

**ĐẠI HỌC QUỐC GIA HÀ NỘI**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**  
**KHOA ĐIỆN TỬ - VIỄN THÔNG**



**XÂY DỰNG, MÔ PHỎNG VÀ  
ĐIỀU KHIỂN ROBOT**

**BÁO CÁO GIỮA KỲ**  
**LẬP TRÌNH ROBOT VỚI ROS**  
**MÃ MÔN HỌC: RBE3017 1**

**Người thực hiện:**

Đào Duy Hưng

22027551

QH-2022-I/CQ-R

Khoa Điện tử Viễn thông

Hà Nội, 2025

## Mục lục

1.	Mở đầu .....	3
2.	Xây dựng mô hình robot .....	3
2.1.	Đề bài .....	3
2.2.	Thiết kế SOLIDWORKS .....	3
2.2.1.	Phần thân xe .....	3
2.2.2.	Phần tay máy .....	4
2.2.3.	Phần bánh xe mecanum .....	7
2.2.4.	Mô hình hoàn chỉnh .....	8
2.3.	Đặt hệ trục tọa độ .....	9
3.	Mô phỏng trong với ROS .....	10
4.	Điều khiển robot .....	18
4.1.	Các cảm biến .....	18
4.2.	Xe di chuyển .....	21
5.	Cấu trúc của dự án .....	24
6.	Các vấn đề và kết luận .....	24

## 1. Mở đầu

Báo cáo giữa kỳ cho bộ môn Lập trình Robot với ROS trình bày quá trình xây dựng, mô phỏng và điều khiển robot với ROS.

Quá trình thực hiện bao gồm:

1. Xây dựng mô hình robot bằng phần mềm SOLIDWORKS
2. Xuất mô hình thành file URDF qua phần mềm SOLIDWORKS
3. Mô phỏng và điều khiển robot trong ROS bằng file URDF.

## 2. Xây dựng mô hình robot

### 2.1. Đề bài

Đề bài cho robot cần được xây dựng:

1. Loại di chuyển: Omnidirectional (mecanum, 4 bánh)
2. Tay máy:
  - a. Khớp 1: Rotation
  - b. Khớp 2: Rotation
3. Cảm biến: LiDAR, IMU, Camera

Robot di chuyển đa hướng với 4 bánh mecanum, kết hợp một cơ cấu tay máy hai bậc tự do và gắn ba cảm biến là LiDAR, IMU, Camera.

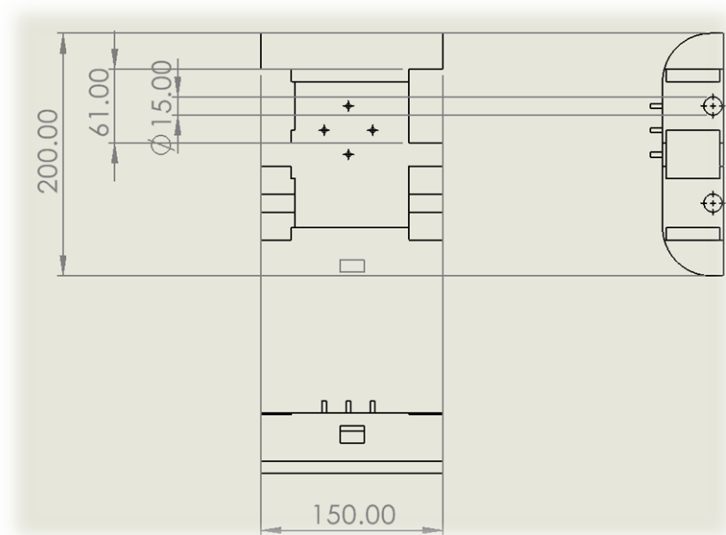
### 2.2. Thiết kế SOLIDWORKS

Robot sẽ bao gồm 3 phần chính: thân xe (base\_link), 2 khớp tay và bánh xe mecanum.

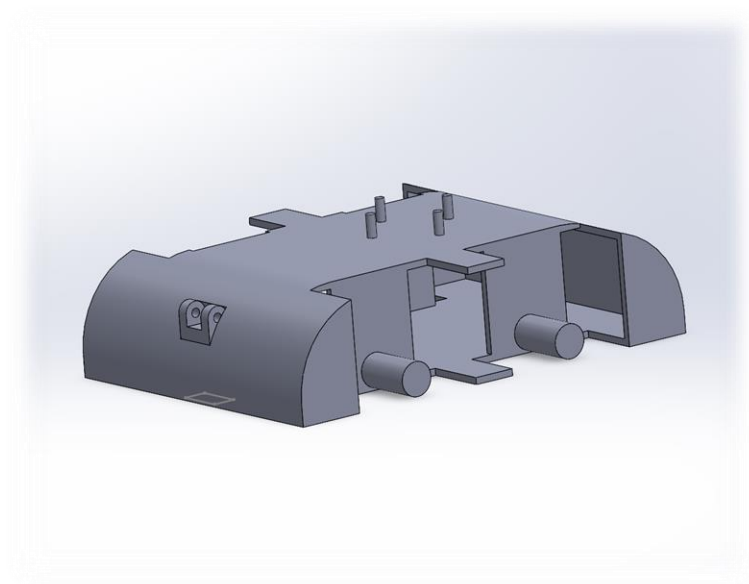
#### 2.2.1. Phần thân xe

Phần thân xe sẽ được thiết kế với bốn trục để gắn bánh xe, phần đầu để gắn tay máy và phần thân để gắn các cảm biến: Camera và LiDAR được gắn trên nóc xe và IMU được gắn trong thân xe.

