Contents

1. Môi trường mô phỏng Robot

Việc mô phỏng robot giúp việc nghiên cứu Robot trở nên dễ dàng hơn. Có thể vận hành Robot mà không cần đến Robot thực tế, có nhiều lợi ích trong việc thử nghiệm Robot trước khi đưa vào sử dụng thực tế.

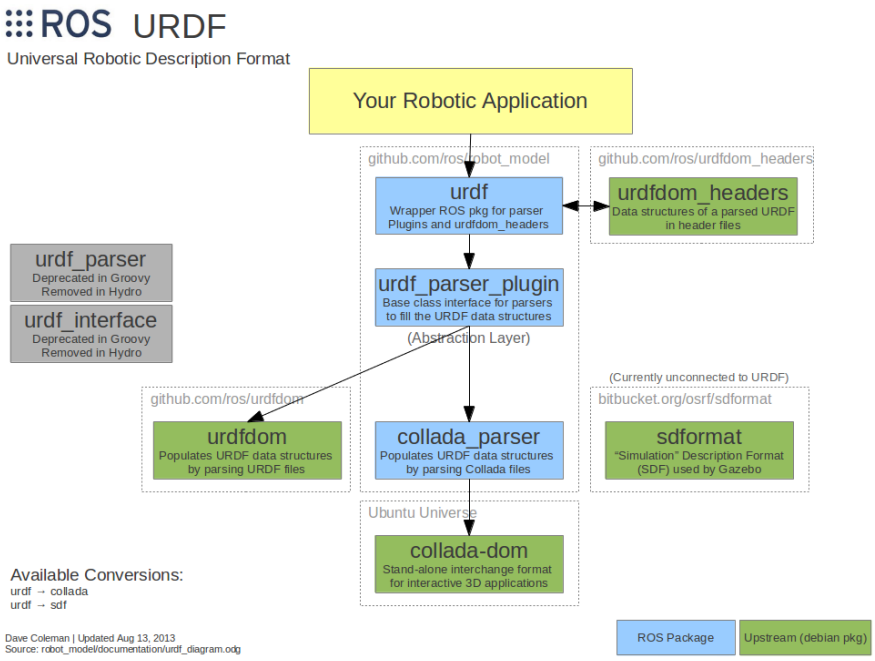
Các tính năng của giả lập Robot:

- Minh họa Robot, mô tả các chuyển động, bản đồ, quỹ đạo di chuyển của Robot (Rviz)
- Mô tả các hiện tượng vật lý, tính chất động học, động lực học, cảm biến, môi trường hoạt động của Robot. (Gazebo)

1.1. Khái niệm về URDF

Là một package có chứa trình phân tích cú pháp C++ nhằm mục đích mô tả Robot - Unified Robot Description Format (URDF), sử dụng định dạng XML để biểu diễn một mô hình Robot. Định dạng này được thiết kế để mô tả nhiều loại Robot khác nhau (Robot dạng người, Robot tự hành với bánh xe, cánh tay Robot). URDF có nhiều điểm tương đồng với Simulation Description Format (SDF), và có thể sử dụng trong môi trường Gazebo.

Source: https://github.com/ros/urdf



Hệ trục tọa độ trong URDF:

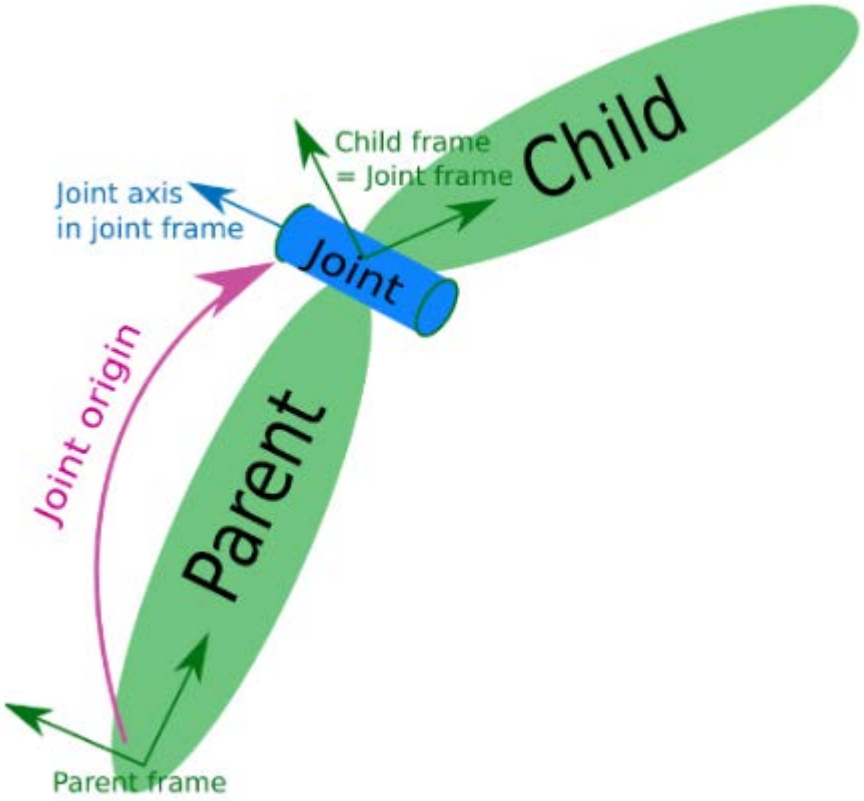
- 3 trục tịnh tiến: x, y, z
- 3 trục quay: r (roll), p (pitch), y (yaw)

Cấu trúc URDF:

Gồm link và joint

Link: Mô tả các thành phần trong Robot

Joint: Là thành phần liên kết các link với nhau, trong đó sẽ có 1 link là mẹ và 1 link là con.



- Các thẻ sử dụng trong URDF:

STT	Tên thẻ	Chức năng
	<robot> </robot>	Mô tả Robot
	<link> </link>	Mô tả 1 link
	<visual> </visual>	Thẻ mô tả hình dáng và màu sắc
	<geometry> </geometry>	Đặt trong visual mô tả hình dáng
	<material> </material>	Mô tả vật liệu
	<color> </color>	Mô tả màu sắc đặt trong thẻ <material>
	<collision />	Va chạm
	<inertial />	Moment quán tính
	<parent />	Link đóng vai trò là mẹ
	<child />	Link đóng vai trò là con
	<origin xyz= rpy= />	Gốc tọa độ theo 3 trục tịnh tiến và ba trục quay
	<axis />	Vector trục
	<gazebo> </gazebo>	Thẻ mô tả trên Gazebo

Ví dụ về 1 link:

```
<?xml version="1.0"?>
<robot name="tortoisebot">
  <link name="base_link">
    <visual>
      <geometry>
        <box size="0.6 0.3 0.3"/>
      </geometry>
      <material name="silver">
        <color rgba="0.75 0.75 0.75 1"/>
      </material>
    </visual>
  </link>
</robot>
```

Trong ví dụ này, thẻ link mô tả base_link như sau:

- base_link có dạng hình hộp kích thước 0.6x0.3x0.3m mô tả trong thẻ geometry
- Có màu bạc (silver) được mô tả trong thẻ material
- Cả hình dáng và màu sắc được mô tả trong thẻ visual

Ví dụ về joint:

```
<link name="front_caster">
  <visual>
    <geometry>
      <box size="0.1 0.1 0.3"/>
    </geometry>
    <material name="silver"/>
  </visual>
</link>

<joint name="front_caster_joint" type="continuous">
  <axis xyz="0 0 1"/>
  <parent link="base_link"/>
  <child link="front_caster"/>
  <origin rpy="0 0 0" xyz="0.3 0 0"/>
</joint>
```

Trong ví dụ này, front-caster_joint được mô tả như sau:

Joint có thể quay trên một trục có sẵn, trong trường hợp này là trục z.

Joint nối front_caster với base_link.

Và có tọa độ: xyz: 0.3 0 0, rpy: 0 0 0

*** Các gói cần thiết khi sử dụng urdf:**

- URDF Display Tutorial:
  - Cài đặt từ Terminal: $ sudo apt-get install ros-kinetic-urdf-tutorial
- Joint state publisher:
  - Cài đặt từ Terminal: $ sudo apt-get install joint-state-publisher
  - Source code: $ git clone https://github.com/ros/joint_state_publisher
- Robot pose publisher:
  - Cài đặt từ Terminal: $ sudo apt-get install robot-pose-publisher
  - Source code: $ git clone https://github.com/GT-RAIL/robot_pose_publisher

1.2. Rviz

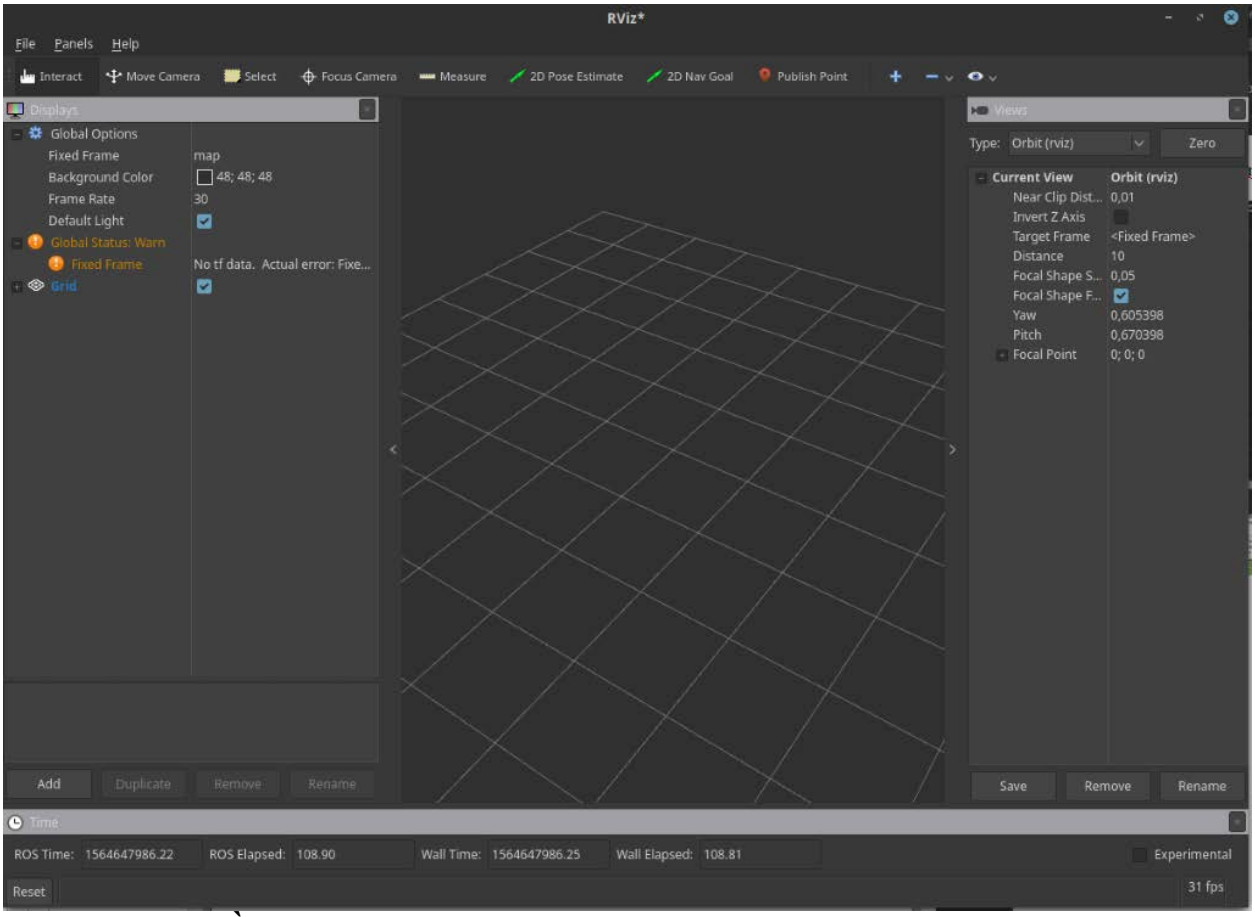
1 công cụ giả lập hình ảnh 3D, có thể mô phỏng khoảng cách từ cảm ứng LDS (Laser Distance Sensor), PCD của các cảm biến khoảng cách 3D như RealSense, Kinect, Xtion.

*** Cài đặt Rviz**

Nếu đã cài đặt bản full của ROS, RViz đã được cài đặt tích hợp trong hệ thống. Nếu chưa, ta cài đặt thông qua lệnh:

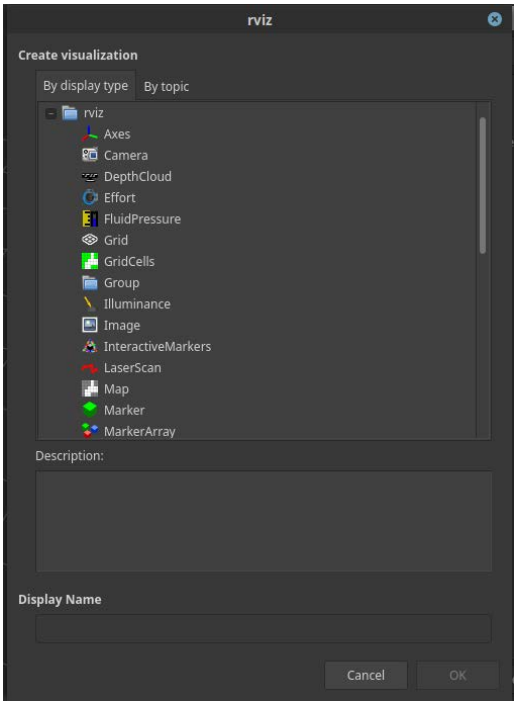
```
$ sudo apt-get install ros-kinetic-rviz
```

Để chạy rviz, đầu tiên chạy roscore, sau đó có thể khởi chạy như 1 node: rosrun rviz rviz hoặc $ rviz



*** Các thành phần trong giao diện Rviz:**

- **3D View:** Vùng màu đen nằm giữa màn hình, cho phép quan sát dữ liệu 3D.
- **Displays:** Thanh Display nằm ở cột trái màn hình, dùng để lựa chọn các dữ liệu chúng ta muốn hiển thị từ vô số các topics. Nếu click [add] ở góc dưới bên trái của thanh màn hình lựa chọn sẽ xuất hiện. Như hình dưới:



- **Menu:** Thanh lựa chọn nằm ở phần trên cùng của màn hình. (Save/Load, …)
- **Tools:** Bộ công cụ được đặt ngay dưới thanh Menu. Chứ các hàm như interact, camera movement (di chuyển cam), lựa chọn – selection, …
- **View:** Cài đặt góc nhìn của 3D View

→ **Orbit:** Xác định 1 điểm được gọi là trung tâm và quỹ đạo quay quanh điểm đó. Đây là lựa chọn mặc định và phổ biến nhất.

→ **FPS:** Góc nhìn thứ nhất

→ **ThirdPersonFollower:** Góc nhìn thứ 3, đi theo 1 đối tượng xác định.

→ **TopDownOrtho:** Sử dụng trục Z làm gốc, và hiển thị 1 hình chiếu của đối tượng lên mặt XY.

→ **XY Orbit:** Tương tự như Orbit nhưng điểm gốc được cố định vào mặt XY và coi tất cả các giá trị Z là 0.

- **Time:** Hiển thị thời gian hiện tại, ROS time và thời gian đã trôi qua kể từ khi khởi động chúng.

*** Rviz Displays**

Axes	- Hiển thị 3 trục XYZ
Camera	- Tạo 1 cửa sổ render từ điểm và các lớp hủ của 1 ảnh từ đầu của nó
DepthCloud	- Hiển thị 1 điểm đám mây dựa trên DepthMap. Hiển thị giá trị khoảng cách đo được từ cảm biến và topic ColorImage

	thành các điểm phủ màu nhận được bởi camera.
Effort	- Hiển thị lực tác dụng lên khớp xoay của Robot.
FluidPressure	- Hiển thị áp lực của các chất có liên kết tự
Grid	- Lưới 2D và 3D
Grid Cells	- Hiển thị các ô trên lưới. Chủ yếu được sử dụng để hiển thị vật cản trong costmap khi điều hướng.
Group	- Cho phép quản lý hiển thị thông qua các group hiển thị.
Illuminance	- Độ sáng.
Image	- Hiển thị ảnh ở cửa sổ render mới. Khác với Camera, không tạo các lớp phủ.
Interactive Markers	- Chúng ta có thể thay đổi vị trí (x, y, z) và quay (roll, Pitch, yaw) với chuột
LaserScan	.- Hiển thị giá trị quét laser.
Map	- Hiển thị bản đồ, sử dụng trong định hướng và ở mặt chiếu bằng.
Marker	- Hiển thị các đánh dấu như mũi tên, vòng tròn, tam giác, tứ giác, trụ, …
MarkArrray	- Hiển thị tập hợp các dấu.
Odometry	- Hiển thị thông tin quan hệ giữa đường đi theo thời gian dưới dạng mũi tên.
Path	- Hiển thị quãng đường mà robot đã sử dụng định hướng.
Point Cloud	- Hiển thị dữ liệu các điểm theo đám mây. Sử dụng dữ liệu từ cảm biến. PointClound2 phù hợp với Thư viện các điểm trong đám mây được cập nhật mới nhất.
Point Cloud2	
Point Stamped	- Hiển thị các điểm thành vòng tròn.
Polygon	- Hiển thị khung đa giác, thường sử dụng để dựng khung đường cho Robot trên mặt phẳng 2D.
Pose	- Hiển thị pose (vị trí + hướng) 3D. Biểu thị bằng mũ tên, gốc là vị trí (tọa độ x, y, z) và hướng của mũi tên. Pose có thể biểu thị vị trí và hướng của 1 model robot 3D, trong khi có thể biểu thị dưới dạng điểm.
Pose Array	- Chuỗi các Pose

Range	- Giả lập vùng đo được từ cảm biến ví dụ như siêu âm hay hồng ngoại.
Relative Humidity	- Hiển thị độ ẩm liên quan.
RobotModel	- Hiển thị model của robot.
TF	- Hiển thị tọa độ biến đổi TF bởi ROS. Nó có thể hiển thị cùng với trục xyz, mỗi trục tọa độ biểu thị cấp với mũi tên đi theo trục tọa độ.
Temperature	- Hiển thị nhiệt độ.
WrenchStamped	- Hiển thị xoắn, với chuyển động xoắn, theo dạng mũi tên (lực), mũi tên + vòng tròn (torque).

1.3. Gazebo và các Plugin

Gazebo là giả lập 3D cung cấp mô hình Robot, cảm biến và môi trường, nó cho phép mô phỏng các tính chất vật lý thực tế. Gazebo rất phổ biến trong những năm gần đây và có nhiều ứng dụng trong lĩnh vực Robot. Gazebo được phát triển và phân phối bởi Open Robotics và được tích hợp trong ROS.

Các tính chất của Gazebo:

- **Giả lập động lực học:** Mới phát triển, hiện tại mới chỉ dừng lại ở ODE (Open Dynamics Engine).
- **Đồ họa 3D:** Gazebo sử dụng OGRE, công nghệ render thường được sử dụng trong game, có thể mô tả không chỉ hình dạng Robot mà còn cả ánh sáng, đổ bóng và màu sắc.
- **Cảm biến và tiếng ồn:** Cảm biến Laser (LRF), 2D/3D camera, depth camera, các cảm biến va chạm, cảm biến lực - quán tính và nhiều loại cảm biến khác được hỗ trợ tương tự như môi trường thật.
- **Hình dạng Robot:** Được hỗ trợ bằng định dạng SDF, người dùng có thể tự tạo mô hình cho riêng mình.
- **Giao thức truyền thông TCP/IP:** Giả lập có thể chạy trên một server từ xa.
- **Cloud:** Gazebo cung cấp giả lập đám mây thông qua môi trường CloudSim được sử dụng trong các môi trường đám mây như Amazon, Softlayer, và OpenStack.
- **Command Line Tool:** Cả GUI và các công cụ GUI được hỗ trợ để kiểm tra và điều khiển trạng thái giả lập.

* **Tổng quan về Plugin Gazebo:**

Plugin được compile như một thư viện được chèn vào giả lập. Plugin có thể được trực tiếp trang bị cho Gazebo thông qua các class C++ tiêu chuẩn.

