TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

ĐẠI HỌC QUỐC GIA HÀ NỘI

BÁO CÁO GIỮA KỲ MÔN

LẬP TRÌNH ROBOT VỚI ROS

ĐỀ TÀI: XE BÁNH XÍCH, 2 KHỚP ROTATION, CÁC CẢM BIẾN CAMERA, LIDAR, GPS

Họ và tên: Lê Hoàng Thanh Phương

Lớp: K67E-RE

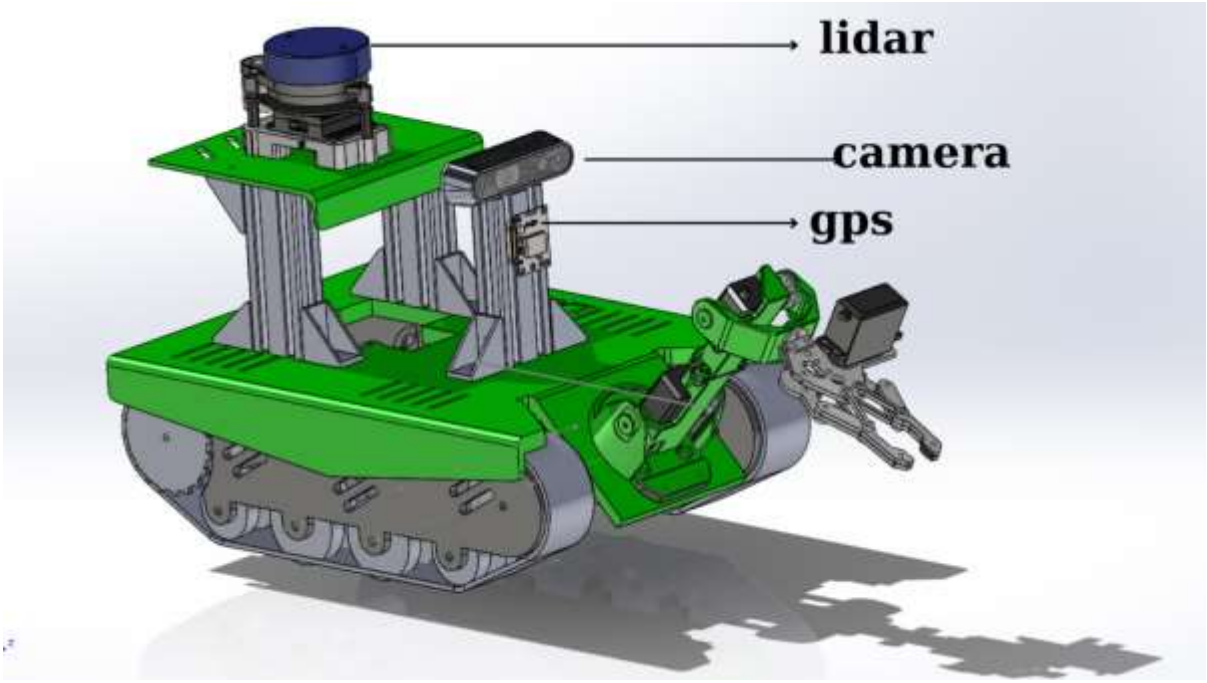
Mã số sinh viên: 22027526

Mục lục

1. Thông tin bài báo cáo

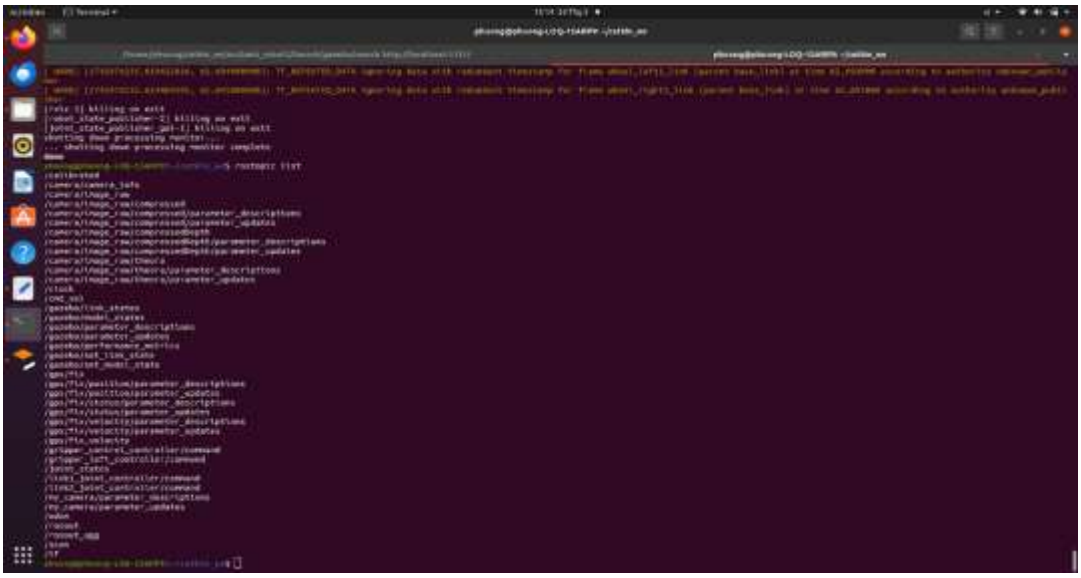
1.1. Thông tin chung

- Mô phỏng robot có cơ cấu chuyển động hệ bánh xích, tích hợp tay máy 2 bậc tự do với 2 trục xoay linh hoạt. Áp dụng mô phỏng và kết hợp giữa các loại cảm biến cụ thể: cảm biến Lidar, GPS, Camera.



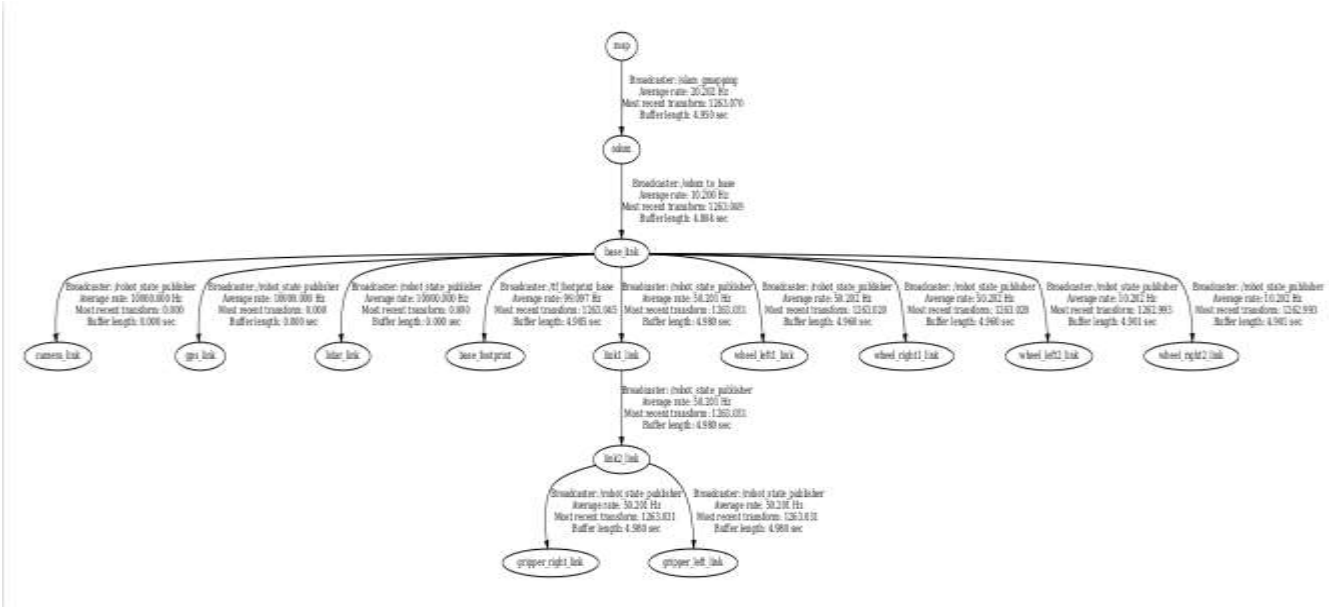
(Bản thiết kế solidworks)

- Để hệ thống mô phỏng hoạt động như mong muốn và có thể kết hợp giữa các loại cảm biến và phần cơ cấu chấp hành cần add plugin của các loại cảm biến và điều khiển vào trong file URDF để hệ thống có thể hoạt động như mong muốn. Mô hình trên để có thể hoạt động được thì cần có 1 số topic sau: < rostopic list >



(ROSTOPIC LIST)

- Thành phần của dự án, (liên kết các link)



- Trong Gazebo, file **URDF (Unified Robot Description Format)** được dùng để mô tả hình dạng, cấu trúc và các thuộc tính vật lý của robot. Nó xác định các **link** (khớp cứng) và **joint** (khớp nối) của robot. Khi sử dụng với Gazebo, URDF có thể mở rộng bằng các **plugin của Gazebo** để mô phỏng vật lý, cảm biến, và điều khiển.