Phần đầu thân xe được gắn thêm một liên kết dùng để liên kết tay máy và thân xe, coi đó là một phần của thân xe.



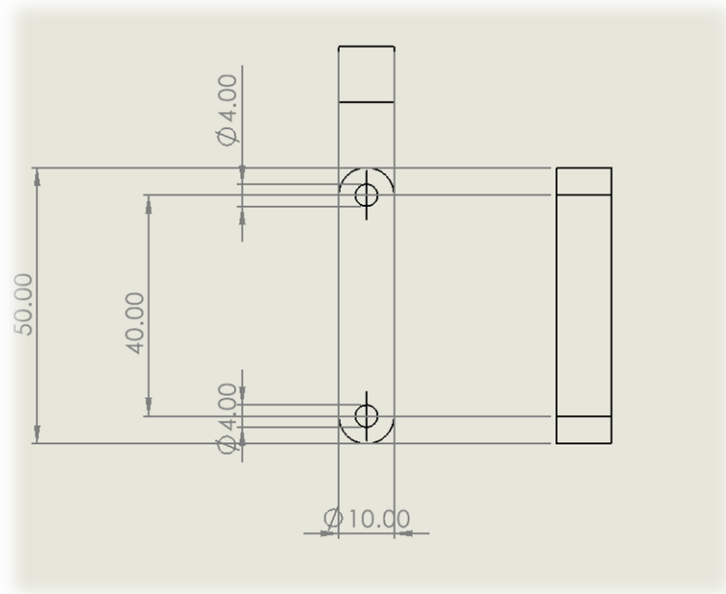
Hình 2.1: Bản vẽ mặt cắt của thân xe



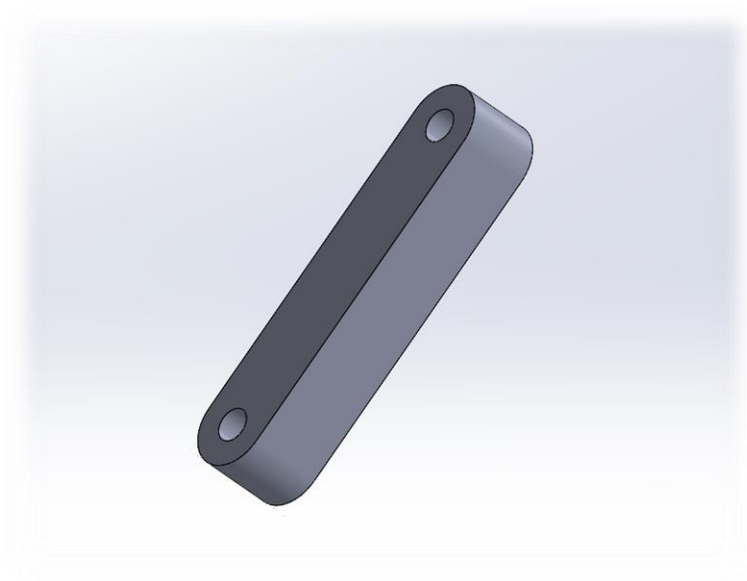
Hình 2.2: Mô hình thân xe

### 2.2.2. Phần tay máy

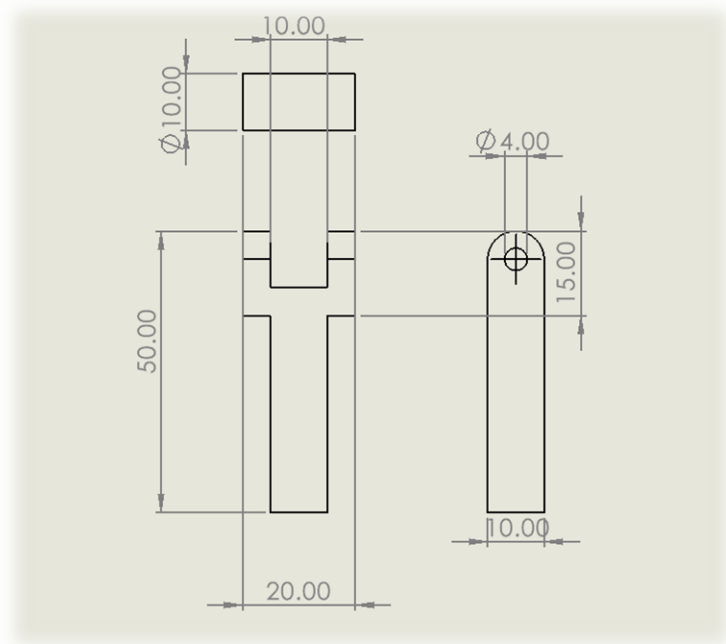
Phần tay máy hai bậc tự do gồm hai thanh là arm\_link1 và arm\_link2. Để hạn chế việc tay máy chắn tầm nhìn của Camera và tầm quét của cảm biến LiDAR, tay máy được đặt ở đầu xe và giới hạn góc quay của hai khớp.



