BÁO CÁO GIỮA KỲ

Họ và tên: Bùi Quang Dương

Msv: 22027512

Mã môn học: RBE3017

Mục lục

Danh mục ảnh

I. Giới thiệu đề tài:

1. Đề tài:

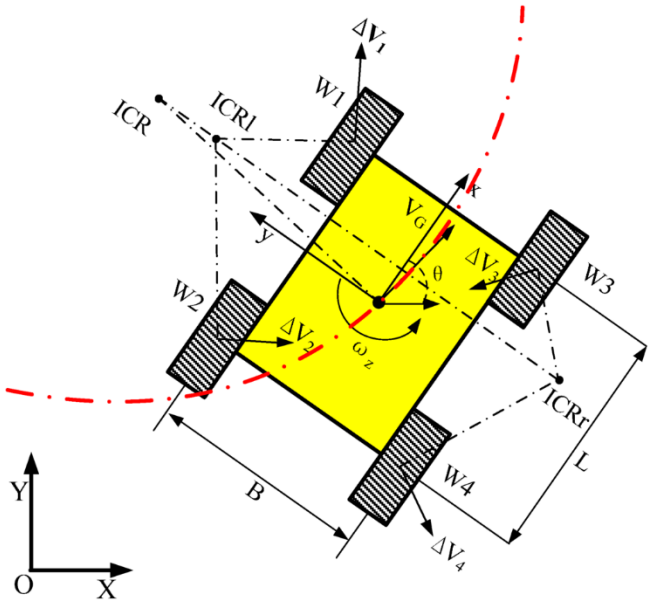
Đề tài giữa kỳ được lựa chọn là điều khiển một robot AGV bằng ROS kèm theo các cảm biến.

Robot di chuyển bằng bánh xích; tay máy hai khớp quay. Ba cảm biến được sử dụng gồm IMU, camera và encoder.

2. Phương án điều khiển:

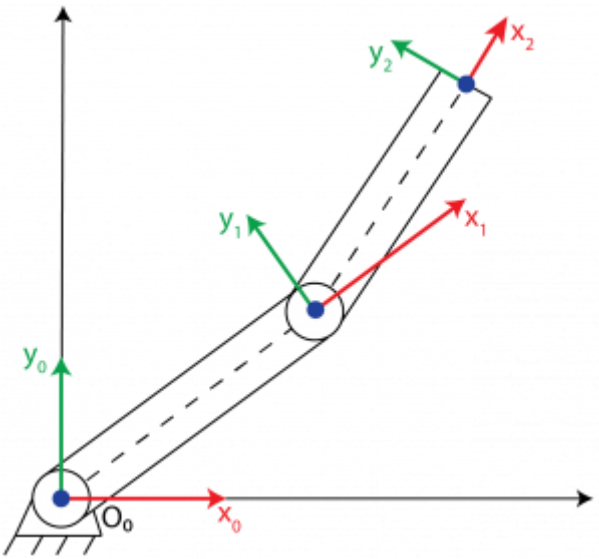
Với yêu cầu robot di chuyển bằng bánh xích, sau khi tham khảo các plugin khác nhau, em quyết định lựa chọn phương thức skid-steering cho mô phỏng robot bánh xích vì các lí do sau:

- Cung cấp mô hình động học phù hợp với bản chất của robot 2 bánh xích, đặc biệt trong việc xử lý các hiện tượng trượt và lực ma sát.
- Đáp ứng tốt các yêu cầu về chính xác chuyển động, đặc biệt trong các tình huống cần quay đột ngột.
- Các tham số của plugin có thể được hiệu chỉnh dễ dàng dựa trên các dữ liệu thực nghiệm, từ đó cải thiện hiệu năng vận hành trong môi trường thực tế.



Hình 1. Minh hoạ động học Skid-Steering.

3. Tay máy:



Hình 2. Mô hình cánh tay robot 2 khớp xoay.

Tay máy 2 bậc tự do có thể điều khiển được.

4. Camera:

Giúp robot quan sát và nhận diện các đối tượng. Có thể phát triển thêm sử lí ảnh trong các nghiên cứu sau.

