

**ĐẠI HỌC QUỐC GIA HÀ NỘI**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



**BÁO CÁO GIỮA KỲ: XÂY DỰNG, MÔ PHỎNG VÀ ĐIỀU  
KHIỂN ROBOT OMNI BA BÁNH**

**Môn học: Lập trình robot với ROS - RBE3017\_1**  
**Tô Vũ Khôi Nguyên - 22027500**

# **MỤC LỤC**

## **1. TỔNG QUAN ĐỀ TÀI**

- 1.1 Mục tiêu dự án
- 1.2 Tổng quan về robot Omni ba bánh

## **2. THIẾT KẾ**

- 2.1. Mô hình robot thiết kế trên SOLIDWORKS
- 2.2. Cách gán hệ trục tọa độ cho các bộ phận
- 2.3. Xuất file URDF từ SOLIDWORKS

## **3. MÔ PHỎNG**

- 3.1. Mô phỏng trên Gazebo
- 3.2. Mô phỏng cảm biến trên Gazebo
- 3.3 Mô phỏng trên Rviz

## **4. ĐIỀU KHIỂN**

- 4.1. Mô hình động học của robot
- 4.2. Điều khiển bánh
- 4.3. Điều khiển tay máy

## **5.TỔNG KẾT**

- 5.1. Kết luận

# **1. Tổng quan đề tài:**

## **1.1 Mục tiêu dự án:**

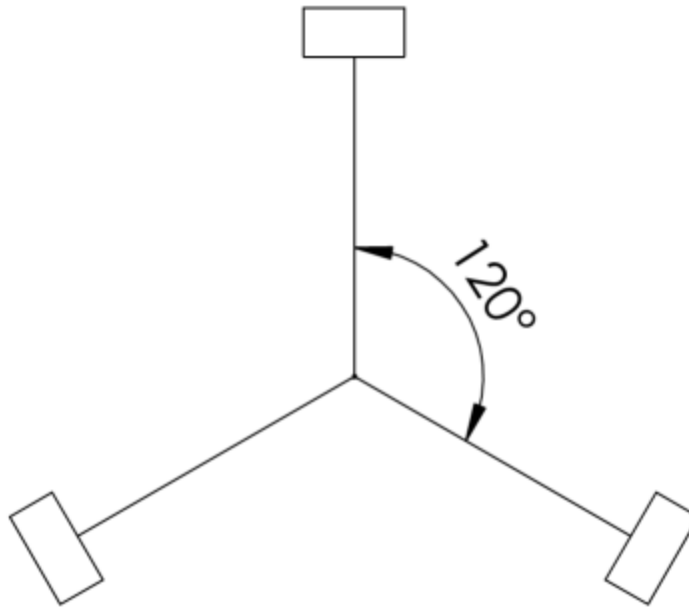
Dự án thiết kế một robot omni ba bánh có khả năng di chuyển linh hoạt trong môi trường giả lập Gazebo. Robot sẽ được trang bị các cảm biến như Camera, IMU, LiDAR và hệ thống tay máy để thực hiện các tác vụ mô phỏng thao tác với vật thể. Ngoài ra, robot phải hỗ trợ điều khiển từ xa thông qua bàn phím hoặc giao diện ROS.

## **1.2 Tổng quan về robot Omni ba bánh:**

Robot Omni 3 bánh là một loại robot sử dụng ba bánh xe đa hướng (omni wheels), cho phép di chuyển linh hoạt theo mọi hướng mà không cần xoay thân. Cấu trúc chính của robot bao gồm:

**Thân chính:** Khung xe gắn các cảm biến, động cơ và tay máy

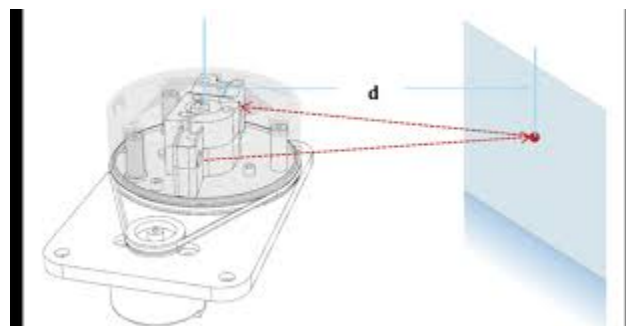
**Ba bánh omni:** Được bố trí lệch nhau theo góc 120 độ như hình, giúp robot có thể di chuyển linh hoạt theo mọi hướng.



**Cảm biến:** Bao gồm camera, IMU và LiDAR để thu thập dữ liệu về môi trường và trạng thái chuyển động của robot.

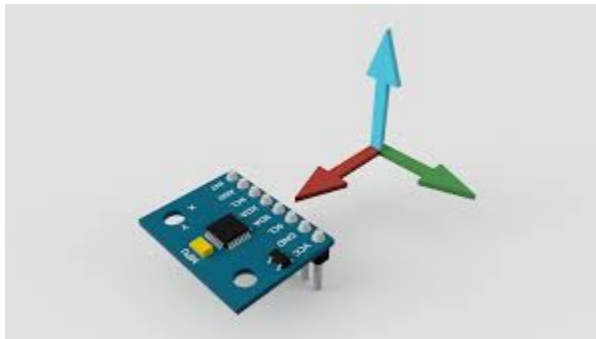
**- Cảm biến LiDAR:**

Cảm biến LiDAR hoạt động bằng cách phát tia laser và đo thời gian phản xạ từ các vật thể để xác định khoảng cách và hình dạng. Quá trình quét laser từ nhiều góc độ giúp tạo ra hình ảnh 3D chính xác của môi trường trong thời gian thực.



## **Cảm biến IMU:**

Cảm biến IMU là thiết bị giúp robot cảm nhận được chuyển động và hướng của mình. Nó đo lường gia tốc, tốc độ góc và phương hướng, cung cấp thông tin quan trọng về vị trí và chuyển động trong không gian, từ đó hỗ trợ robot duy trì thăng bằng và điều khiển chính xác trong quá trình di chuyển.



## **Cảm biến Camera:**

Cảm biến Camera giúp robot quan sát và thu thập thông tin hình ảnh về môi trường từ đó hỗ trợ xử lý các tác vụ khác.



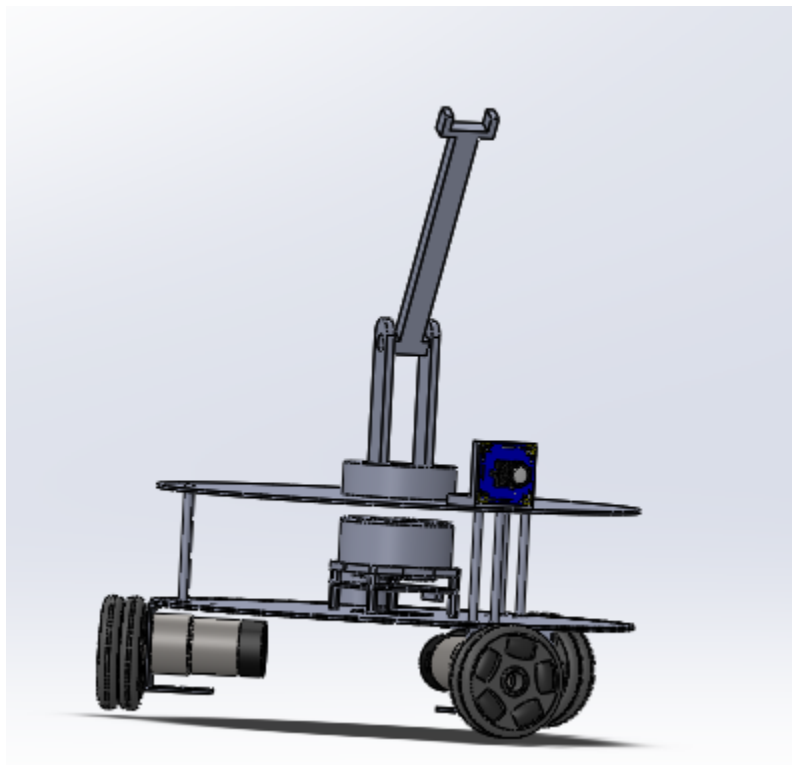
**Tay máy:** Tay máy 2 khớp quay có thể tự do quay các khớp