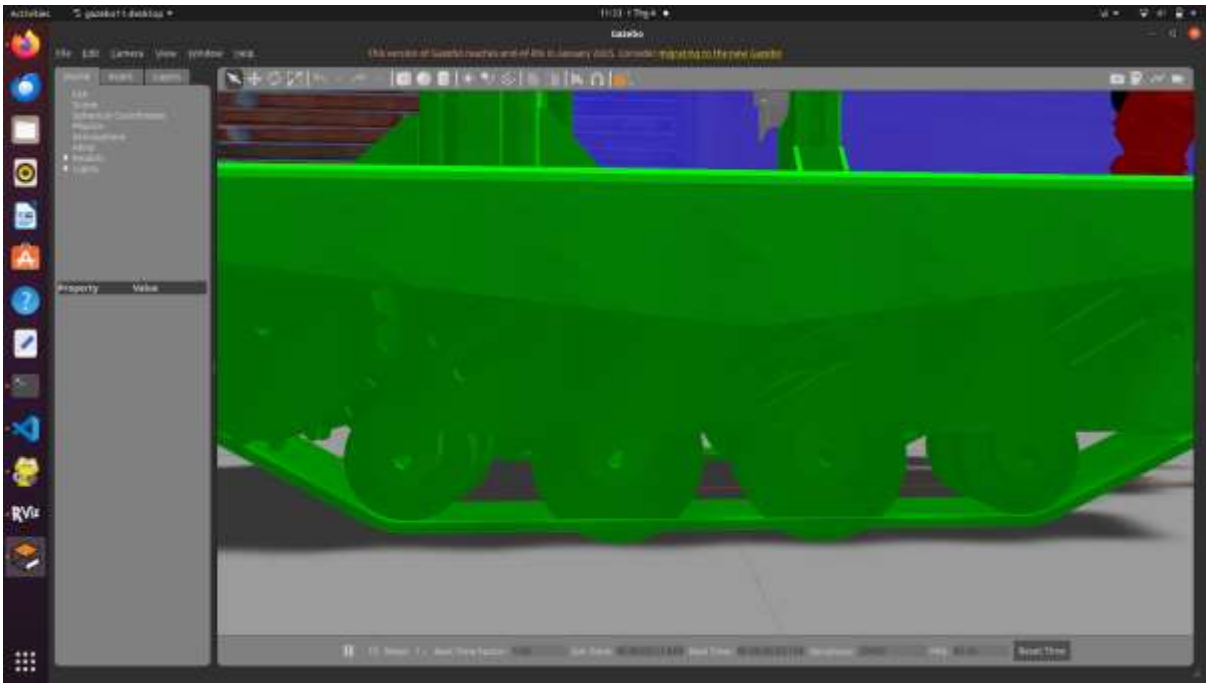
1.2. Động học của hệ bánh xích.

1.2.1. Cơ chế hoạt động của bánh xích.

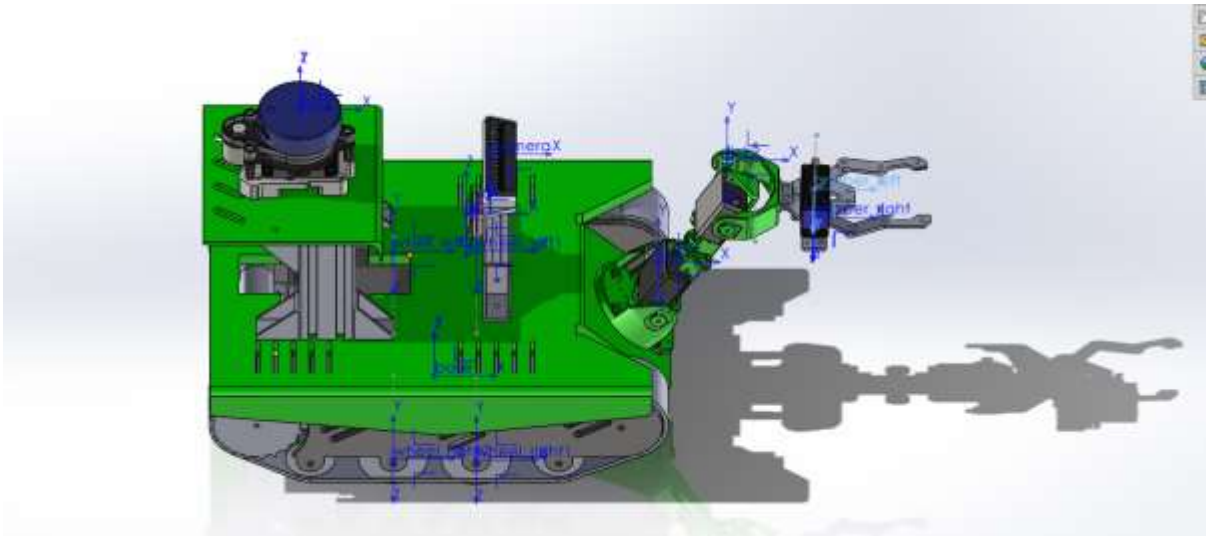
- **Chuyển động bám dính:** Không có sự trượt giữa bánh xích và mặt đất ở điều kiện lý tưởng.
- **Cơ chế quay:** Khi hai bên bánh xích quay với tốc độ khác nhau, phương tiện có thể xoay tại chỗ hoặc theo một đường cong.
- **Tiếp xúc theo đường thẳng dài:**Ổn định hơn trên các bề mặt mềm hoặc địa hình không bằng phẳng.

1.2.2. Mô phỏng.

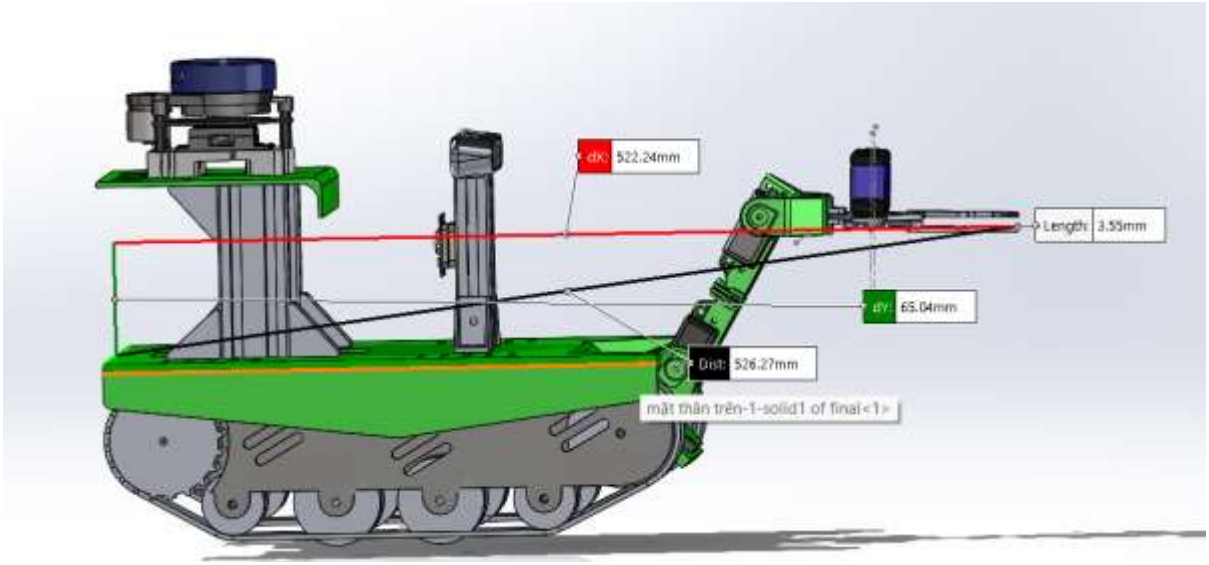
- Để đơn giản hóa quá trình mô phỏng bánh xích ta mô phỏng bánh xích truyền động với cơ chế hoạt động của các bánh giống với bánh xích với 4 bánh xích bị truyền động và tốc độ của 2 bánh mỗi bên là luôn bằng nhau.



1.3. Kích thước Robot



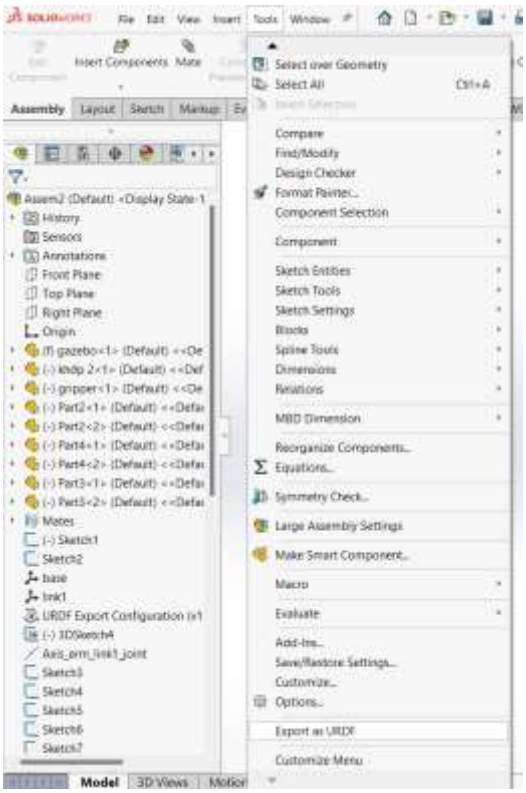
(Ví dụ gắn trục cho các khớp của các link)



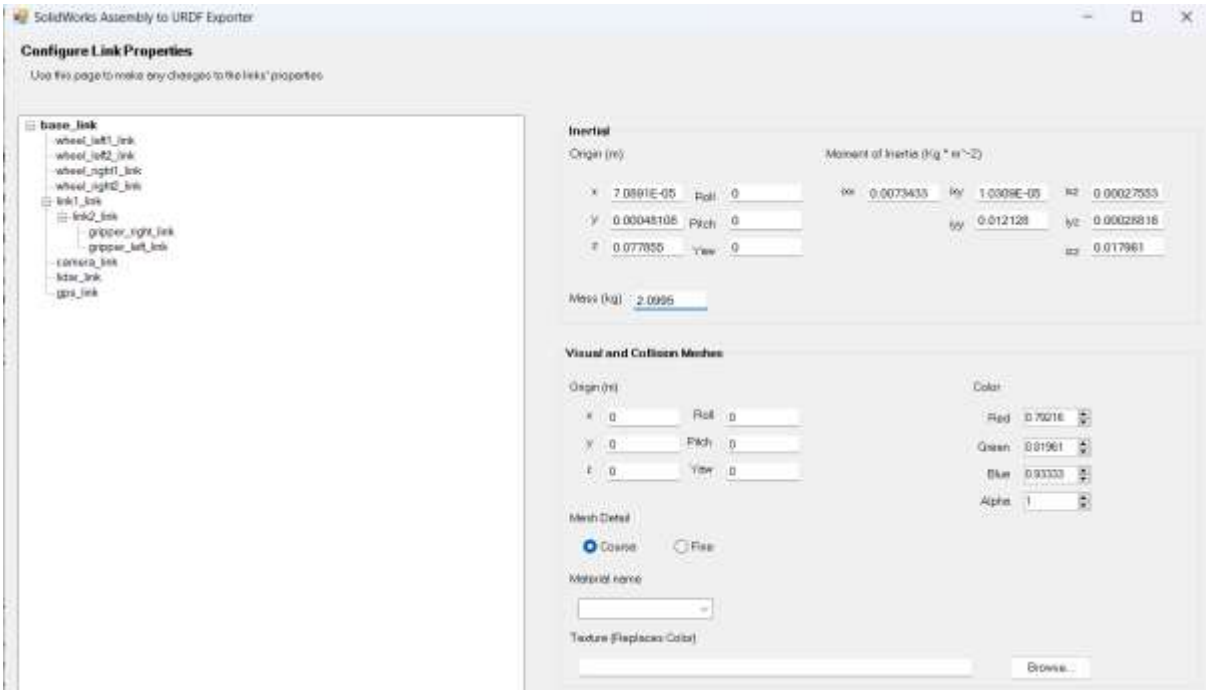
- Sau khi đã có các trục của các khớp xoay, tịnh tiến và các thành phần sử dụng export as URDF với setting cha, con như sau:

2. Thiết lập URDF cho mô hình mô phỏng robot.

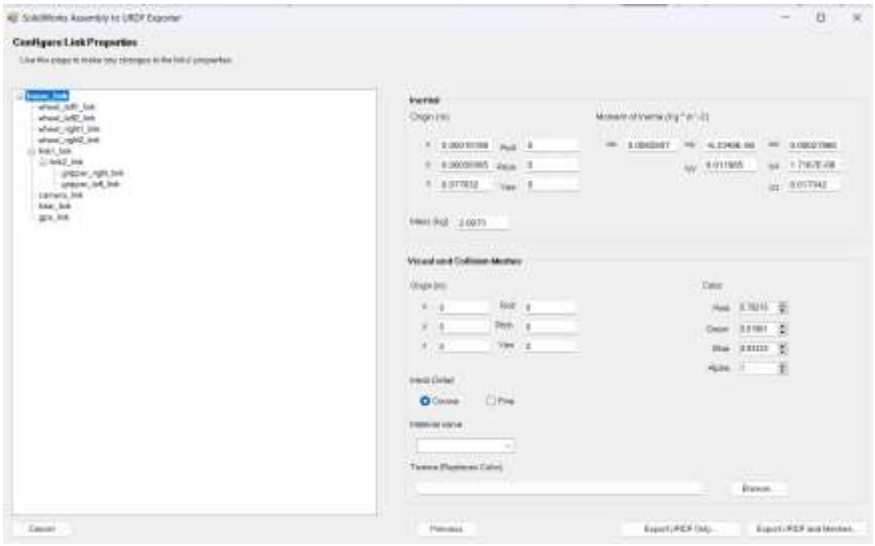
- Sử dụng tool cài đặt trên solidworks để export sang urdf:



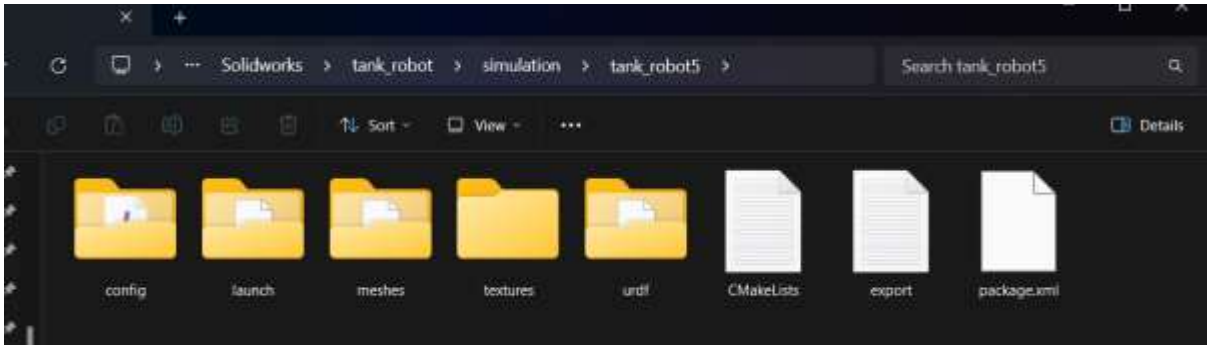
[Link cài đặt tool]



– Để sử dụng được tool cần đặt các trục xyz để định nghĩa các trục xoay (bánh xe, khớp tay máy), tịnh tiến, định nghĩa các loại cảm biến và các thành phần muốn chuyển sang urdf.



– Sau khi setting xong tool sẽ tự động chuyển sang file URDF và có thể sử dụng mô phỏng, file được tạo chứa các thành phần:



– Bây giờ có thể sử dụng mô hình tĩnh với display.launch và gazebo.launch.

- **Cấu trúc tổng quát của một mô hình robot trong URDF**
- **Link (Bộ phận của robot)**

– Mỗi link đại diện cho một phần của robot, như thân chính (base_link), bánh xe (wheel_link), hoặc cảm biến (lidar_link).
– Link có thể có hình dạng như hình hộp, hình trụ, hoặc hình cầu.
– Mỗi link có thể có thuộc tính vật lý như khối lượng, quán tính, và ma sát.

- **Joint (Khớp nối)**

  – Joint kết nối hai link với nhau.
  – Có nhiều loại khớp nối:
    - **Cố định (fixed)**: Hai link không thể di chuyển tương đối với nhau.

        - **Quay (revolute)**: Một link có thể quay quanh một trục (như bánh xe quay quanh trục).
        - **Liên tục (continuous)**: Giống revolute nhưng không có giới hạn góc quay (ví dụ như bánh xe robot di chuyển không giới hạn).
        - **Trượt (prismatic)**: Một link có thể di chuyển tịnh tiến theo một hướng (như piston).
    - Mỗi joint có thể có **giới hạn góc quay/tịnh tiến, lực tác động, và vận tốc tối đa**.
    - **Tích hợp với Gazebo**
        - Gazebo mở rộng URDF bằng cách thêm **plugin** giúp mô phỏng vật lý, điều khiển và cảm biến.
        - Các link có thể được gắn cảm biến như **LiDAR, camera, IMU** để thu thập dữ liệu.
        - Có thể định nghĩa **động lực học** cho robot như trọng lượng, lực kéo, và mô-men xoắn.

3. Cơ chế điều khiển trên gazebo

3.1. Mô hình động học và động lực học

- Robot trong Gazebo được mô tả bằng **URDF/Xacro**, trong đó định nghĩa các liên kết (**link**) và khớp (**joint**).
- Nếu muốn mô phỏng vật lý, ta sử dụng mô hình động lực học (thường dùng plugin **gazebo_ros_control** để liên kết với ROS).

3.2. Hệ thống plugin điều khiển

Các plugin được sử dụng để điều khiển robot, chẳng hạn như:

- **gazebo_ros_control**: Cho phép sử dụng ROS để gửi lệnh điều khiển robot.
- **diff_drive_controller**: Điều khiển robot di chuyển bằng hệ thống bánh xe vi sai.
- **joint_trajectory_controller**: Điều khiển cánh tay robot theo quỹ đạo.
- **gazebo_ros_force**: Áp dụng lực trực tiếp lên robot.

3.3. ROS Control và Controller Manager

- **Controller Manager**: Quản lý các bộ điều khiển của robot (ví dụ: PID cho các động cơ).
- **Các bộ điều khiển phổ biến:**
    - velocity_controllers/JointVelocityController: Điều khiển tốc độ của các khớp.
    - effort_controllers/JointEffortController: Điều khiển lực tác động lên khớp.

- position_controllers/JointPositionController: Điều khiển vị trí của các khớp.

3.4. Giao tiếp điều khiển

- Robot có thể nhận lệnh từ ROS thông qua các topic:
  - /cmd_vel: Điều khiển vận tốc tuyến tính và góc của robot.
  - /joint_states: Theo dõi trạng thái của các khớp.
  - /tf: Cung cấp thông tin vị trí và hướng của robot.
- Các node quan trọng trong ROS:
  - **teleop_twist_keyboard**: Dùng bàn phím để điều khiển robot.
  - **move_base**: Dùng trong điều hướng tự động.