5. IMU:

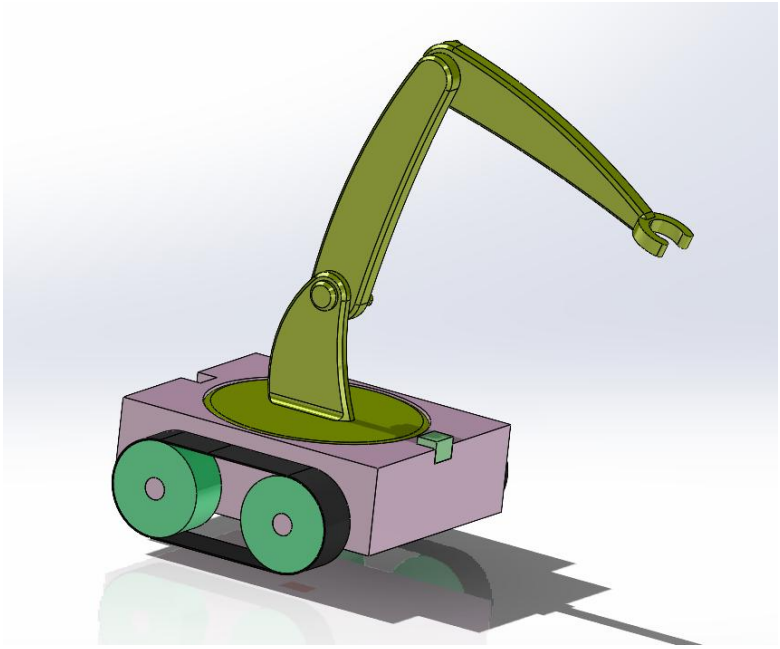
Cảm biến đo gia tốc và vận tốc góc. Giúp robot xác định được trạng thái chuyển động.

6. Encoder:

Gắn vào các cơ cấu truyền động ở trong nghiên cứu này cụ thể là động cơ để đo tốc độ quay và quang đường đã đi.

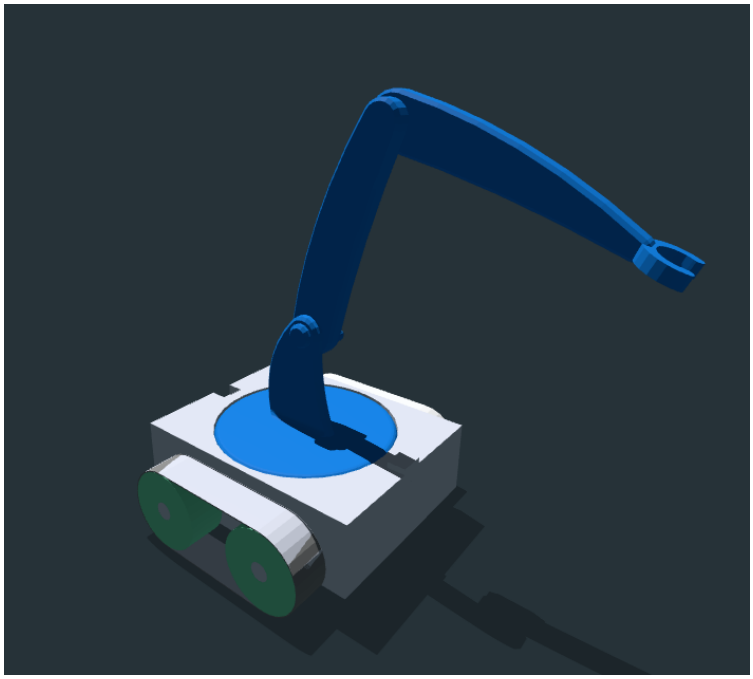
II. Quá trình thực hiện:

1. Thiết kế mô hình 3D:



Hình 3. Mô hình tổng thể của robot.

Sau khi đã có được mô hình tổng thể robot, tiến hành đặt trục toạ độ và chuyển sang URDF.



Hình 4. Mô hình sau khi được chuyển sang URDF

Cần bổ sung 1 link là dummy_root_link để sửa lỗi việc gazebo định nghĩa base_link không có mô-men quán tính.