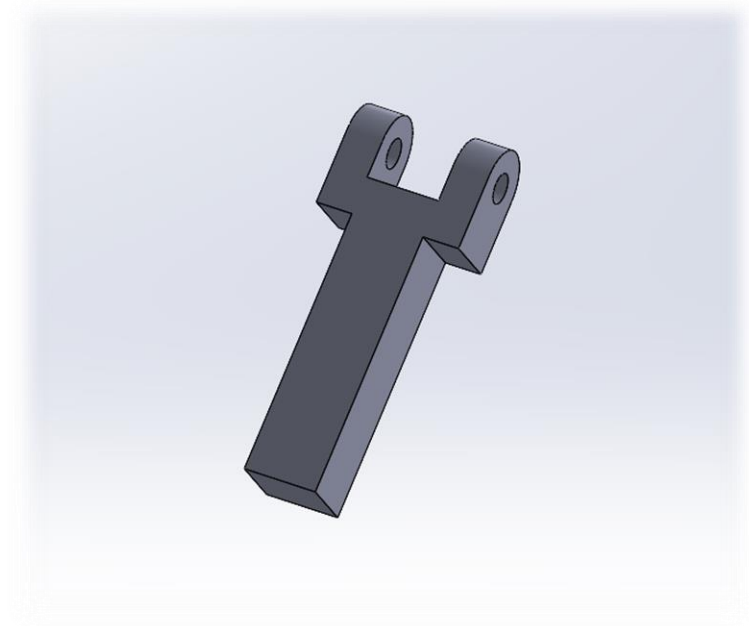
Hình 2.3: Bản vẽ mặt cắt của arm\_link1



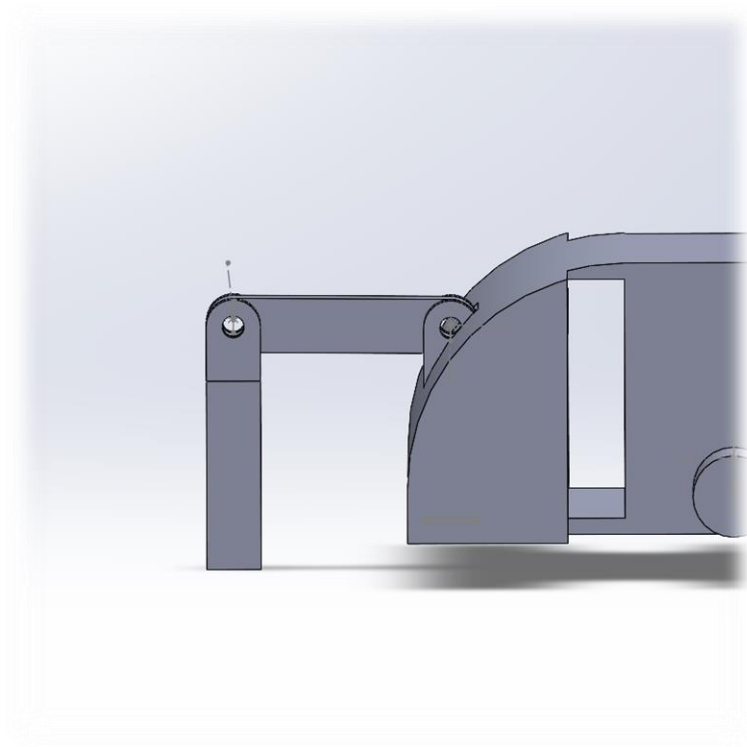
Hình 2.4: Mô hình arm\_link1



Hình 2.4: Bản vẽ mặt cắt của arm\_link2



Hình 2.5: Mô hình arm\_link2



Hình 2.5: Mô hình xe gắn tay máy

Vị trí đặt tay máy trong Hình 2.5 là vị trí gốc của tay máy. Từ vị trí ban đầu này, arm\_link1 có giới hạn góc quay là  $-0.53$  đến  $0.53$  (rad), arm\_link2 có giới hạn góc quay là  $-1.57$  đến  $0$  (rad).

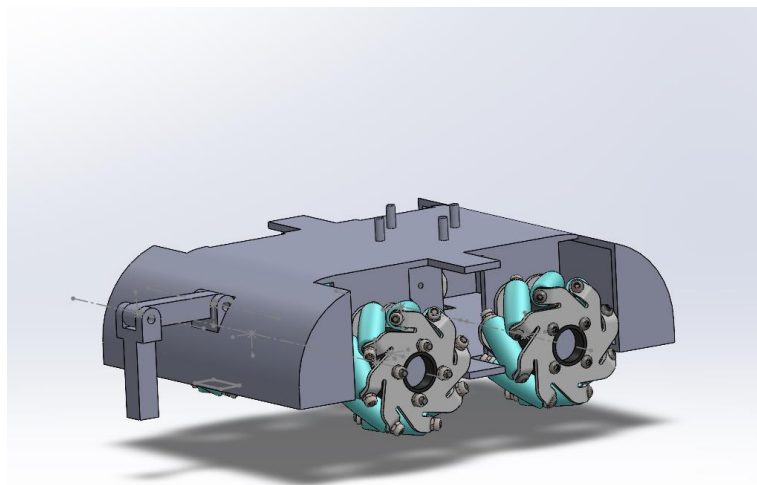
### 2.2.3. Phần bánh xe mecanum

Bánh xe mecanum phục vụ cho di chuyển đa hướng.