## **2. Thiết kế robot Omni ba bánh:**

### **2.1. Mô hình robot thiết kế trên SOLIDWORKS:**

#### **Thiết kế thân chính:**

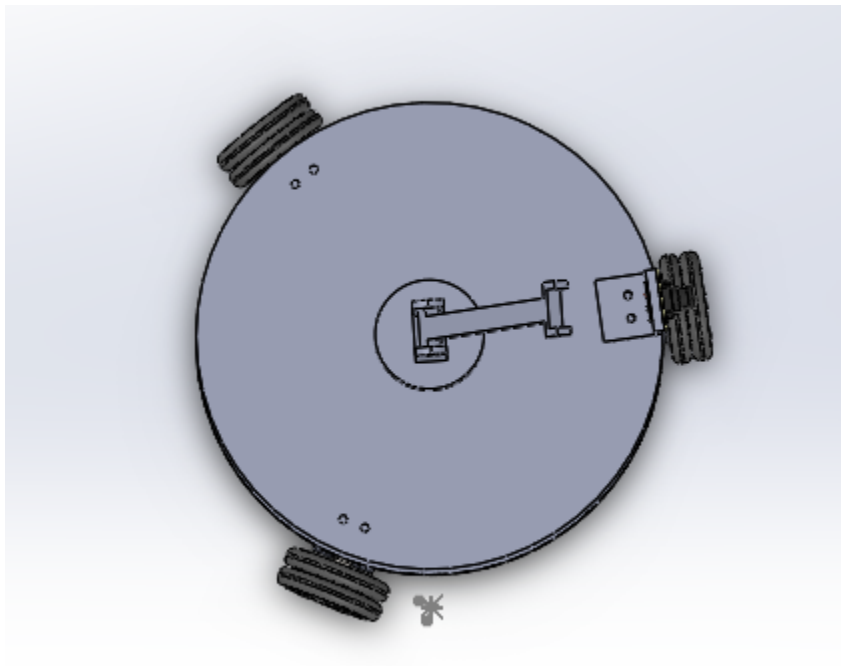
Robot có thiết kế hình tròn với hai tầng: tầng dưới được trang bị cảm biến LiDAR để quét và thu thập dữ liệu môi trường xung quanh, trong khi tầng trên chứa tay máy có hai khớp quay, cho phép thực hiện các thao tác như nâng, di chuyển và xoay vật thể.



Mô hình robot trên SOLIDWORKS

#### **Bánh xe:**

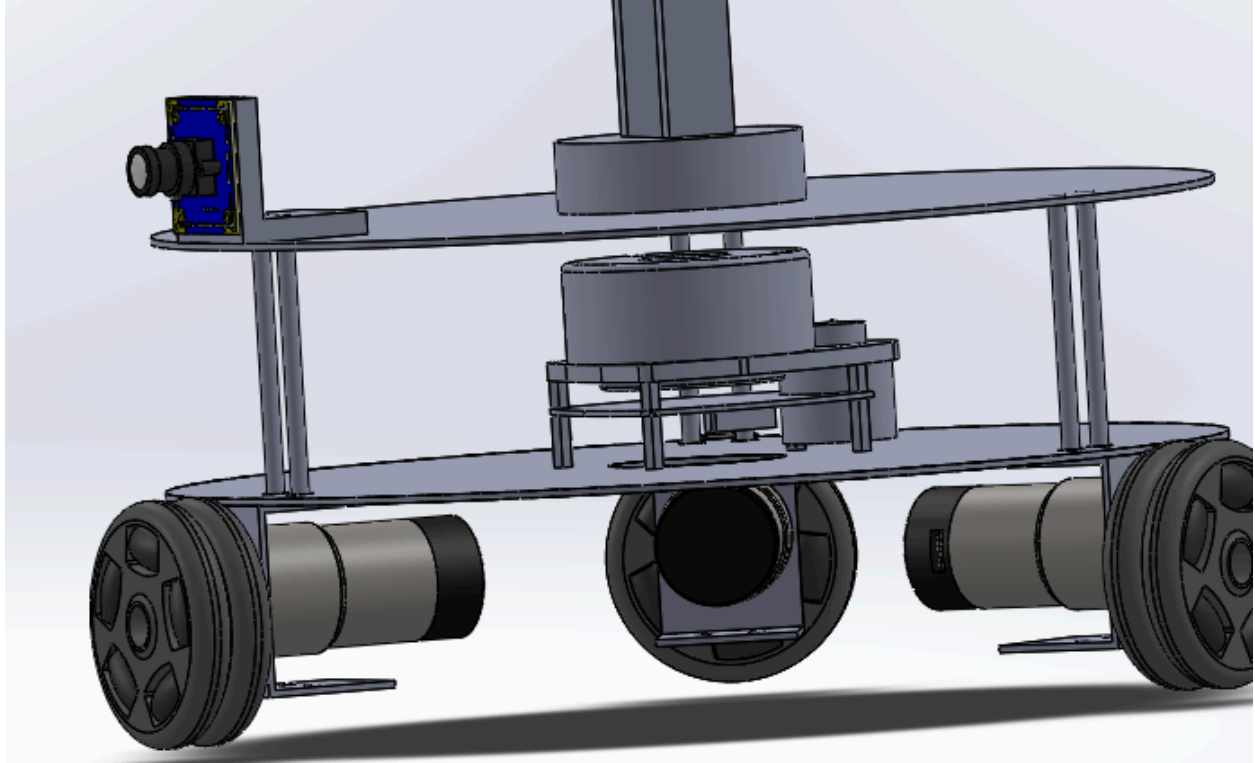
Để thực hiện khả năng di chuyển như mục tiêu của đề tài, robot phải có 3 bậc tự do trong mặt phẳng di chuyển gồm di chuyển độc lập trong mặt phẳng oxy và xoay quanh trục z. Do đó, cần tối thiểu 3 bánh xe đa hướng với các động cơ điều khiển độc lập, vì mỗi động cơ cho khả năng điều khiển một bậc tự do. Các bánh xe được gắn lên robot và lệch nhau một góc 120 độ như hình



### **Cảm biến:**

Cảm biến LiDAR được đặt ở trung tâm tầng dưới để có thể quét được bản đồ mà không bị cản bởi tay máy

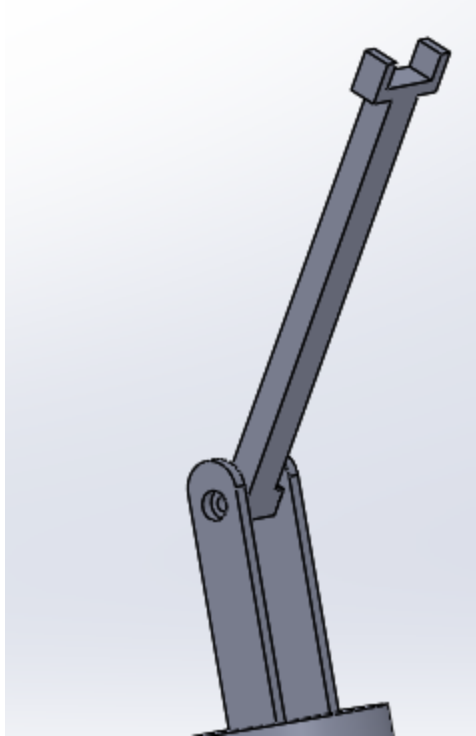
Cảm biến IMU và Camera được đặt ở tầng trên cùng tay máy để tránh cản trở phạm vi quét của LiDAR



### **Tay máy:**

Tay máy 2 khớp quay linh hoạt đặt ở trung tâm tầng trên để có thể thoải mái trong việc di chuyển và thực hiện các thao tác đơn giản như gấp thả vật