4. Mô phỏng các loại cảm biến

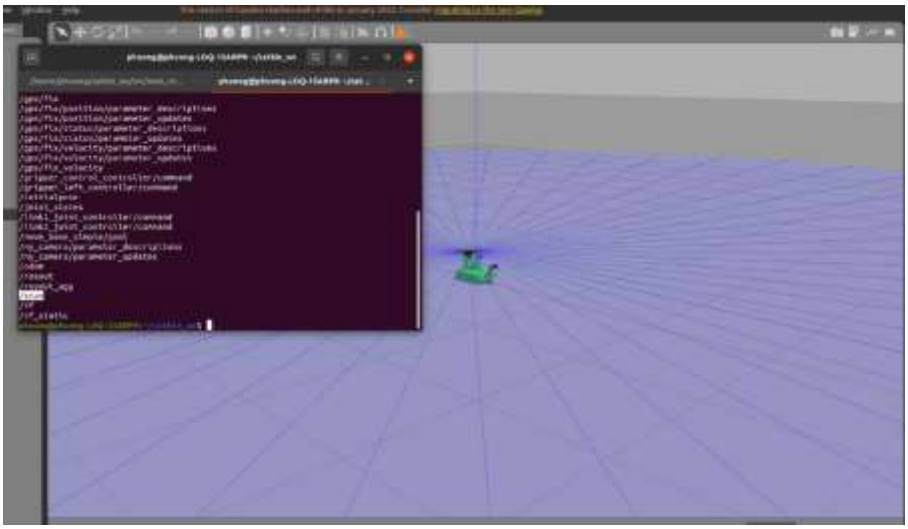
4.1. Cảm biến lidar

- Đây là 1 loại cảm biến cơ bản được sử dụng và ứng dụng nhiều trong robot tự hành, slam để di chuyển trong map, địa hình phức tạp. Để sử dụng cảm biến Lidar ta cần thêm 1 plugin cần thiết vào urdf:

```
<gazebo reference="lidar_link"> <!-- Tham chiếu đến liên kết (link) có tên là "lidar_link" trong mô hình URDF/SDF  -->
  <material>Gazebo/FlatBlack</material> <!-- Thiết lập vật liệu hiển thị trong Gazebo là màu đen phẳng-->

  <sensor type="ray" name="head_rplidar_sensor"> <!-- Khai báo cảm biến loại "ray" (tia quét) với tên là "head_rplidar_sensor" -->
    <pose>0 0 0 0 0 0</pose> <!-- Xác định vị trí và hướng của cảm biến trong hệ tọa độ của "lidar_link" (x, y, z, roll, pitch, yaw) -->
    <visualize>true</visualize> <!-- Cho phép hiển thị cảm biến trong giao diện Gazebo -->
    <update_rate>30.0</update_rate> <!-- Tần số cập nhật cảm biến là 30 lần mỗi giây (30 Hz) -->

    <ray> <!-- Bắt đầu cấu hình cho cảm biến tia quét -->
      <scan> <!-- Xác định thông số quét của cảm biến -->
        <horizontal> <!-- Cài đặt cho quét theo phương ngang -->
          <samples>50</samples> <!-- Số lượng tia quét trong mỗi vòng quét (50 tia) -->
          <resolution>1</resolution> <!-- Độ phân giải góc, mỗi tia quét sẽ có giá trị riêng lẻ (1: không nội suy giữa các tia) -->
          <min_angle>0.0</min_angle> <!-- Góc quét tối thiểu theo radian (0 rad, tương ứng 0 độ) -->
```

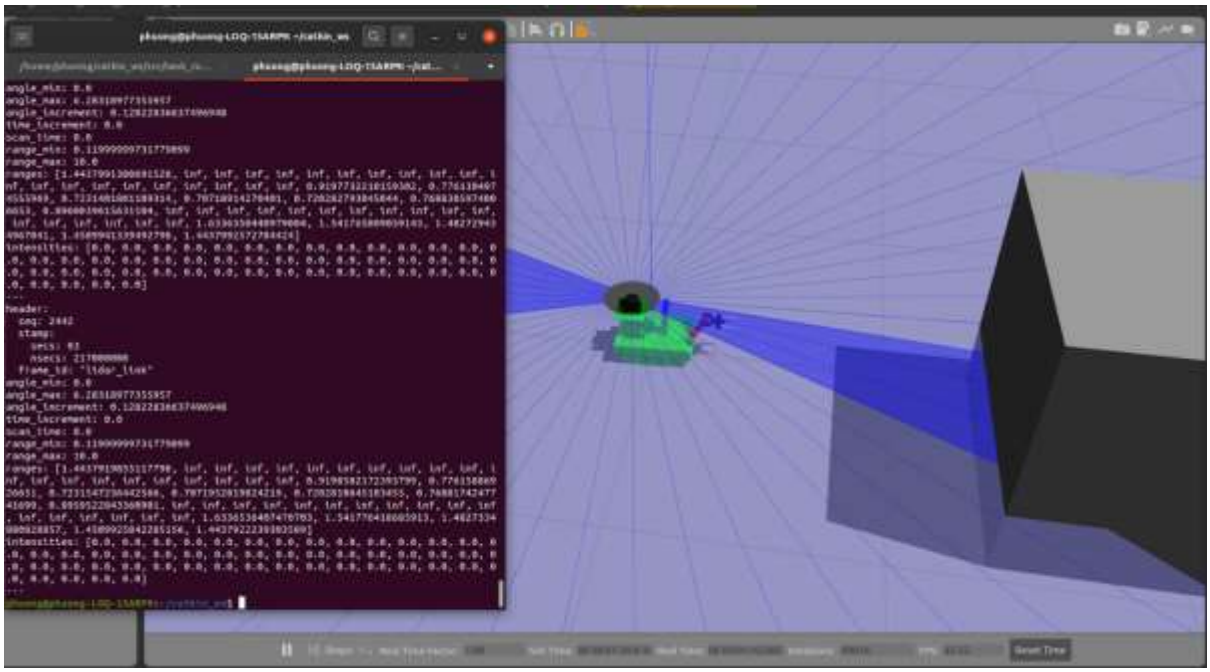
```
          <max_angle>6.28319</max_angle>  <!-- Góc quét tối đa theo radian (~6.28319 rad,
tương ứng 360 độ) -->
        </horizontal> <!-- Kết thúc phần cấu hình quét ngang  -->
      </scan> <!-- Kết thúc phần thông số quét -->

      <range> <!-- Thiết lập phạm vi đo lường của cảm biến -->
        <min>0.120</min>  <!-- Khoảng cách đo tối thiểu là 0.12 m (12 cm) -->
        <max>10</max>  <!-- Khoảng cách đo tối đa là 10 m -->
        <resolution>0.015</resolution> <!-- Độ phân giải của khoảng cách là 0.015 m (15 mm)  --
>
      </range> <!-- Kết thúc phần thiết lập phạm vi -->
    </ray> <!-- Kết thúc phần cấu hình cảm biến tia quét -->

    <plugin name="gazebo_ros_head_rplidar_controller" filename="libgazebo_ros_laser.so"> <!-
- Khai báo plugin ROS cho cảm biến lidar -->
      <topicName>scan</topicName>  <!-- Dữ liệu quét của Lidar sẽ được xuất bản lên topic
ROS có tên "scan" -->
      <frameName>lidar_link</frameName>  <!-- Tên khung tọa độ (frame) của cảm biến là
"lidar_link" -->
      <gaussianNoise>0.01</gaussianNoise>  <!-- Mức độ nhiễu Gaussian (nhiễu ngẫu nhiên) là
0.01 -->
    </plugin> <!-- Kết thúc phần cấu hình plugin ROS -->
  </sensor> <!-- Kết thúc phần cấu hình cảm biến -->
</gazebo> <!-- Kết thúc khối cấu hình Gazebo -->
```

- Sau khi thêm plugin xong khởi động mô hình roslaunch tank_robot5 gazebo.launch, cảm biến sẽ hiện lên trên mô hình.
- Kiểm tra topic của cảm biến: rostopic list --> sẽ thấy topic liên quan đến /scan nghĩa là cảm biến đã hoạt động:

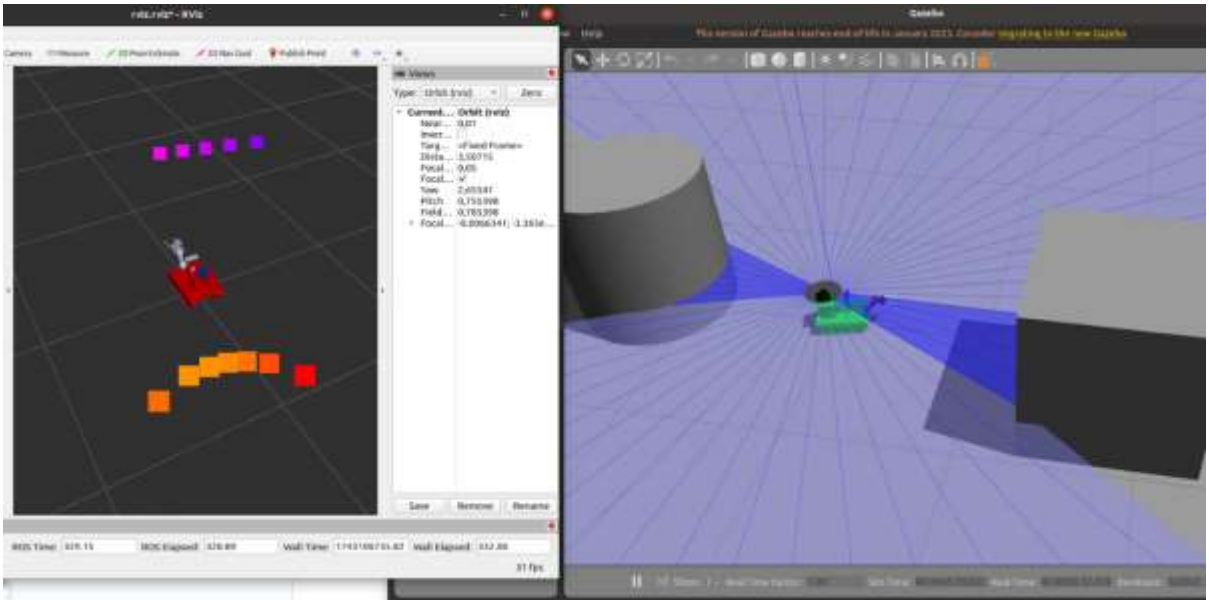


(Cảm biến lidar hoạt động với topic /scan)

– Kiểm tra giá trị trả về của cảm biến lidar: rostopc echo /scan



– Cảm biến trả về các giá trị khoảng cách đo được với đơn vị là (m) với **inf**: Vô cực, tức là không phát hiện được vật thể (ngoài phạm vi đo).