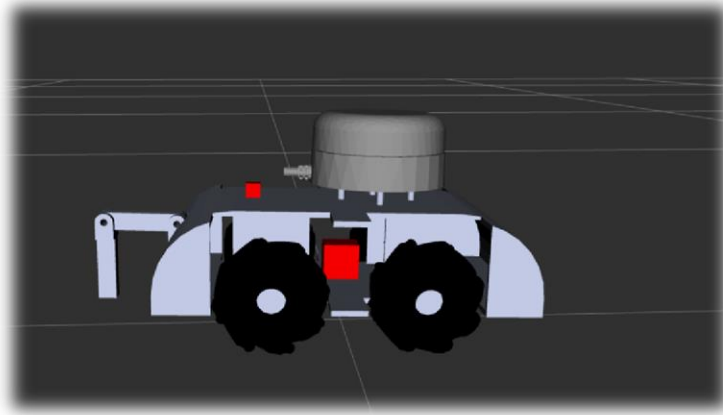
Hình 2.6: Mô hình bánh xe mecanum

#### 2.2.4. Mô hình hoàn chỉnh



Hình 2.7: Mô hình hoàn chỉnh của robot

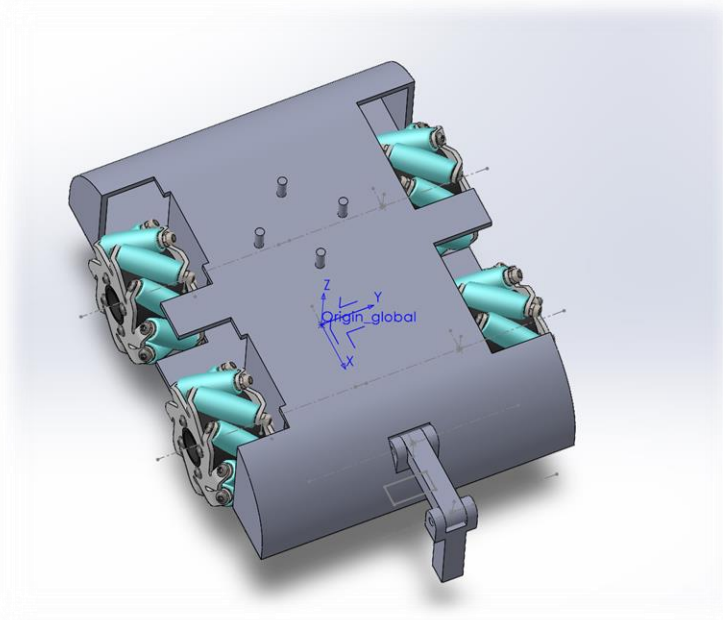




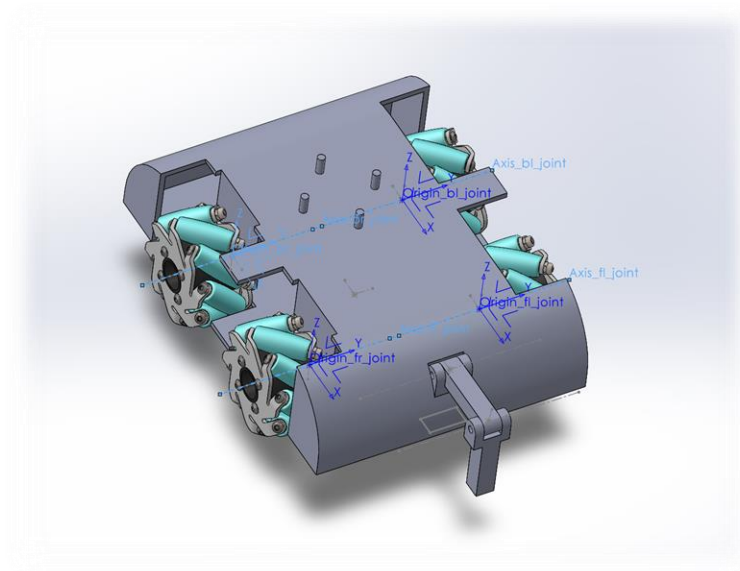
Mô hình robot hoàn chỉnh sau khi gắn các cảm biến: trong đó Camera gắn ở phía trên trước xe, LiDAR ở phía sau và IMU được gắn bên trong thân xe.

