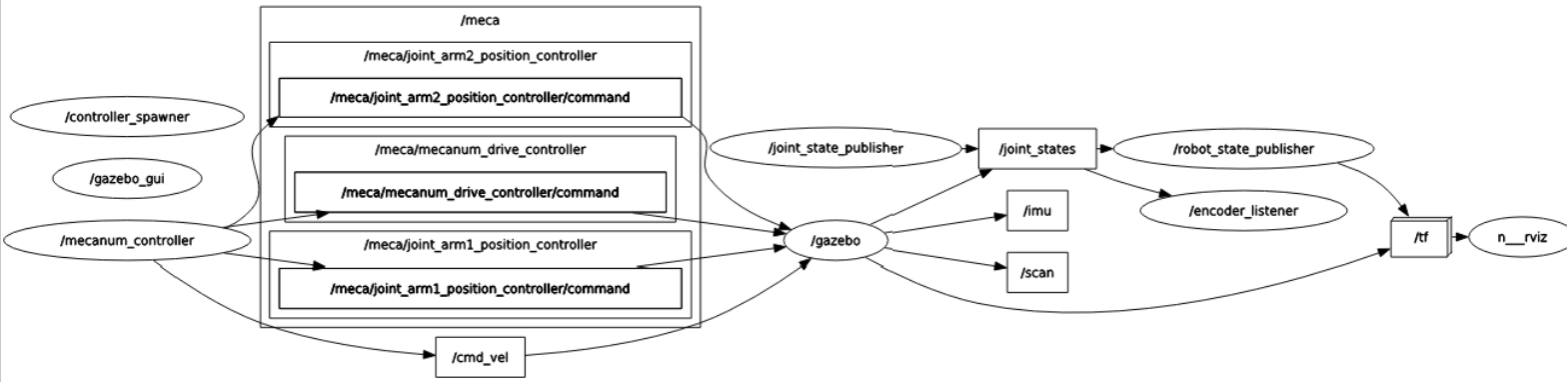
```
rc > mecanum_steering_control > config > ! mecanum_encoder_controller.yaml
meca:
  mecanum_encoder_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50
    joints:
      - encoder_left_forward_joint
      - encoder_left_back_joint
      - encoder_right_back_joint
      - encoder_right_forward_joint
```

```
meca:
  joint_arm1_position_controller:
    type: position_controllers/JointPositionController
    joint: joint_arm1

  joint_arm2_position_controller:
    type: position_controllers/JointPositionController
    joint: joint_arm2
```

-Bây giờ chúng ta sẽ pub vào vào các wordspace này để điều khiển động cơ hoặc tay máy

VD:Rospy.Publisher("/meca/mecanum_drive_controller/command", Float64MultiArray, queue_size=10) để điề khiển 4 bánh xe



-Sơ đồ node:

1.Cụm điều khiển chính /meca:

- /meca/mecanum_drive_controller:

    - Điều khiển hệ thống bánh xe mecanum.
    - Nhận lệnh từ **/cmd_vel** và gửi đến topic **/meca/mecanum_drive_controller/command**.
- /meca/joint_arm1_position_controller:
    - Điều khiển vị trí của khớp thứ nhất của cánh tay robot.
    - Gửi lệnh đến **/meca/joint_arm1_position_controller/command**
- /meca/joint_arm2_position_controller:
    - Điều khiển vị trí của khớp thứ hai của cánh tay robot.
    - Gửi lệnh đến **/meca/joint_arm2_position_controller/command**
- Node này xuất lệnh vận tốc **/cmd_vel** để gửi đến Gazebo.