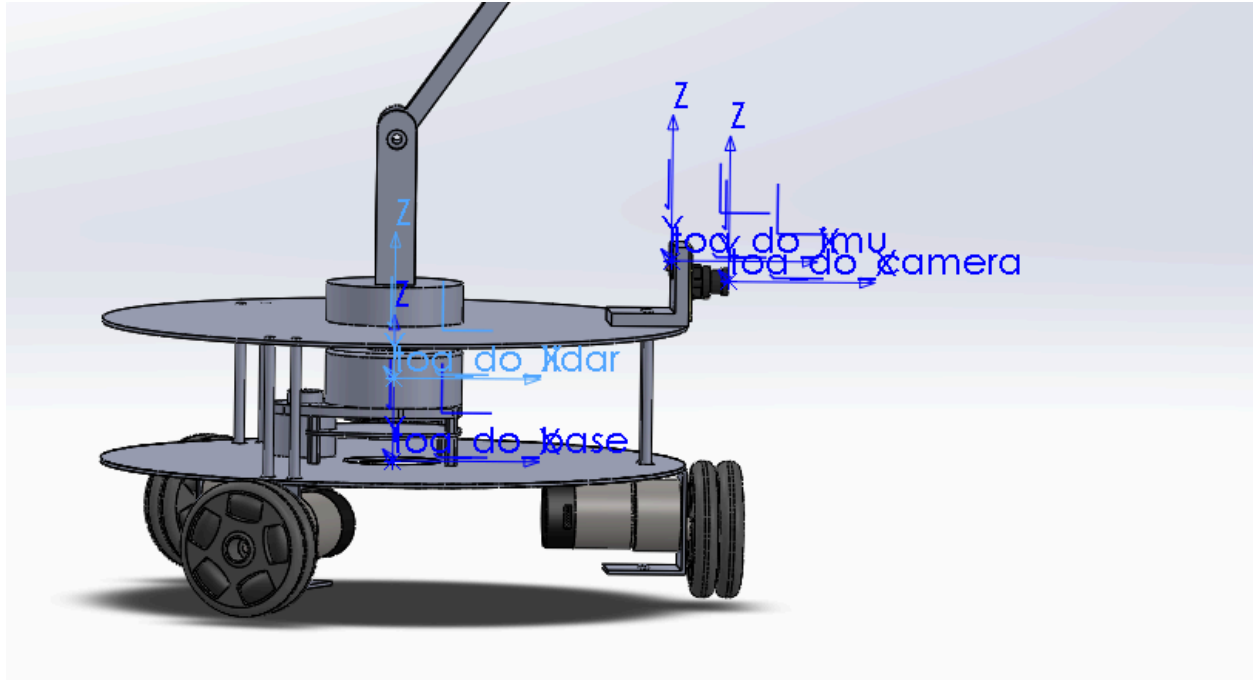
## **2.2 Gắn trục tọa độ**

Hệ trục tọa độ của Robot và các cảm biến được xác định:

Trục X: Hướng về phía trước của robot.

Trục Z: Hướng lên trong không gian 3D.

Trục Y: Được xác định theo trục X và trục Z theo quy tắc bàn tay phải

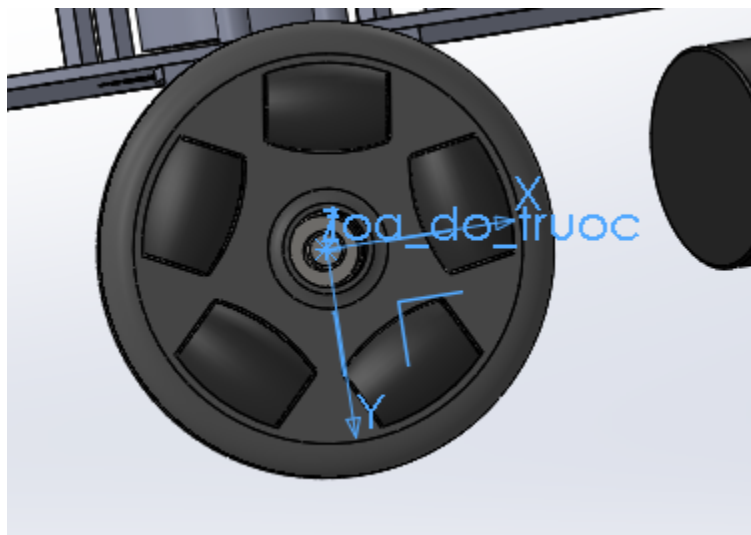


Hệ trục tọa độ của bánh xe:

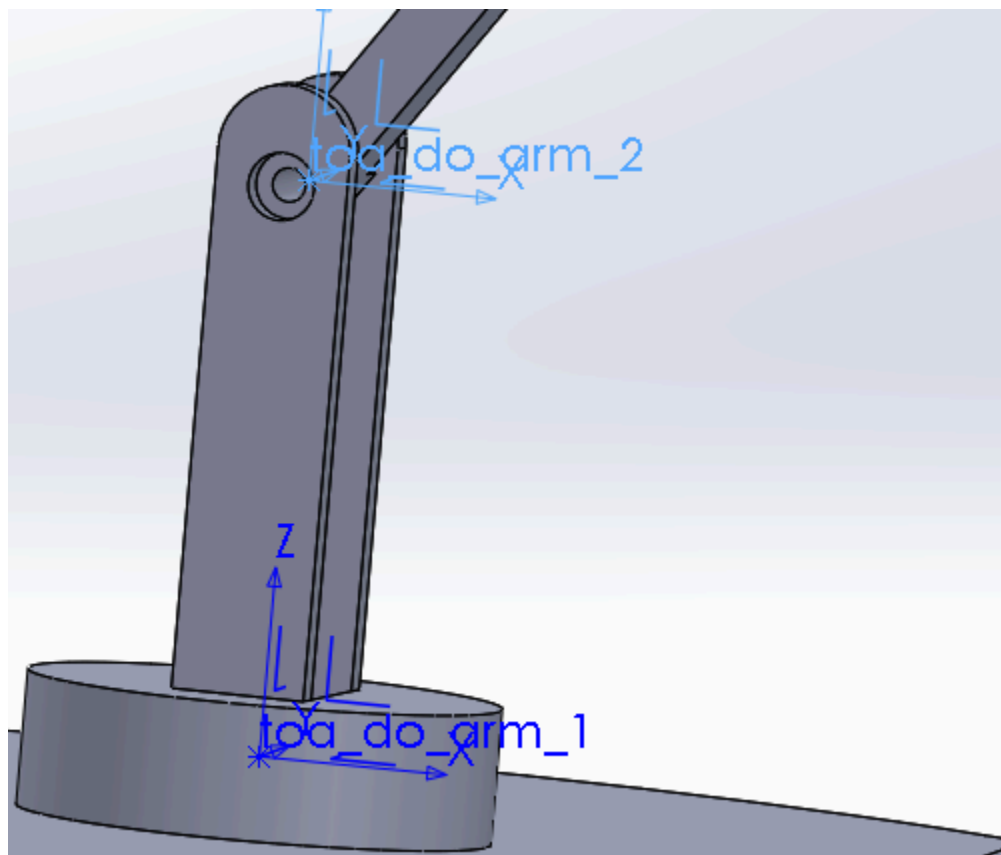
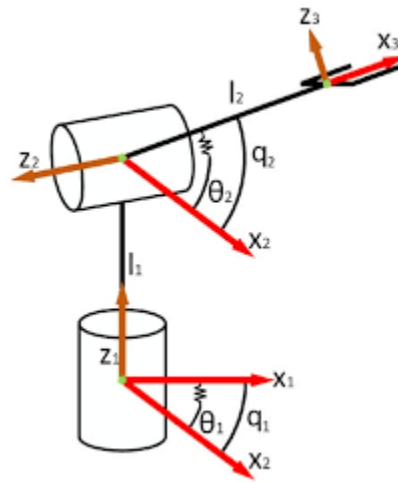
Trục Z: Hướng theo trục quay của bánh về có chiều hướng vào tâm của bánh xe

Trục X: Vuông góc với trục Z và nằm trên mặt phẳng bánh xe

Trục Y: Được xác định theo trục X và trục Z theo quy tắc bàn tay phải



Cách đặt trục tay máy dựa theo quy tắc Denavit-Hartenberg:



## **2.3. Xuất file URDF từ SOLIDWORKS**

Mô hình robot được thiết kế trên SolidWorks, đảm bảo đúng tỷ lệ thực tế và có thể xuất sang URDF để tích hợp vào ROS.

### **- Cấu trúc URDF**

config/: Chứa các file cấu hình

launch/: Chứa các file launch để khởi chạy mô phỏng, RViz, Gazebo và node điều khiển.