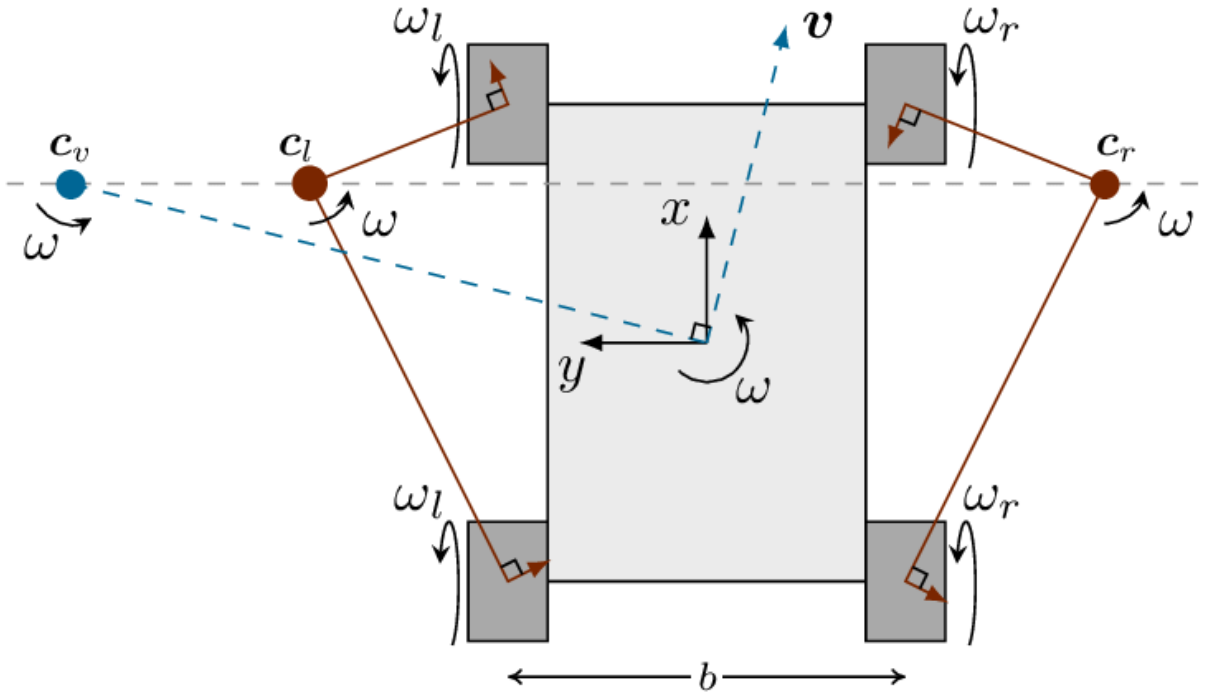
```
<link name="dummy_root_link"/>
 <joint name="dummy_joint" type="fixed">
    <parent link="dummy_root_link"/>
```

```
    <child link="base_link"/>
    <origin xyz="0 0 0" rpy="0 0 0"/>
  </joint>
```

**Lưu ý: vì cấu trúc của file URDF theo kiểu parent – child không phù hợp với bản chất của bánh xích có dạng chuỗi nối nhau, nên từ đây trở đi, mô phỏng chỉ quan tâm chuyển động của xe được gây ra bởi 4 bánh xe và bỏ qua các đại lượng vật lí của xích.*

2. Thiết kế chương trình điều khiển:

2.1. Điều khiển bánh xe:



Hình 5. Động học của robot sử dụng skid-steering.

v : Tốc độ tuyến tính, di chuyển thẳng về phía trước.

ω : Tốc độ quay, phản ánh khả năng xoay tại chỗ hoặc di chuyển cong.

c_v, c_l ,c_r :điểm quay tổng thể, trái, phải khi xe quay.

- Vận tốc tuyến tính: v=r·(ωr+ωl)/2, với r là bán kính bánh xe, ωr và ωl là vận tốc góc của bánh xe bên phải và bên trái.
- Vận tốc góc: ω=r·(ωr−ωl)/d, với d là khoảng cách giữa hai bánh xe (khoảng cách bánh xe).
- Tốc độ bánh xe được tính như: Tốc độ bánh xe bên phải = v+ω·d/2, tốc độ bánh xe bên trái = v−ω·d/2.