2. Các node điều khiển chính

- /mecanum_controller:
    - Xuất lệnh **/cmd_vel** (tốc độ di chuyển của robot).
    - Kết nối với **/meca/mecanum_drive_controller** để điều khiển robot di chuyển.
- /controller_spawner:
    - Dùng để khởi tạo các controller trong **ros_control**.
- /gazebo:
    - Mô phỏng robot trong môi trường ảo.
    - Nhận lệnh từ các controller và gửi thông tin trạng thái của robot ra các topic như **/joint_states**, **/imu**, **/scan**

3. Dữ liệu cảm biến và trạng thái robot

- **/joint_states:**
    - Được xuất bởi /joint_state_publisher và Gazebo.
    - Dùng để theo dõi vị trí các khớp của robot.
- **/imu:**
    - Cảm biến gia tốc
- **/scan:**
    - Dữ liệu quét từ cảm biến LiDAR.
- **/encoder_listener:**
    - Lắng nghe dữ liệu encoder từ /joint_states.
- **/robot_state_publisher:**
    - Chuyển đổi trạng thái robot thành thông tin TF (biến đổi tọa độ)

6. Các Thành Phần Chính Của Code, Structure Folder

a. Các Thành Phần Chính Của Code

***Code điều khiển**

```
class TeleopMecanumArm:
    def __init__(self):
        rospy.init_node("teleop_mecanum_arm", anonymous=True)

        self.wheel_pub = rospy.Publisher("/meca/mecanum_drive_controller/command", Float64MultiArra
        self.cmd_vel_pub = rospy.Publisher("/cmd_vel", Twist, queue_size=10)
        self.arm1_pub = rospy.Publisher('/meca/joint_arm1_position_controller/command', Float64, qu
        self.arm2_pub = rospy.Publisher('/meca/joint_arm2_position_controller/command', Float64, qu

        self.arm1_angle = 0.0
        self.arm2_angle = 0.0
        self.arm_step = 0.01

        rospy.loginfo("Teleop Mecanum Arm Node Started")
```

- pub vào các topic de điều khiển

```
    wheels = Float64MultiArray(data=[0.0, 0.0, 0.0, 0.0])
    cmd_vel = Twist()
    while not rospy.is_shutdown():
        key = self.get_key()



        if key == "w":
            wheels.data = [2, 2, 2, 2]
            cmd_vel.linear.x = 0.04
            cmd_vel.linear.y = 0.0
        elif key == "x":
            wheels.data = [-2, -2, -2, -2]
            cmd_vel.linear.x = -0.04
            cmd_vel.linear.y = 0.0
        elif key == "q":
            wheels.data = [-6, 6, -6, 6]
            cmd_vel.linear.x = 0.0
            cmd_vel.linear.y = 0.0
        elif key == "e":
            wheels.data = [6, -6, 6, -6]
            cmd_vel.linear.x = 0.0
            cmd_vel.linear.y = 0.0
        elif key == "a":
            wheels.data = [-6, 6, 6, -6]
            cmd_vel.linear.y = 0.04
            cmd_vel.linear.x = 0.0
```

- Truyền giá trị tương ứng khi ấn nút

*Code tạo topic encoder

```
    def __init__(self):
        rospy.init_node("joint_states_listener", anonymous=True)
        rospy.Subscriber("/joint_states", JointState, self.joint_states_callback)

        # Publisher cho encoder velocities
        self.encoder_pub = rospy.Publisher("/encoder_velocities", JointState, queue_size=10)

        # Lưu giá trị vận tốc cũ
        self.encoder_velocities = {
            "encoder_left_forward_joint": 0.0,
            "encoder_left_back_joint": 0.0,
            "encoder_right_back_joint": 0.0,
            "encoder_right_forward_joint": 0.0
        }

        self.last_log_time = rospy.Time.now()
        rospy.loginfo("Listening to /joint_states topic...")
```

- Một publisher được tạo với tên topic /encoder_velocities. Publisher này sẽ phát ra thông điệp kiểu JointState, chứa các vận tốc của các encoder joints.

-Sử dụng một dictionary self.encoder_velocities để lưu trữ các giá trị vận tốc của bốn encoder joints

```
def joint_states_callback(self, msg):
    current_time = rospy.Time.now()


    if (current_time - self.last_log_time).to_sec() >= 1.0:
        rospy.loginfo("Received joint states:")

        for joint_name in self.encoder_velocities.keys():
            if joint_name in msg.name:
                index = msg.name.index(joint_name)
                if len(msg.velocity) > index:
                    velocity = msg.velocity[index]
                else:
                    velocity = 0.0
            else:
                velocity = 0.0
            self.encoder_velocities[joint_name] = velocity
            rospy.loginfo(f"{joint_name}: Velocity={velocity:.6f}")


        self.last_log_time = current_time


        encoder_msg = JointState()
        encoder_msg.name = list(self.encoder_velocities.keys())
        encoder_msg.velocity = list(self.encoder_velocities.values())
        encoder_msg.effort = [0.0] * len(encoder_msg.name)
        self.encoder_pub.publish(encoder_msg)
```