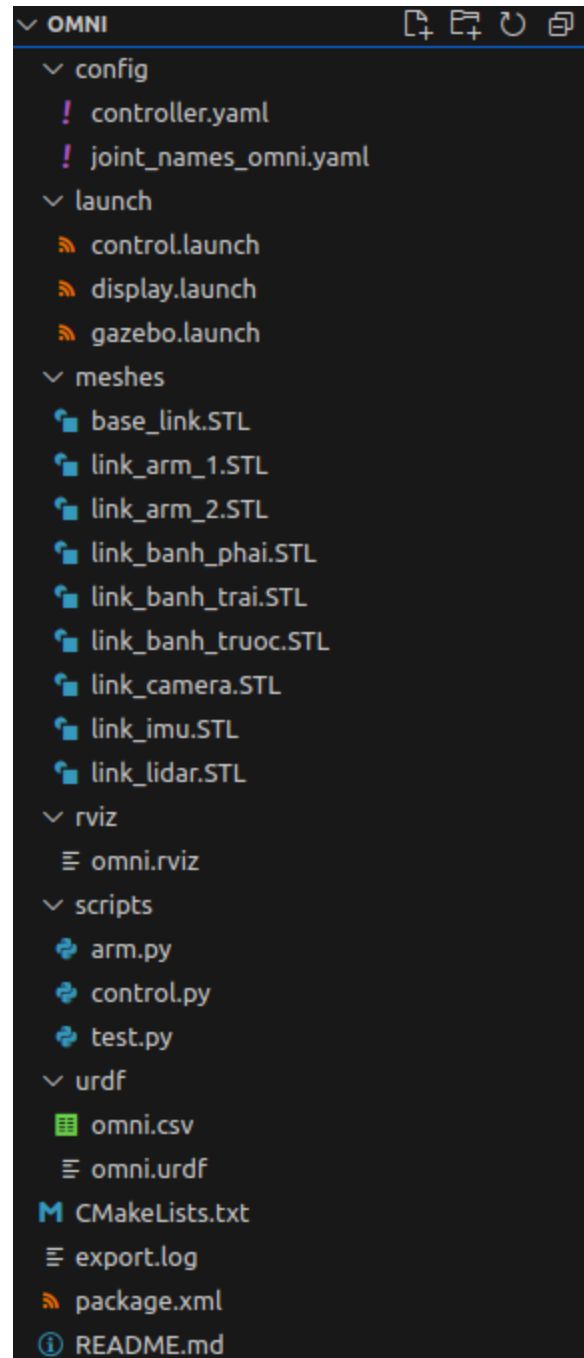
meshes/: Chứa các file mô hình 3D ( ở dạng .stl) của robot.

rviz/: Chứa các cấu hình hiển thị cho RViz.

scripts/: Chứa các script Python dùng để điều khiển robot và tay máy.

urdf/: Chứa file mô tả robot URDF/Xacro.

CMakeLists.txt và package.xml: Các file cần thiết để build package trong ROS.



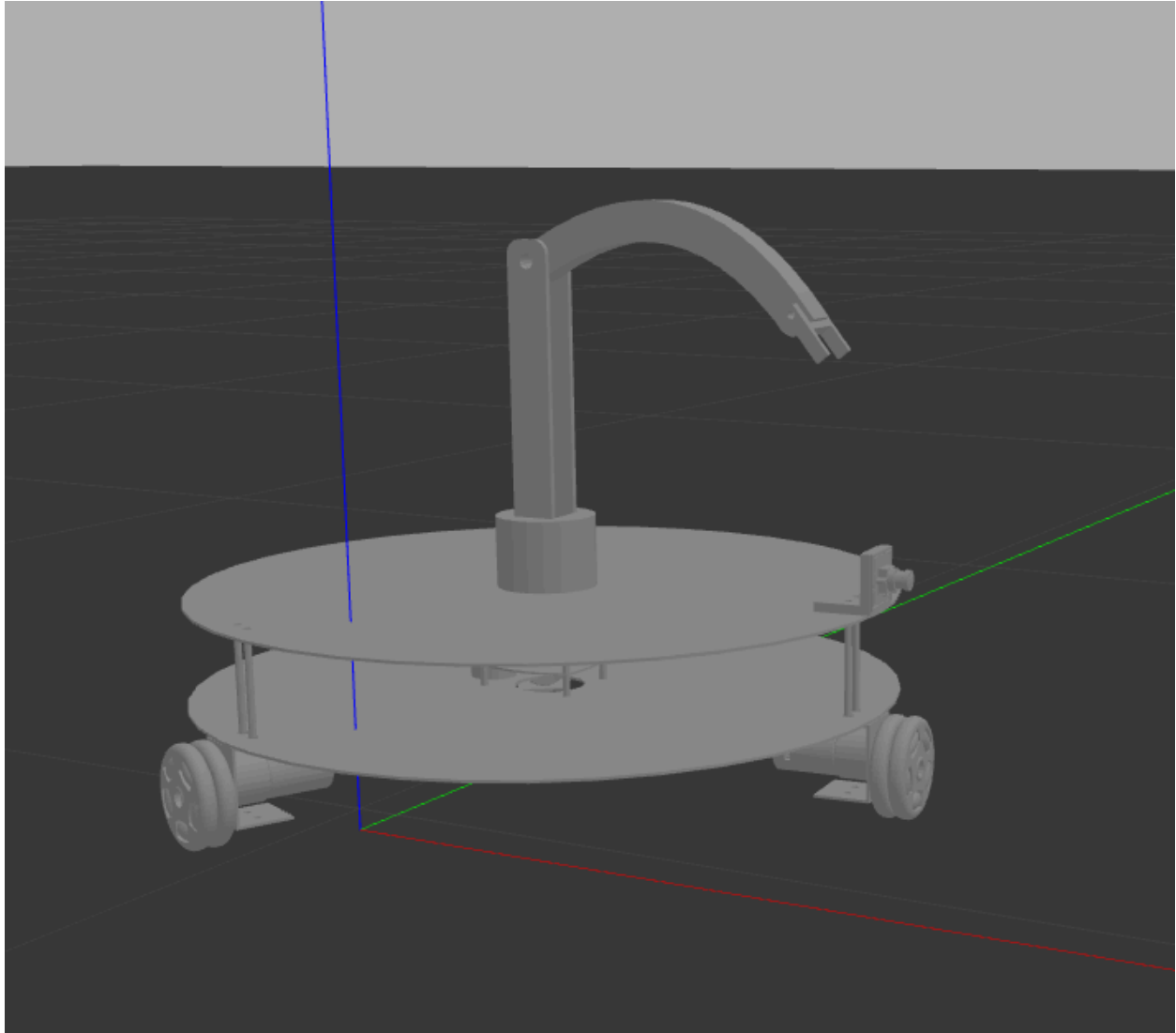
Xuất file URDF trên online trước khi chuyển sang Gazebo để đảm bảo không có lỗi



### **3. Mô phỏng robot Omni ba bánh:**

#### **4.1 Mô phỏng trong Gazebo**

- Chạy file gazebo.launch để hiển thị robot trong môi trường gazebo



Robot đã xuất hiện trong môi trường Gazebo

### 3.2 Mô phỏng cảm biến trên gazebo

- **Plugin camera:** Cho phép robot quan sát môi trường xung quanh trong mô phỏng

```
<gazebo reference="link_camera">  
  <material>Gazebo/Red</material>  
  <sensor name="camera1" type="camera">  
    <pose>0 0 0 0 0 0</pose>  
    <visualize>true</visualize>
```

```

<update_rate>30.0</update_rate>

<camera name="head">
  <horizontal_fov>1.089</horizontal_fov>
  <image>
    <width>800</width>
    <height>800</height>
    <format>R8G8B8</format>
  </image>
  <clip>
    <near>0.02</near>
    <far>8.0</far>
  </clip>
  <noise>
    <type>gaussian</type>
    <mean>0.0</mean>
    <stddev>0.007</stddev>
  </noise>
</camera>

<plugin name="camera_controller" filename="libgazebo_ros_camera.so">
  <alwaysOn>true</alwaysOn>
  <updateRate>0.0</updateRate>
  <cameraName>rrbot/camera1</cameraName>
  <imageTopicName>/camera/image_raw</imageTopicName>
  <cameraInfoTopicName>/camera/camera_info</cameraInfoTopicName>
  <frameName>link_camera</frameName>
  <hackBaseline>0.07</hackBaseline>
  <distortionK1>0.0</distortionK1>
  <distortionK2>0.0</distortionK2>
  <distortionK3>0.0</distortionK3>
  <distortionT1>0.0</distortionT1>
  <distortionT2>0.0</distortionT2>
</plugin>
</sensor>
</gazebo>