### 2.3. Đặt hệ trục tọa độ

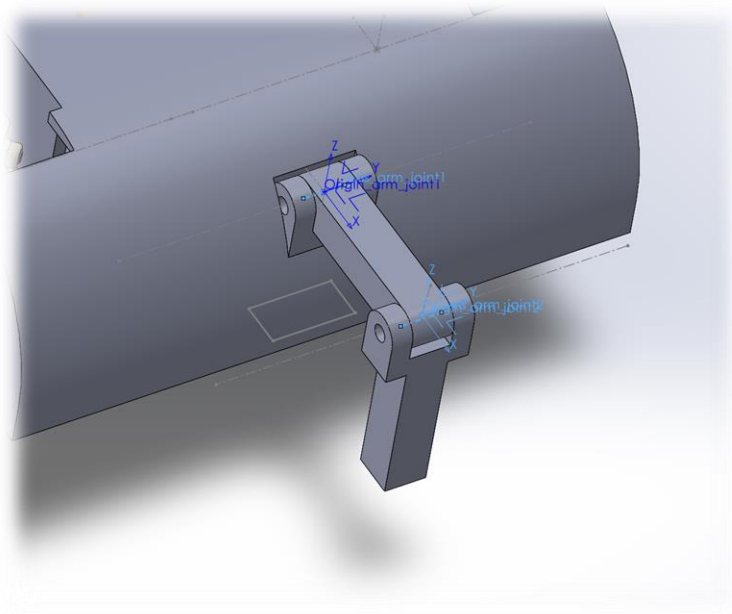
Ta đặt hệ trục tọa độ sao cho xe tiến tiến theo phương trục x, trục quay của bánh xe và tay máy có phương trùng với phương của trục y. Ta có hệ tọa độ base\_link như Hình 2.8 cùng các hệ tọa độ, trục quay của bánh xe và tay máy lần lượt ở Hình 2.9 và Hình 2.10.



Hình 2.8: Hệ tọa độ base\_link



Hình 2.9: Hệ tọa độ và trục quay của 4 bánh xe



Hình 2.10: Hệ tọa độ và trục quay của 2 khớp tay máy

### 3. Mô phỏng trong với ROS

Folder URDF chứa file [omnifinal.xacro](#) và file [omnifinal.gazebo](#). File ,xacro được xây dựng từ file gốc omnifinal.urdf định nghĩa robot trong ros và file .gazebo để cấu hình thông số vật lý, plugin và môi trường mô của robot trong Gazebo. Cả hai file đều cần thiết cho quá trình mô phỏng robot trong ROS, Gazebo.

Đầu tiên ta cần nhúng file .gazebo vào file .xacro để phục vụ cho việc mô phỏng trong Gazebo.

```
<!-- Include the Gazebo-specific configurations -->  
<xacro:include filename="$(find omnifinal)/urdf/omnifinal.gazebo" />
```

File .xacro mô tả robot với link gốc là base\_link với các tag <inertial> cho biết quán momen quán tính của base\_link, <mass> cho biết khối lượng, <mesh> nhúng file mô hình 3D, <material> / <color> định nghĩa chất liệu, màu sắc. Trong file .xacro base\_link được nối với các link khác bằng các joint được định nghĩa cho các link đó.

Base\_link nối với base\_footprint (hình chiếu của base\_link lên mặt phẳng) bằng base\_footprint\_joint.

```
<!-- root link, on the ground just below the model origin -->
<link name="base_footprint"></link>

  <joint name="base_footprint_joint" type="fixed">
    <origin xyz="0 0 0" rpy="0 0 0" />
    <parent link="base_footprint" />
    <child link="base_link" />
  </joint>
```

```
<link
  name="base_link">
  <inertial>
    <origin
      xyz="0.00211440695779285 -0.000172345786933302 0.0214923159791835"
      rpy="0 0 0" />
    <mass
      value="0.213265827988128" />
    <inertia
      ixx="0.000459624639615948"
      ixy="-6.64436047700312E-08"
      ixz="4.28549936785738E-06"
      iyy="0.00096819045435354"
      iyz="1.42696555292235E-07"
      izz="0.00127034015073323" />
    </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://omnifinal/meshes/base_link.STL" />
      </geometry>
    <material
      name="gray">
      <color
        rgba="0.792156862745098 0.819607843137255 0.933333333333333 1" />
      </material>
    </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://omnifinal/meshes/base_link.STL" />
      </geometry>
    </collision>
  </link>
```

Và cứ thế base\_link nối với 4 bánh xe và 2 khớp tay máy là: fl\_link, fr\_link, bl\_link, br\_link, arm\_link1, arm\_link2 lần lượt bằng fl\_joint, fr\_joint, bl\_joint, br\_joint, arm\_joint1, arm\_joint2. Đối với các khớp xoay như bánh xe và tay máy trong tag <joint> còn được định nghĩa loại joint type = “continuous” với bánh xe và type = “revolute” với tay máy.

```
<joint
  name="br_joint"
  type="continuous">
  <origin
    xyz="-0.04 -0.0474 0.0085"
    rpy="0 0 0" />
  <parent
    link="base_link" />
  <child
    link="br_link" />
  <axis
    xyz="0 1 0" />
</joint>
```

```
<joint
  name="arm_joint1"
  type="revolute">
  <origin
    xyz="0.092 0 0.036"
    rpy="0 0 0" />
  <parent
    link="base_link" />
  <child
    link="arm_link1" />
  <axis
    xyz="0 1 0" />
  <limit
    lower="-0.53"
    upper="0.53"
    effort="0.3"
    velocity="0.2" />
</joint>
```