(Hiển thị dữ liệu thu được từ gazebo lên Rviz)

4.2. Camera

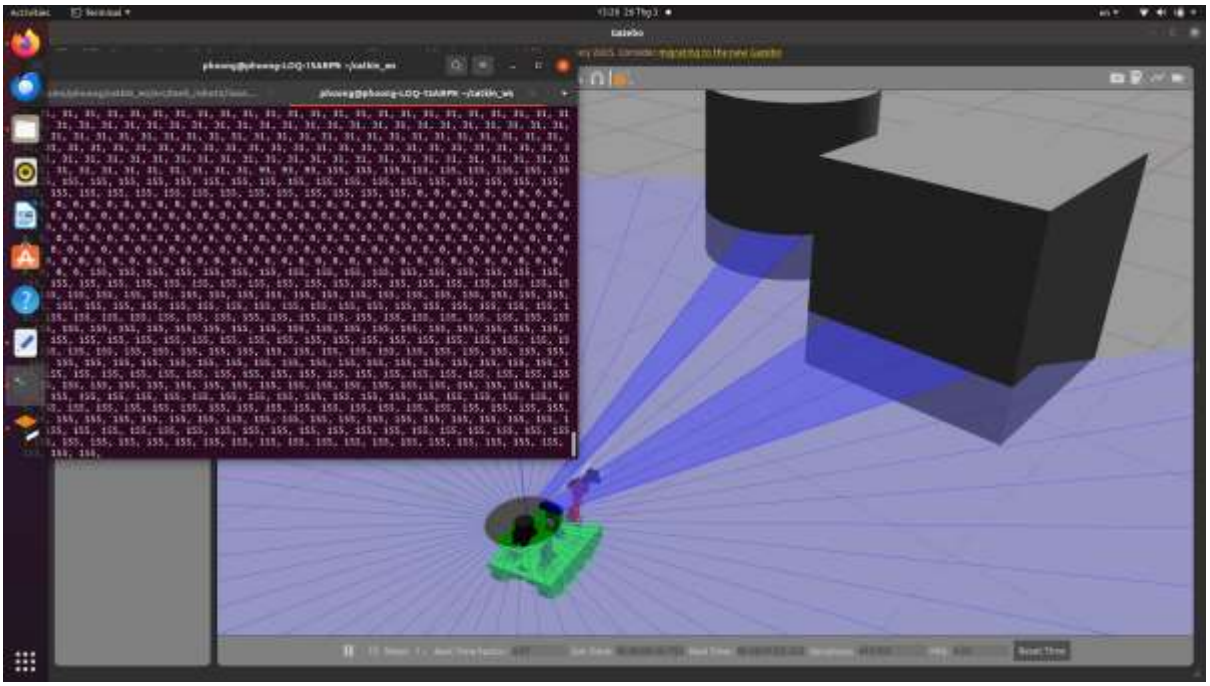
- Camera cũng là một trong những thiết bị quan trọng trong việc sử dụng AI xử lý của robot, giúp robot nhận biết và phân tích vật thể tùy vào mục đích sử dụng sẽ sử dụng 1 số loại camera khác nhau.
- Để sử dụng được camera cần add thêm plugin vào urdf để có thể sử dụng:

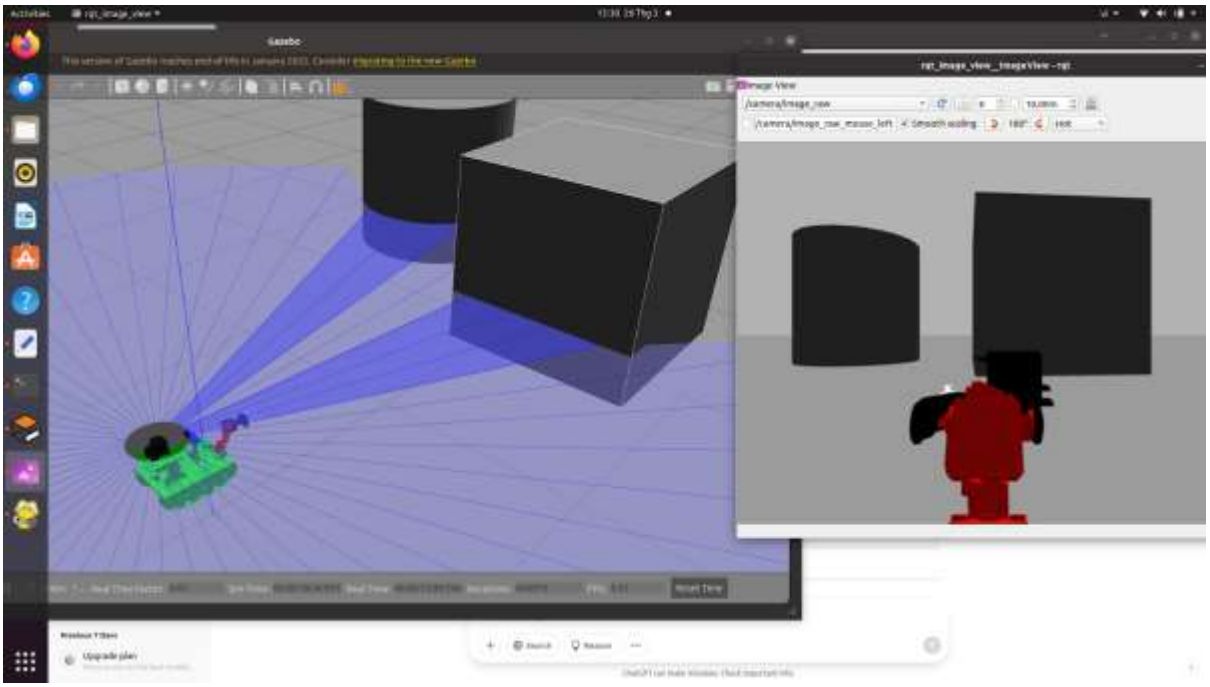
```
<!-- Plugin Camera -->
<gazebo reference="camera_link">  <!-- Gắn cảm biến vào liên kết (link) có tên 'camera_link' -->
 <sensor type="camera" name="camera_sensor"> <!-- Khai báo một cảm biến loại camera với tên 'camera_sensor' -->

  <update_rate>30</update_rate> <!-- Tần suất cập nhật của camera là 30 Hz (30 khung hình/giây) -->

  <camera> <!-- Bắt đầu cấu hình các thông số cho camera -->

   <horizontal_fov>1.3962634</horizontal_fov> <!-- Góc nhìn ngang (Field of View) là 1.3962634 rad (~80 độ) -->

   <image> <!-- Cài đặt thông số ảnh mà camera chụp được -->
    <width>640</width> <!-- Chiều rộng ảnh là 640 pixel -->
    <height>480</height>  <!-- Chiều cao ảnh là 480 pixel -->
    <format>R8G8B8</format>  <!-- Định dạng ảnh là RGB (8-bit cho mỗi kênh màu: đỏ, lục, lam) -->
```

```
    </image>

  </camera> <!-- Kết thúc cấu hình camera -->

  <!-- Cấu hình plugin ROS cho camera -->
  <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
    <!-- 'name' là tên plugin, 'filename' là file thư viện động (.so) giúp camera giao tiếp với ROS -
->

    <alwaysOn>true</alwaysOn> <!-- Camera luôn hoạt động (không bị tắt khi không cần thiết) -
->

    <updateRate>30.0</updateRate> <!-- Tần suất gửi dữ liệu hình ảnh là 30 Hz -->

    <cameraName>my_camera</cameraName> <!-- Tên camera được sử dụng trong ROS -->

    <frameName>camera_link</frameName> <!-- Tên frame (hệ tọa độ) gắn với camera -->

    <imageTopicName>/camera/image_raw</imageTopicName> <!-- Topic ROS phát hình ảnh
thô (raw image) -->

    <cameraInfoTopicName>/camera/camera_info</cameraInfoTopicName> <!-- Topic ROS
phát thông tin hiệu chỉnh camera -->

  </plugin> <!-- Kết thúc cấu hình plugin ROS -->

  <pose>0 0 0 0 0 0</pose> <!-- Vị trí và hướng của camera so với 'camera_link':
                    (x, y, z, roll, pitch, yaw) = (0, 0, 0, 0, 0, 0) -->
 </sensor> <!-- Kết thúc khai báo cảm biến camera -->
</gazebo> <!-- Kết thúc cấu hình cho Gazebo -->
```

(Sau thi thêm plugin vào urdf giờ đây ta có thể sử dụng camera)

- Kiểm tra topic của camera vừa tạo: rostopic list sẽ có 1 số topic liên quan đến /camera/image_raw và topic /camera/image_raw sẽ trả về giá trị từ 0 - 255 là các điểm ảnh theo màu RGB.