Plugin rất tiện dụng bởi:

- Có thể điều khiển mọi thứ của Gazebo

- Dễ dàng chia sẻ
- Có thể chèn và xóa khỏi một hệ thống đang hoạt động

Tiền thân của plugin Gazebo xuất phát từ các Vi Điều khiển thực tế. Plugin mô tả lại phương thức hoạt động này. Plugin nên được sử dụng khi chúng ta muốn lập trình trực tiếp trên giả lập.

* **Các loại Plugin:**

- World
- Model
- Sensor
- System
- Visual
- GUI

Mỗi plugin đóng một vai trò khác nhau trong Gazebo. Ví dụ, Model plugin là những model cụ thể trên Gazebo, World plugin mô tả một môi trường, Sensor plugin mô tả các cảm biến. System plugin mô tả command line, và được chạy đầu tiên khi khởi động Gazebo.

Chi tiết: http://gazebosim.org/tutorials?tut=plugins_hello_world&cat=write_plugin

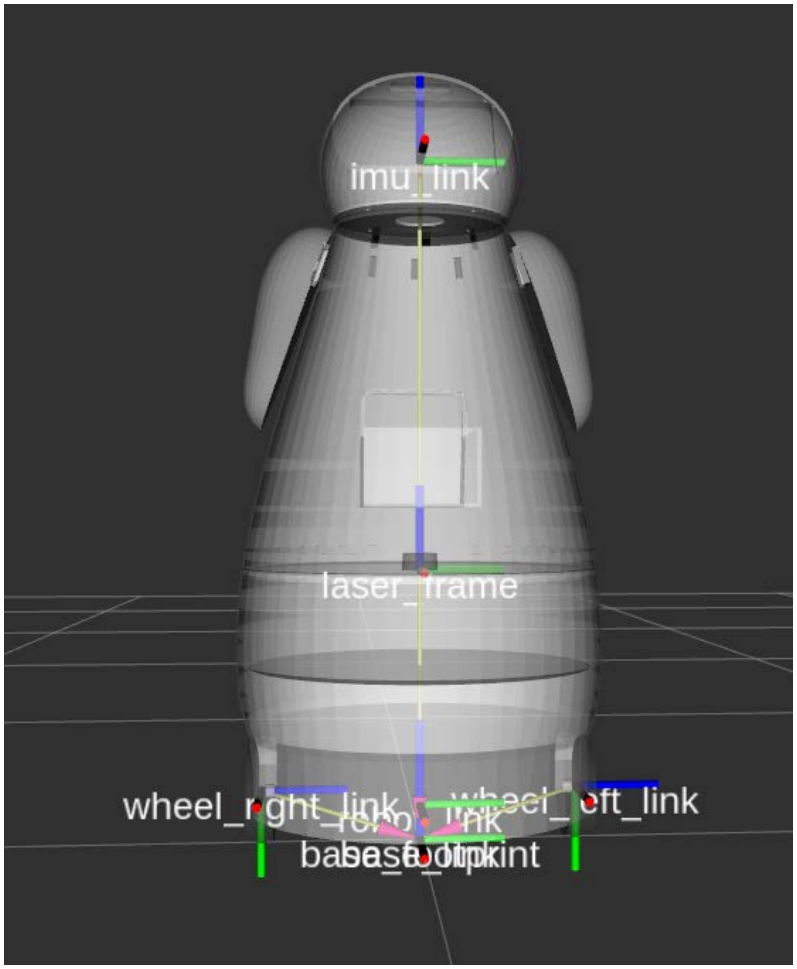
2. Các thành phần Robot

2.1. Đế Robot

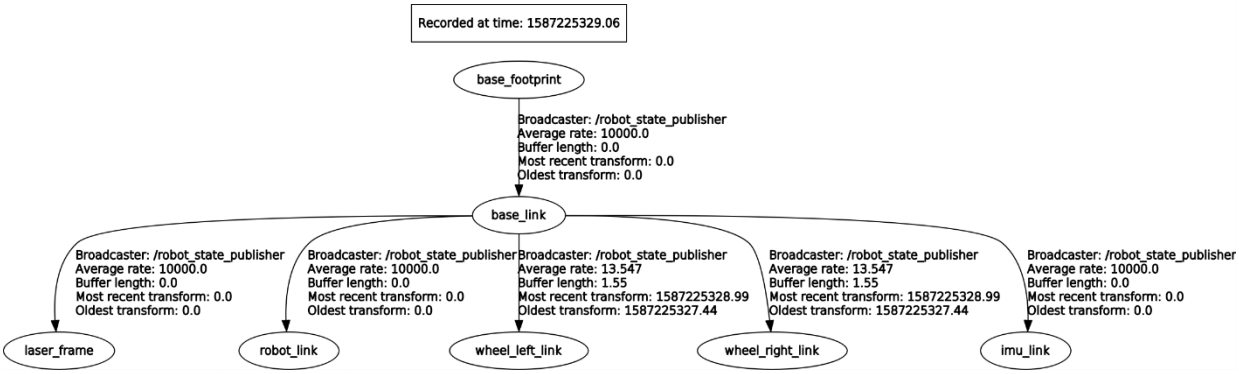
2.2. Thân Robot

2.3. Bánh xe Robot

2.4. LiDar



Danh sách các link và joint tương ứng Robot hiện tại:



STT	Tên Link/Joint	Chức năng
	base_footprint	Gốc tọa độ Robot
	base_link	Đế Robot
	robot_link	Thân Robot
	wheel_left_link	bánh trái
	wheel_right_link	bánh phải
	imu_link	IMU
	laser_frame	LiDar
	base_joint	Nối đế vào base_footprint

	robot_joint	Nối thân vào đế
	wheel_left_joint	Nối bánh trái vào đế
	wheel_right_joint	Nối bánh phải vào đế
	imu_joint	Nối IMU và đế
	scan_joint	Nối LiDar vào đế

3. Mô phỏng Robot

3.1. Mô hình hóa Robot với URDF

*** Phần mềm cần chuẩn bị:**

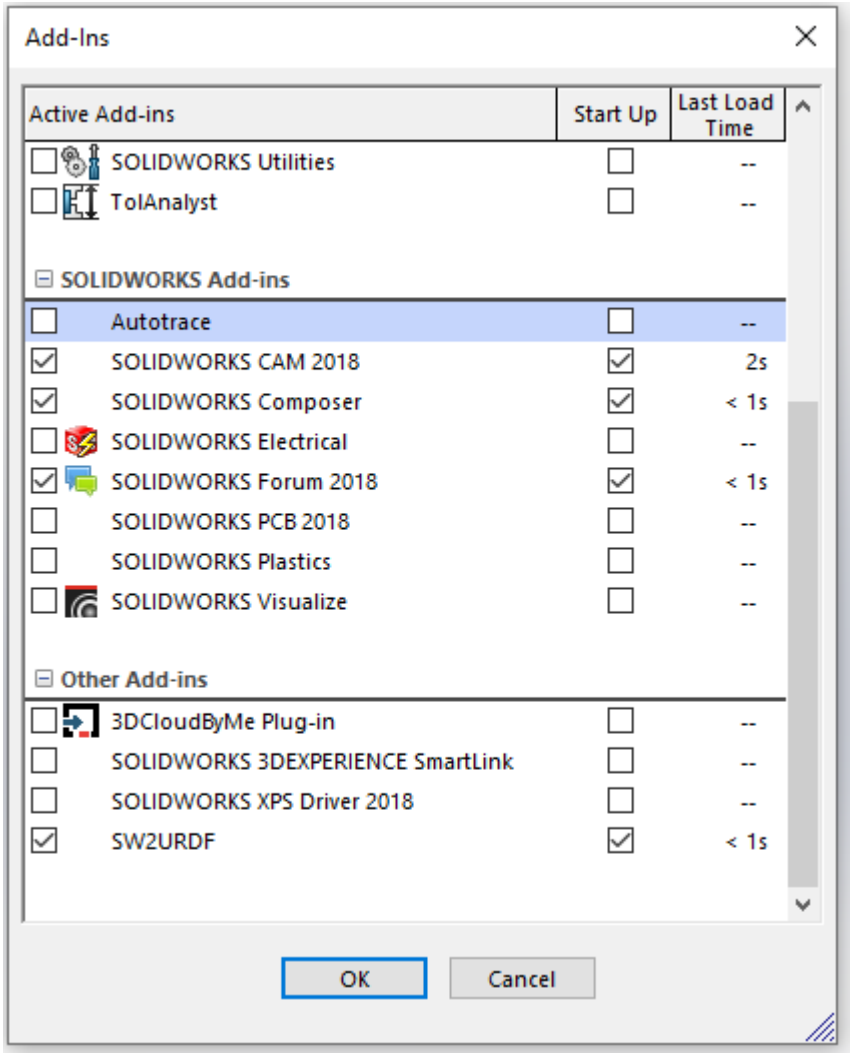
Solidworks (Tốt nhất từ 2018 SP5 trở lên, các phiên bản còn lại vẫn làm việc được nhưng có thể có lỗi).

Add-in SolidWorks to URDF exporter.

Link tải Add-in: https://github.com/ros/solidworks_urdf_exporter/releases

*** Cài đặt:**

- .Net Framework 4.7.2 trở lên
- Chạy installer của Add-in đã tải
- Mở Solidworks, Tools -> Add-in chúng ta sẽ thấy SW2URDF.

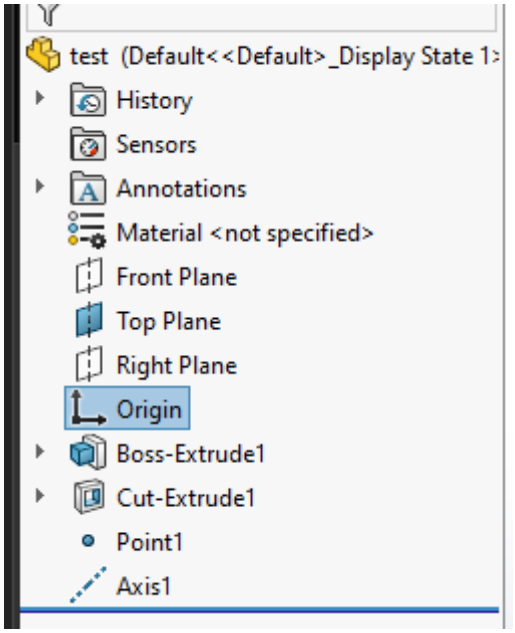


*** Note:**

Có thể sử dụng Visual Studio và C# để compile source code sau khi xuất. Đọc thêm tại: http://wiki.ros.org/sw_urdf_exporter

1. Tạo URDF với 1 chi tiết

Đối với 1 chi tiết, trước hết chúng ta nên vẽ tâm chi tiết dựa trên gốc tọa độ của Solidworks.

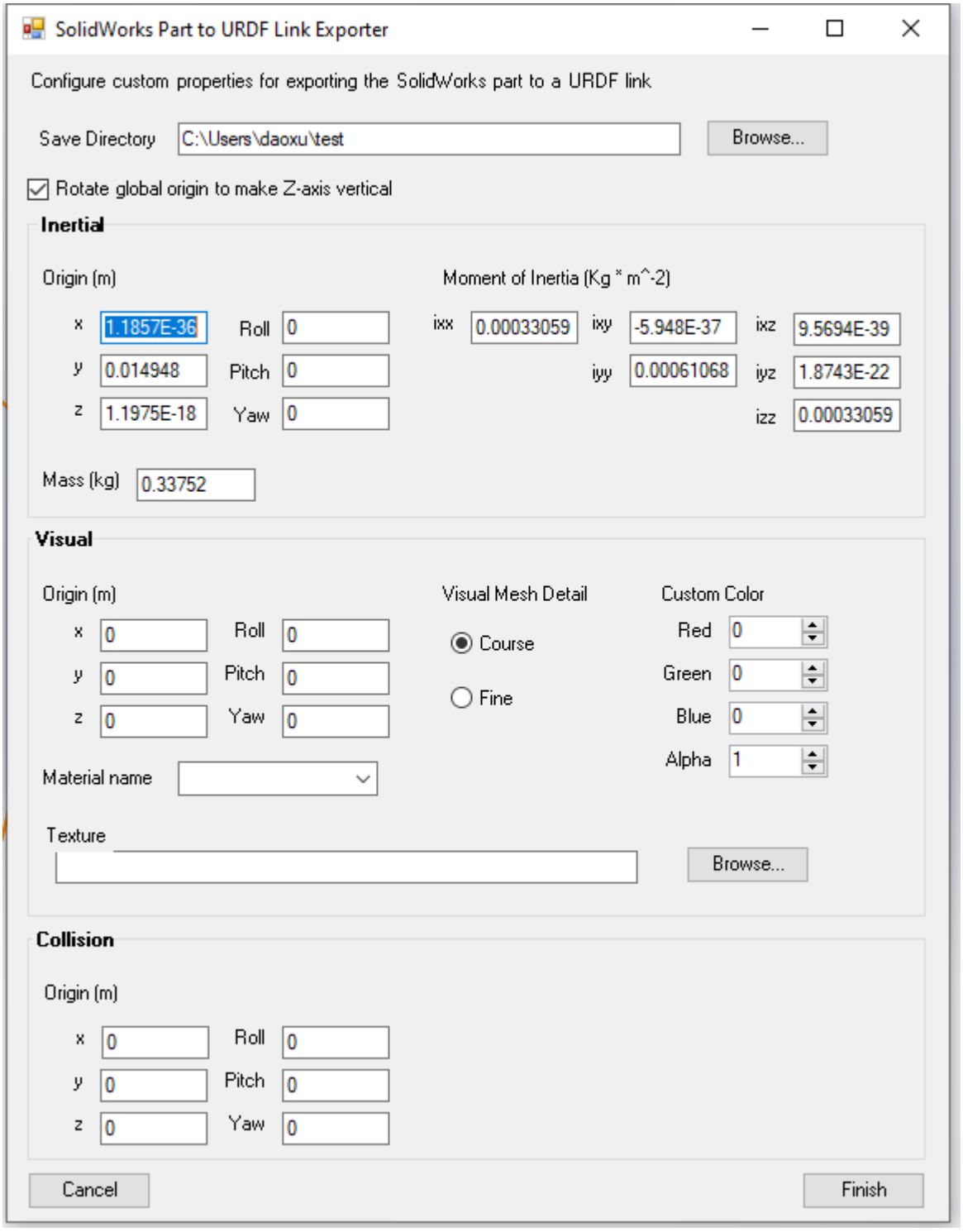


Như vậy khi Add-in tạo trục tọa độ nó có thể dễ dàng bắt vào tâm của tọa độ.

Các bước thực hiện:

- Mở chi tiết bằng Solidworks
- Vào File -> Export as URDF

Bảng chọn hiện ra như sau:

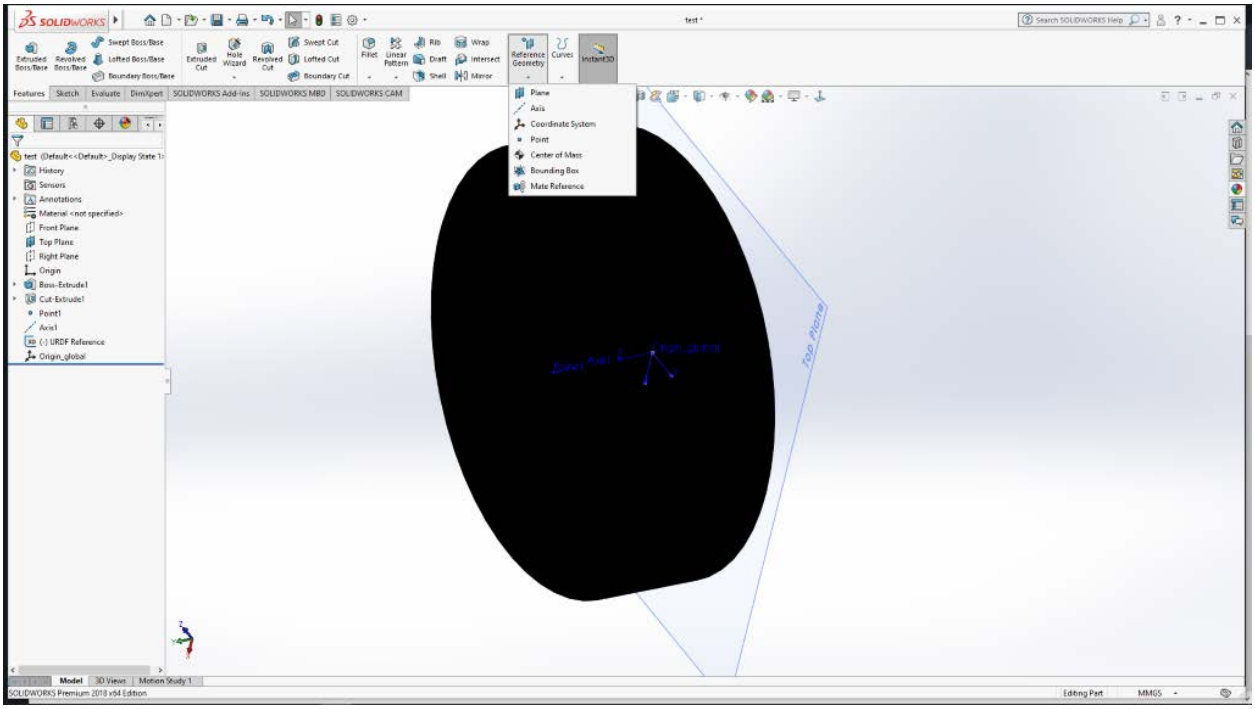


Chúng ta có thể chọn thư mục lưu, tick vào ô “Rotate global origin to make Z-axis vertical” để chi tiết khi xuất ra URDF sẽ tự xoay để trục Z trở thành phương thẳng đứng. (Solidworks thường sử dụng trục Y làm phương thẳng đứng).

- Nhấn Finish để kết thúc.

*** Tùy chỉnh trục tọa độ:**

Để tùy chỉnh trục tọa độ của chi tiết này chúng ta có thể tạo các reference point/axis trước đó ở mục reference geometry:

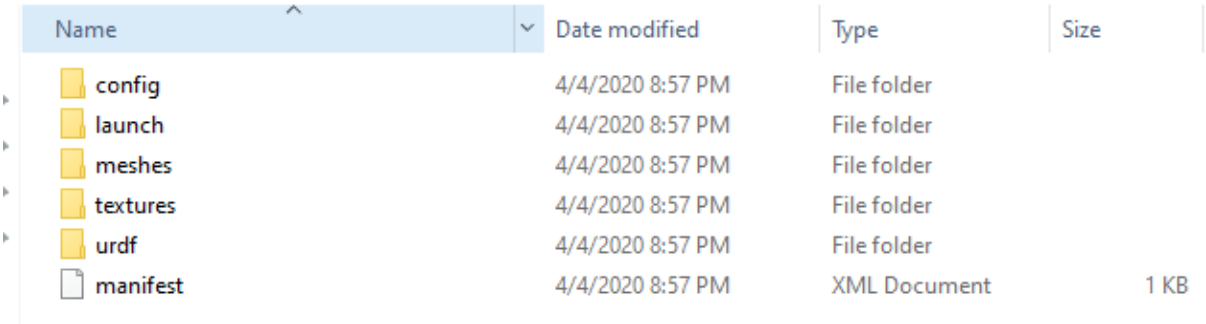


Sau đó **Edit feature** đối với Origin_Global để thay đổi gốc tọa độ:

Tại đây chúng ta có thể xóa các reference gốc mà Add-in tự tạo bằng gốc hoặc trục tọa độ mà chúng ta tự tạo.

Sau khi thực hiện xong, thực hiện lại thao tác Export as URDF để cập nhật trục tọa độ mới.