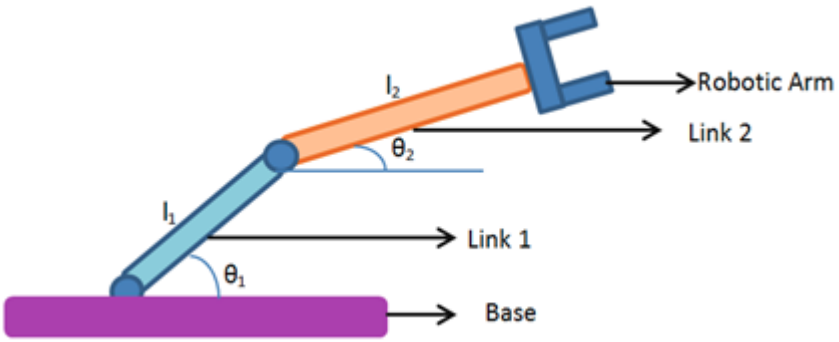
Dựa trên lí thuyết trên, ta sử dụng plugin để điều khiển xe:

```
  <gazebo>
    <plugin name="skid_steer_drive" filename="libga-
zebo_ros_skid_steer_drive.so">
      <robotNamespace>/</robotNamespace>
      <commandTopic>cmd_vel</commandTopic>
      <odometryTopic>odom</odometryTopic>
      <odometryFrame>odom</odometryFrame>
      <robotBaseFrame>base_link</robotBaseFrame>
```

```
      <torque>10.0</torque>
      <covariance_x>0.001</covariance_x>
      <covariance_y>0.001</covariance_y>
      <covariance_yaw>0.01</covariance_yaw>

      <leftFrontJoint>wheel3_joint</leftFrontJoint>
      <rightFrontJoint>wheel2_joint</rightFrontJoint>
      <leftRearJoint>wheel4_joint</leftRearJoint>
      <rightRearJoint>wheel1_joint</rightRearJoint>
      <wheelSeparation>0.19</wheelSeparation>
      <wheelDiameter>0.065</wheelDiameter>

      <broadcastTF>1</broadcastTF>
      <updateRate>100.0</updateRate>
    </plugin>
  </gazebo>
```

Đoạn mã trên đã được khai báo các giá trị phù hợp với mô hình xe.

2.2. Điều khiển tay máy:



Hình 6. Mô hình điều khiển 2 khớp quay.

Trong chương trình này chỉ điều khiển tay máy đơn thuần bằng việc tăng thêm hoặc giảm đi giá trị của 2 góc θ.

Ví dụ khi điều khiển Link 1:

```
elif key == 'r' or key == 'R':
                arm1_pos = constrain(arm1_pos + ARM_STEP, -
arm_limit, arm_limit)
                print(f"Arm1: {arm1_pos:.2f} rad")
            elif key == 'f' or key == 'F':
                arm1_pos = constrain(arm1_pos - ARM_STEP, -
arm_limit, arm_limit)
                print(f"Arm1: {arm1_pos:.2f} rad")
```

2.3. Các cảm biến:

Các cảm biến được sử dụng là: camera, imu và encoder.

Trong đó:

- Camera: sử dụng 1 plugin

```
<!-- plugin cho camera -->
  <gazebo reference="camera">
    <sensor name="camera_sensor" type="camera">
      <always_on>1</always_on>
      <update_rate>30</update_rate>
      <camera>
        <horizontal_fov>1.0472</horizontal_fov>
        <image_width>640</image_width>
        <image_height>480</image_height>
        <near>0.1</near>
        <far>100</far>
      </camera>
      <plugin name="camera_plugin" filename="libgazebo_ros_cam-
era.so">
        <topicName>/camera/image_raw</topicName>
        <cameraName>camera</cameraName>
        <frameName>camera</frameName>
        <camera_info_topic_name>/camera/camera_info</cam-
era_info_topic_name>
        <rotation>1.5708</rotation> <!-- Xoay 90 độ (1.5708 ra-
dian) -->
      </plugin>
    </sensor>
  </gazebo>
```

Như vậy, dữ liệu ảnh thô sẽ được xuất ra và được RVIZ tiếp nhận rồi hiển thị.

- IMU: tương tự camera, chương trình cũng sử dụng 1 plugin cho IMU.

```
  <!-- plugin cho imu -->
  <gazebo reference="imu">
    <gravity>true</gravity>
    <sensor name="imu_sensor" type="imu">
      <always_on>true</always_on>
      <update_rate>100</update_rate>
      <visualize>true</visualize>
      <topic>__default_topic__</topic>
      <plugin filename="libgazebo_ros_imu_sensor.so"
name="imu_plugin">
        <topicName>imu</topicName>
        <bodyName>imu_link</bodyName>
```