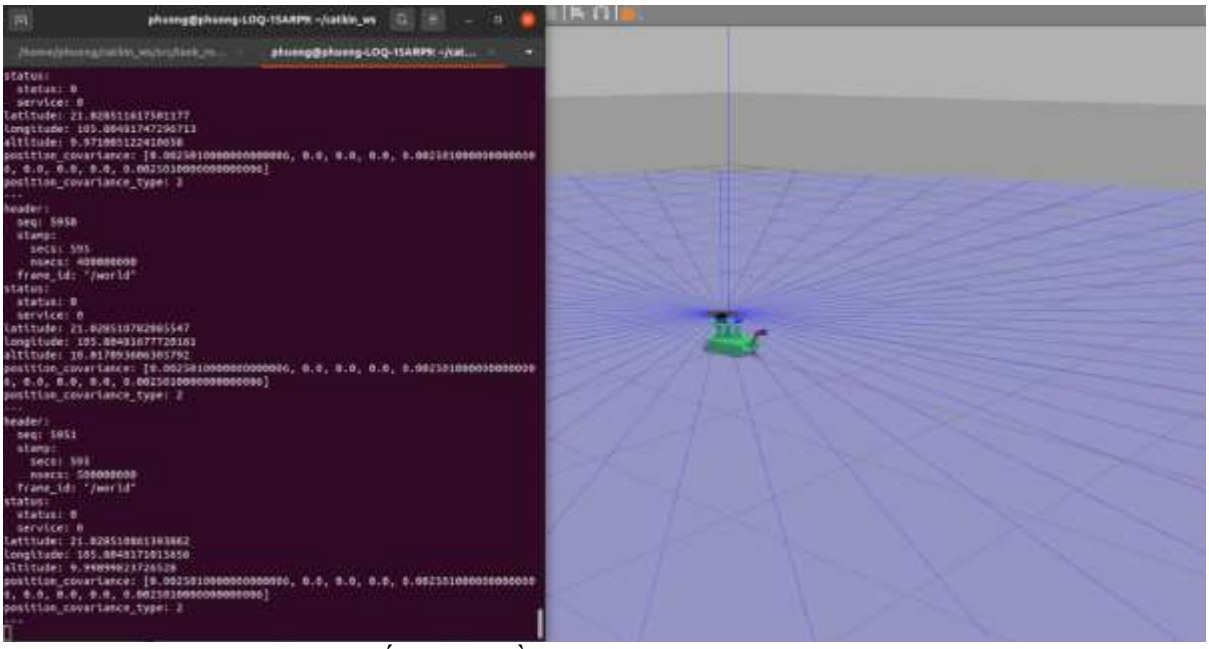


4.3. GPS

- GPS là 1 module quan trọng trong việc định hướng và vị trí của robot đặc biệt là 1 số robot phải thực hiện 1 số nhiệm vụ ngoài trời .
- Để có thể sử dụng GPS ta cần thêm plugin vào urdf:

```
<gazebo>
 <!-- Bắt đầu khai báo cấu hình cho Gazebo -->
  <plugin name="gps_plugin"  filename="libhector_gazebo_ros_gps.so">
   <!-- Khai báo plugin GPS sử dụng thư viện "libhector_gazebo_ros_gps.so" -->

   <alwaysOn>true</alwaysOn>
   <!-- Luôn kích hoạt plugin GPS, ngay cả khi không có client nào subscribe -->

   <updateRate>10.0</updateRate>
   <!-- Tần suất cập nhật dữ liệu GPS là 10 Hz (10 lần/giây) -->

   <bodyName>base_link</bodyName>
   <!-- Xác định liên kết (link) mà cảm biến GPS gắn vào (ở đây là "base_link") -->

   <topicName>/gps/fix</topicName>
   <!-- Tên topic ROS xuất ra dữ liệu vị trí GPS (kiểu sensor_msgs/NavSatFix) -->

   <velocityTopicName>/gps/fix_velocity</velocityTopicName>
   <!-- Tên topic ROS xuất ra dữ liệu vận tốc (kiểu geometry_msgs/Vector3Stamped) -->

   <referenceLatitude>21.028511</referenceLatitude>
   <!-- Giá trị vĩ độ gốc (ở đây là 21.028511 - tọa độ ở Hà Nội, Việt Nam) -->

   <referenceLongitude>105.804817</referenceLongitude>
   <!-- Giá trị kinh độ gốc (ở đây là 105.804817 - tọa độ ở Hà Nội, Việt Nam) -->

   <referenceAltitude>10.0</referenceAltitude>
   <!-- Độ cao gốc so với mực nước biển (10 mét) -->

   <drift>0.001 0.001 0.001</drift>
   <!-- Mô phỏng độ lệch (trôi) của tín hiệu GPS theo các trục X, Y, Z (đơn vị: mét) -->

   <gaussianNoise>0.05 0.05 0.05</gaussianNoise>
   <!-- Mô phỏng nhiễu Gauss (ngẫu nhiên) trên các trục X, Y, Z (đơn vị: mét) -->

  </plugin>
  <!-- Kết thúc khai báo plugin GPS -->
 </gazebo>
<!-- Kết thúc phần cấu hình cho Gazebo -->
```

- Sau khi thêm plugin ta kiểm tra topic của gps: GPS thường sẽ trả về 1 số topic như /gps/fix
- Lưu ý để có thể sử dụng gps sẽ phải Xác định liên kết (link) mà cảm biến GPS gắn vào (ở đây là "base_link").
- Kiểm tra dữ liệu từ topic:



(Cảm biến sẽ trả về các giá trị: Vĩ độ, kinh độ, cao độ)

4.4. Điều khiển động cơ 4 bánh hệ bánh xích

– Để điều khiển được tốc độ của bánh xe cần thêm plugin

```
<!-- Plugin điều khiển di chuyển trong Gazebo -->
<gazebo>
 <!-- Khai báo plugin diff_drive_controller sử dụng file thư viện libgazebo_ros_diff_drive.so -->
 <plugin name="diff_drive_controller" filename="libgazebo_ros_diff_drive.so">
  <ros>
  <!-- Thiết lập namespace cho các topic ROS -->
  <namespace>/</namespace>

  <!-- Đổi tên topic từ cmd_vel thành /cmd_vel -->
  <remap>
     <from>cmd_vel</from>
     <to>/cmd_vel</to>
  </remap>

  <!-- Thiết lập cấp độ log (có thể là INFO, WARN, ERROR, DEBUG) -->
  <rosDebugLevel>INFO</rosDebugLevel>
  </ros>

  <!-- Khai báo các joint (khớp) của bánh xe trái và phải -->
  <leftJoint>wheel_left1_joint</leftJoint>  <!-- Bánh xe trái 1 -->
  <leftJoint>wheel_left2_joint</leftJoint>  <!-- Bánh xe trái 2 -->
  <rightJoint>wheel_right1_joint</rightJoint>  <!-- Bánh xe phải 1 -->
```

```
  <rightJoint>wheel_right2_joint</rightJoint>  <!-- Bánh xe phải 2 -->

  <!-- Các thông số kỹ thuật của bánh xe -->
  <wheelSeparation>0.1566</wheelSeparation>  <!-- Khoảng cách giữa hai bánh xe -->
  <wheelDiameter>0.07</wheelDiameter>          <!-- Đường kính của bánh xe -->
  <wheelAcceleration>1.0</wheelAcceleration>  <!-- Gia tốc tối đa của bánh xe -->
  <wheelTorque>10</wheelTorque>                <!-- Lực mô-men xoắn cực đại -->

  <!-- Chủ đề (topic) và khung TF liên quan đến robot -->
  <commandTopic>cmd_vel</commandTopic>       <!-- Chủ đề nhận lệnh vận tốc -->
  <odometryTopic>odom</odometryTopic>       <!-- Chủ đề xuất dữ liệu odometry -->
  <odometryFrame>odom</odometryFrame>       <!-- Tên khung odometry -->
  <robotBaseFrame>base_footprint</robotBaseFrame>  <!-- Khung gốc của robot -->

  <!-- Cài đặt xuất dữ liệu TF và trạng thái bánh xe -->
  <publishWheelTF>true</publishWheelTF>        <!-- Xuất TF của bánh xe -->
  <publishOdomTF>true</publishOdomTF>          <!-- Xuất TF của odometry -->
  <publishWheelJointState>true</publishWheelJointState>  <!-- Xuất trạng thái joint của bánh
xe -->
  <publishTf>1</publishTf>                  <!-- Xuất thông tin TF với tần suất thiết lập -->

  <!-- Tần số cập nhật plugin (Hz) -->
  <updateRate>100</updateRate>  <!-- Tần số cập nhật là 100Hz -->

  <!-- Chọn nguồn dữ liệu odometry (0 = ground truth, 1 = encoder) -->
  <odometrySource>1</odometrySource>  <!-- Sử dụng dữ liệu encoder -->

  <!-- Thiết lập cấp độ log cho ROS -->
  <rosDebugLevel>info</rosDebugLevel>

 </plugin>
</gazebo>
```

→ Khi đã có plugin để có thể điều khiển được tốc độ bánh xe bằng mã python hoặc các thư viện điều khiển khác.

4.5. Điều khiển các khớp của tay máy

- Tương tự để điều khiển được các khớp của tay máy ta cần có plugin điều khiển riêng cho các link:

```
  <!-- Truyền động (Transmission) cho khớp link1 -->
<transmission name="link1_joint_transmission">
  <!-- Kiểu truyền động đơn giản (SimpleTransmission) -->
  <type>transmission_interface/SimpleTransmission</type>
```

```
  <!-- Khai báo khớp (joint) link1_joint -->
  <joint name="link1_joint">
    <!-- Giao diện phần cứng: Điều khiển vị trí (PositionJointInterface) -->
    <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
  </joint>

  <!-- Khai báo bộ chấp hành (actuator) link1_servo -->
  <actuator name="link1_servo">
    <!-- Tỉ lệ giảm cơ khí: 1.0 (không giảm tốc) -->
    <mechanicalReduction>1.0</mechanicalReduction>
    <!-- <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface> -->
  </actuator>
</transmission>

<!-- Truyền động cho khớp link2 -->
<transmission name="link2_joint_transmission">
  <!-- Kiểu truyền động đơn giản -->
  <type>transmission_interface/SimpleTransmission</type>

  <!-- Khai báo khớp link2_joint -->
  <joint name="link2_joint">
    <!-- Điều khiển vị trí của khớp -->
    <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
  </joint>

  <!-- Khai báo bộ chấp hành link2_servo -->
  <actuator name="link2_servo">
    <!-- Tỉ lệ giảm cơ khí: 1.0 (không giảm tốc) -->
    <mechanicalReduction>1.0</mechanicalReduction>
    <!-- <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface> -->
  </actuator>
</transmission>

<!-- Truyền động cho khớp điều khiển Gripper -->
<transmission name="gripper_transmission">
  <!-- Kiểu truyền động đơn giản -->
  <type>transmission_interface/SimpleTransmission</type>

  <!-- Khai báo khớp gripper_control_joint -->
  <joint name="gripper_control_joint">
    <!-- Điều khiển vị trí của khớp -->
    <hardwareInterface>PositionJointInterface</hardwareInterface>
  </joint>

  <!-- Khai báo bộ chấp hành gripper_motor -->
```

```
  <actuator name="gripper_motor">
    <!-- Tỉ lệ giảm cơ khí: 1 (không giảm tốc) -->
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<!-- Truyền động cho khớp Gripper bên trái -->
<transmission name="gripper_left_transmission">
  <!-- Kiểu truyền động đơn giản -->
  <type>transmission_interface/SimpleTransmission</type>

  <!-- Khai báo khớp gripper_left_joint -->
  <joint name="gripper_left_joint">
    <!-- Điều khiển vị trí của khớp -->
    <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
  </joint>

  <!-- Khai báo bộ chấp hành gripper_left_motor -->
  <actuator name="gripper_left_motor">
    <!-- Tỉ lệ giảm cơ khí: 1.0 (không giảm tốc) -->
    <mechanicalReduction>1.0</mechanicalReduction>
  </actuator>
</transmission>

<!-- Plugin ros_control để kết nối Gazebo với ROS -->
<gazebo>
  <!-- Khai báo plugin gazebo_ros_control -->
  <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
    <!-- Thiết lập không gian tên của robot -->
    <robotNamespace>/</robotNamespace>
  </plugin>
</gazebo>
```