*** Cấu trúc thư mục được tạo ra:**



- Launch: Chứa launcher của Rviz và Gazebo
- Meshes: Chứa tệp STL 3D Model của chi tiết.
- Texture: Màu sắc/chất liệu đặc biệt cho chi tiết nếu có.
- Urdf: chứa tệp URDF

Chúng ta có thể sử dụng cả package này để đưa vào hoặc chỉ copy tệp STL và các đoạn code cần thiết trong URDF nếu có. URDF được tạo ra như dưới đây:

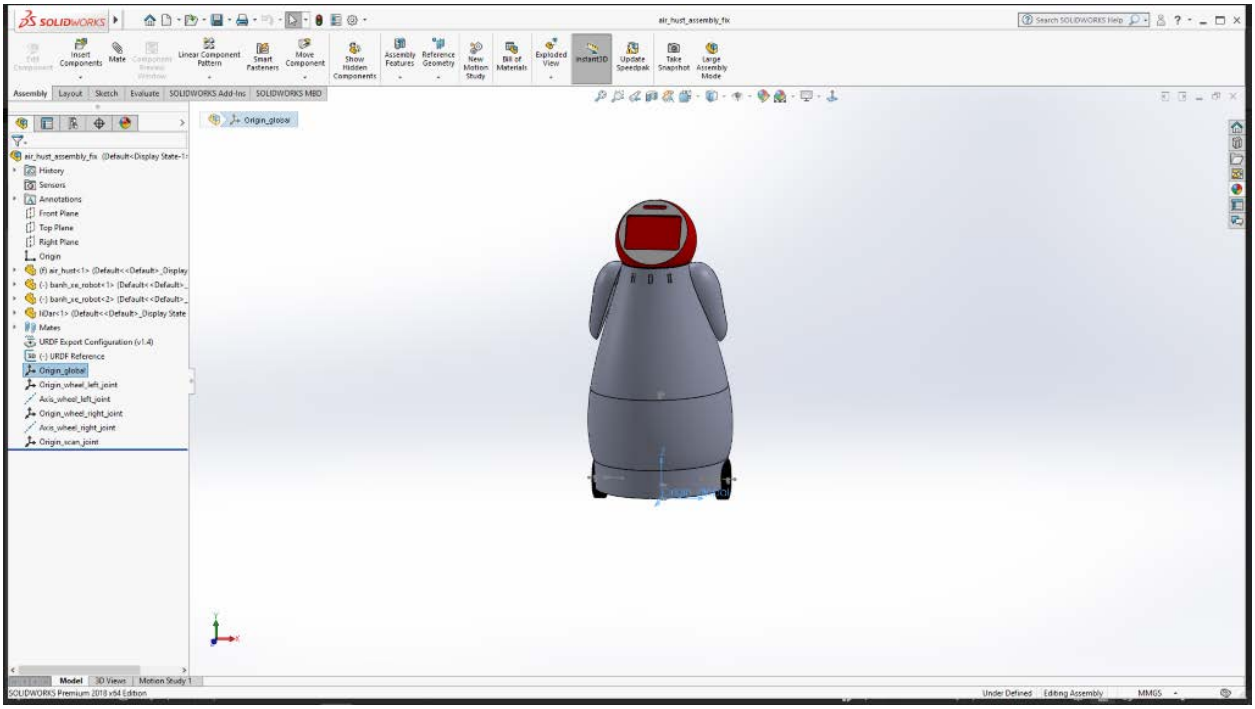
```
1. <?xml version="1.0" encoding="utf-8"?>
2. <!-- This URDF was automatically created by SolidWorks to URDF
   Exporter! Originally created by Stephen Brawner (brawner@gmail.com)
3.      Commit Version: 1.5.1-0-g916b5db  Build Version:
   1.5.7152.31018
4.      For more information, please see
   http://wiki.ros.org/sw_urdf_exporter -->
5. <robot
6.   name="test">
7.   <link
8.     name="test">
9.     <inertial>
10.        <origin
11.          xyz="1.1857E-36 -2.82229211153155E-19 0.014948"
12.          rpy="0 0 0" />
13.        <mass
14.          value="0.33752" />
15.        <inertia
16.          ixx="0.00033059"
17.          ixy="-9.56939999999996E-39"
18.          ixz="-5.948E-37"
19.          iyy="0.00033059"
20.          iyz="1.73374296821055E-20"
21.          izz="0.00061068" />
22.      </inertial>
23.      <visual>
24.        <origin
25.          xyz="0 0 0"
26.          rpy="1.5707963267949 0 0" />
```

```
      <geometry>
        <mesh
          filename="package://test/meshes/test.STL" />
      </geometry>
      <material
        name="">
        <color
          rgba="0 0 0 1" />
        <texture
          filename="package://test/textures/" />
      </material>
    </visual>
    <collision>
      <origin
        xyz="0 0 0"
        rpy="1.5707963267949 0 0" />
      <geometry>
        <mesh
          filename="package://test/meshes/test.STL" />
      </geometry>
    </collision>
  </link>
</robot>

```

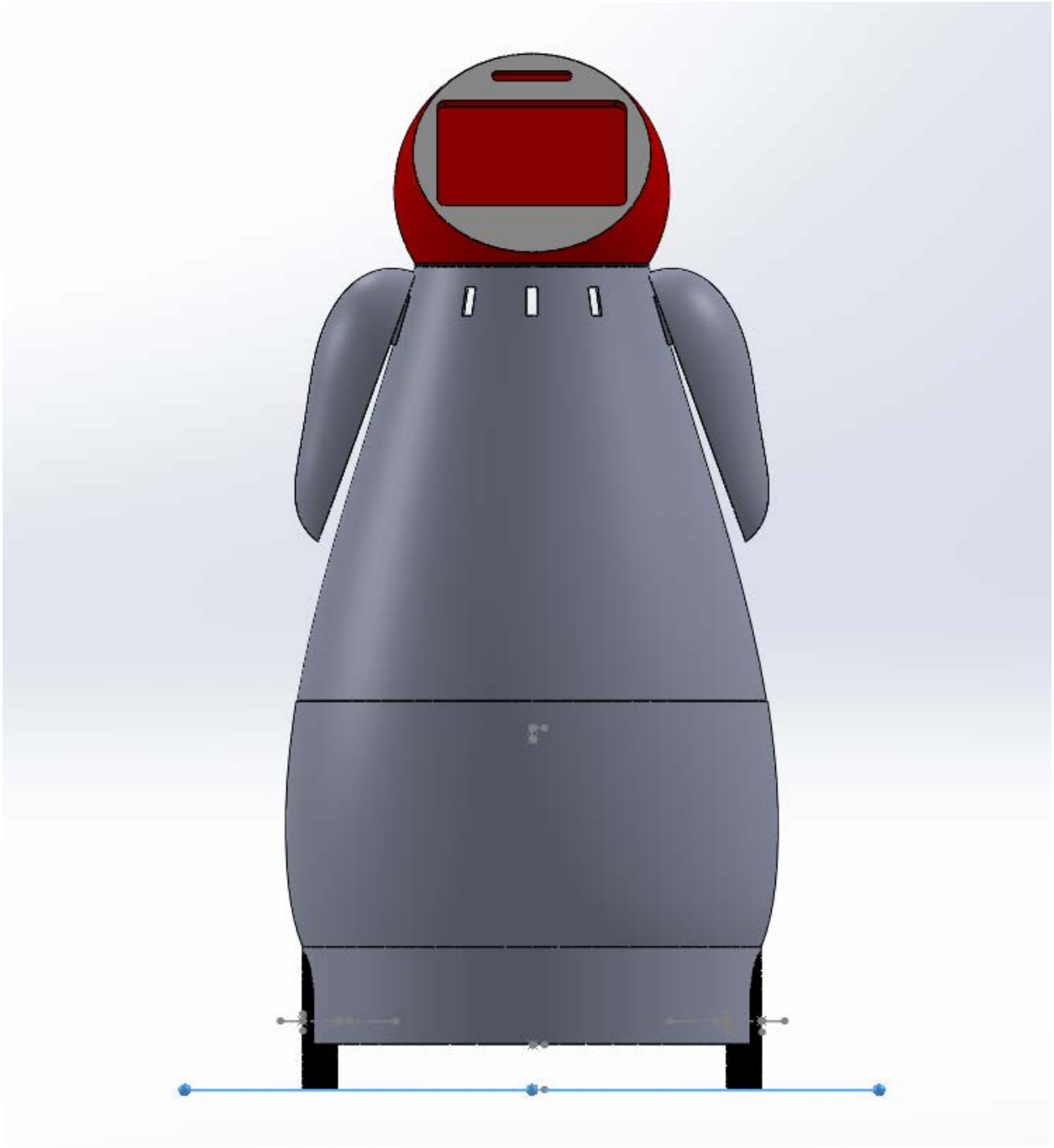
*** Đối với mô hình Robot:**

Đối với Robot, thường có nhiều hơn 1 chi tiết. Tuy nhiên chúng ta cần giảm thiểu tối đa số lượng chi tiết cần sử dụng, gộp các khối chi tiết vào một chi tiết. Cuối cùng ta có:

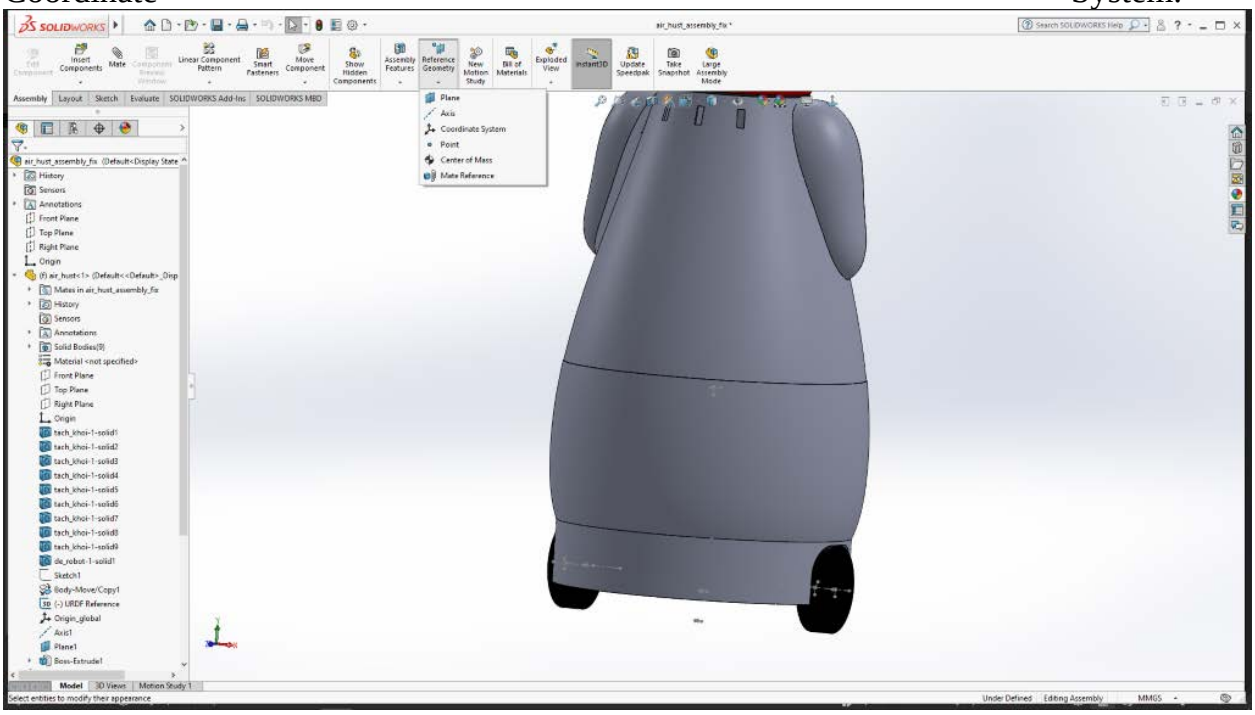
- Thân Robot: base_link
- 2 bánh xe: left/right_wheel_link
- LiDar: Laser Frame



Robot nên được đặt thẳng hàng gốc tọa độ của bản vẽ lắp theo trục Z, 2 bánh xe của Robot tiếp tuyến với mặt phẳng Top Plane (mặt phẳng này sẽ tương tự như sàn nhà).

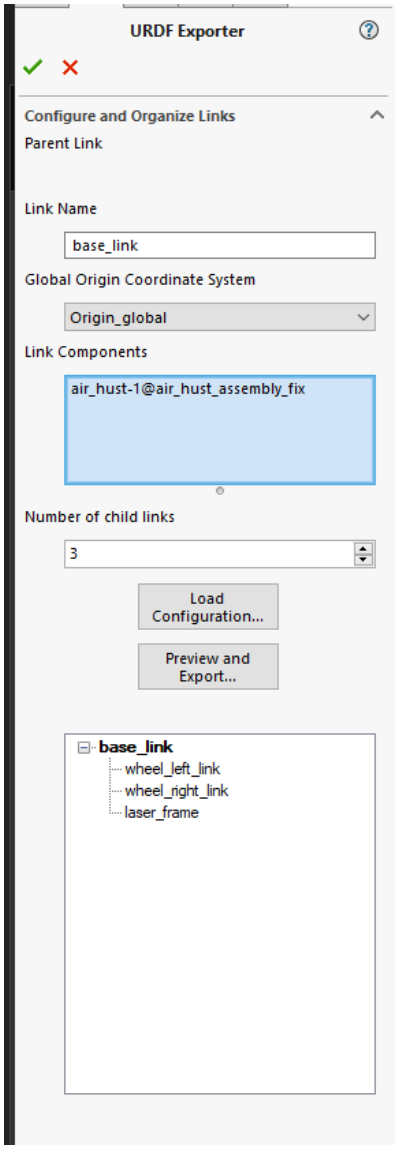


Chúng ta có thể tạo trước các trục tọa độ cần thiết bằng Reference Geometry -> Coordinate System.

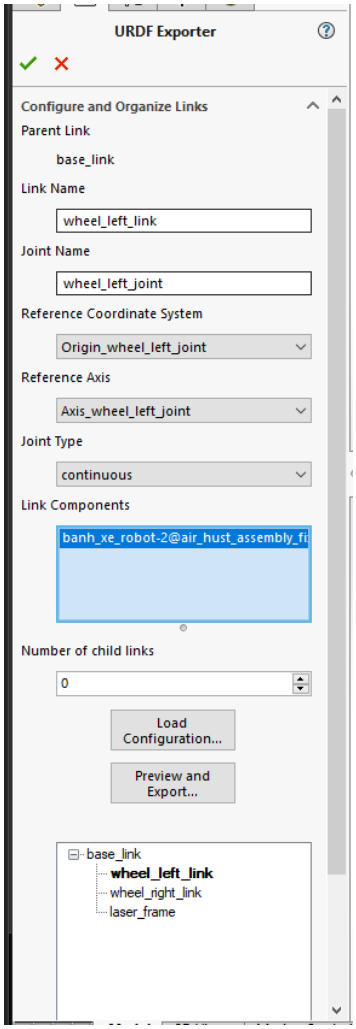


Với Robot, chúng ta sẽ có 1 hệ trục tọa độ gốc Global_origin, 2 hệ trục tọa độ của bánh xe (wheel_left/right_joint) và 2 trục quay của bánh xe, 1 hệ trục tọa độ của LiDar.

- Truy cập File -> Export as URDF

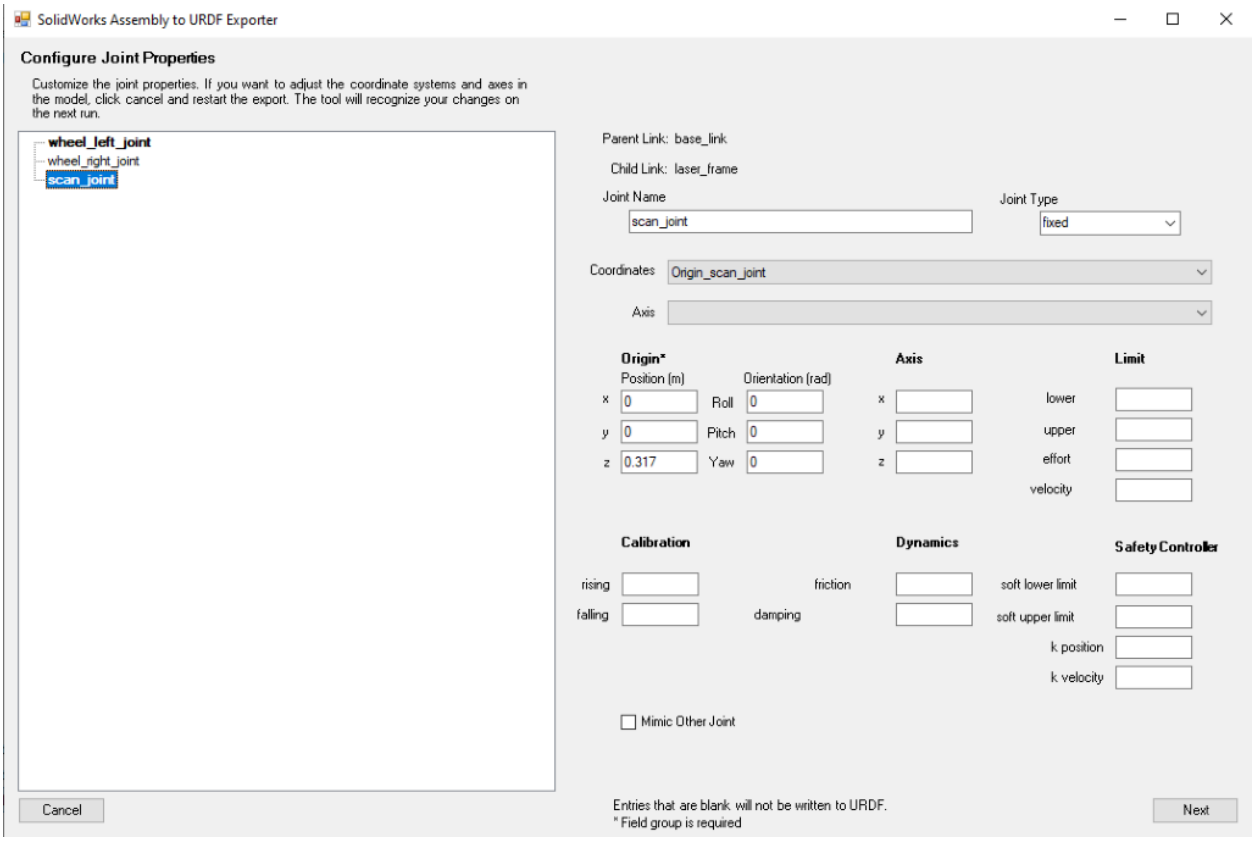


- Đặt tên link.
- Chọn hệ trục tọa độ.
- Chọn chi tiết tương ứng
- Điền số link con (Child links)
- Sau đó Click vào các link con để lựa chọn tương tự cho các link này và chọn kiểu kết nối cho joint. (fixed/continuous …)

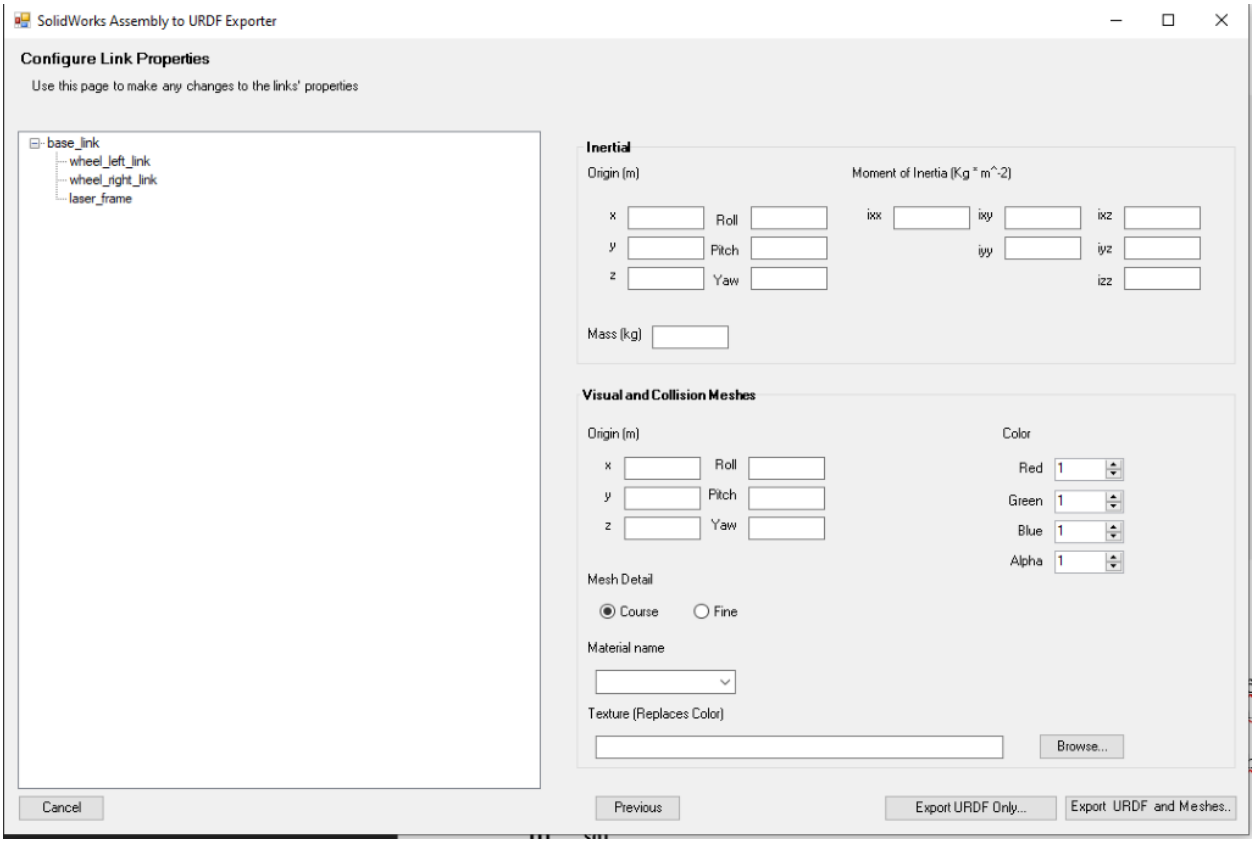


Chọn Preview and Export để xuất ra URDF.