```

- **Plugin IMU:** Cho phép robot lấy vị trí trong mô phỏng



```

<gazebo reference="link_imu">
<sensor name="imu_sensor" type="imu">
  <always_on>true</always_on>
  <update_rate>1000</update_rate>
  <visualize>true</visualize>
  <scale>0.005 0.005 0.005</scale>
  <plugin filename="libgazebo_ros_imu_sensor.so" name="imu_plugin">
    <topicName>/imu</topicName>
    <bodyName>link_imu</bodyName>
    <frameName>link_imu</frameName>
    <gaussianNoise>0.01</gaussianNoise>
    <xyzOffset>0 0 0</xyzOffset>
    <rpyOffset>0 0 0</rpyOffset>
    <initialOrientationAsReference>false</initialOrientationAsReference>
  </plugin>
  <pose>0 0 0 0 0 0</pose>
</sensor>
</gazebo>

```

## - Plugin LiDAR: Cho phép robot quét môi trường 3D

```

<gazebo reference="link_lidar">
  <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>-3.1416</min_angle>
          <max_angle>3.1416</max_angle>
        </horizontal>
      </scan>
      <range>
        <min>0.1</min>
        <max>30.0</max>
        <resolution>0.01</resolution>
      </range>
    </ray>
  </sensor>
</gazebo>

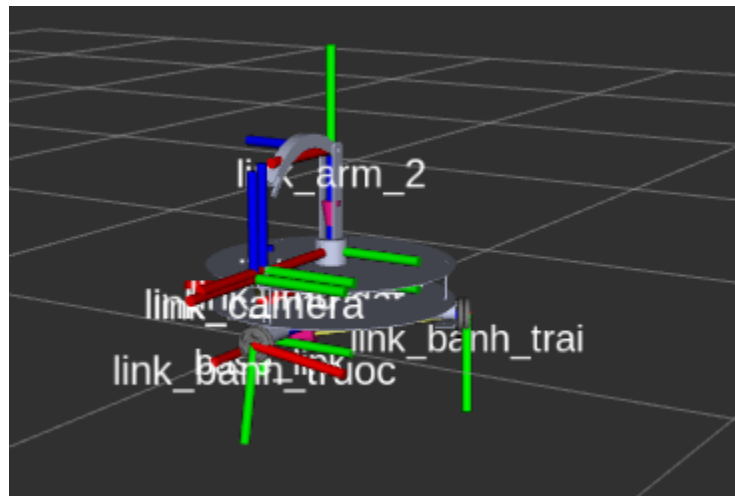
```

```

</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>link_lidar</frameName>
</plugin>
</sensor>
</gazebo>

```

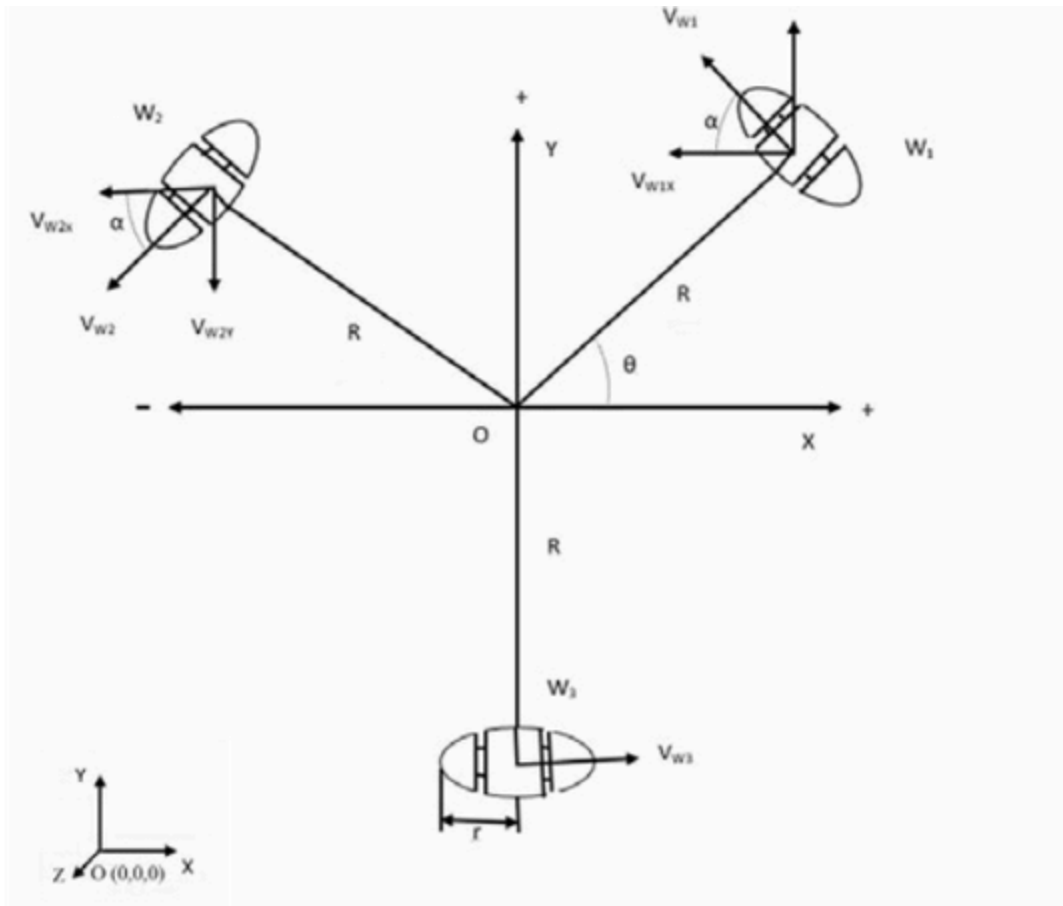
### 3.3 Mô phỏng trên RViz



Mô hình đã được hiển thị thành công trên Rviz, đồng thời TF cũng đã xuất hiện đầy đủ.

## 4. Điều khiển Robot:

### 4.1 Mô hình động học của robot ba bánh Omni



Robot omni 3 bánh sử dụng ba bánh xe đặt cách nhau  $120^\circ$ .  
 Công thức vận tốc được biểu diễn dưới dạng ma trận:

$$\begin{bmatrix} V_{wx} \\ V_{wy} \\ V_0 \end{bmatrix} = \begin{bmatrix} -\cos(\alpha) & -\cos(\alpha) & 1 \\ \sin(\alpha) & -\sin(\alpha) & 0 \\ \frac{1}{R} & \frac{1}{R} & \frac{1}{R} \end{bmatrix} \begin{bmatrix} V_{w1} \\ V_{w2} \\ V_{w3} \end{bmatrix}$$

Trong đó:

$V_{wx}, V_{wy}$  là vận tốc của robot theo trục x và y.

$V_0$  là tốc độ quay quanh tâm.

Vw1,Vw2,Vw3 là vận tốc góc của các bánh xe.  
R là khoảng cách từ tâm robot đến bánh xe.