Các link của tay máy cần có các hàm truyền động để có thể điều khiển được góc cho các link. Khi đã plugin muốn hoạt động được tay máy cần có bộ điều khiển ros với các tham số PID cho từng link của tay máy trong config/joint_names_tank_robot5.yaml

```
link1_joint_controller:
 type: "position_controllers/JointPositionController"
 joint: "link1_joint"
 pid: {p: 100.0, i: 0.1, d: 5.0}

link2_joint_controller:
 type: "position_controllers/JointPositionController"
 joint: "link2_joint"
 pid: {p: 100.0, i: 0.1, d: 5.0}
```

```
gripper_control_controller:
 type: "position_controllers/JointPositionController"
 joint: "gripper_control_joint"
 pid: {p: 100.0, i: 0.1, d: 5.0}

gripper_left_controller:
 type: "position_controllers/JointPositionController"
 joint: "gripper_left_joint"
 pid: {p: 100.0, i: 0.1, d: 5.0}

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

→ Cuối cùng trong file launch cần thêm đường dẫn tới file yaml để có thể nhận các thông số:

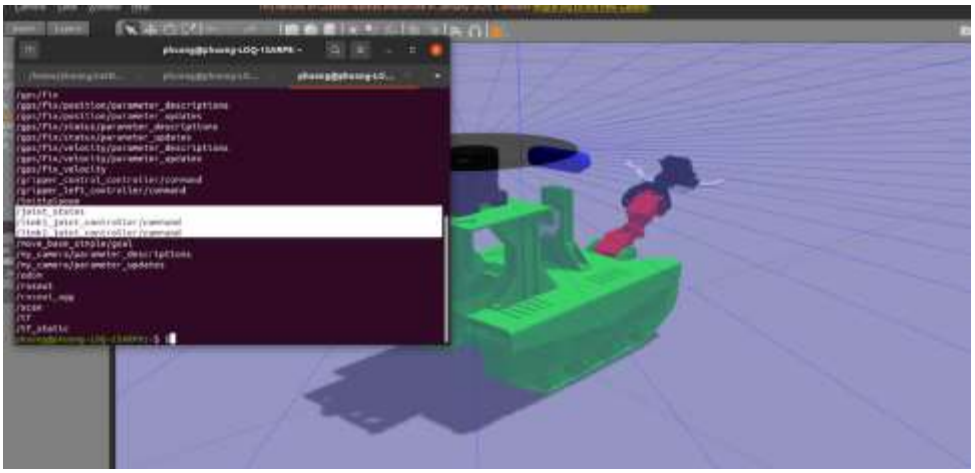
```
<!-- Tải thông số joint -->
  <rosparam command="load"  file="$(find tank_robot5)/config/joint_names_tank_robot5.yaml"
/>
```

và khởi động bộ điều khiển:

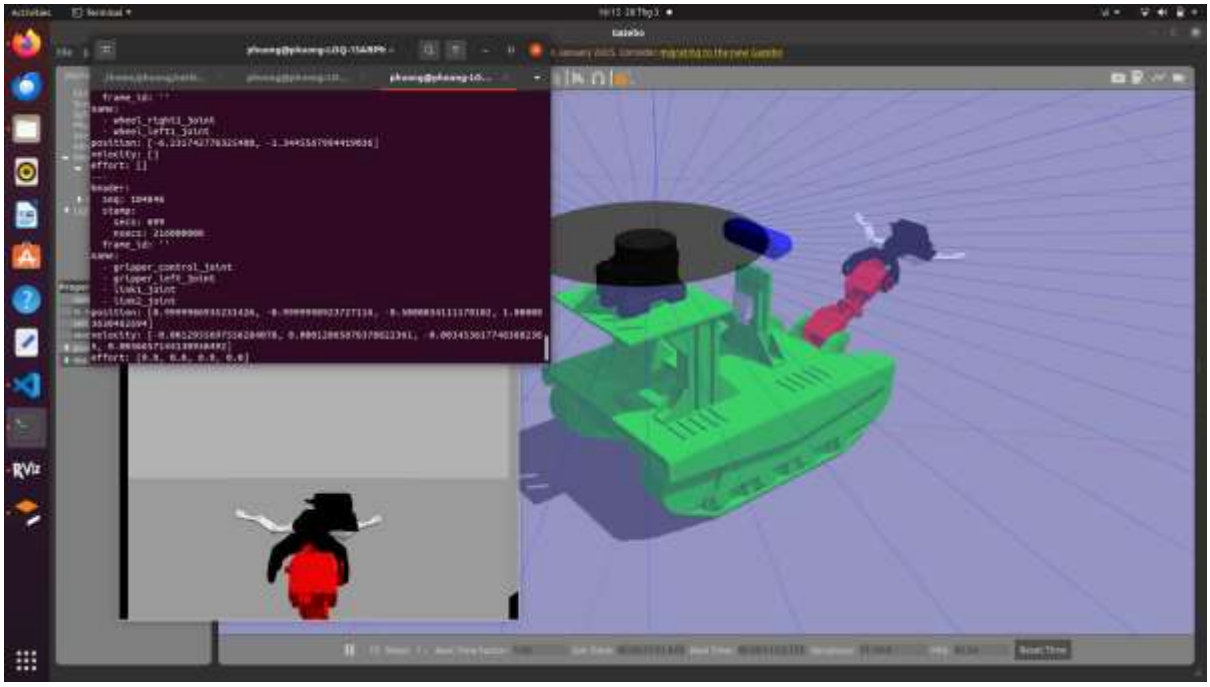
```
  <!-- Khởi động bộ điều khiển -->
  <node
     name="two_dof_arm_controller_spawner"
     pkg="controller_manager"
     type="spawner"
     args="joint_state_controller link1_joint_controller link2_joint_controller" />
```

(Các link khác tương tự)

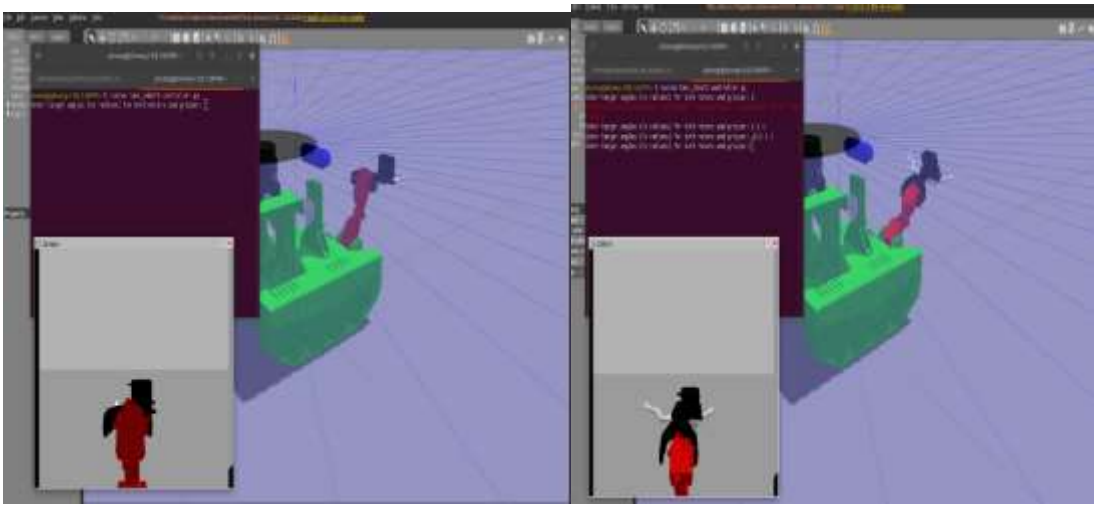
– Viết 1 đoạn code python đơn giản để gửi lệnh điều khiển tới các joint của tay máy, trước hết cần kiểm tra topic điều khiển và vị trí của tay máy đang hoạt động bằng cách rostopic echo /(topic cần kiểm tra)



(/joint_state; /link1,2_joint_controller/command và /gripper_control)



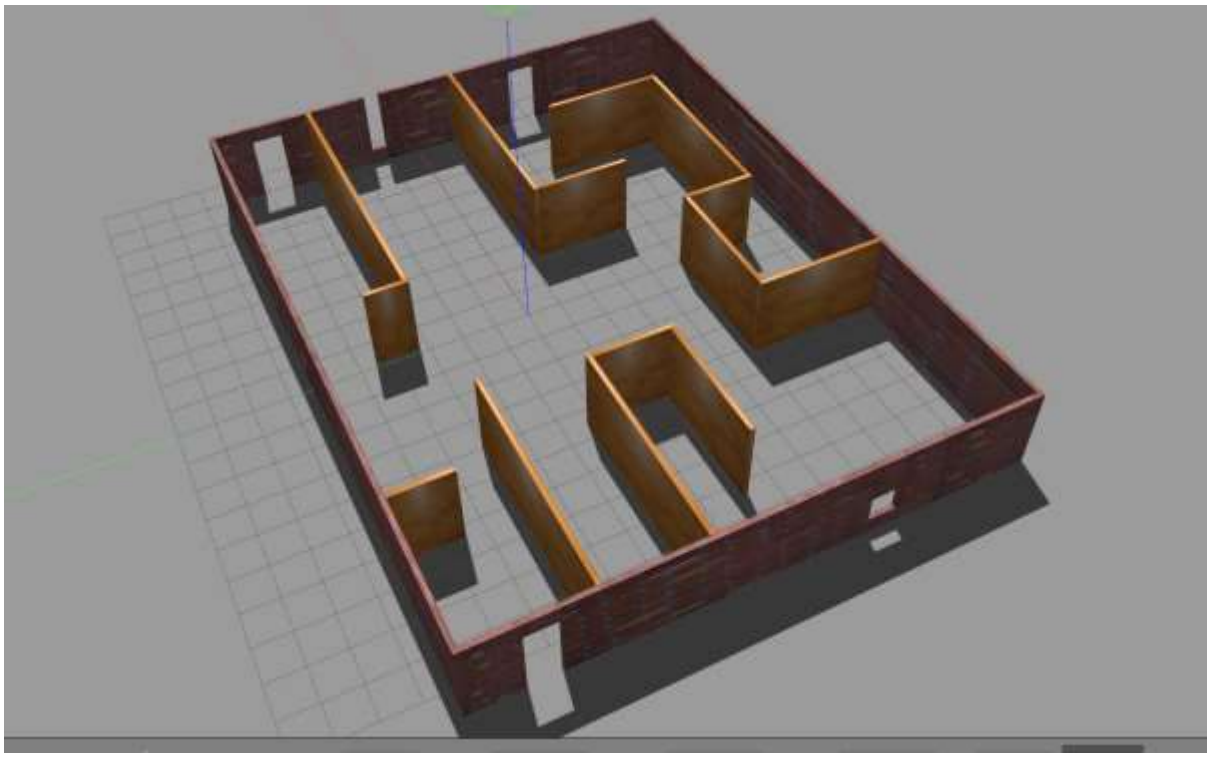
(Thông tin vị trí các khớp)



(Vị trí các khớp sau khi thay đổi)

5. Môi trường mô phỏng:

- Khi các cảm biến, trục bánh xe và các khớp của tay máy đã hoạt động cần tạo môi trường trên gazebo để có thể kiểm thử và kết hợp các loại cảm biến và điều khiển:
- Xây dựng map với chức năng Building Editer và lưu dưới dạng .world.

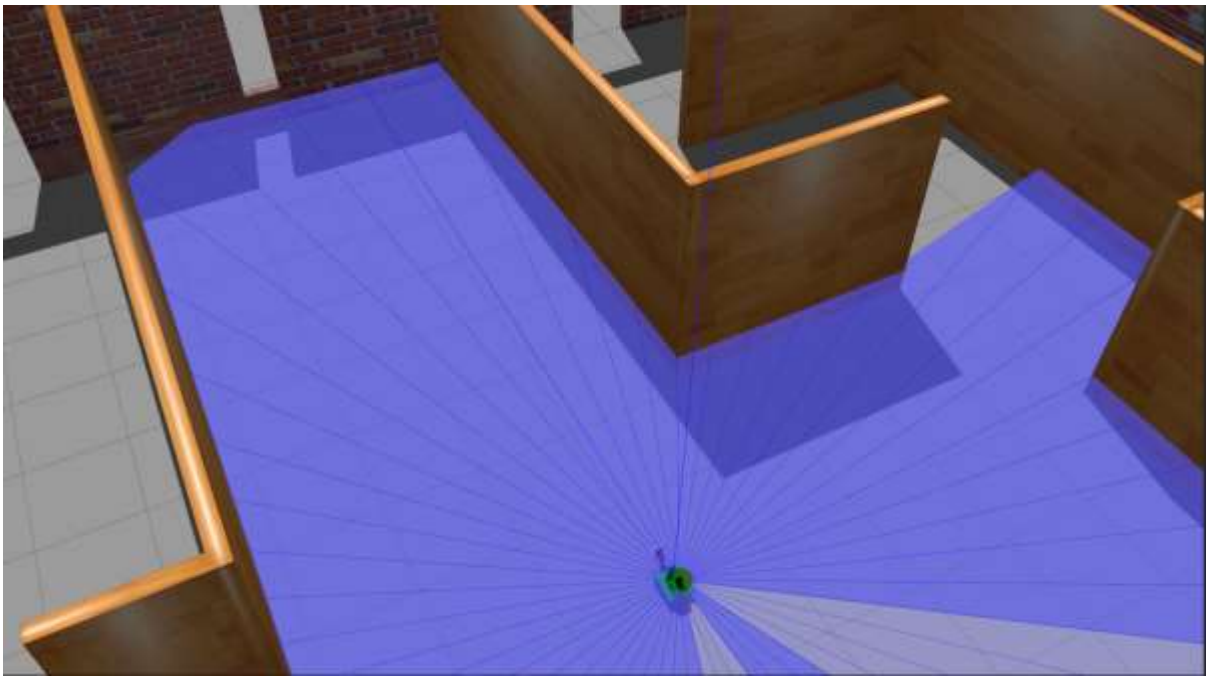


(Map gazebo)

– Khi đã có môi trường mô phỏng add thêm robot vào môi trường đó: Thêm môi trường trong file launch:

```
<!-- Đường dẫn tới mô hình robot -->
  <arg name="model" value="$(find tank_robot5)/urdf/tank_robot5.urdf" />

  <!-- Chuyển đổi file Xacro thành URDF -->
  <param name="robot_description" command="$(find xacro)/xacro $(arg model)" />

  <!-- Đường dẫn tới file world -->
  <arg name="world" default="$(find tank_robot5)/urdf/mapper.world"/>

  <!-- Khởi động Gazebo với map đã tạo -->
  <include file="$(find gazebo_ros)/launch/empty_world.launch">
     <arg name="world_name" value="$(arg world)" />
     <arg name="paused" value="false" />
     <arg name="use_sim_time" value="true" />
     <arg name="gui" value="true" />
  </include>
```

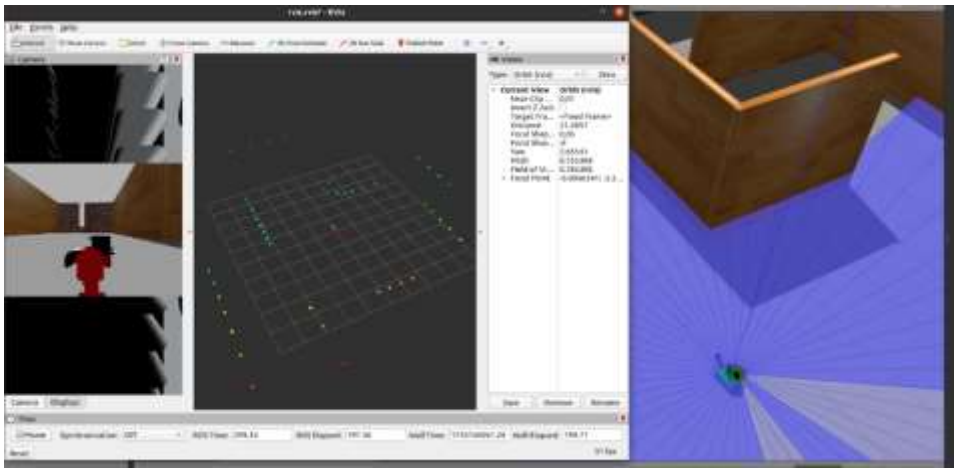
– Mở file lauch vừa thêm map:

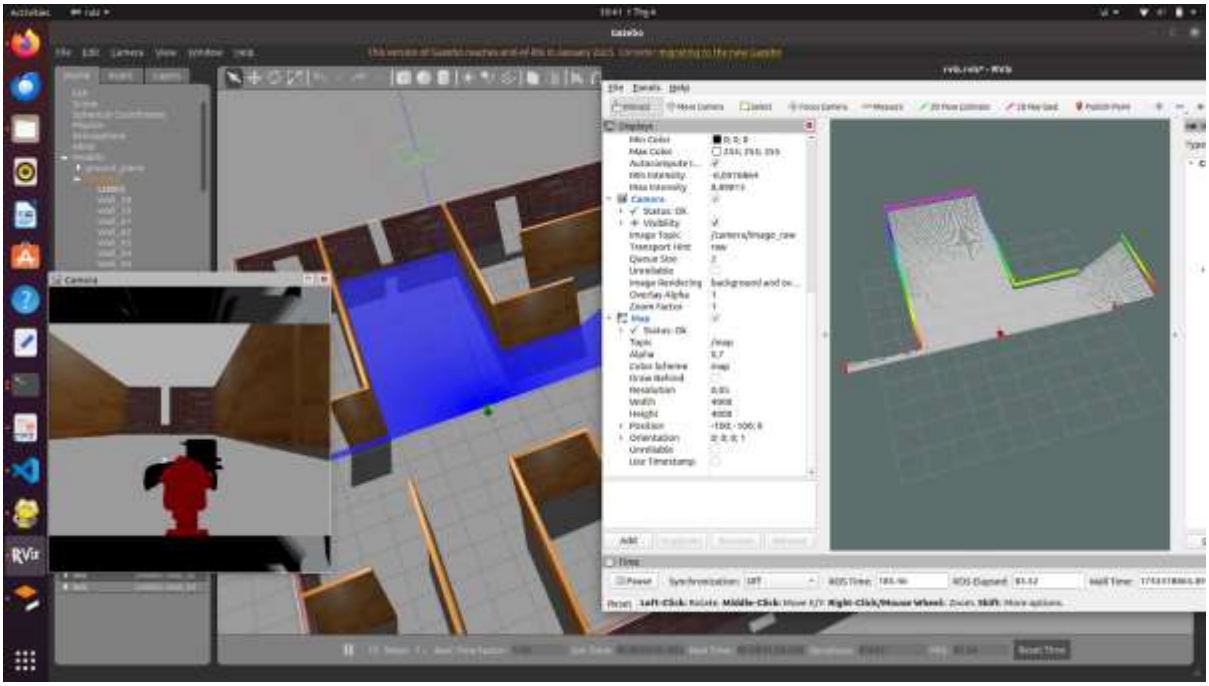


– Mở thêm Rviz để để lấy thông tin từ các loại cảm biến bổ sung vào file launch đường dẫn tới Rviz:

```
<!-- Khởi động RViz với file cấu hình -->
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find tank_robot5)/rviz/rviz.rviz" />
```

```
<!-- Load robot_description từ file URDF -->
<param name="robot_description" textfile="$(arg model)" />
```





(Scan với gmappingapping)