- Đầu tiên là các Joint.



- Nhấn Next để kiểm tra các Link



• Chọn 2 option: Export URDF Only (Chỉ tệp URDF), Export URDF and Meshes (Kèm theo URDF là tệp STL 3D Model tương ứng của các chi tiết).

Thư mục tạo ra như sau:

Chúng ta có thể sử dụng luôn package này trở thành package description cho Robot.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This URDF was automatically created by SolidWorks to URDF Exporter! Originally created by Stephen Brawner (brawner@gmail.com)
     Commit Version: 1.5.1-0-g916b5db  Build Version: 1.5.7152.31018
     For more information, please see http://wiki.ros.org/sw_urdf_exporter -->
<robot
  name="air_hust_assembly_fix">
  <link
    name="base_link">
    <inertial>
      <origin
        xyz="-0.0013095577982326 -1.42252840205461E-07 0.21109485688867"
        rpy="0 0 0" />
      <mass
        value="26.9938115434281" />
      <inertia
        ixx="0.332984272931578"
        ixy="8.77326290539876E-08"
        ixz="0.00145514000464255"
        iyy="0.348533804196212"
        iyz="2.9605652900359E-07"
        izz="0.511517733519956" />
    </inertial>
    <visual>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://air_hust_assembly_fix/meshes/base_link.STL" />
      </geometry>
      <material
        name="">
        <color
          rgba="0.792156862745098 0.819607843137255 0.933333333333333 1" />
      </material>
    </visual>
    <collision>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
```

```
43.
   filename="package://air_hust_assembly_fix/meshes/base_link.STL" />
44.         </geometry>
45.       </collision>
46.     </link>
47.     <link
48.       name="wheel_left_link">
49.       <inertial>
50.         <origin
51.           xyz="-7.99314656839132E-18 2.77555756156289E-17
   0.0150523560209424"
52.           rpy="0 0 0" />
53.         <mass
54.           value="0.337524860720053" />
55.         <inertia
56.           ixx="0.000330592489924653"
57.           ixy="-8.77397368906768E-22"
58.           ixz="-6.70483305224477E-20"
59.           iyy="0.000330592489924653"
60.           iyz="5.14817284350815E-19"
61.           izz="0.000610675910880328" />
62.       </inertial>
63.       <visual>
64.         <origin
65.           xyz="0 0 0"
66.           rpy="0 0 0" />
67.         <geometry>
68.           <mesh
69.
   filename="package://air_hust_assembly_fix/meshes/wheel_left_link.ST
   L" />
70.         </geometry>
71.         <material
72.           name="">
73.           <color
74.             rgba="0 0 0 1" />
75.         </material>
76.       </visual>
77.       <collision>
78.         <origin
79.           xyz="0 0 0"
80.           rpy="0 0 0" />
81.         <geometry>
82.           <mesh
83.
   filename="package://air_hust_assembly_fix/meshes/wheel_left_link.ST
   L" />
84.         </geometry>
85.       </collision>
86.     </link>
87.     <joint
88.       name="wheel_left_joint"
89.       type="continuous">
90.       <origin
91.         xyz="0 0.17 0.06"
92.         rpy="-1.5708 0 0" />
93.       <parent
94.         link="base_link" />
95.       <child
```

```
96.       link="wheel_left_link" />
97.     <axis
98.       xyz="0 0 -1" />
99.   </joint>
100.  <link
101.    name="wheel_right_link">
102.    <inertial>
103.      <origin
104.        xyz="9.03360152944636E-18 -6.93889390390723E-18 -
    0.0150523560209426"
105.        rpy="0 0 0" />
106.      <mass
107.        value="0.337524860720053" />
108.      <inertia
109.        ixx="0.000330592489924653"
110.        ixy="1.00794714908295E-20"
111.        ixz="5.80328772926542E-22"
112.        iyy="0.000330592489924653"
113.        iyz="-1.75965083994712E-19"
114.        izz="0.000610675910880328" />
115.    </inertial>
116.    <visual>
117.      <origin
118.        xyz="0 0 0"
119.        rpy="0 0 0" />
120.      <geometry>
121.        <mesh
122.
    filename="package://air_hust_assembly_fix/meshes/wheel_right_link.S
    TL" />
123.      </geometry>
124.      <material
125.        name="">
126.        <color
127.          rgba="0 0 0 1" />
128.      </material>
129.    </visual>
130.    <collision>
131.      <origin
132.        xyz="0 0 0"
133.        rpy="0 0 0" />
134.      <geometry>
135.        <mesh
136.
    filename="package://air_hust_assembly_fix/meshes/wheel_right_link.S
    TL" />
137.      </geometry>
138.    </collision>
139.  </link>
140.  <joint
141.    name="wheel_right_joint"
142.    type="continuous">
143.    <origin
144.      xyz="0 -0.17 0.06"
145.      rpy="-1.5708 0 0" />
146.    <parent
147.      link="base_link" />
148.    <child
149.      link="wheel_right_link" />
```

```
    <axis
      xyz="0 0 1" />
  </joint>
  <link
    name="laser_frame">
    <inertial>
      <origin
        xyz="5.35723653090511E-35 -2.67936355302359E-19
  0.00982265813218092"
        rpy="0 0 0" />
      <mass
        value="0.0377610772290107" />
      <inertia
        ixx="5.31207031318074E-06"
        ixy="-7.08442480412977E-38"
        ixz="-1.63970877998851E-38"
        iyy="5.31207031318075E-06"
        iyz="1.6630178786121E-23"
        izz="8.11285579512851E-06" />
    </inertial>
    <visual>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
  filename="package://air_hust_assembly_fix/meshes/laser_frame.STL"
  />
      </geometry>
      <material
        name="">
        <color
          rgba="0.298039215686275 0.298039215686275
  0.298039215686275 1" />
      </material>
    </visual>
    <collision>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
  filename="package://air_hust_assembly_fix/meshes/laser_frame.STL"
  />
      </geometry>
    </collision>
  </link>
  <joint
    name="scan_joint"
    type="fixed">
    <origin
      xyz="0 0 0.317"
      rpy="0 0 0" />
    <parent
      link="base_link" />
    <child
      link="laser_frame" />
```

```
203.     <axis
204.       xyz="0 0 0" />
205.   </joint>
206. </robot>
```

Copy toàn package vừa rồi vào thư mục src workspace. (~/dashgo_ws/src)

- Mở terminal:
- $ cd ~/dashgo_ws
- $ catkin_make
- Hiển thị trên Rviz:
- $ roslaunch urdf_tutorial display.launch model:='$(find air_hust_assembly_fix)/urdf/air_hust_assembly_fix.urdf'
- Bây giờ chúng ta có thể sử dụng nó làm description cho Robot.

*** Hệ thống hóa bằng các tệp xacro:**

Do tệp mà chúng ta tạo ra khá dài, khó hiểu, chúng ta nên hệ thống hóa lại bằng các tệp xacro.

- dashgo.urdf.xacro
- common.properties.xacro
- dashgo.gazebo.xacro

Trong đó:

- dashgo.urdf.xacro: Là tệp tin chính chỉ chứa link, joint và các tọa độ, các thẻ cần thiết.
- common.properties.xacro: Chứa Mã màu của các vật liệu của Robot
- dashgo.gazebo.xacro: Chứa các thẻ plugin tương ứng trên gazebo.

common.properties.xacro và dashgo.gazebo.xacro được include vào tệp tin chính nhờ lệnh sau:

```
1. <xacro:include filename="$(find
   dashgo_description)/urdf/dashgobase/common_properties.xacro"/>
2. <xacro:include filename="$(find
   dashgo_description)/urdf/dashgobase/dashgo.gazebo.xacro"/>
```

Tệp dashgo.urdf.xacro:

```
1.  <?xml version="1.0" ?>
2. <robot name="dashgobot" xmlns:xacro="http://ros.org/wiki/xacro">
3.   <xacro:include filename="$(find
   dashgo_description)/urdf/dashgobase/common_properties.xacro"/>
4.   <xacro:include filename="$(find
   dashgo_description)/urdf/dashgobase/dashgo.gazebo.xacro"/>
5.
6.   <link name="base_footprint"/>
7.
8.   <joint name="base_joint" type="fixed">
9.     <parent link="base_footprint"/>
```

```
      <child link="base_link"/>
      <origin xyz="0.0 0.0 0" rpy="0 0 0"/>
    </joint>

    <link name="base_link">
      <visual>
        <origin xyz="0 0 0.04" rpy="0 0 0"/>
        <geometry>
          <mesh
   filename="package://dashgo_description/urdf/dashgobase/meshes/bases
   /base_link.STL"/>
        </geometry>
        <material name="white"/>
      </visual>

      <collision>
        <origin
          xyz="0 0 0"
          rpy="0 0 0" />
        <geometry>
          <mesh

   filename="package://dashgo_description/urdf/dashgobase/meshes/bases
   /base_link.STL" />
        </geometry>
      </collision>

      <inertial>
        <origin
          xyz="-5.3591E-07 -4.6799E-09 0.097992"
          rpy="0 0 0" />
        <mass
          value="22.293" />
        <inertia
          ixx="0.31906"
          ixy="7.5186E-09"
          ixz="1.6457E-06"
          iyy="0.3349"
          iyz="4.7391E-09"
          izz="0.45765" />
      </inertial>
    </link>

    <joint name="robot_joint" type="fixed">
      <parent link="base_link"/>
      <child link="robot_link"/>
      <origin xyz="0.0 0.0 0.041" rpy="0 0 0"/>
    </joint>

   <link name="robot_link">
      <visual>
        <origin xyz="0 0 0.0" rpy="0 0 0"/>
        <geometry>
          <mesh
   filename="package://dashgo_description/urdf/dashgobase/meshes/bases
   /robot_link.STL"/>
        </geometry>
        <material name="white"/>
      </visual>
```

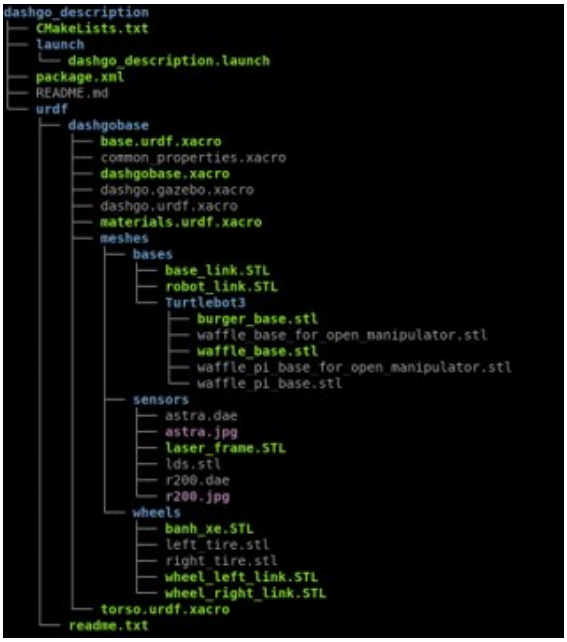
```
63.
64.      <inertial>
65.        <origin
66.          xyz="-0.008188 -8.839E-07 0.27619"
67.          rpy="0 0 0" />
68.        <mass
69.          value="4.3162" />
70.        <inertia
71.          ixx="0.047961"
72.          ixy="8.0458E-08"
73.          ixz="0.001459"
74.          iyy="0.04767"
75.          iyz="2.9481E-07"
76.          izz="0.046024" />
77.      </inertial>
78.
79.      <collision>
80.        <origin
81.          xyz="0 0 0"
82.          rpy="0 0 0" />
83.        <geometry>
84.          <mesh
   filename="package://dashgo_description/urdf/dashgobase/meshes/bases
   /robot_link.STL"/>
85.        </geometry>
86.      </collision>
87.
88.    </link>
89.
90.    <joint name="wheel_left_joint" type="continuous">
91.      <parent link="base_link"/>
92.      <child link="wheel_left_link"/>
93.      <origin xyz="0.0 0.185 0.06" rpy="-1.57 0 0"/>
94.      <axis xyz="0 0 1"/>
95.    </joint>
96.
97.    <link name="wheel_left_link">
98.      <visual>
99.        <origin xyz="-0.06 -0.06 0.015" rpy="-1.57 0 0"/>
100.       <geometry>
101.         <mesh
   filename="package://dashgo_description/urdf/dashgobase/meshes/wheel
   s/banh_xe.STL" scale="0.001 0.001 0.001"/>
102.       </geometry>
103.       <material name="dark"/>
104.     </visual>
105.
106.     <collision>
107.       <origin xyz="0 0 0" rpy="0 0 0"/>
108.       <geometry>
109.         <cylinder length="0.036" radius="0.066"/>
110.       </geometry>
111.     </collision>
112.
113.     <inertial>
114.       <origin xyz="0 0 0" />
115.       <mass value="2.8498940e-02" />
116.       <inertia ixx="1.1175580e-05" ixy="-4.2369783e-11" ixz="-
   5.9381719e-09"
```

```
               iyy="1.1192413e-05" iyz="-1.4400107e-11"
               izz="2.0712558e-05" />
      </inertial>
  </link>

  <joint name="wheel_right_joint" type="continuous">
    <parent link="base_link"/>
    <child link="wheel_right_link"/>
    <origin xyz="0.0 -0.185 0.06" rpy="-1.57 0 0"/>
    <axis xyz="0 0 1"/>
  </joint>

  <link name="wheel_right_link">
    <visual>
      <origin xyz="-0.06 0.06 -0.015" rpy="1.57 0 0"/>
      <geometry>
        <mesh
  filename="package://dashgo_description/urdf/dashgobase/meshes/wheel
  s/banh_xe.STL" scale="0.001 0.001 0.001"/>
      </geometry>
      <material name="dark"/>
    </visual>

    <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <cylinder length="0.036" radius="0.066"/>
      </geometry>
    </collision>

    <inertial>
      <origin xyz="0 0 0" />
      <mass value="2.8498940e-02" />
      <inertia ixx="1.1175580e-05" ixy="-4.2369783e-11" ixz="-
  5.9381719e-09"
               iyy="1.1192413e-05" iyz="-1.4400107e-11"
               izz="2.0712558e-05" />
      </inertial>
  </link>

  <joint name="caster_back_joint" type="fixed">
    <parent link="base_link"/>
    <child link="caster_back_link"/>
    <origin xyz="-0.162 0.0 0.025" rpy="-1.57 0 0"/>
  </joint>

  <link name="caster_back_link">
    <visual>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <cylinder length="0.02" radius="0.02"/>
      </geometry>
      <material name="orange"/>
    </visual>
    <collision>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
          <cylinder length="0.02" radius="0.02"/>
        </geometry>
```

```
    </collision>
    <inertial>
      <origin xyz="0 0 0" />
      <mass value="0.005" />
      <inertia ixx="0.001" ixy="0.0" ixz="0.0"
               iyy="0.001" iyz="0.0"
               izz="0.001" />
    </inertial>
  </link>

  <joint name="caster_front_joint" type="fixed">
    <parent link="base_link"/>
    <child link="caster_front_link"/>
    <origin xyz="0.162 0.0 0.025" rpy="-1.57 0 0"/>
  </joint>

  <link name="caster_front_link">
    <visual>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <cylinder length="0.02" radius="0.02"/>
      </geometry>
      <material name="orange"/>
    </visual>

    <collision>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
          <cylinder length="0.02" radius="0.02"/>
        </geometry>
    </collision>

    <inertial>
      <origin xyz="0 0 0" />
      <mass value="0.005" />
      <inertia ixx="0.001" ixy="0.0" ixz="0.0"
               iyy="0.001" iyz="0.0"
               izz="0.001" />
    </inertial>
  </link>

  <joint name="imu_joint" type="fixed">
    <parent link="base_link"/>
    <child link="imu_link"/>
    <origin xyz="0.0 0 0.80" rpy="0 0 0"/>
  </joint>

  <link name="imu_link"/>

  <joint name="scan_joint" type="fixed">
    <parent link="base_link"/>
    <child link="laser_frame"/>
    <origin xyz="0.0 0 0.317" rpy="0 0 0"/>
  </joint>