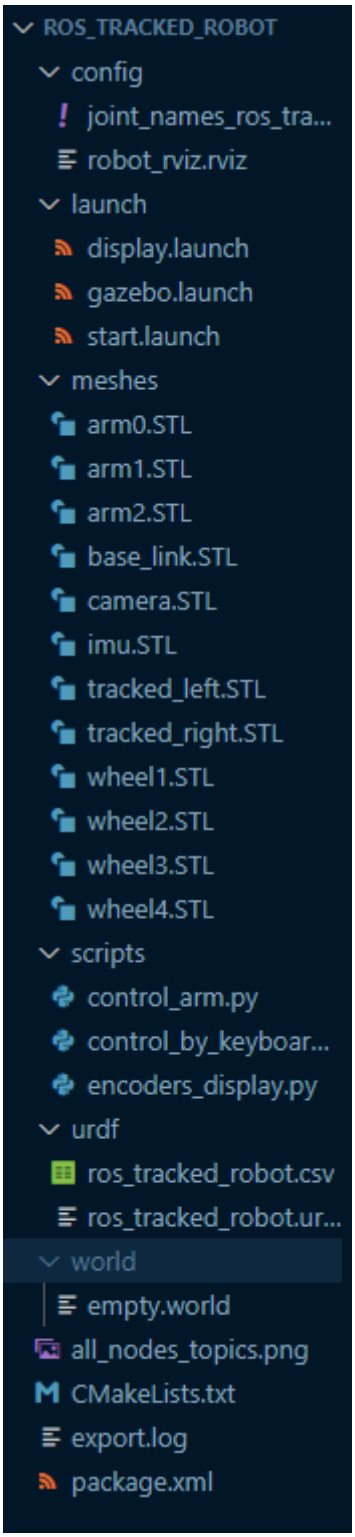
```
        <updateRateHZ>10.0</updateRateHZ>
        <gaussianNoise>0.0</gaussianNoise>
        <xyzOffset>0 0 0</xyzOffset>
        <rpyOffset>0 0 0</rpyOffset>
        <frameName>imu</frameName>
        <initialOrientationAsReference>false</initialOrienta-
tionAsReference>
      </plugin>
      <pose>0 0 0 0 0 0</pose>
    </sensor>
  </gazebo>
```

- Encoder:

Đối với encoder, sử dụng trực tiếp topic /joint_states để đọc giá trị vận tốc và vị trí từ các bánh xe và khớp xoay.

```
def joint_state_callback(msg):
    # Danh sách các khớp cần hiển thị, khớp với tên trong URDF
    joints_to_display = ['arm_1_joint', 'arm_2_joint',
'wheel1_joint', 'wheel2_joint', 'wheel3_joint', 'wheel4_joint']

    # Duyệt qua danh sách khớp từ thông điệp JointState
    for i, joint_name in enumerate(msg.name):
        if joint_name in joints_to_display:
            position = msg.position[i]
            # Kiểm tra xem velocity có tồn tại không, nếu không
thì gán 0.0
            velocity = msg.velocity[i] if i < len(msg.velocity)
else 0.0
            rospy.loginfo("%s: Position = %.3f rad, Velocity =
%.3f rad/s", joint_name, position, velocity)

def encoder_display():
    # Khởi tạo node ROS
    rospy.init_node('encoder_display', anonymous=True)

    # Đăng ký subscriber để đọc topic /joint_states
    rospy.Subscriber('/joint_states', JointState,
joint_state_callback)

    # Thông báo node đang chạy và giữ nó hoạt động
    rospy.loginfo("Hiển thị giá trị encoder của các khớp...")
    rospy.spin()
```

2.4. Cấu trúc tổng thể của package:

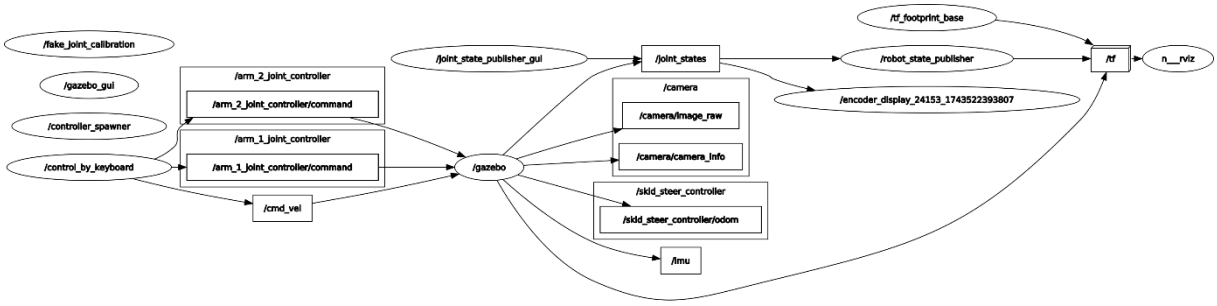


Hình 7. Cấu trúc của package mô phỏng và điều khiển robot.

Cấu trúc thư mục:

- config/: Chứa các file cấu hình như thông số PID, cấu hình hiển thị của RVIZ.
- launch/: Chứa các file .launch để khởi chạy mô phỏng, RViz, Gazebo, hoặc node điều khiển.
- meshes/: Chứa các file mô hình 3D (ở dạng .stl) của robot.