Trong file .xacro còn chứa thông tin về các cảm biến được gắn vào robot. Cũng tương tự như bánh xe hay tay máy. Mỗi cảm biến được nối với base\_link bằng các joint được định nghĩa riêng. Và trong đó các tag quan trọng là tag <origin> nằm trong tag <joint> cho biết tọa độ vị trí của cảm biến; <parent link> và <child link> cho biết hai link được liên kết với joint, trong đó parent thường là base\_link và child là các link cảm biến. Ta ví dụ với cảm biến LiDAR như sau:

```

    <!-- rplidar Laser joint -->
    <joint name="rplidar_joint" type="fixed">
        <axis xyz="0 1 0" />
        <origin xyz="-0.02 0 0.06" rpy="0 0 0"/>
        <parent link="base_link"/>
        <child link="laser"/>
    </joint>

    <!-- rplidar Laser link -->
    <link name="laser">
        <collision>
            <origin xyz="-0.1 0 0.055" rpy="1.5707 0 3.1412"/>
            <geometry>
                <mesh filename="package://omnifinal/meshes/rplidar.dae" scale="0.001 0.001 0.001" />
            </geometry>
        </collision>

        <visual>
            <origin xyz="0 0 0" rpy="1.5707 0 4.71"/>
            <geometry>
                <mesh filename="package://omnifinal/meshes/rplidar.dae" scale="0.001 0.001 0.001" />
            </geometry>
        </visual>
        <material name="black"/>
    </link>

    <inertial>
        <mass value="{1e-6*scale}" />
        <origin xyz="0 0 0.058" rpy="1.5707 0 4.71"/>
        <inertia ixx="0" ixy="0" ixz="0" iyy="0" iyz="0" izz="0" />
    </inertial>
</link>

```

File .gazebo là một file để cấu hình mô phỏng robot trong Gazebo. Bao gồm từ ngoại hình, màu sắc, môi trường của các link cho đến các plugin cho cảm biến và điều khiển.

Ta gọi các tag <gazebo reference> để áp dụng cấu hình cho các link của robot: <material> là màu sắc...



```

<gazebo reference="base_link">
  <material>Gazebo/Gray</material>
  <turnGravityOff>>false</turnGravityOff>
</gazebo>
<gazebo reference="fl_link">
  <material>Gazebo/Black</material>
  <turnGravityOff>>false</turnGravityOff>
</gazebo>
<gazebo reference="fr_link">
  <material>Gazebo/Black</material>
  <turnGravityOff>>false</turnGravityOff>
</gazebo>
<gazebo reference="bl_link">
  <material>Gazebo/Black</material>
  <turnGravityOff>>false</turnGravityOff>
</gazebo>
<gazebo reference="br_link">
  <material>Gazebo/Black</material>
  <turnGravityOff>>false</turnGravityOff>
</gazebo>
<gazebo reference="arm_link1">
  <material>Gazebo/Gray</material>
  <turnGravityOff>>false</turnGravityOff>
</gazebo>
<gazebo reference="arm_link2">
  <material>Gazebo/Gray</material>
  <turnGravityOff>>false</turnGravityOff>
</gazebo>

```

Plugin cho cảm biến Camera, IMU, LiDAR ta quan tâm đến các tag <topicName> cho biết tên các topic mà dữ liệu từ các cảm biến sẽ được publish đến.

```

<plugin name="imu_plugin" filename="libgazebo_ros_imu.so">
  <alwaysOn>true</alwaysOn>
  <bodyName>base_link</bodyName>
  <topicName>/imu/data</topicName>
  <serviceName>/imu/service</serviceName>
  <gaussianNoise>0.001</gaussianNoise>
  <updateRate>50.0</updateRate>
</plugin>

```

```

</noise>
</ray>
<plugin name="gazebo_ros_head_rplidar_controller" filename="libgazebo_ros_laser.so">
  <topicName>scan</topicName>
  <frameName>laser</frameName>
</plugin>