  <link name="laser_frame">
    <visual>
      <origin xyz="0 0 0.0" rpy="0 0 0"/>
      <geometry>
```

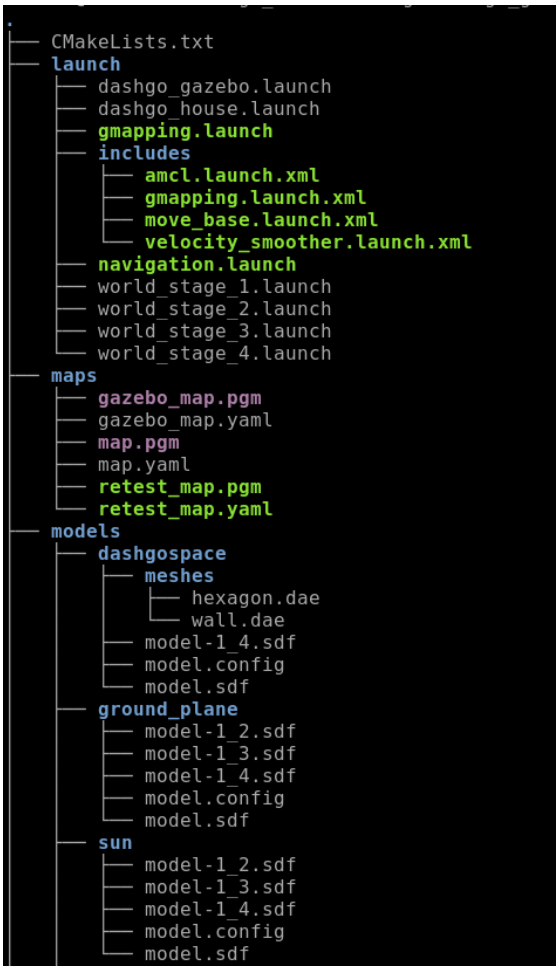
```
232.         <mesh
   filename="package://dashgo_description/urdf/dashgobase/meshes/senso
   rs/laser_frame.STL"/>
233.       </geometry>
234.       <material name="dark"/>
235.     </visual>
236.
237.     <collision>
238.       <origin xyz="0 0 0" rpy="0 0 0"/>
239.       <geometry>
240.         <cylinder length="0.063" radius="0.055"/>
241.       </geometry>
242.     </collision>
243.
244.     <inertial>
245.       <mass value="0.114" />
246.       <origin xyz="0 0 0" />
247.       <inertia ixx="0.001" ixy="0.0" ixz="0.0"
248.                iyy="0.001" iyz="0.0"
249.                izz="0.001" />
250.     </inertial>
251.   </link>
252.
253.   <joint name="sonar_joint" type="fixed">
254.     <origin xyz="0.2 0 0.125" rpy="0 0 0"/>
255.     <parent link="base_link" />
256.     <child link="sonar_link" />
257. </joint>
258.
259. <link name="sonar_link" />
260.
261. <!-- this is only a hack until gazebo fixes their SonarSensor
   which points towards -z per default (incompatible with rviz which
   points towards x) -->
262.
263. <joint name="sonar1_joint_fake" type="fixed">
264.     <origin rpy="0 ${-pi/2} 0"/>
265.     <parent link="sonar_link" />
266.     <child link="sonar_link_fake" />
267. </joint>
268.
269. <link name="sonar_link_fake"/>
270. </robot>
```

Từ đó chúng ta có cấu trúc thư mục như hình dưới:

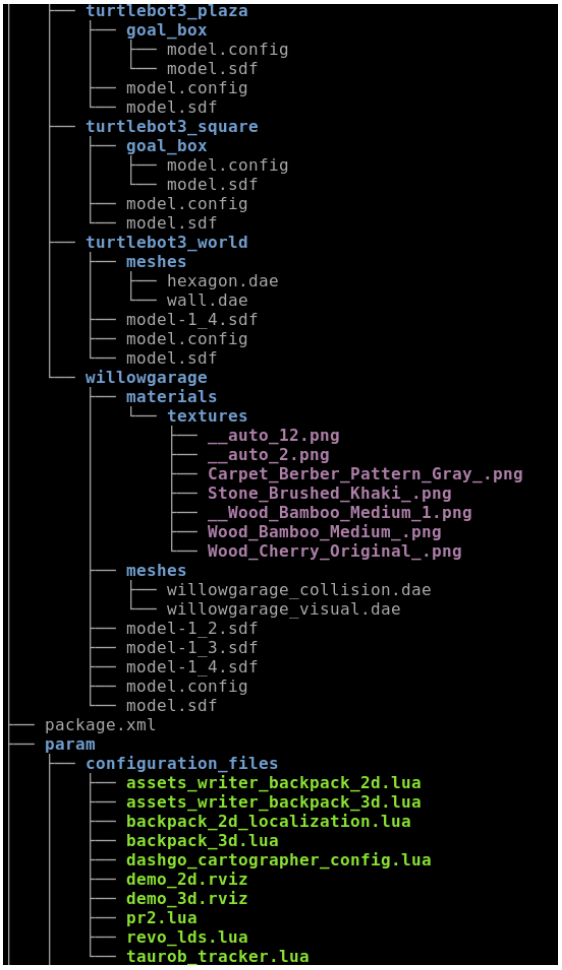


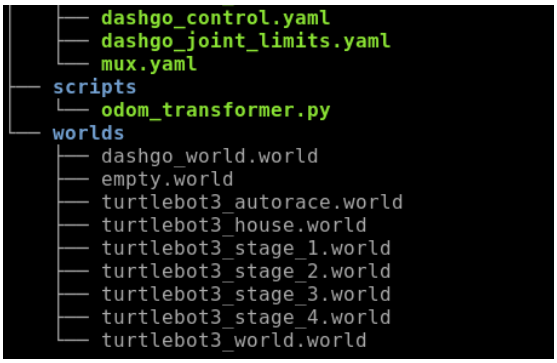
3.2. Mô phỏng với Gazebo

* Cấu trúc thư mục dashgo_gazebo:




turtlebot3_autorace
course
materials
scripts
course.material
textures
course.png
model.config
model.sdf
traffic_bar_down
materials
scripts
traffic_bar.material
textures
traffic_bar.png
model.config
model.sdf
traffic_bar_up
materials
scripts
traffic_bar.material
textures
traffic_bar.png
model.config
model.sdf
traffic_light_green
model.config
model.sdf
traffic_light_red
model.config
model.sdf
traffic_light_yellow
model.config
model.sdf
traffic_parking
materials
scripts
traffic_parking.material
textures
traffic_parking.png
model.config
model.sdf
traffic_stop
materials
scripts
traffic_stop.material
textures
traffic_stop.png
model.config
model.sdf





- Gói dashgo_gazebo là gói vận hành Robot
- Cấu trúc thư mục:
  - launch: 3 launcher chính (gmapping, navigation và world (dashgo_gazebo)
    - include (các thông số và gói chạy kèm được mô tả bằng tệp xml
  - maps: lưu bản đồ mặc định
  - models: Model sử dụng
  - param: Các thông số cần thiết
  - scripts: Chứa odom_transformer thay thế cho tf_broadcaster
  - words: Chứa world của Robot

3.2.1. Tạo môi trường với Gazebo:

*** Các khái niệm cơ bản:**

- **World:** Tập hợp gồm Robot và các đối tượng, môi trường xung quanh Robot (Nhà cửa, bàn, ghế, đèn điện,...) và các đối tượng tự nhiên như bầu trời, ánh sáng và các tính chất vật lý.
- **Static:** Các thành phần cố định (Gắn trong thẻ <static>true</static> trong SDF). Tất cả các đối tượng không di chuyển sẽ đượng gắn nhãn static.
- **Dynamic:** Các thành phần động học (không kèm theo thẻ <static> hoặc có giá trị false trong thẻ), là các đối tượng có quán tính và va chạm vật lý.

*** Xây dựng World:**

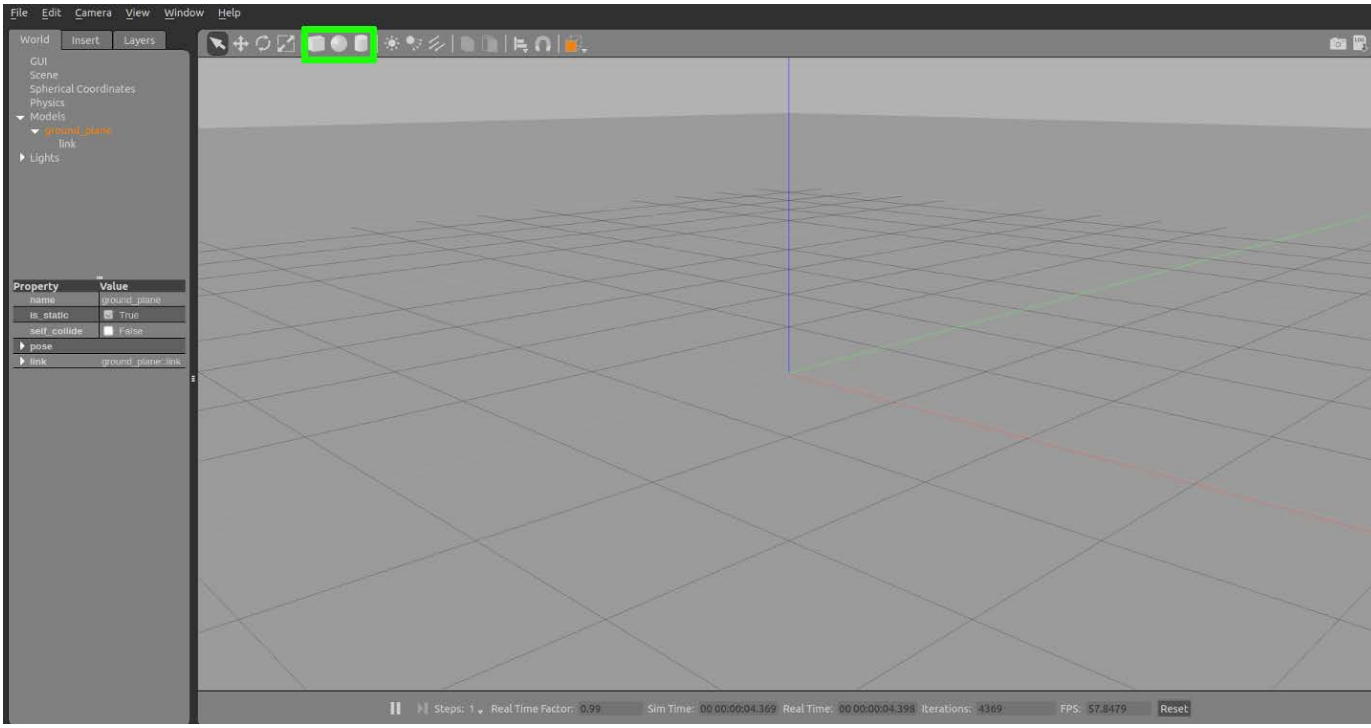
Bước 1: Cài đặt Gazebo

Bước 2: Khởi chạy Gazebo với lệnh:

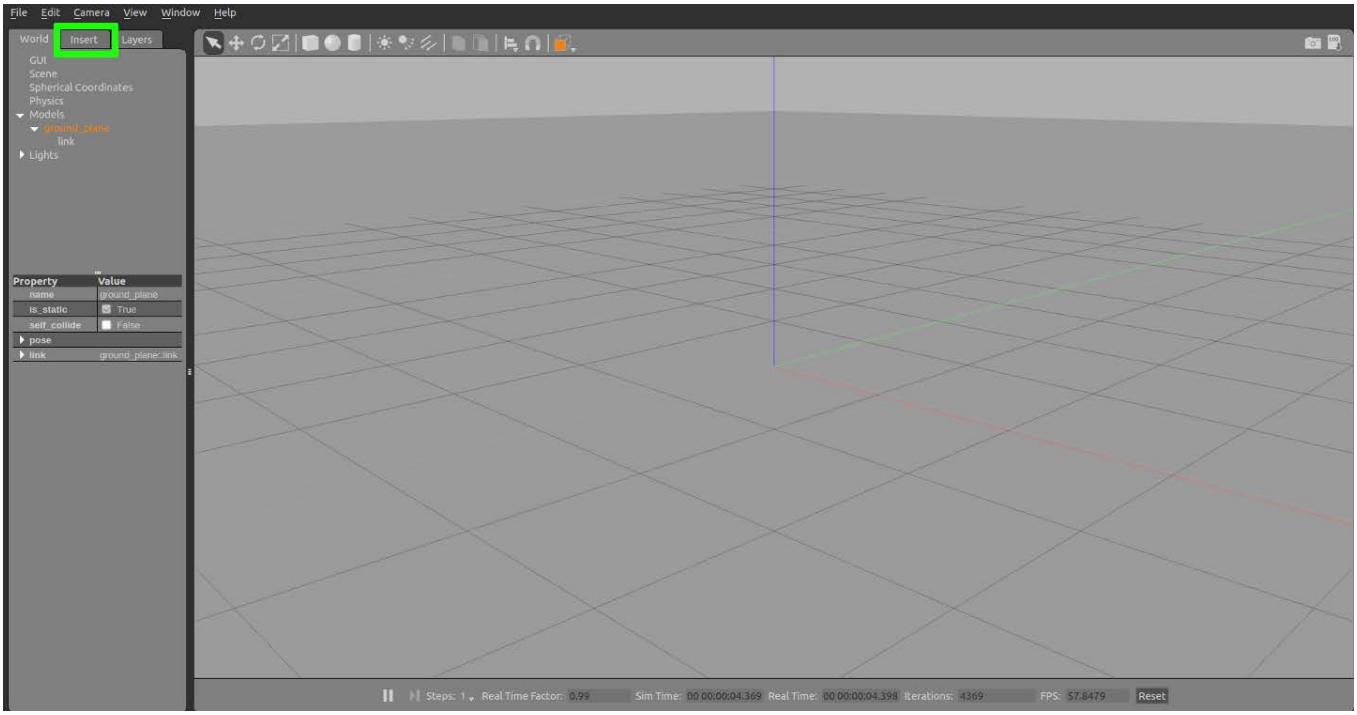
```
1. $ gazebo
```

Bước 3: Thêm các đối tượng:

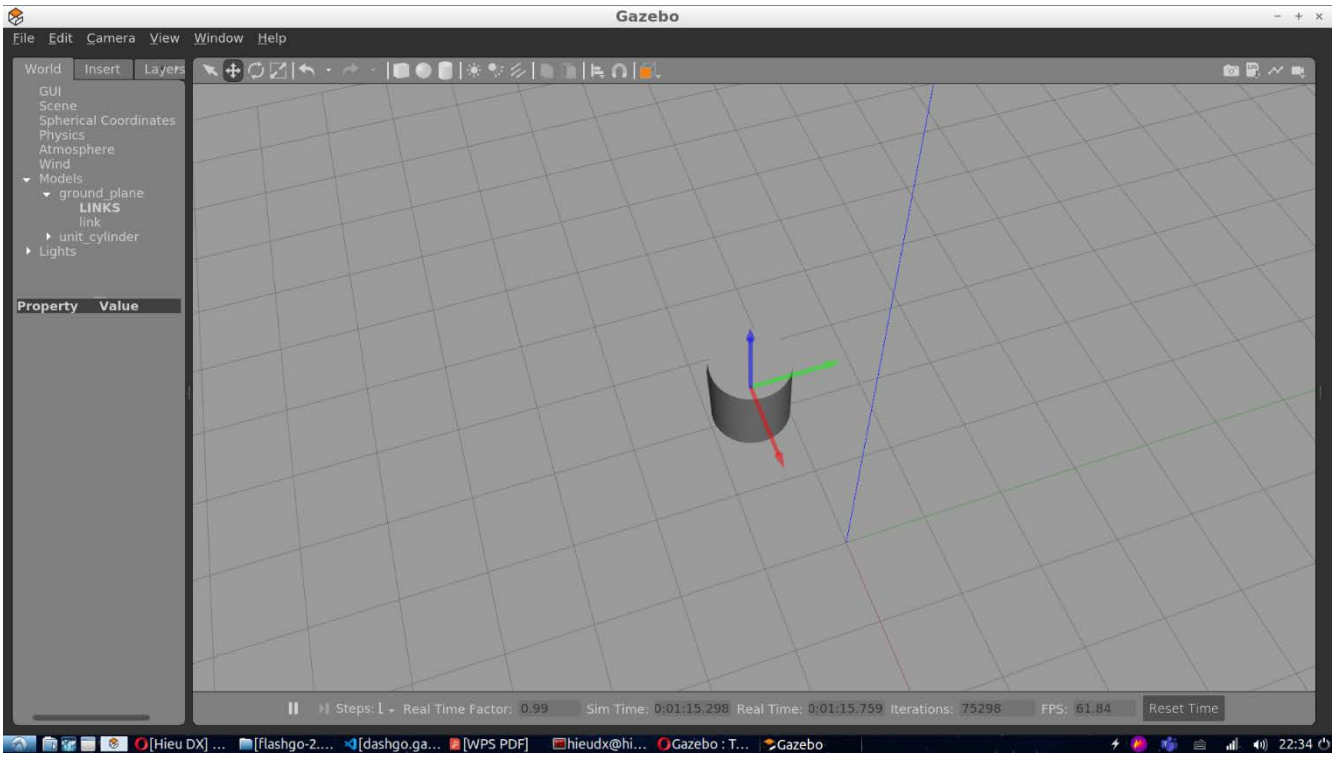
- Các hình khối cơ bản:



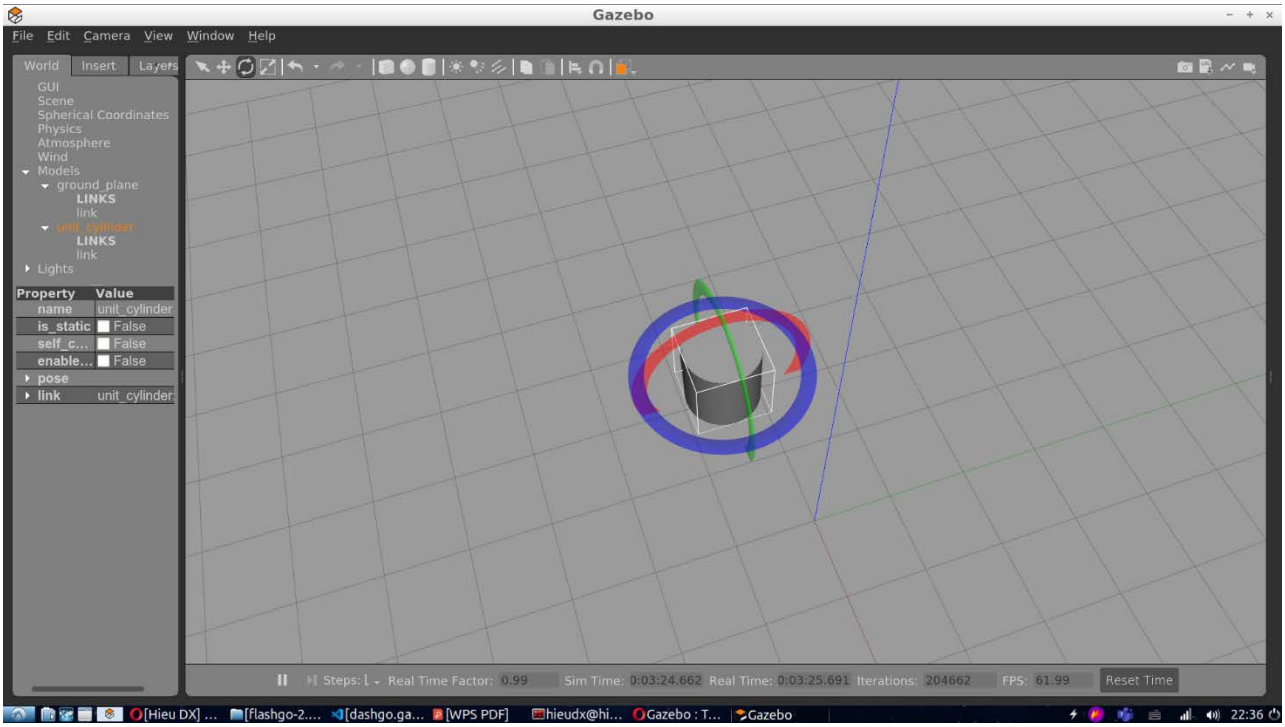
- Các model sẵn có với database:



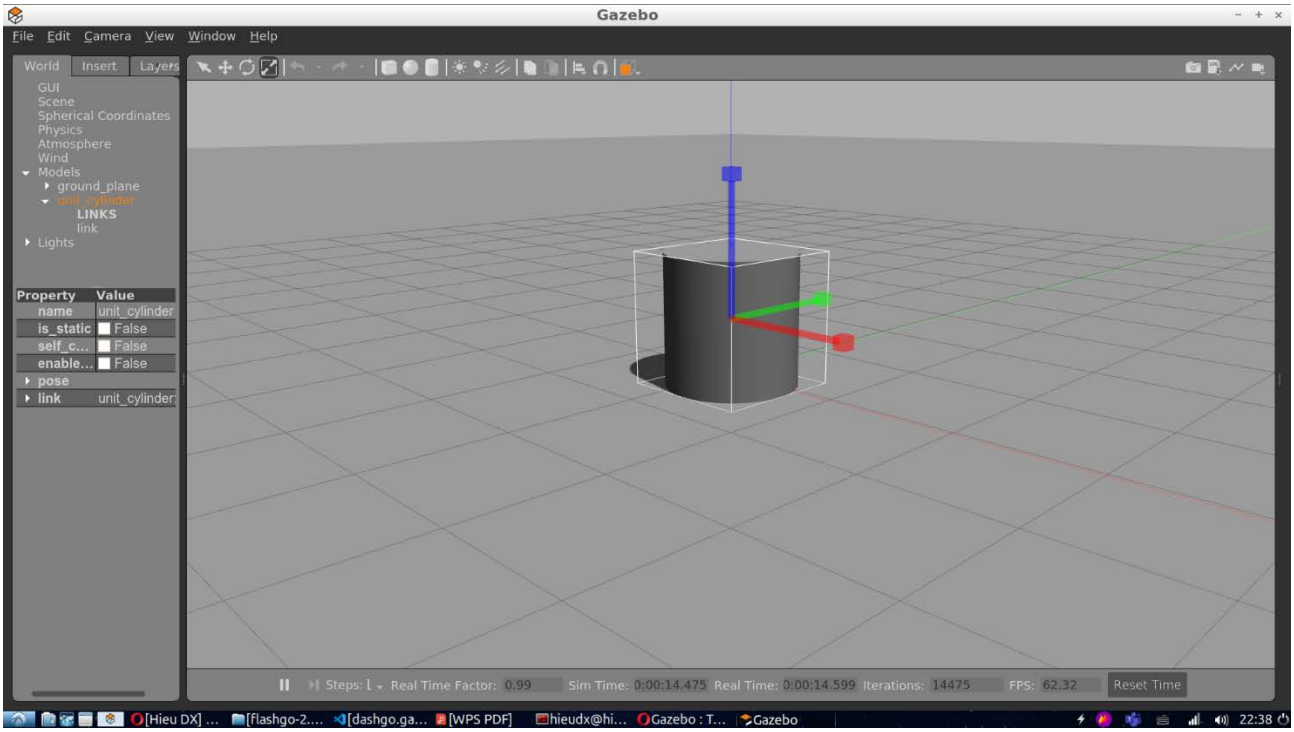
- Dịch chuyển theo 3 trục x, y, z:



- Xoay theo 3 trục x, y, z:



- Lấy tỉ lệ:



- Xóa đối tượng: Chọn đối tượng và nhấn delete.
- Lưu world:
  - File/Save World As -> lưu tên tệp.
- Mở 1 world sẵn có:

Gõ lệnh trên terminal với đường dẫn tới tệp world sau gazebo:

$ gazebo ~/dashgo_ws/src/dashgo/dashgo_gazebo/worlds/dashgo_world.world

*** Tạo launch để khởi chạy các world:**