## 4.2 Điều khiển bánh:

File .yaml định nghĩa các bộ điều khiển cho các khớp của robot

```
joint_state_controller:
  type: "joint_state_controller/JointStateController"
  publish_rate: 50

left_wheel_joint_velocity_controller:
  type: "velocity_controllers/JointVelocityController"
  joint: "joint_banh_trai"
  pid: {p: 100.0, i: 0.1, d: 10.0}

right_wheel_joint_velocity_controller:
  type: "velocity_controllers/JointVelocityController"
  joint: "joint_banh_phai"
  pid: {p: 100.0, i: 0.1, d: 10.0}

front_wheel_joint_velocity_controller:
  type: "velocity_controllers/JointVelocityController"
  joint: "joint_banh_truoc"
  pid: {p: 100.0, i: 0.1, d: 10.0}

arm_1_joint_controller:
  type: "position_controllers/JointPositionController"
  joint: "joint_arm_1"
  pid: {p: 100.0, i: 0.05, d: 1.0}

arm_2_joint_controller:
  type: "position_controllers/JointPositionController"
  joint: "joint_arm_2"
  pid: {p: 100.0, i: 0.05, d: 1.0}
```

Sử dụng plugin `gazebo_ros_control` trong Gazebo kết nối hệ thống điều khiển ROS với mô phỏng Gazebo, cho phép điều khiển robot qua các controller ROS, tạo các topic như `/robot/joint_states` và `/robot/controller_manager`, giúp ROS gửi lệnh điều khiển tới các joint robot. Ngoài ra, plugin này còn hỗ trợ mô phỏng động học, bao gồm lực, vận tốc và vị trí của các khớp khi điều khiển robot.

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

#### - Transmission cho các bánh:

```
<transmission name="left_wheel_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint_banh_trai">

<hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>

  </joint>
  <actuator name="left_wheel_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<transmission name="right_wheel_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint_banh_phai">
```

```

<hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>
  </joint>
  <actuator name="right_wheel_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<transmission name="front_wheel_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint_banh_truoc">

<hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>
  </joint>
  <actuator name="front_wheel_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

```

```

    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

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

```

- Code điều khiển bánh:

Ta sẽ viết file code control.py để điều khiển robot với ba bánh xe omni bằng cách nhận lệnh từ bàn phím hoặc từ topic /cmd\_vel của ROS. Các lệnh di chuyển như tiến, lùi, xoay trái/phải, và dịch chuyển trái/phải được chuyển thành các vận tốc của các bánh xe và gửi đến các controller tương ứng

- Thiết lập ROS node và tạo các publisher để điều khiển ba bánh xe của robot omni-wheel

```
rospy.init_node('omni_wheel_control', anonymous=True)

self.wheel_front_pub =
rospy.Publisher('/front_wheel_joint_velocity_controller/command',
Float64, queue_size=10)

self.wheel_left_pub =
rospy.Publisher('/left_wheel_joint_velocity_controller/command',
Float64, queue_size=10)

self.wheel_right_pub =
rospy.Publisher('/right_wheel_joint_velocity_controller/command',
Float64, queue_size=10)

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

rospy.sleep(1)
```

- Hàm xử lý thông tin từ topic /cmd\_vel và tính toán tốc độ cho ba bánh xe dựa trên các thông số Vx, Vy và Wz.

```
def cmd_vel_callback(self, msg):

    Vx = msg.linear.x

    Vy = msg.linear.y

    Wz = msg.angular.z

    if Wz != 0:

        self.wheel_front_vel = Wz

        self.wheel_left_vel = Wz
```

```

        self.wheel_right_vel = Wz

    elif Vx != 0:

        self.wheel_front_vel = -Wz * self.R

        self.wheel_left_vel = Vx - Wz * self.R

        self.wheel_right_vel = -Vx - Wz * self.R

    elif Vy > 0:

        self.wheel_front_vel = -Vy

        self.wheel_left_vel = 0.0

        self.wheel_right_vel = Vy

    elif Vy < 0:

        self.wheel_front_vel = -Vy

        self.wheel_left_vel = Vy

        self.wheel_right_vel = 0.0

    self.wheel_front_vel = max(min(self.wheel_front_vel,
self.speed_limit), -self.speed_limit)

    self.wheel_left_vel = max(min(self.wheel_left_vel,
self.speed_limit), -self.speed_limit)

    self.wheel_right_vel = max(min(self.wheel_right_vel,
self.speed_limit), -self.speed_limit)

    self.publish_wheel_velocities()

```

- Xuất tốc độ bánh xe lên các publisher.

```

def publish_wheel_velocities(self):

    self.wheel_front_pub.publish(self.wheel_front_vel)

```



```

self.wheel_left_pub.publish(self.wheel_left_vel)

self.wheel_right_pub.publish(self.wheel_right_vel)

rospy.loginfo(f"Front: {self.wheel_front_vel:.2f}, Left: {self.wheel_left_vel:.2f}, Right: {self.wheel_right_vel:.2f}")

```

- Đọc phím nhấn từ bàn phím.

```

def get_key(self):
    fd = sys.stdin.fileno()
    old_settings = termios.tcgetattr(fd)
    try:
        tty.setraw(fd)
        key = sys.stdin.read(1)
    finally:
        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
    return key

```

- Nhận các lệnh điều khiển từ bàn phím và gửi lệnh điều khiển đến các bánh

```

def run(self):
    rospy.loginfo("w: Tiến, s: Lùi, a: Trái, d: Phải, q: Xoay trái, e: Xoay phải, k: Dừng, x: Thoát")

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

        rospy.loginfo(f"Phím đang nhấn: {key}")

        Vx, Vy, Wz = 0.0, 0.0, 0.0

```

```
if key == 'w':  
    Vx = self.speed_limit  
  
elif key == 's':  
    Vx = -self.speed_limit  
  
elif key == 'a':  
    Vy = self.speed_limit  
  
elif key == 'd':  
    Vy = -self.speed_limit  
  
elif key == 'q':  
    Wz = -self.speed_limit  
  
elif key == 'e':  
    Wz = self.speed_limit  
  
elif key == 'k':  
    self.stop_robot()  
    continue  
  
elif key == 'x':  
    self.stop_robot()  
    self.running = False  
    break  
  
twist_msg = Twist()  
  
twist_msg.linear.x = Vx
```

```

        twist_msg.linear.y = Vy

        twist_msg.angular.z = Wz

        self.cmd_vel_callback(twist_msg)

        rospy.sleep(0.1)

    self.cmd_vel_callback(Twist())

```

## 4.3 Điều khiển tay máy:

### Sử dụng các plugin transmission cho tay máy

```

<transmission name="arm_1_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint_arm_1">

<hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>

  </joint>
  <actuator name="arm_1_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<transmission name="arm_2_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint_arm_2">

<hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>

  </joint>
  <actuator name="arm_2_motor">

```

Viết file code arm.py điều khiển tay máy thông qua các phím từ bàn phím, cho phép điều chỉnh góc của hai khớp tay máy

- Khởi tạo một node có tên arm\_teleop\_control và tạo các publisher để gửi lệnh điều khiển tới các khớp của tay máy qua các topic /arm\_1\_joint\_controller/command và /arm\_2\_joint\_controller/command

```
rospy.init_node('arm_teleop_control', anonymous=True)

self.joint_arm_1_pub =
rospy.Publisher('/arm_1_joint_controller/command', Float64, queue_size=10)

self.joint_arm_2_pub =
rospy.Publisher('/arm_2_joint_controller/command', Float64, queue_size=10)
```

- Hàm get\_key đọc phím từ terminal và hàm cmd\_callback xử lý các lệnh điều khiển từ bàn phím, cập nhật giá trị góc của các khớp và gửi các giá trị này đến các publisher.

```
def get_key(self):

    fd = sys.stdin.fileno()

    old_settings = termios.tcgetattr(fd)

    try:

        tty.setraw(fd)

        select.select([sys.stdin], [], [], 0)

        key = sys.stdin.read(1)

    finally:

        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
```

```

        return key

    def cmd_callback(self, key):

        if key == 'w':

            self.joint_arm_1_angle = min(self.joint_arm_1_angle +
self.angle_step, self.angle_limit)

        elif key == 's':

            self.joint_arm_1_angle = max(self.joint_arm_1_angle -
self.angle_step, -self.angle_limit)

        elif key == 'a':

            self.joint_arm_2_angle = min(self.joint_arm_2_angle +
self.angle_step, self.angle_limit)

        elif key == 'd':

            self.joint_arm_2_angle = max(self.joint_arm_2_angle -
self.angle_step, -self.angle_limit)

        elif key == 'x':

            self.running = False

        self.joint_arm_1_pub.publish(self.joint_arm_1_angle)

        self.joint_arm_2_pub.publish(self.joint_arm_2_angle)

        rospy.loginfo(f"Joint Arm 1: {self.joint_arm_1_angle:.2f},
Joint Arm 2: {self.joint_arm_2_angle:.2f}")