```

```

<updateRate>30.0</updateRate>
<cameraName>omnirobot/camera1</cameraName>
<imageTopicName>/camera/image_raw</imageTopicName>
<cameraInfoTopicName>/camera/camera_info</cameraInfoTopicName>
<frameName>camera_link</frameName>
<hackBaseline>0.07</hackBaseline>

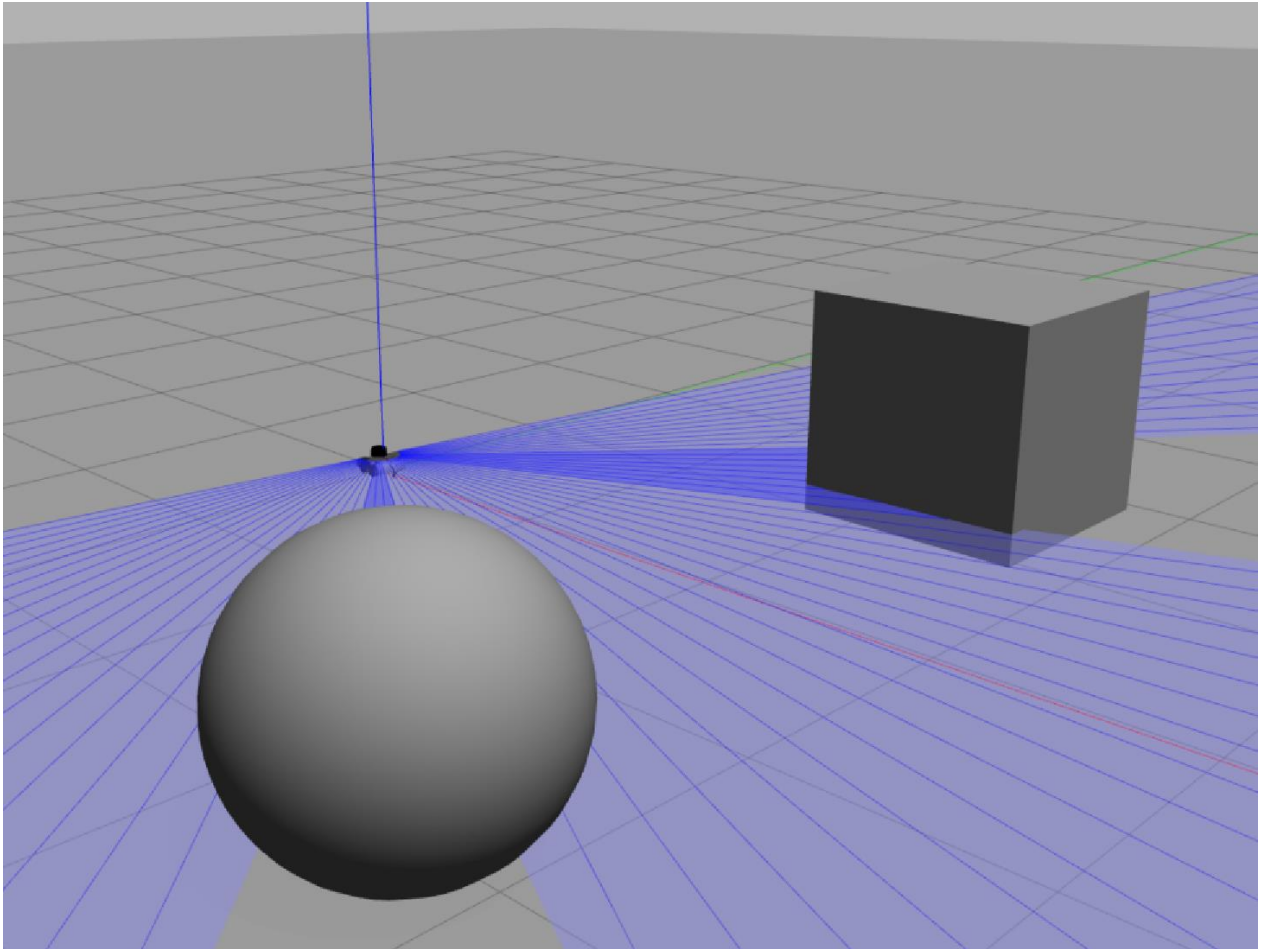
```

## 4. Điều khiển robot

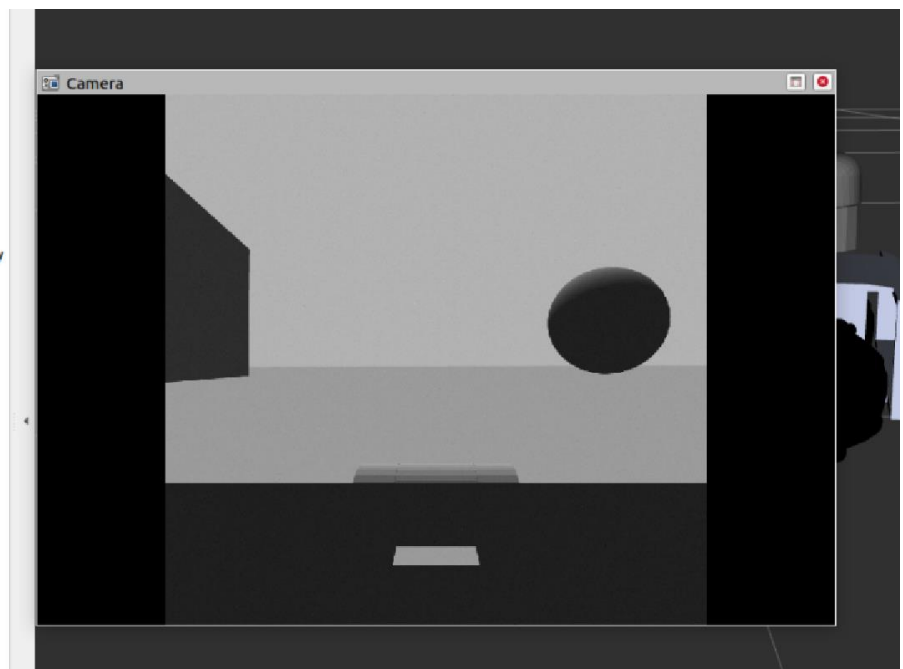
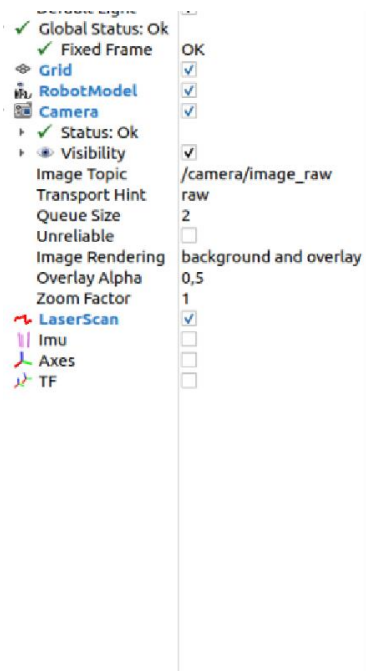
### 4.1. Các cảm biến