```
1. <launch>
2.   <param name="use_sim_time" value="true"/>
3.   <arg name="x_pos" default="-2.0"/>
4.   <arg name="y_pos" default="-0.5"/>
5.   <arg name="z_pos" default="0.0"/>
6.   <include file="$(find gazebo_ros)/launch/empty_world.launch">
7.     <arg name="world_name" value="$(find
   dashgo_gazebo)/worlds/dashgo_world.world"/>
8.     <arg name="paused" value="false"/>
9.     <arg name="use_sim_time" value="true"/>
10.      <arg name="gui" value="true"/>
11.      <arg name="headless" value="false"/>
12.      <arg name="debug" value="false"/>
13.    </include>
14.
15.    <node pkg="nodelet" type="nodelet"
   name="mobile_base_nodelet_manager" args="manager"/>
16.    <node pkg="nodelet" type="nodelet" name="cmd_vel_mux"
17.          args="load yocs_cmd_vel_mux/CmdVelMuxNodelet
   mobile_base_nodelet_manager">
18.      <param name="yaml_cfg_file" value="$(find
   dashgo_gazebo)/param/mux.yaml" />
19.      <remap from="cmd_vel_mux/output"
   to="mobile_base/commands/velocity"/>
20.    </node>
21.
22.    <node name="robot_state_publisher" pkg="robot_state_publisher"
   type="robot_state_publisher"
23.      respawn="false" output="screen">
24.      <param name="publish_frequency" value="50.0" />
25.      <param name="ignore_timestamp" value="true" />
26.      <!-- <remap from="/joint_states" to="/dashgo/joint_states" />
   -->
27.    </node>
28.
29.    <param name="robot_description" command="$(find xacro)/xacro --
   inorder $(find
   dashgo_description)/urdf/dashgobase/dashgo.urdf.xacro" />
30.
31.    <node pkg="gazebo_ros" type="spawn_model" name="spawn_urdf"
   args="-urdf -model dashgobase -x $(arg x_pos) -y $(arg y_pos) -z
   $(arg z_pos) -param robot_description" />
32.  </launch>
```

Với:

```
1.   <param name="use_sim_time" value="true"/>
2.   <arg name="x_pos" default="-2.0"/>
3.   <arg name="y_pos" default="-0.5"/>
4.   <arg name="z_pos" default="0.0"/>
5.   <include file="$(find gazebo_ros)/launch/empty_world.launch">
6.     <arg name="world_name" value="$(find
   dashgo_gazebo)/worlds/dashgo_world.world"/>
```

```
7.      <arg name="paused" value="false"/>
8.      <arg name="use_sim_time" value="true"/>
9.      <arg name="gui" value="true"/>
10.       <arg name="headless" value="false"/>
11.       <arg name="debug" value="false"/>
12.     </include>
```

Là vị trí mặc định khi khởi động giả lập Robot và liên kết với world đã tạo sẵn.

Để cập nhật trạng thái của các joint trong Robot, liên kết với description và URDF.

```
1. <node name="robot_state_publisher" pkg="robot_state_publisher"
   type="robot_state_publisher"
2.      respawn="false" output="screen">
3.      <param name="publish_frequency" value="50.0" />
4.      <param name="ignore_timestamp" value="true" />
5.      <!-- <remap from="/joint_states" to="/dashgo/joint_states" /> -
   ->
6.    </node>
7.
8.    <param name="robot_description" command="$(find xacro)/xacro --
   inorder $(find
   dashgo_description)/urdf/dashgobase/dashgo.urdf.xacro" />
9.
10.     <node pkg="gazebo_ros" type="spawn_model" name="spawn_urdf"
   args="-urdf -model dashgobase -x $(arg x_pos) -y $(arg y_pos) -z
   $(arg z_pos) -param robot_description" />
```

3.2.2. Chuyển đổi từ URDF sang SDF cho Gazebo

Trong khi Robot được mô tả bằng URDF. URDF chỉ có thể mô tả tính chất động học và động lực học của Robot trong môi trường riêng biệt, cũng không thể mô tả được vị trí và hướng của Robot trong môi trường cụ thể.

Để giải quyết các vấn đề này, định dạng mới gọi là Simulation Description Format (SDF) được tạo ra để sử dụng trong Gazebo, giải quyết những hạn chế của URDF.

Và có cách tương ứng để chuyển đổi định dạng từ URDF có sẵn sang SDF cho Gazebo.

*** Bắt buộc:**

- Trong mỗi <link> phải có các thẻ <inertia> được điều chỉnh tương ứng

*** Tùy chọn, tùy vào link:**

- Thêm thẻ <gazebo> vào các <link>
  - Chuyển đổi màu sắc sang định dạng Gazebo
  - Chuyển đổi tệp stl sang dae để có được màu sắc tốt hơn.
  - Thêm các cảm biến.
- Thêm thẻ <gazebo> vào các <joint>
  - Thêm các thông số vật lí cho các chuyển động
  - Thêm plugin điều khiển cho các cơ cấu chuyển động

- Thêm thẻ <gazebo> cho thẻ <robot>
- Thêm <link name="world"/> để link robot với môi trường.

*** Các thẻ cần thiết cho link:**

Tên	**Kiểu dữ liệu**	**Mô tả**
material	value	Biểu diễn vật liệu màu sắc
gravity	bool	Sử dụng trọng lực
dampingFactor	double	Hàm lũy thừa suy giảm tốc độ của tốc độ của link - lấy giá trị và nhân với tốc độ của link trước đó với (1- dampingFactor)
maxVel	double	Tốc độ tối đa có thể đạt được ở các tiếp xúc trong thời gian ngắn
minDepth	double	Độ sâu cho phép tối thiểu trước khi có động lực ở các chỗ tiếp xúc
mu1 mu2	double	Hệ số ma sát cho các hướng của các liên kết dọc theo bề mặt của liên kết được định nghĩa bởi Open Dynamics Engine (ODE)
fdir1	string	3 chiều mô tả hướng của mu1 trong va chạm
kp kd	double	Độ cứng kp và hệ số cản nhớt kd trong liên kết định nghĩa bởi ODE.
selfCollide	bool	Nếu kết quả là true, liên kết có thẻ va chạm với các link khác trong không gian.
maxContacts	int	Số lượng liên kết tối đa giữa 2 thành phần.
laserRetro	double	Độ dày được trả về bởi Laser

Các hệ số này dùng để mô tả các thành phần có chuyển động như 2 bánh xe:

```
1.   <gazebo reference="wheel_left_link">
2.     <mu1>0.1</mu1>
3.     <mu2>0.1</mu2>
4.     <kp>500000.0</kp>
5.     <kd>10.0</kd>
6.     <minDepth>0.001</minDepth>
7.     <maxVel>0.1</maxVel>
8.     <fdir1>1 0 0</fdir1>
9.     <material>Gazebo/FlatBlack</material>
10.    </gazebo>
11.
12.    <gazebo reference="wheel_right_link">
13.      <mu1>0.1</mu1>
14.      <mu2>0.1</mu2>
15.      <kp>500000.0</kp>
16.      <kd>10.0</kd>
17.      <minDepth>0.001</minDepth>
18.      <maxVel>0.1</maxVel>
19.      <fdir1>1 0 0</fdir1>
20.      <material>Gazebo/FlatBlack</material>
21.    </gazebo>
```

*** Các plugin cơ bản được sử dụng:**

- Cảm biến gia tốc góc IMU: Trả kết quả về topic imu.

```
<gazebo>
  <plugin name="imu_plugin" filename="libgazebo_ros_imu.so">
    <alwaysOn>true</alwaysOn>
    <bodyName>imu_link</bodyName>
    <frameName>imu_link</frameName>
    <topicName>imu</topicName>
    <serviceName>imu_service</serviceName>
    <gaussianNoise>0.0</gaussianNoise>
    <updateRate>200</updateRate>
      <imu>
        <noise>
          <type>gaussian</type>
          <rate>
            <mean>0.0</mean>
            <stddev>2e-4</stddev>
            <bias_mean>0.0000075</bias_mean>
            <bias_stddev>0.0000008</bias_stddev>
          </rate>
          <accel>
            <mean>0.0</mean>
            <stddev>1.7e-2</stddev>
            <bias_mean>0.1</bias_mean>
            <bias_stddev>0.001</bias_stddev>
          </accel>
        </noise>
      </imu>
    </plugin>
  </gazebo>
```

- Cảm biến laser "": Trả kết quả về topic scan:

```
<gazebo reference="laser_frame">
    <material>Gazebo/FlatBlack</material>
    <sensor type="ray" name="lds_lfcd_sensor">
      <pose>0 0 0 0 0 0</pose>
      <visualize>$(arg laser_visual)</visualize>
      <update_rate>8</update_rate>
      <ray>
        <scan>
          <horizontal>
            <samples>500</samples>
            <resolution>1</resolution>
            <min_angle>-3.14</min_angle>
            <max_angle>3.14</max_angle>
          </horizontal>
        </scan>
        <range>
          <min>0.08</min>
          <max>12.0</max>
          <resolution>0.01</resolution>
        </range>
        <noise>
          <type>gaussian</type>
```

```
23.            <mean>0.0</mean>
24.            <stddev>0.01</stddev>
25.          </noise>
26.        </ray>
27.        <plugin name="gazebo_ros_lds_lfcd_controller"
   filename="libgazebo_ros_laser.so">
28.          <topicName>scan</topicName>
29.          <frameName>laser_frame</frameName>
30.        </plugin>
31.      </sensor>
32.    </gazebo>
```

Sensor Noise

Hầu hết các cảm biến đều có nhiễu. Camera, cảm biến siêu âm, laser đều có những nhiễu loạn không chính xác khi đọc dữ liệu. Chúng ta có thể them các nhiễu này vào dữ liệu thu được từ mô phỏng để có thể gần với dữ liệu thực tế mà cảm biến nhận được. (Thẻ <noise>

Gazebo xây dựng mô hình nhiễu dựa trên nhiễu Gaussian. Mặc dù không thực tế, nhưng nó là phương pháp xấp xỉ tốt nhất để mô tả nhiễu.

- Bộ điều khiển Robot Differential Drive - thay thế cho bộ điều khiển Robot:
  - Lấy thông số từ topic cmd_vel
  - Lấy gốc Robot từ base_footprint
  - Lấy thông số từ Odom
  - Trả các kết quả lên tf

```
1.     <gazebo>
2.    <plugin name="dashgo_driver"
   filename="libgazebo_ros_diff_drive.so">
3.       <commandTopic>cmd_vel</commandTopic>
4.       <odometryTopic>odom</odometryTopic>
5.       <odometryFrame>odom</odometryFrame>
6.       <odometrySource>world</odometrySource>
7.       <publishOdomTF>true</publishOdomTF>
8.       <robotBaseFrame>base_footprint</robotBaseFrame>
9.       <publishWheelTF>false</publishWheelTF>
10.        <publishTf>true</publishTf>
11.        <publishWheelJointState>true</publishWheelJointState>
12.        <legacyMode>false</legacyMode>
13.        <updateRate>20</updateRate>
14.        <leftJoint>wheel_left_joint</leftJoint>
15.        <rightJoint>wheel_right_joint</rightJoint>
16.        <wheelSeparation>0.32</wheelSeparation>
17.        <wheelDiameter>0.12</wheelDiameter>
18.        <wheelAcceleration>1</wheelAcceleration>
19.        <wheelTorque>10</wheelTorque>
20.        <rosDebugLevel>na</rosDebugLevel>
21.      </plugin>
22.    </gazebo>
```

*** Tệp mô tả Gazebo:**

```
1. <?xml version="1.0"?>
2. <robot name="dashgobot_sim"
   xmlns:xacro="http://ros.org/wiki/xacro">
3.
4. <xacro:arg name="laser_visual" default="true"/>
5. <xacro:arg name="imu_visual" default="true"/>
6. <xacro:arg name="sonar_visual" default="true"/>
7.
8. <gazebo reference="base_link">
9. <material>Gazebo/White</material>
10.  </gazebo>
11.
12.  <gazebo reference="robot_link">
13.  <material>Gazebo/White</material>
14.  </gazebo>
15.
16.  <gazebo reference="wheel_left_link">
17.  <mu1>0.1</mu1>
18.  <mu2>0.1</mu2>
19.  <kp>500000.0</kp>
20.  <kd>10.0</kd>
21.  <minDepth>0.001</minDepth>
22.  <maxVel>0.1</maxVel>
23.  <fdir1>1 0 0</fdir1>
24.  <material>Gazebo/FlatBlack</material>
25.  </gazebo>
26.
27.  <gazebo reference="wheel_right_link">
28.  <mu1>0.1</mu1>
29.  <mu2>0.1</mu2>
30.  <kp>500000.0</kp>
31.  <kd>10.0</kd>
32.  <minDepth>0.001</minDepth>
33.  <maxVel>0.1</maxVel>
34.  <fdir1>1 0 0</fdir1>
35.  <material>Gazebo/FlatBlack</material>
36.  </gazebo>
37.
38.  <gazebo reference="caster_back_link">
39.  <mu1>0.1</mu1>
40.  <mu2>0.1</mu2>
41.  <kp>1000000.0</kp>
42.  <kd>100.0</kd>
43.  <minDepth>0.001</minDepth>
44.  <maxVel>1.0</maxVel>
45.  <material>Gazebo/FlatBlack</material>
46.  </gazebo>
47.
48.  <gazebo reference="caster_front_link">
49.  <mu1>0.1</mu1>
50.  <mu2>0.1</mu2>
51.  <kp>1000000.0</kp>
52.  <kd>100.0</kd>
53.  <minDepth>0.001</minDepth>
54.  <maxVel>1.0</maxVel>
55.  <material>Gazebo/FlatBlack</material>
56.  </gazebo>
57.
58.  <gazebo reference="imu_link">
```

```
59.  <sensor type="imu" name="imu">
60.  <always_on>true</always_on>
61.  <visualize>$(arg imu_visual)</visualize>
62.  </sensor>
63.  <material>Gazebo/FlatBlack</material>
64.  </gazebo>
65.
66.  <gazebo>
67.  <plugin name="dashgo_driver"
   filename="libgazebo_ros_diff_drive.so">
68.  <commandTopic>cmd_vel</commandTopic>
69.  <odometryTopic>odom</odometryTopic>
70.  <odometryFrame>odom</odometryFrame>
71.  <odometrySource>world</odometrySource>
72.  <publishOdomTF>true</publishOdomTF>
73.  <robotBaseFrame>base_footprint</robotBaseFrame>
74.  <publishWheelTF>false</publishWheelTF>
75.  <publishTf>true</publishTf>
76.  <publishWheelJointState>true</publishWheelJointState>
77.  <legacyMode>false</legacyMode>
78.  <updateRate>20</updateRate>
79.  <leftJoint>wheel_left_joint</leftJoint>
80.  <rightJoint>wheel_right_joint</rightJoint>
81.  <wheelSeparation>0.32</wheelSeparation>
82.  <wheelDiameter>0.12</wheelDiameter>
83.  <wheelAcceleration>1</wheelAcceleration>
84.  <wheelTorque>10</wheelTorque>
85.  <rosDebugLevel>na</rosDebugLevel>
86.  </plugin>
87.  </gazebo>
88.
89.  <gazebo>
90.  <plugin name="imu_plugin" filename="libgazebo_ros_imu.so">
91.  <alwaysOn>true</alwaysOn>
92.  <bodyName>imu_link</bodyName>
93.  <frameName>imu_link</frameName>
94.  <topicName>imu</topicName>
95.  <serviceName>imu_service</serviceName>
96.  <gaussianNoise>0.0</gaussianNoise>
97.  <updateRate>200</updateRate>
98.  <imu>
99.  <noise>
100. <type>gaussian</type>
101. <rate>
102. <mean>0.0</mean>
103. <stddev>2e-4</stddev>
104. <bias_mean>0.0000075</bias_mean>
105. <bias_stddev>0.0000008</bias_stddev>
106. </rate>
107. <accel>
108. <mean>0.0</mean>
109. <stddev>1.7e-2</stddev>
110. <bias_mean>0.1</bias_mean>
111. <bias_stddev>0.001</bias_stddev>
112. </accel>
113. </noise>
114. </imu>
115. </plugin>
116. </gazebo>
```

```
117.
118. <gazebo reference="laser_frame">
119. <material>Gazebo/FlatBlack</material>
120. <sensor type="ray" name="lds_lfcd_sensor">
121. <pose>0 0 0 0 0 0</pose>
122. <visualize>$(arg laser_visual)</visualize>
123. <update_rate>8</update_rate>
124. <ray>
125. <scan>
126. <horizontal>
127. <samples>500</samples>
128. <resolution>1</resolution>
129. <min_angle>-3.14</min_angle>
130. <max_angle>3.14</max_angle>
131. </horizontal>
132. </scan>
133. <range>
134. <min>0.08</min>
135. <max>12.0</max>
136. <resolution>0.01</resolution>
137. </range>
138. <noise>
139. <type>gaussian</type>
140. <mean>0.0</mean>
141. <stddev>0.01</stddev>
142. </noise>
143. </ray>
144. <plugin name="gazebo_ros_lds_lfcd_controller"
   filename="libgazebo_ros_laser.so">
145. <topicName>scan</topicName>
146. <frameName>laser_frame</frameName>
147. </plugin>
148. </sensor>
149. </gazebo>
150.
151. <gazebo reference="sonar_link_fake">
152.
153. <sensor type="ray" name="TeraRanger">
154. <pose>0 0 0 0 0 0</pose>
155. <visualize>true</visualize>
156. <update_rate>50</update_rate>
157. <ray>
158. <scan>
159. <horizontal>
160. <samples>10</samples>
161. <resolution>1</resolution>
162. <min_angle>-0.14835</min_angle>
163. <max_angle>0.14835</max_angle>
164. </horizontal>
165. <vertical>
166. <samples>10</samples>
167. <resolution>1</resolution>
168. <min_angle>-0.14835</min_angle>
169. <max_angle>0.14835</max_angle>
170. </vertical>
171. </scan>
172. <range>
173. <min>0.08</min>
174. <max>10</max>
```

```
175. <resolution>0.02</resolution>
176. </range>
177. </ray>
178. <plugin filename="libgazebo_ros_range.so"
   name="gazebo_ros_range">
179. <gaussianNoise>0.005</gaussianNoise>
180. <alwaysOn>true</alwaysOn>
181. <updateRate>50</updateRate>
182. <topicName>sonar0</topicName>
183. <frameName>sonar_link</frameName>
184. <radiation>INFRARED</radiation>
185. <fov>0.2967</fov>
186. </plugin>
187. </sensor>
188. </gazebo>
189.
190. </robot>
```

3.3. Điều khiển và vận hành Robot

Source code: https://github.com/AIR-Hust/dashgo_simulations

Hướng dẫn cài đặt:

Tải về và chép vào: ~/dashgo_ws/src/dashgo

Mở terminal:

```
1. $ cd ~/dashgo_ws
2. $ catkin_make
3. $ source devel/setup.bash
```

3.3.1. Các topic mà Robot subscribed:

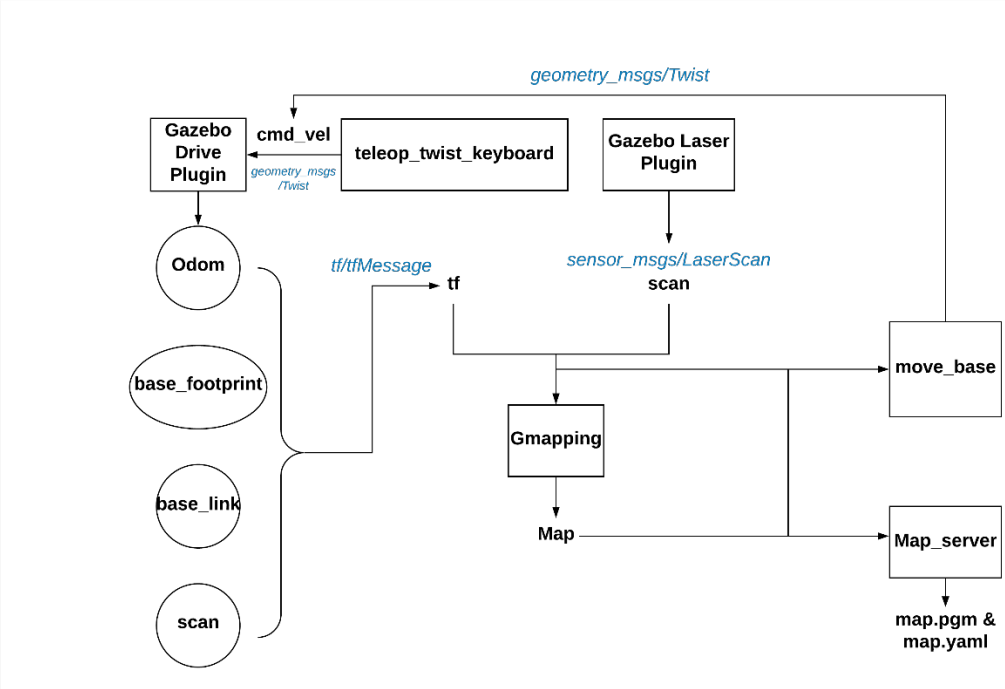
Tên Topic	Định dạng	Chức năng
motor_power	std_msgs/Bool	Bật/Tắt moment xoắn động học
cmd_vel	Geometry_msgs/Twist	Điều khiển chuyển động tịnh tiến và xoay của Robot, đươn vị m/s và rad/s

3.3.2. Các topic mà Robot published:

Tên Topic	Định dạng	Chức năng
joint_states	Sensor_msgs/JointState	Kiểm tra vị trí của Robot (m), tốc độ (m/s) và Moment (N.m) với bánh xe là các joint
vmd_vel	Geometry_msgs/Twist	Điều khiển chuyển động tịnh tiến và xoay của Robot, đươn vị m/s và rad/s
scan	sensor_msgs/LaserScan	Topic chứa giá trị scan ddwwojc của Lidar
imu	sensor_msgs/imu	Topic
odom	nav_msgs/Odometry	Odom của Robot tính toán dựa ten encoder và IMU
tf	tf2_msgs/TFMessage	Hệ trục tọa độ biến đổi của Robot dựa trên basefootprint và Odom

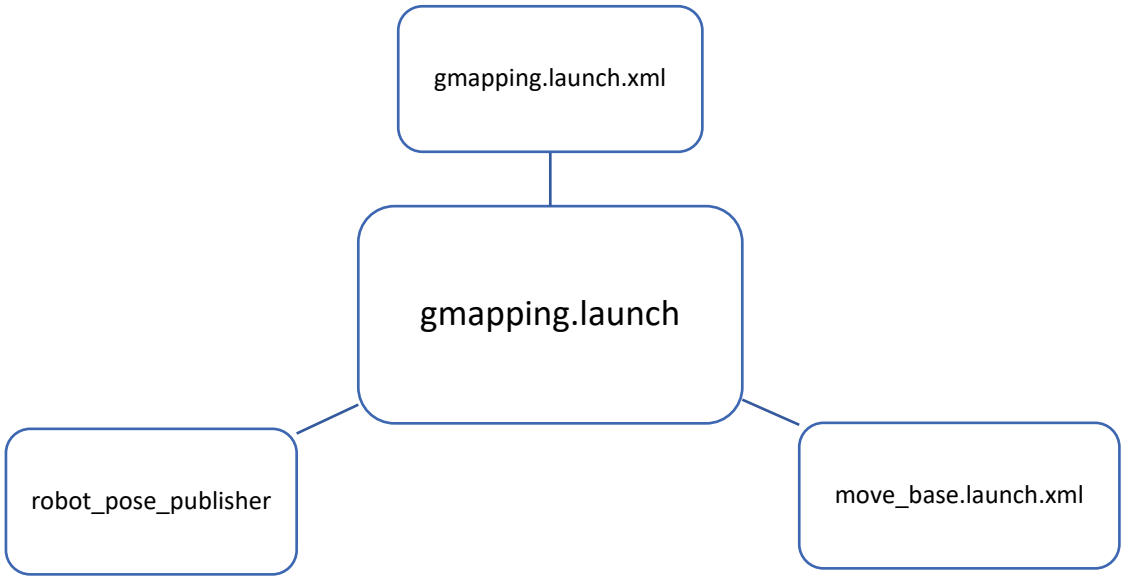
3.3.3. Gmapping

Mô hình thuật toán như sau:



*** Xây dựng gmapping với giả lập:**

Để thực hiện Gmapping, gói gmapping được xây dựng như sau:

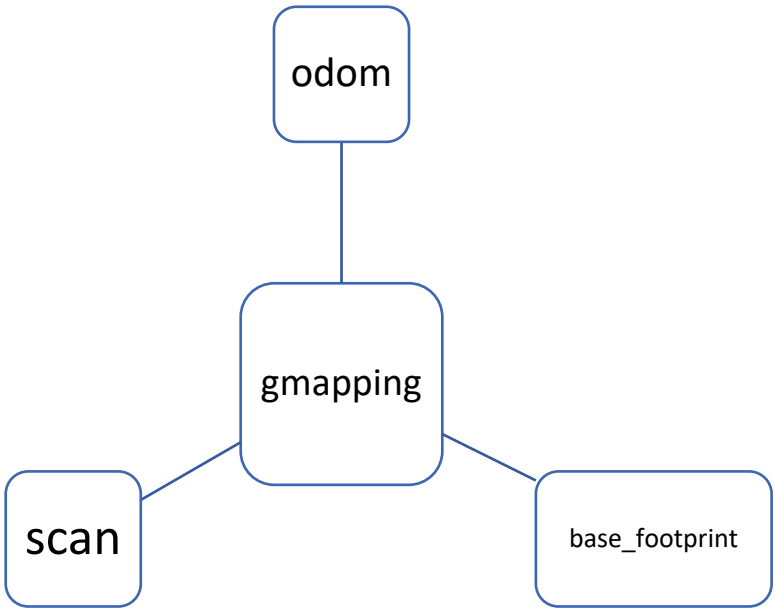


- gmapping.launch.xml: Chạy package gmapping và các thông số khi thực hiện gmapping

- move_base.launch.xml: Chạy package move_base và các thông số liên quan đến move base.
- robot_pose_publisher: Chạy tf và cập nhật vị trí của Robot.

```
1. <launch>
2.   <arg name="with_teb"   default="true"/>
3.   <!-- Gmapping -->
4.   <include file="$(find
   dashgo_gazebo)/launch/includes/gmapping.launch.xml">
5.     <arg name="scan_topic" value="scan"/>
6.   </include>
7.
8.   <!-- Move base -->
9.   <include file="$(find
   dashgo_gazebo)/launch/includes/move_base.launch.xml">
10.      <arg name="teb_move_base"   default="$(arg with_teb)"/>
11.    </include>
12.   <node name="robot_pose_publisher" pkg="robot_pose_publisher"
   type="robot_pose_publisher" />
13.  </launch>
```

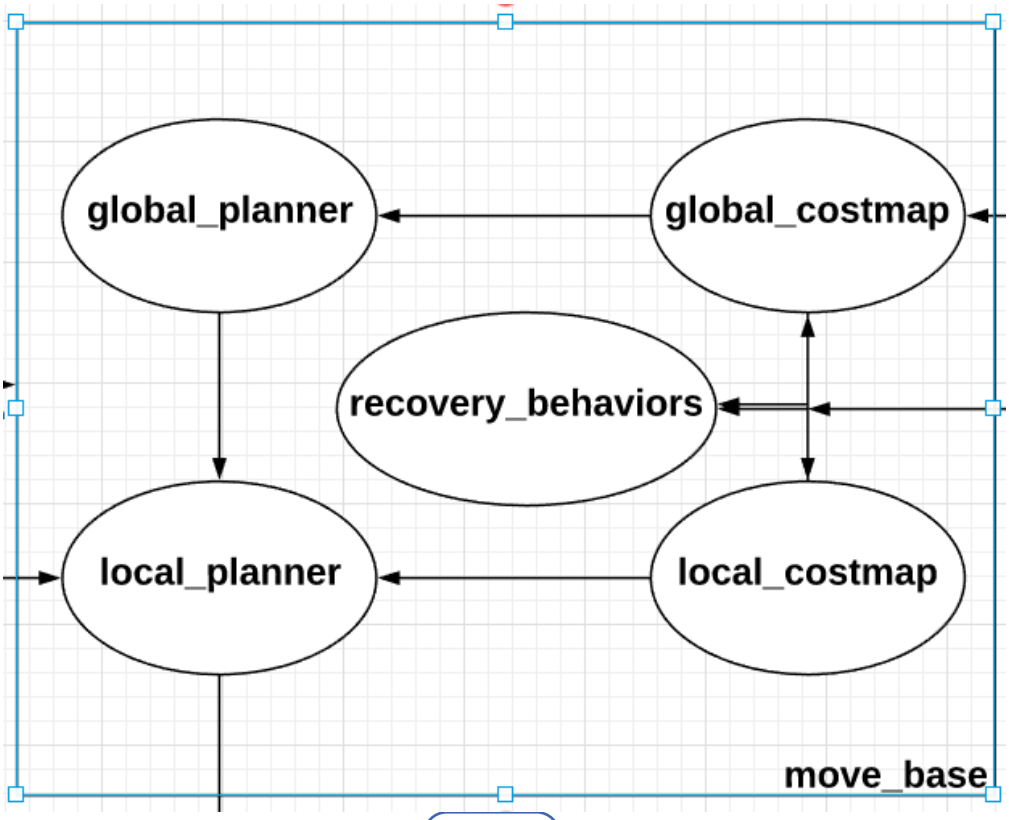
*** gmapping.launch.xml:** Lấy thông số từ các topic odom, base_footprint và scan.

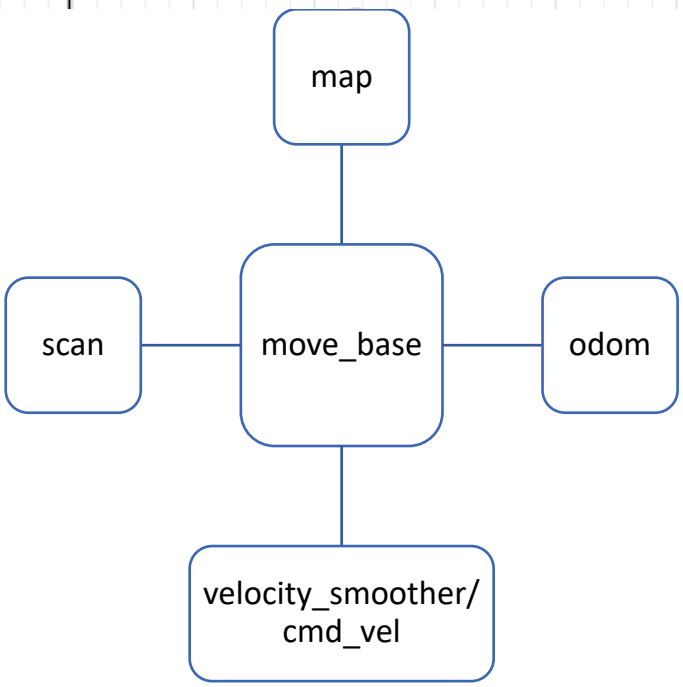



```
1. <launch>
2.   <arg name="scan_topic"  default="scan" />
3.   <arg name="base_frame"  default="base_footprint"/>
4.   <arg name="odom_frame"  default="odom"/>
5.
6.   <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping"
   output="screen">
7.     <param name="base_frame" value="$(arg base_frame)"/>
8.     <param name="odom_frame" value="$(arg odom_frame)"/>
9.     <param name="map_update_interval" value="5.0"/>
10.      <param name="maxUrange" value="6.0"/>
```

```
    <param name="maxRange" value="8.0"/>
    <param name="sigma" value="0.05"/>
    <param name="kernelSize" value="1"/>
    <param name="lstep" value="0.05"/>
    <param name="astep" value="0.05"/>
    <param name="iterations" value="5"/>
    <param name="lsigma" value="0.075"/>
    <param name="ogain" value="3.0"/>
    <param name="lskip" value="0"/>
    <param name="minimumScore" value="30"/>
    <param name="srr" value="0.01"/>
    <param name="srt" value="0.02"/>
    <param name="str" value="0.01"/>
    <param name="stt" value="0.02"/>
    <param name="linearUpdate" value="0.5"/>
    <param name="angularUpdate" value="0.436"/>
    <param name="temporalUpdate" value="-1.0"/>
    <param name="resampleThreshold" value="0.5"/>
    <param name="particles" value="8"/>
  <!--
    <param name="xmin" value="-50.0"/>
    <param name="ymin" value="-50.0"/>
    <param name="xmax" value="50.0"/>
    <param name="ymax" value="50.0"/>
  make the starting size small for the benefit of the Android
 client's memory...
  -->
    <param name="xmin" value="-1.0"/>
    <param name="ymin" value="-1.0"/>
    <param name="xmax" value="1.0"/>
    <param name="ymax" value="1.0"/>

    <param name="delta" value="0.05"/>
    <param name="llsamplerange" value="0.01"/>
    <param name="llsamplestep" value="0.01"/>
    <param name="lasamplerange" value="0.005"/>
    <param name="lasamplestep" value="0.005"/>
    <remap from="scan" to="$(arg scan_topic)"/>
  </node>
</launch>
```

*** move_base.launch.xml:**





- Thay thế cho cả move_base và teb_move_base giúp điều khiển Robot di chuyển.
- teb_move_base: Điều hướng trong bản đồ đã được tạo khi chạy navigation.
- move_base: Điều hướng trong những vùng bản đồ đã đọc được khi thực hiện gmapping.

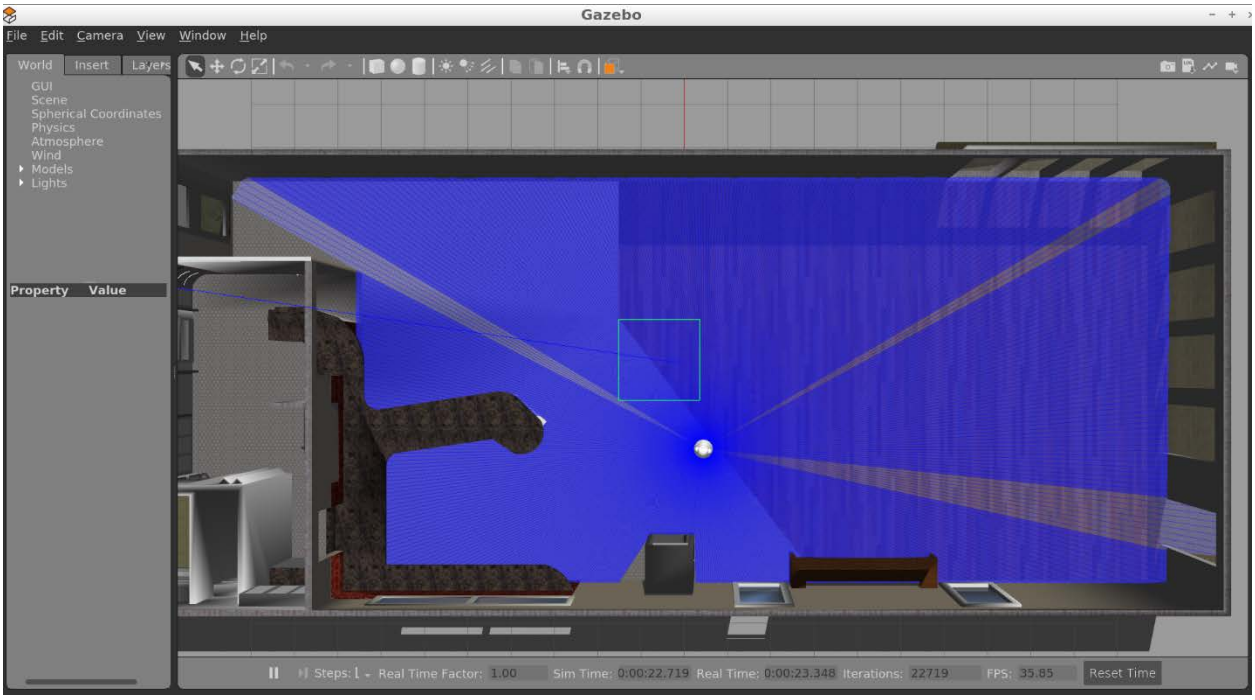
```
1. <!--
```

```
2.     ROS navigation stack with velocity smoother and safety
   (reactive) controller
3. -->
4. <launch>
5.   <include file="$(find
   dashgo_gazebo)/launch/includes/velocity_smoother.launch.xml"/>
6.
7.   <arg name="teb_move_base"   default="true"/>
8.   <arg name="navi_cmd_vel"    default="cmd_vel_mux/input/navi"/>
9.   <arg name="odom_frame_id"   default="odom"/>
10.    <arg name="base_frame_id"   default="base_footprint"/>
11.    <arg name="global_frame_id" default="map"/>
12.    <arg name="odom_topic"      default="odom"/>
13.    <arg name="laser_topic"     default="scan"/>
14.    <arg name="with_imu"        default="false"/>
15.    <arg name="cmd_vel_topic" default="/cmd_vel" />
16.  <!--  <arg name="custom_param_file" default="$(find
   turtlebot_navigation)/param/dummy.yaml"/>-->
17.
18.    <group if="$(arg teb_move_base)">
19.      <!-- switch to teb_move_base -->
20.      <node pkg="move_base" type="move_base" respawn="false"
   name="move_base" output="screen" clear_params="true">
21.        <rosparam file="$(find
   dashgo_nav)/config/odom/costmap_common_params.yaml" command="load"
   ns="global_costmap" />
22.        <rosparam file="$(find
   dashgo_nav)/config/odom/costmap_common_params.yaml" command="load"
   ns="local_costmap" />
23.        <rosparam file="$(find
   dashgo_nav)/config/odom/local_costmap_params.yaml" command="load"
   />
24.        <rosparam file="$(find
   dashgo_nav)/config/odom/global_costmap_params.yaml" command="load"
   />
25.        <rosparam file="$(find
   dashgo_nav)/config/odom/base_global_planner_param.yaml"
   command="load" />
26.
27.        <rosparam file="$(find
   dashgo_nav)/config/odom/teb_local_planner_params.yaml"
   command="load" />
28.
29.        <param name="base_global_planner"
   value="global_planner/GlobalPlanner"/>
30.        <param name="planner_frequency" value="1.0" />
31.        <param name="planner_patience" value="5.0" />
32.
33.        <param name="base_local_planner"
   value="teb_local_planner/TebLocalPlannerROS" />
34.        <param name="controller_frequency" value="5.0" />
35.        <param name="controller_patience" value="15.0" />
36.
37.        <remap from="cmd_vel" to="$(arg navi_cmd_vel)"/>
38.        <remap from="cmd_vel" to="$(arg cmd_vel_topic)"/>
39.        <remap from="odom" to="$(arg odom_topic)"/>
40.        <remap from="scan" to="$(arg laser_topic)"/>
41.      </node>
42.    </group>
```