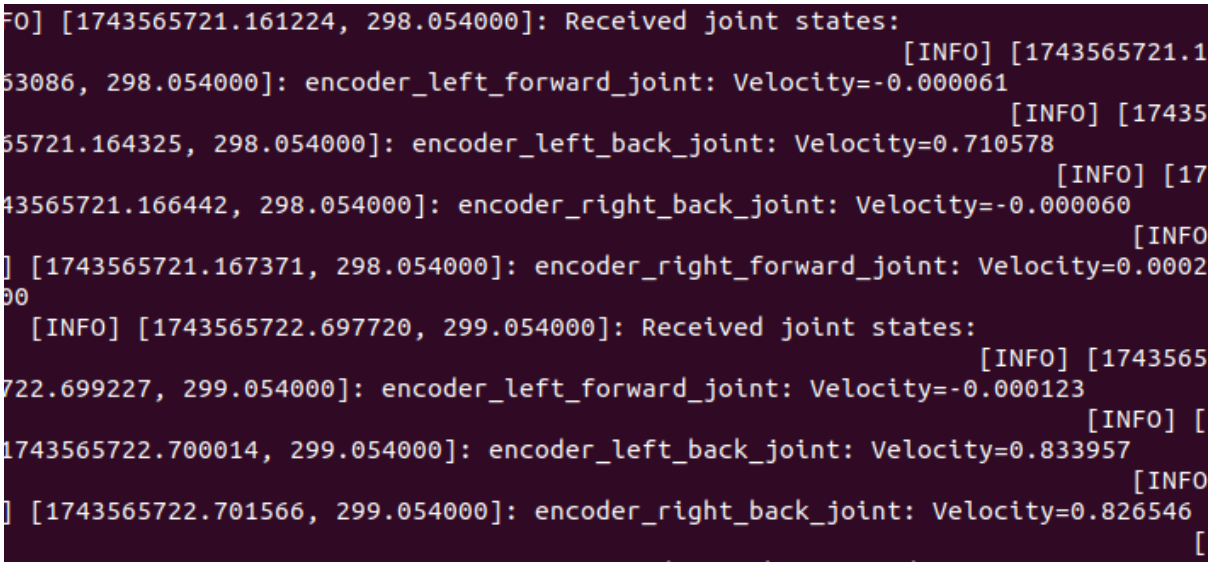
-Trong mỗi lần gọi callback, hàm này sẽ kiểm tra xem có thông điệp mới từ /joint_states hay không và tính toán vận tốc.

-Nếu thời gian giữa các lần nhận dữ liệu vượt quá 1 giây, nó sẽ in ra các vận tốc của các encoder joints vào log.

-Sau đó, hàm sẽ tạo một đối tượng JointState mới, trong đó name là danh sách tên các encoder joints và velocity là các giá trị vận tốc tương ứng.

-Cuối cùng, đối tượng JointState này được phát ra trên topic /encoder_velocities

*Kết quả là cứ 1s encoder sẽ được in giá trị len terminal :



b. **Structure Folder**

-Gồm 2 Fordel

- Meca: chứa urdf,mesh và xacro
  - URDF: Chứa file URDF
  - Xacro: Chứa các file XACRO
  - Meshe: chứa bản vẽ 3D STL
  - Words: chứa file gazebo đã tạo
- Mecanum_steering_control: chứa các file điều khiển:
  - Launch: File launch tự động khởi chạy mô phỏng
  - Config: Chưa các file YAML cài đặt điều khiển
  - Scripts: Code điều khiển và tạo encoder
  - Rviz: chứa file rviz đã lưu

7. Kết Luận

Dự án đã hoàn thành việc thiết kế và mô phông robot 4 bánh Mecanum trong Gazebo và RViz. Robot đã điều khiển được bằng bàn phím, có cài đặt encoder, và hiển thị các thông số cần thiết.