```

## 5. Kết quả:



**Link github:** <https://github.com/ToVuKhoiNguyen/omni>

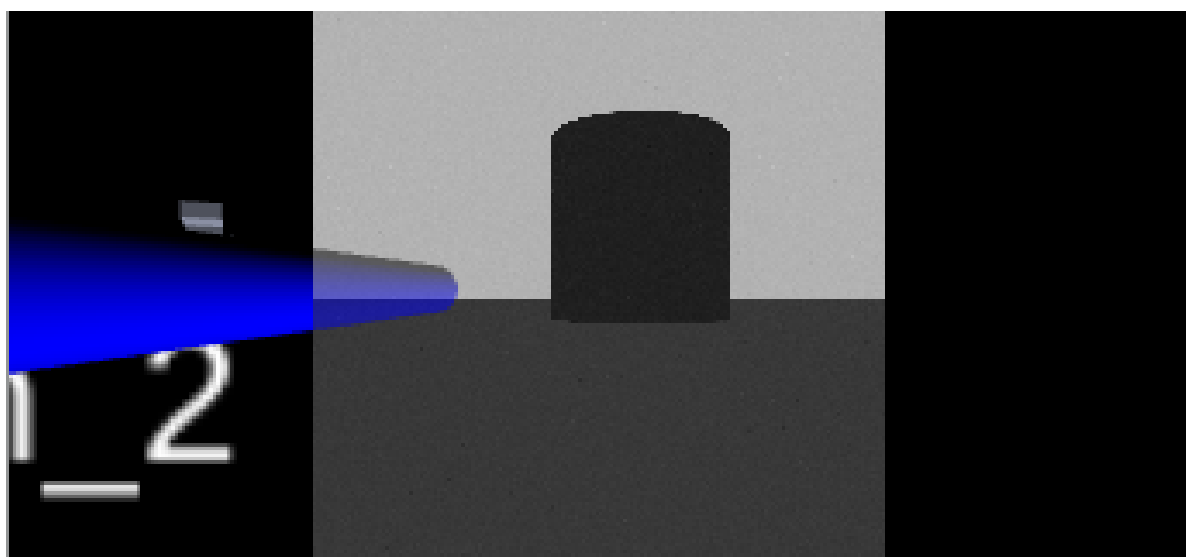
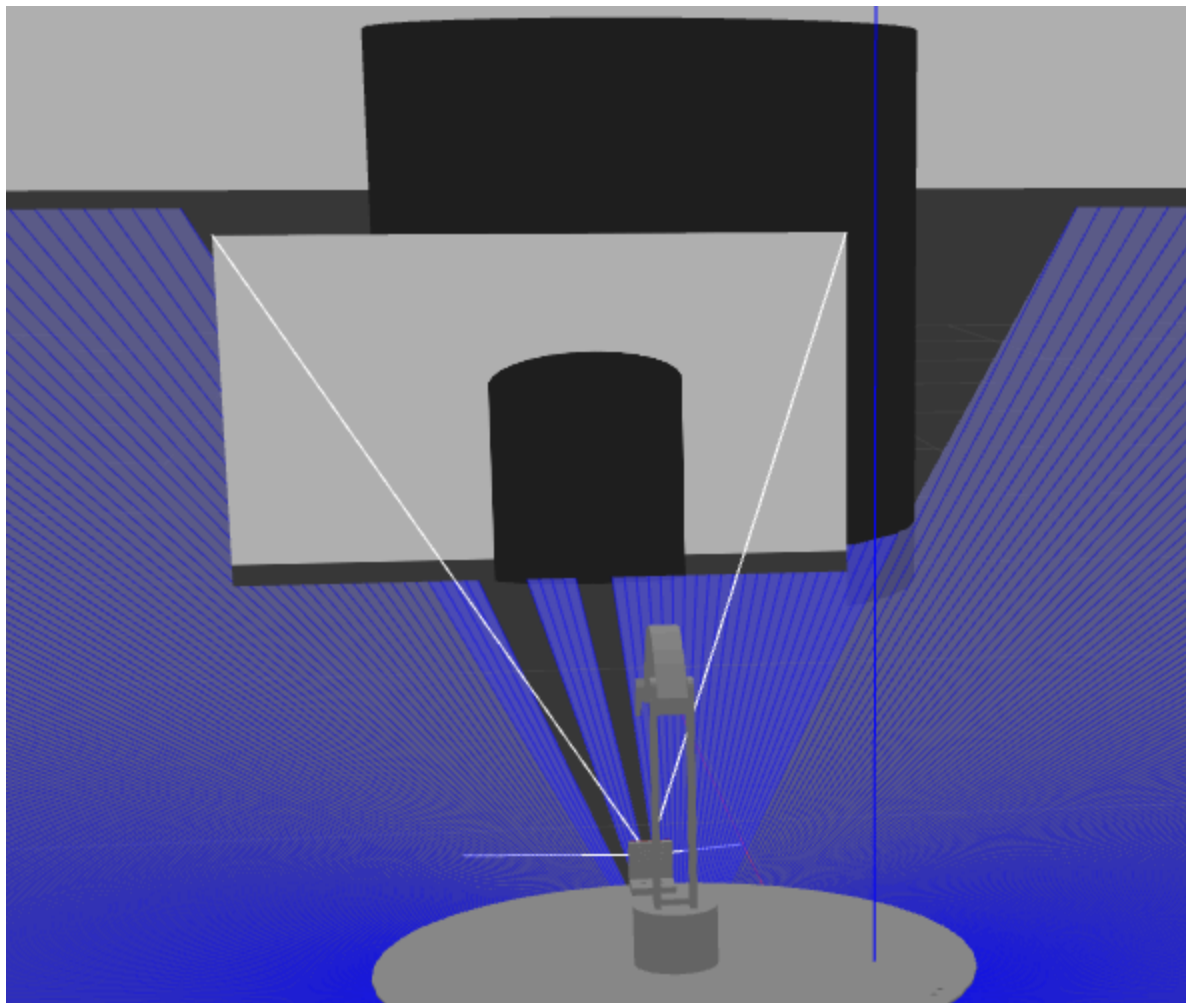
Chạy lệnh `rostopic list` để hiển thị các topic:

```
nguyen@nguyen-Vostro-14-5410:~$ rostopic list
/arm_1_joint_controller/command
/arm_2_joint_controller/command
/camera/camera_info
/camera/image_raw
/camera/image_raw/compressed
/camera/image_raw/compressed/parameter_descriptions
/camera/image_raw/compressed/parameter_updates
/camera/image_raw/compressedDepth
/camera/image_raw/compressedDepth/parameter_descriptions
/camera/image_raw/compressedDepth/parameter_updates
/camera/image_raw/theora
/camera/image_raw/theora/parameter_descriptions
/camera/image_raw/theora/parameter_updates
/clicked_point
/clock
/cmd_vel
/front_wheel_joint_velocity_controller/command
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/performance_metrics
/gazebo/set_link_state
/gazebo/set_model_state
/imu
/initialpose
/joint_states
/left_wheel_joint_velocity_controller/command
/move_base_simple/goal
/right_wheel_joint_velocity_controller/command
/rosout
/rosout_agg
/rrbot/camera1/parameter_descriptions
/rrbot/camera1/parameter_updates
/scan
/tf
/tf_static
```