```
43.
44.    <group unless="$(arg teb_move_base)">
45.      <!-- previous move_base -->
46.      <node pkg="move_base" type="move_base" respawn="false"
   name="move_base" output="screen">
47.        <rosparam file="$(find
   dashgo_nav)/config/odom/costmap_common_params.yaml" command="load"
   ns="global_costmap" />
48.        <rosparam file="$(find
   dashgo_nav)/config/odom/costmap_common_params.yaml" command="load"
   ns="local_costmap" />
49.        <rosparam file="$(find
   dashgo_nav)/config/odom/local_costmap_params.yaml" command="load"
   />
50.        <rosparam file="$(find
   dashgo_nav)/config/odom/global_costmap_params.yaml" command="load"
   />
51.        <rosparam file="$(find
   dashgo_nav)/config/odom/base_local_planner_params.yaml"
   command="load" />
52.
53.        <rosparam file="$(find
   dashgo_nav)/config/nav_obstacles_params.yaml" command="load" />
54.
55.        <remap from="cmd_vel" to="$(arg navi_cmd_vel)"/>
56.        <remap from="cmd_vel" to="$(arg cmd_vel_topic)"/>
57.        <remap from="odom" to="$(arg odom_topic)"/>
58.        <remap from="scan" to="$(arg laser_topic)"/>
59.      </node>
60.    </group>
61.  </launch>
```

*** Thực hiện Gmapping:**

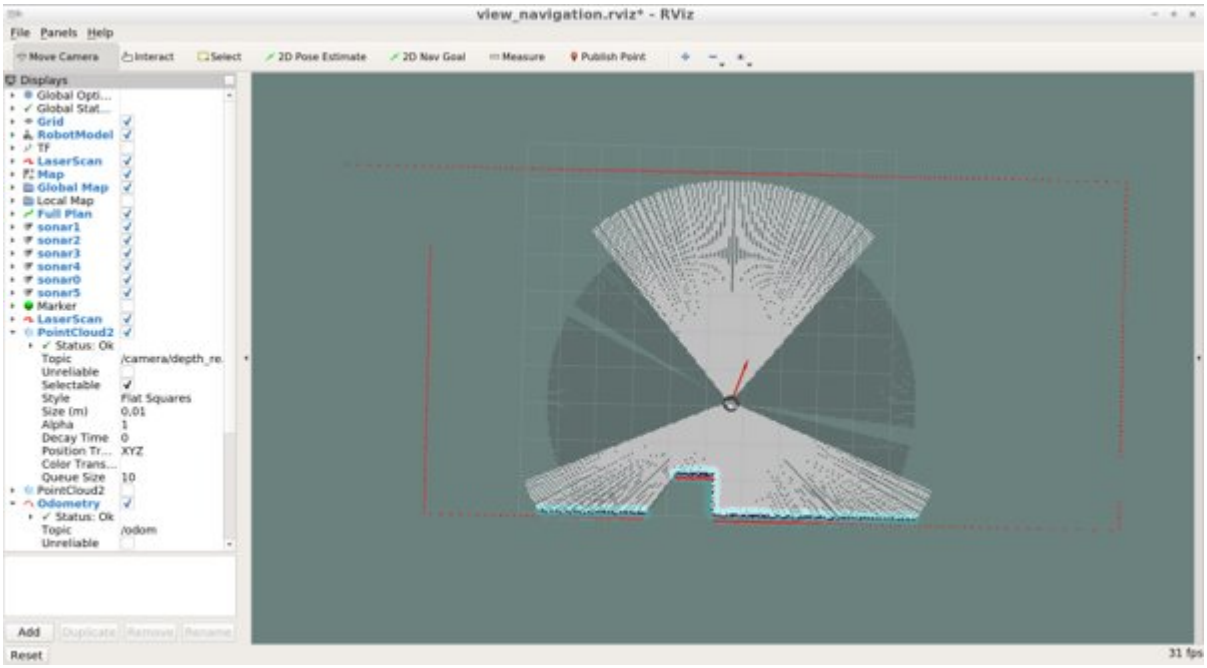
- Khởi động world: roslaunch dashgo_gazebo dashgo_gazebo.launch



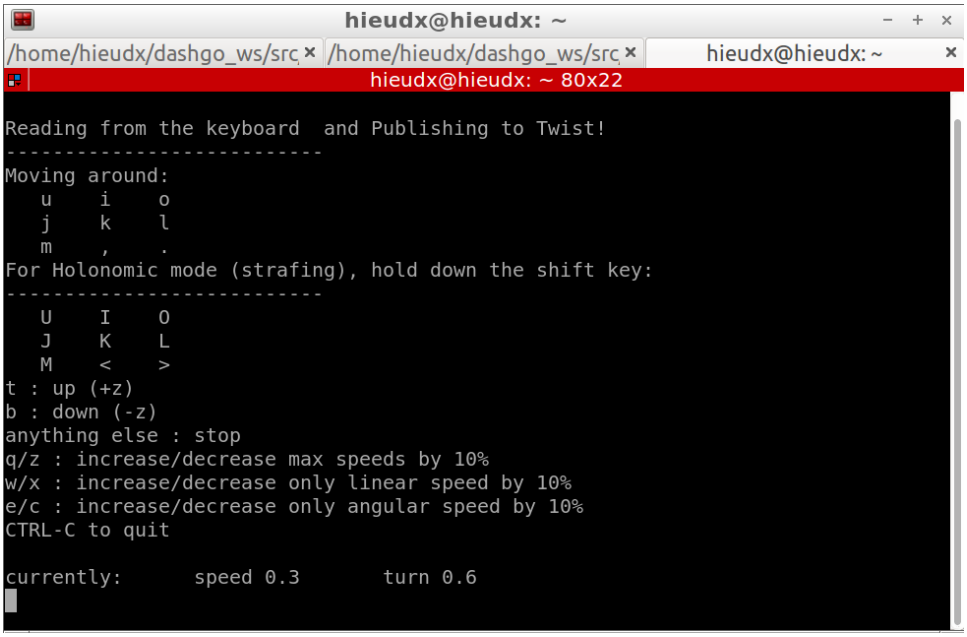
o Dashgo_gazebo.launch có thể thay thế bằng các world sau đây:

- dashgo_house.launch
- world_stage_1.launch
- world_stage_2.launch
- world_stage_3.launch
- world_stage_4.launch

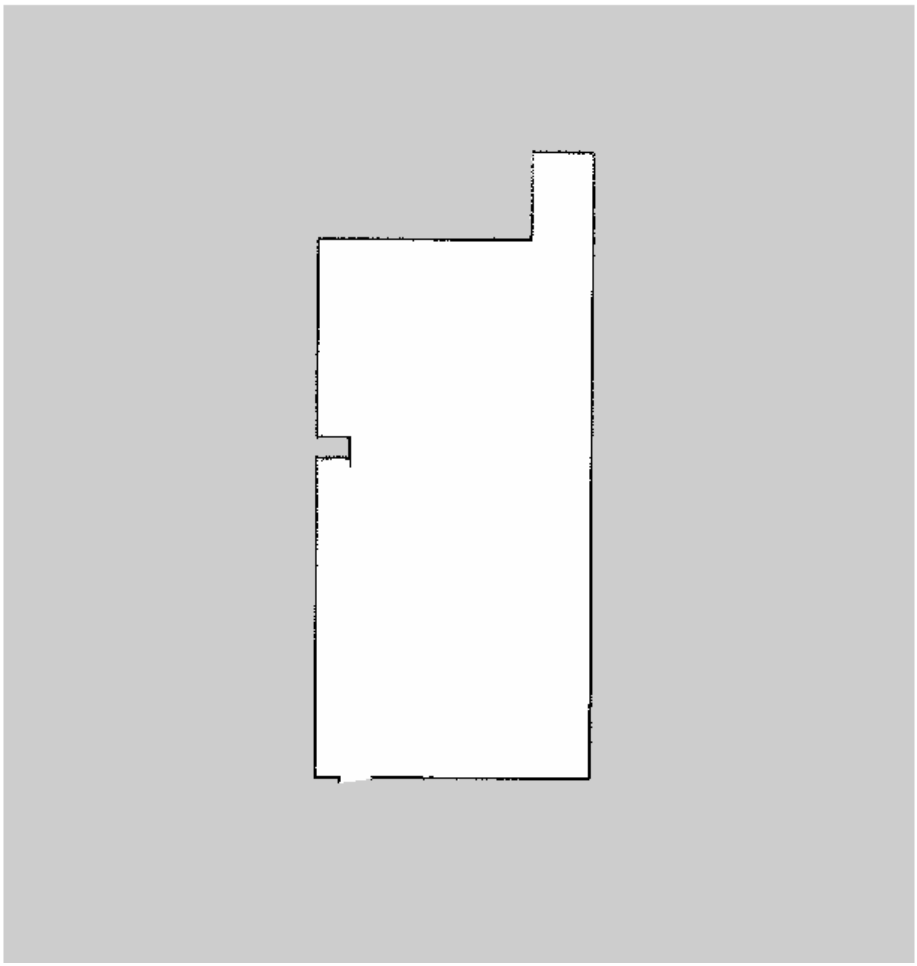
- Khởi động gmapping: roslaunch dashgo_gazebo gmapping.launch
- Rviz: roslaunch dashgo_rviz view_navigation.launch



- Phím điều khiển: rosrun dashgo_tools teleop_twist_keyboard.py

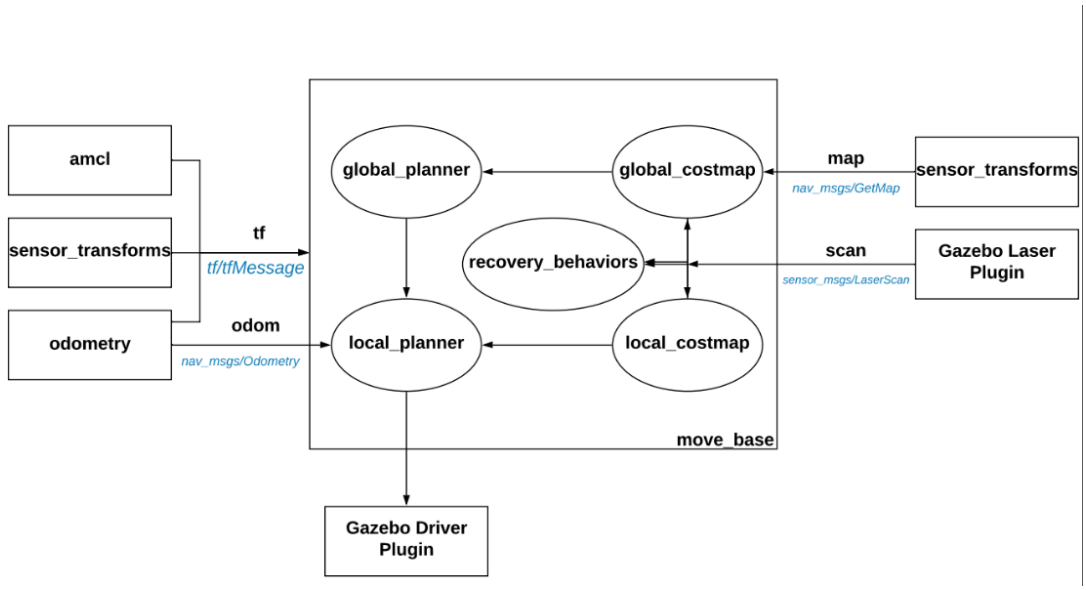


- Lưu bản đồ: rosrun map_server map_saver -f ~/dashgo_ws/map



3.3.4. Navigation

*** Mô hình thuật toán như sau:**



*

*** Xây dựng gói Navigation cho giả lập:**

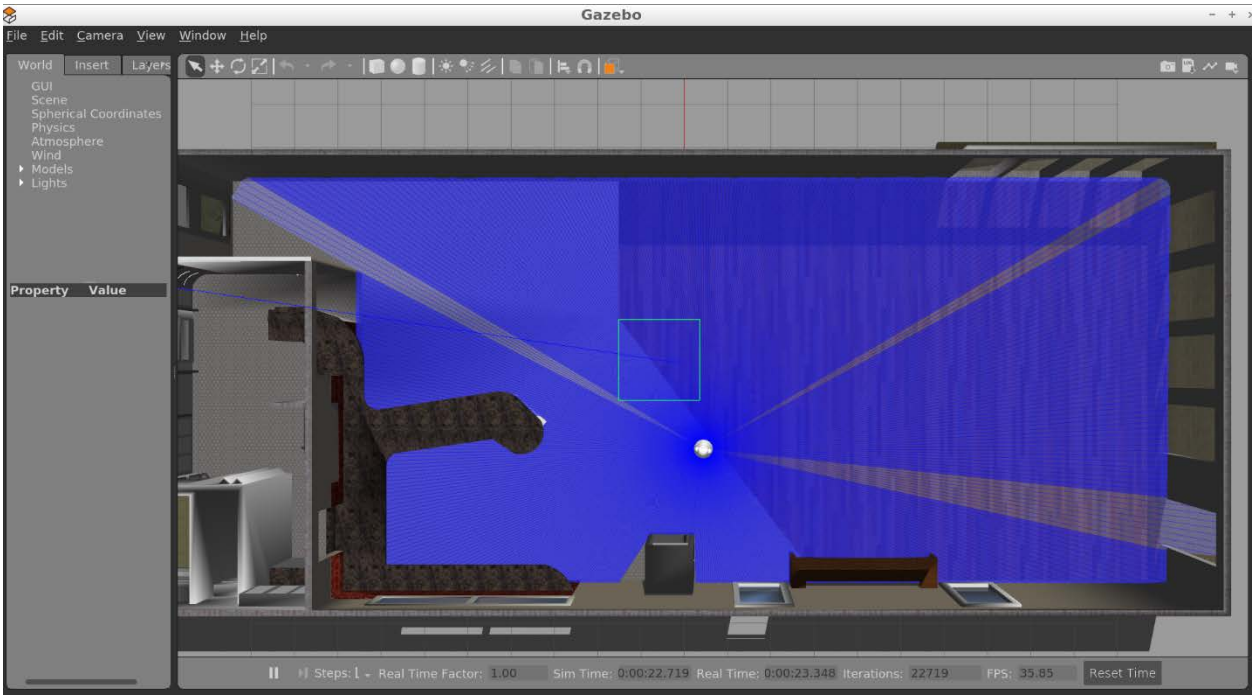
Gói navigation được xây dựng như sau:

navigation	map_server
	amcl
	move_base
	tf_broadcaster (Thay thế bằng script odom transformer)
	robot_pose_publisher

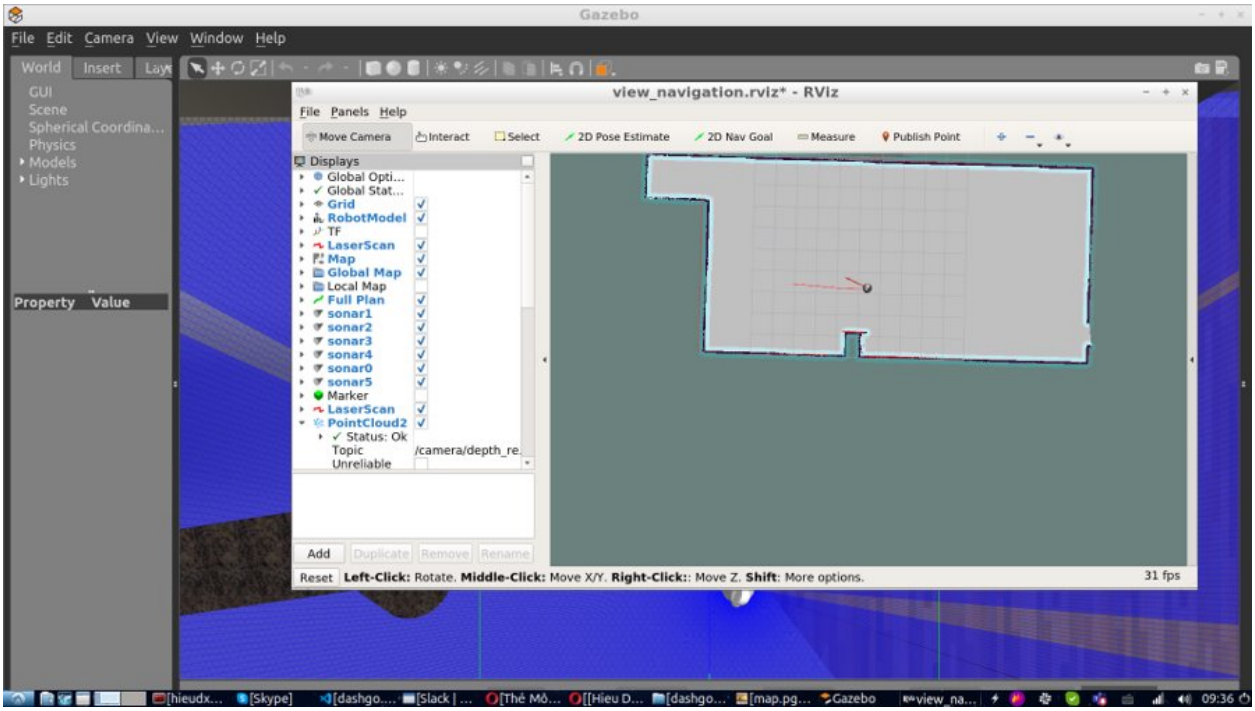
```
1. <launch>
2.   <arg name="teb" default="true"/>
3.   <arg name="imu" default="false"/>
4.   <arg name="map_file" default="$(find
   dashgo_gazebo)/maps/retest_map.yaml"/>
5.   <node name="map_server" pkg="map_server" type="map_server"
   args="$(arg map_file)" />
6.
7.   <arg name="initial_pose_x" default="0.0"/>
8.   <arg name="initial_pose_y" default="0.0"/>
9.   <arg name="initial_pose_a" default="0.0"/>
10.    <include file="$(find
   dashgo_gazebo)/launch/includes/amcl.launch.xml">
11.      <arg name="initial_pose_x" value="$(arg initial_pose_x)"/>
12.      <arg name="initial_pose_y" value="$(arg initial_pose_y)"/>
13.      <arg name="initial_pose_a" value="$(arg initial_pose_a)"/>
14.    </include>
15.
16.    <include file="$(find
   dashgo_gazebo)/launch/includes/move_base.launch.xml">
17.      <arg name="teb_move_base"   default="$(arg teb)"/>
18.      <arg name="with_imu"   default="$(arg imu)"/>
19.    </include>
20.
21.    <!-- add a tool to fix tf broadcaster missing -->
22.    <node pkg="dashgo_gazebo" type="odom_transformer.py"
   name="odom_transformer" output="screen"/>
23.
24.    <node name="robot_pose_publisher" pkg="robot_pose_publisher"
   type="robot_pose_publisher" />
25.  </launch>
```

*** Thực hiện navigation:**

- Khởi động world: roslaunch dashgo_gazebo dashgo_gazebo.launch



- Khởi động gmapping: roslaunch dashgo_gazebo navigation.launch map_file:=$HOME/dashgo_ws/gazebo_map.yaml
- Rviz: roslaunch dashgo_rviz view_navigation.launch



3.4. Kết luận

*** Tổng hợp lại các gói/phần mềm phụ trợ:**

- URDF Exporter: https://github.com/ros/solidworks_urdf_exporter/releases
- URDF Display Tutorial, Robot Pose Publisher, Joint State Publisher:

```
1. $ sudo apt-get install ros-kinetic-urdf-tutorial robot-pose-
   publisher joint-state-publisher
```

- Dashgo_description và dashgo_gazebo: https://github.com/AIR-Hust/dashgo_simulations

*** Đánh giá:**

Ưu điểm:

- Robot thực hiện tốt việc tạo bản đồ, di chuyển, điều hướng.
- Mô phỏng đa dạng, tùy biến cao dễ điều chỉnh.
- Có nhiều trang hỗ trợ từ cộng đồng.

Nhược điểm:

- Do chiều cao của Robot và thiếu các cảm biến tầng cao, Robot gặp khó khan trước các đối tượng như bàn/ghế có chiều cao tương đương với Robot.
- Có thể thêm các đối tượng di chuyển trong môi trường nhưng chỉ them được từ phiên bản Gazebo 8 trở lên. Phiên bản Gazebo mặc định của ROS Kinetic là Gazebo 7, có thể upgrade nhưng quá trình phức tạp và phải gỡ bỏ hoàn toàn ROS trước khi cài lại.