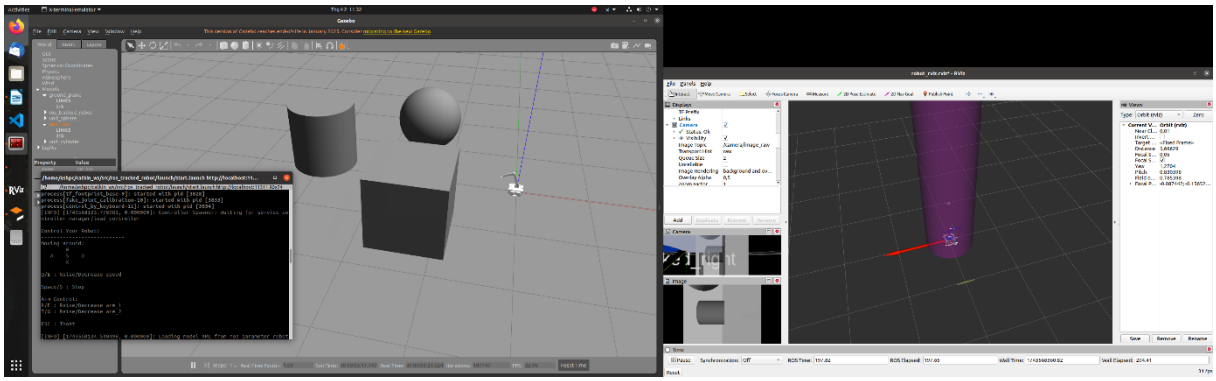
- scripts/: Chứa các chương trình Python, có thể dùng để điều khiển hoặc test robot.
- urdf/: Chứa file mô tả robot URDF/Xacro.
- world/: Chứa thế giới gazebo được sử dụng để mô phỏng.
- CMakeLists.txt & package.xml: Các file cần thiết để build package trong ROS.



Hình 8. Sơ đồ biểu diễn tất cả các node và topic của chương trình.

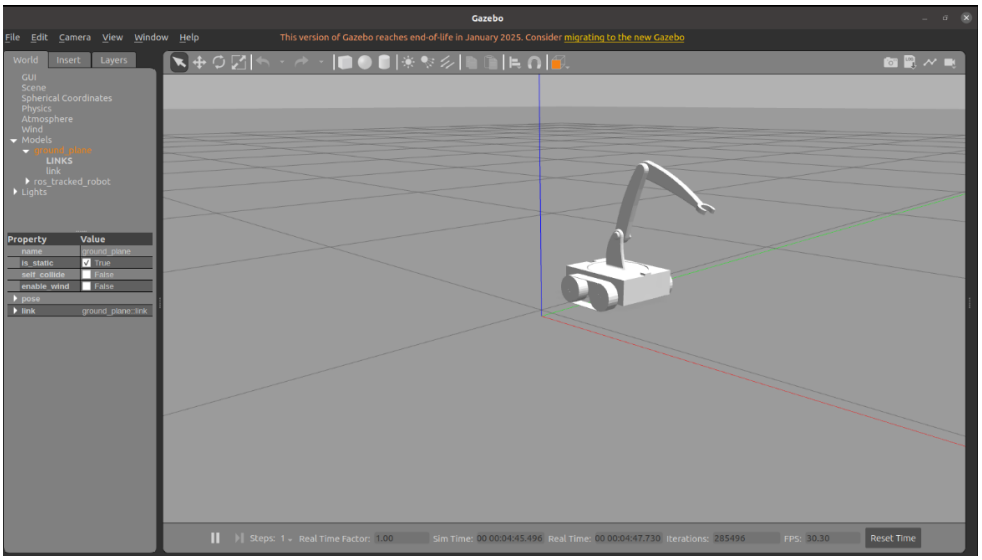
III. Kết quả thu được:

1. Khởi chạy chương trình:

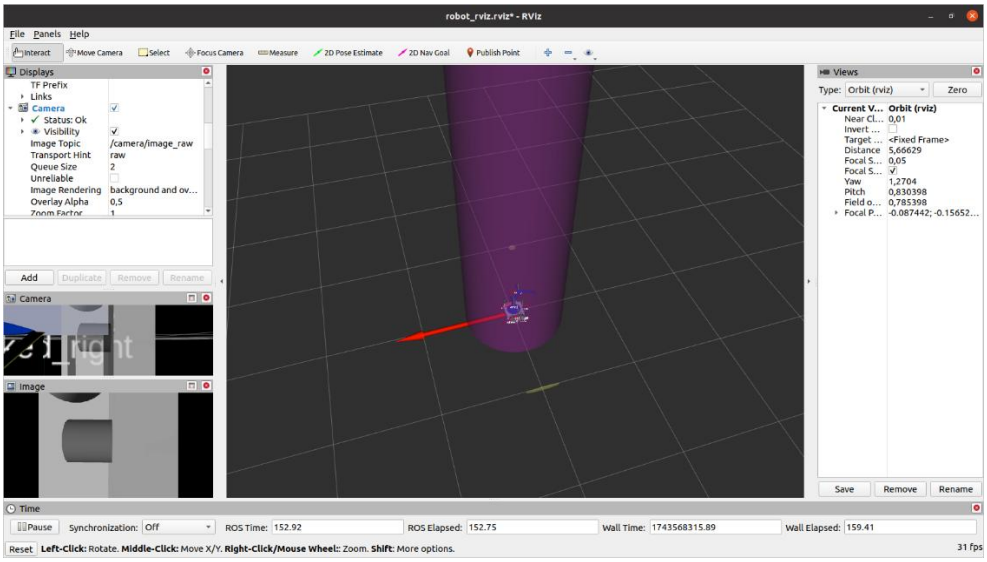


Hình 9. Khởi chạy chương trình.

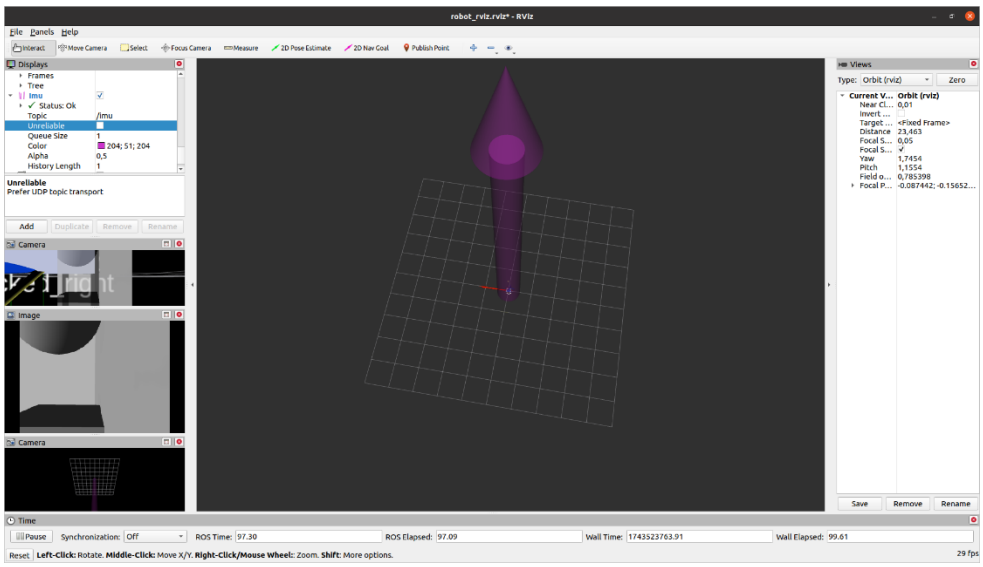
2. Hiển thị trong mô phỏng:



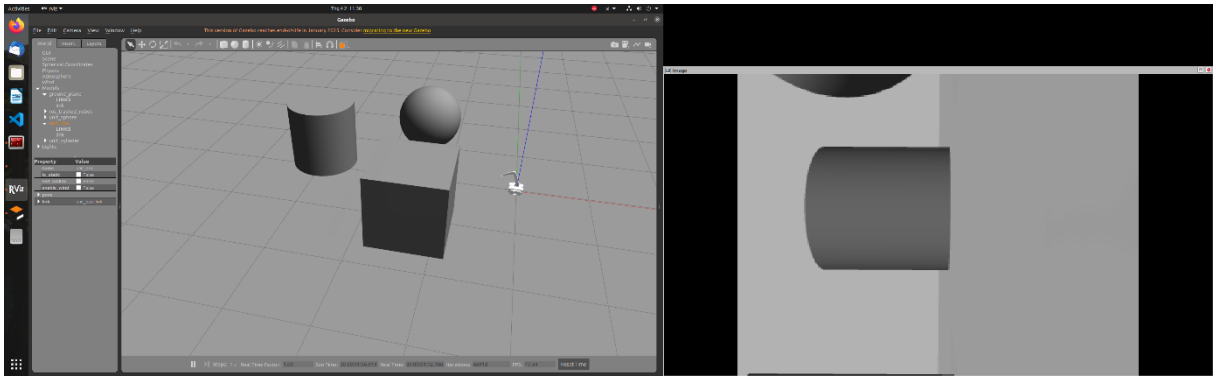
Hình 10. Robot hiển thị trong Gazebo.