Ta thấy đã xuất hiện đầy đủ các topic được yêu cầu

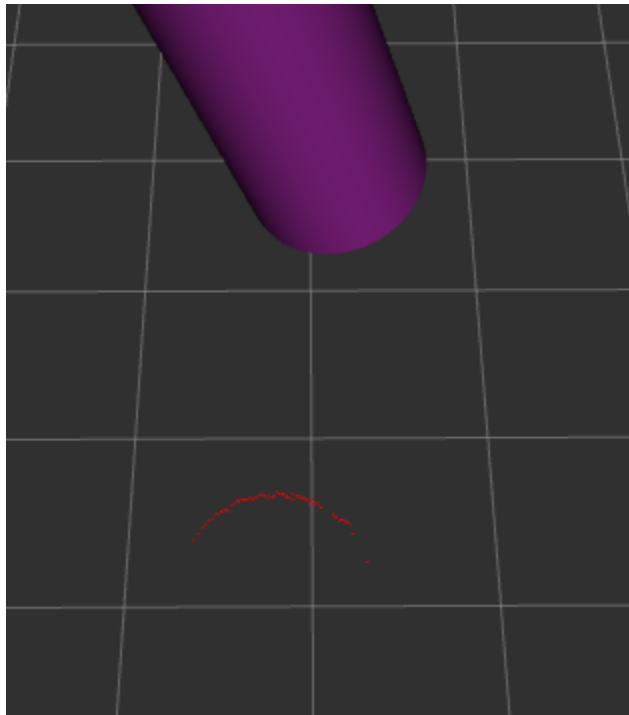
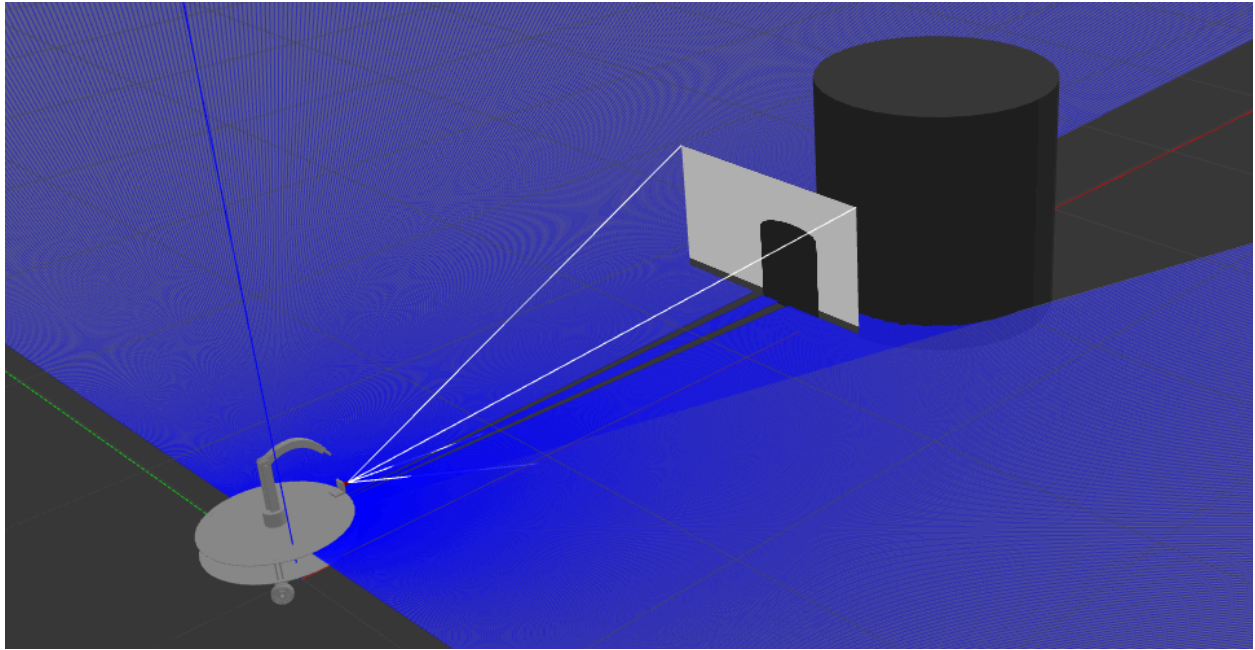
Đã hiện ra các topic trên Rviz

<div> <div></div> <div>Imu</div> </div> <div> <div>✓ Status: Ok</div> <div>Topic</div> <div>Unreliable</div> <div>Queue Size</div> <div>Color</div> <div>Alpha</div> <div>History Length</div> </div>	<div>✓</div> <div>/imu</div> <div><input type="checkbox"/></div> <div>10</div> <div> 204; 51; 204</div> <div>1</div> <div>1</div>
<div> <div></div> <div>Camera</div> </div> <div> <div>✓ Status: Ok</div> <div> <div>👁 Visibility</div> <div>Image Topic</div> <div>Transport Hint</div> <div>Queue Size</div> <div>Unreliable</div> <div>Image Rendering</div> <div>Overlay Alpha</div> <div>Zoom Factor</div> </div> </div>	<div>✓</div> <div>✓</div> <div>/camera/image_raw</div> <div>raw</div> <div>2</div> <div><input type="checkbox"/></div> <div>background and ov...</div> <div>0,5</div> <div>1</div>
<div> <div></div> <div>LaserScan</div> </div> <div> <div>✓ Status: Ok</div> <div> <div>Topic</div> <div>Unreliable</div> <div>Queue Size</div> <div>Selectable</div> <div>Style</div> <div>Size (m)</div> <div>Alpha</div> <div>Decay Time</div> <div>Position Transf...</div> <div>Color Transfor...</div> <div>Channel Name</div> <div>Use rainbow</div> <div>Invert Rainbow</div> <div>Min Color</div> <div>Max Color</div> <div>Autocompute I...</div> </div> </div>	<div>✓</div> <div>/scan</div> <div><input type="checkbox"/></div> <div>10</div> <div>✓</div> <div>Flat Squares</div> <div>0,01</div> <div>1</div> <div>0</div> <div>XYZ</div> <div>Intensity</div> <div>intensity</div> <div>✓</div> <div><input type="checkbox"/></div> <div> 0; 0; 0</div> <div><input type="checkbox"/> 255; 255; 255</div> <div>✓</div>





Ta thấy camera đã quan sát được vật cản ở trong môi trường Gazebo lẫn Rviz



Ta thấy LiDAR đã quét được vật cản trong cả môi trường Gazebo lẫn Rviz. IMU cũng đã xuất hiện trong Rviz

Gọi các lệnh rostopic echo /imu và rostopic echo /scan:

```
orientation_covariance: [0.0001, 0.0, 0.0, 0.0, 0.0001, 0.0, 0.0, 0.0, 0.0001]
angular_velocity:
  x: 0.002163424854699912
  y: -0.003474405631108091
  z: 0.012968632490998097
angular_velocity_covariance: [0.0001, 0.0, 0.0, 0.0, 0.0001, 0.0, 0.0, 0.0, 0.0001]
linear_acceleration:
  x: 1.549875756192258
  y: -2.702132700319549
  z: 11.005428561587033
linear_acceleration_covariance: [0.0001, 0.0, 0.0, 0.0, 0.0001, 0.0, 0.0, 0.0, 0.0001]
---
header:
  seq: 148
  stamp:
    secs: 149
    nsecs: 13000000
  frame_id: "link_imu"
orientation:
  x: 0.013863705975319773
  y: 0.007398946226722373
  z: 0.008958326479559787
  w: 0.9828983825197571
orientation_covariance: [0.0001, 0.0, 0.0, 0.0, 0.0001, 0.0, 0.0, 0.0, 0.0001]
angular_velocity:
  x: 0.028165082243373774
  y: 0.009997741913997696
  z: 0.0003380490920613752
angular_velocity_covariance: [0.0001, 0.0, 0.0, 0.0, 0.0001, 0.0, 0.0, 0.0, 0.0001]
linear_acceleration:
  x: 1.5616346226215116
  y: -2.7120283219961263
  z: 10.990058540142318
linear_acceleration_covariance: [0.0001, 0.0, 0.0, 0.0, 0.0001, 0.0, 0.0, 0.0, 0.0001]
---
header:
  seq: 149
  stamp:
    secs: 150
    nsecs: 14000000
  frame_id: "link_imu"
orientation:
  x: -0.02369442808540491
  y: 0.0051980099594259015
  z: 0.011216578590535983
  w: 1.0050654495584819
orientation_covariance: [0.0001, 0.0, 0.0, 0.0, 0.0001, 0.0, 0.0, 0.0, 0.0001]
angular_velocity:
  x: -0.024105164455998255
```

[illegible]

Khi chạy các file control.py và arm.py, ta nhận thấy robot di chuyển các hướng tiến, lùi, sang trái, sang phải, quay trái, quay phải một cách khá ổn định và cánh tay máy cũng hoạt động mượt mà với khả năng điều khiển các khớp theo chiều tiến lùi trong cả Gazebo và Rviz