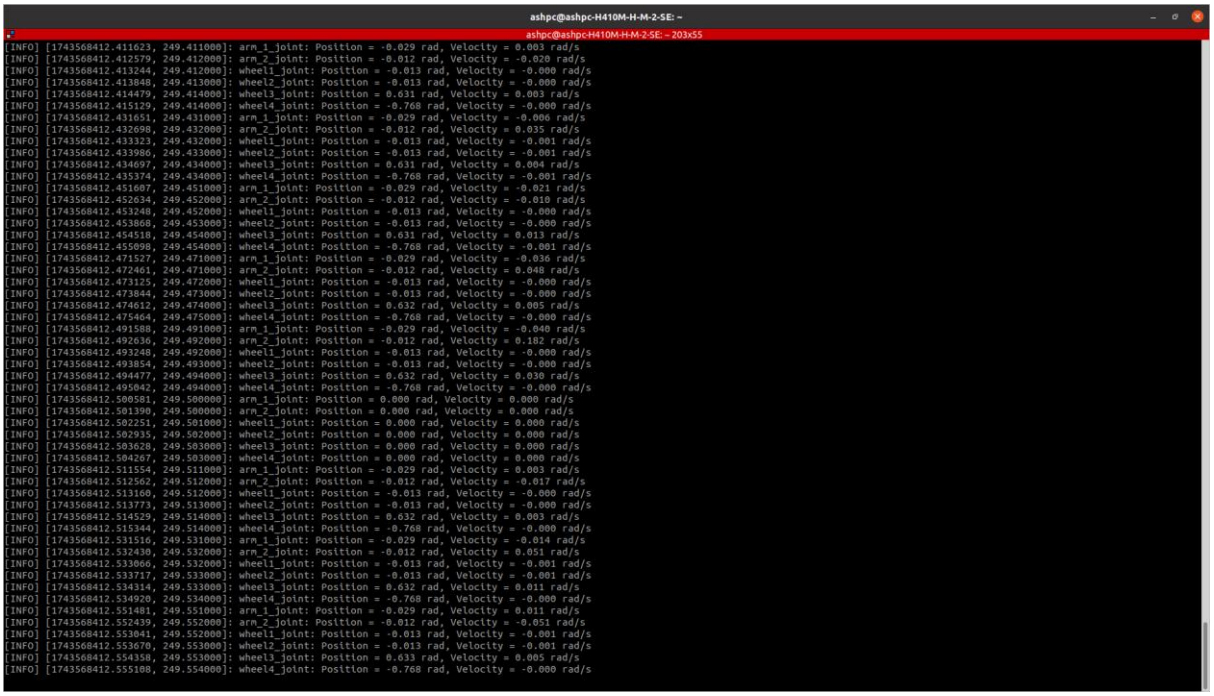
Hình 11. Robot hiển thị trong RVIZ.



Hình 12.Topic IMU.



Hình 13. Hình ảnh thu được từ topic Camera (phải) và hình ảnh thực tế (trái)



Hình 14. Giá trị trả về bởi encoder.

IV. Kết luận:

1. Đã thực hiện:

- Thiết kế robot.
- Mô phỏng trên Gazebo và RVIZ.
- Điều khiển được robot di chuyển các bánh và tay máy.
- Thu được kết quả từ các cảm biến.

2. Công việc cần hoàn thiện:

- Điều khiển di chuyển bánh xe của robot chính xác.
- Cải tiến cách di chuyển của các khớp hay bánh xe để robot không bị khưng hay giật khi di chuyển.

V. Tài liệu tham khảo:

1.https://www.researchgate.net/publication/371500870_MO_PHONG_XE_TU_HANH_HOAT_DONG_DUOC_TREN_MOI_DIA_HINH

2. https://webthesis.biblio.polito.it/14638/1/tesi.pdf

3.https://www.researchgate.net/publication/350956682_Turning_dynamics_analysis_of_a_heavy_articulated_vehicle_by_the_vehicle_planar_dynamic_model_with_two_steering_input_signals

4. https://media.neliti.com/media/publications/450177-designing-adaptive-controllesr-for-robot-69417fdf.pdf

5. https://youtu.be/FuJuLnJy93g?si=WWwbNA09RatP6hpK