Đối với các cảm biến, sử dụng các plugin để điều khiển và lấy dữ liệu. Camera sẽ gửi hình ảnh qua topic /camera/image\_raw, IMU gửi dữ liệu qua topic /imu/data và LiDAR sẽ gửi dữ liệu qua topic /scan.

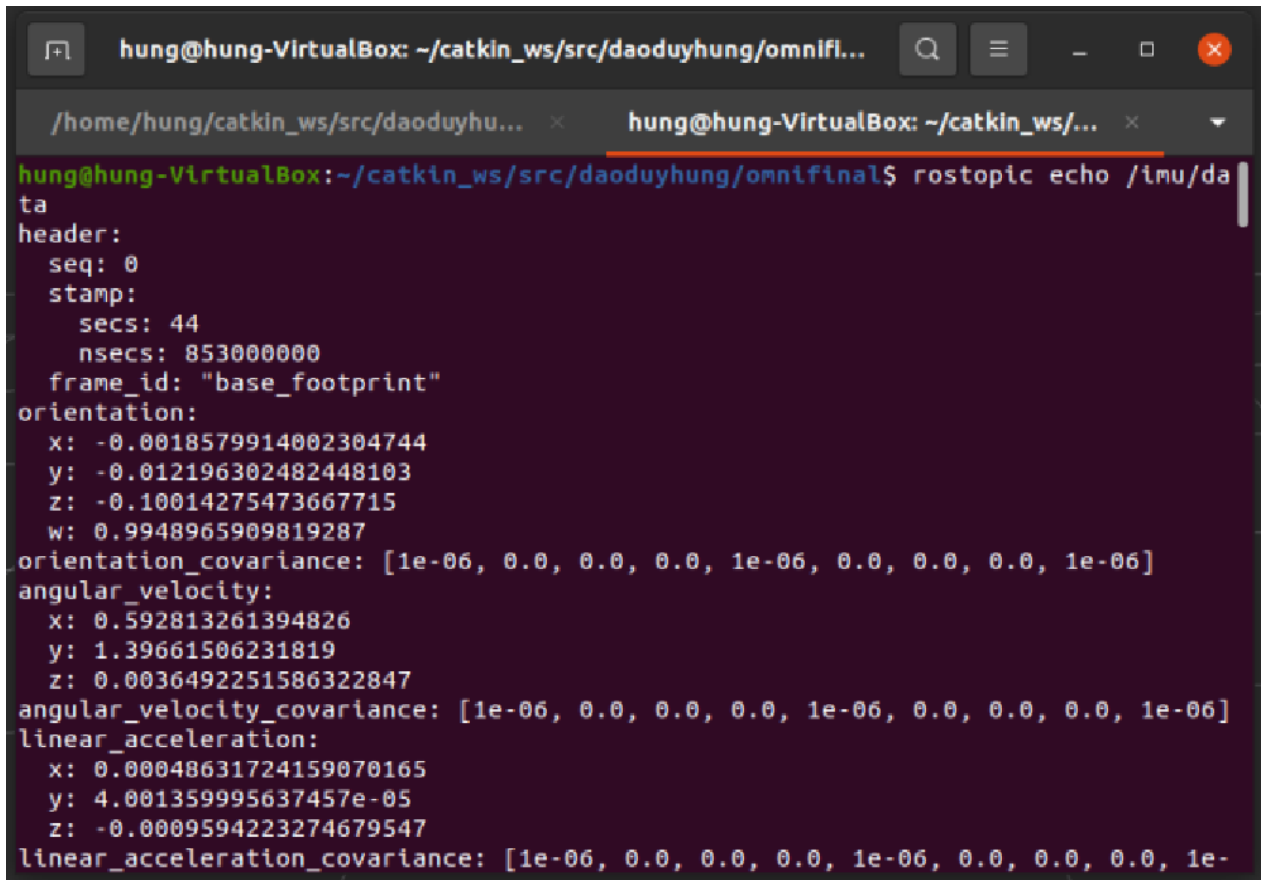
Kết quả của mô phỏng các cảm biến như sau:



Hình 4.1: Thiết lập trong Gazebo

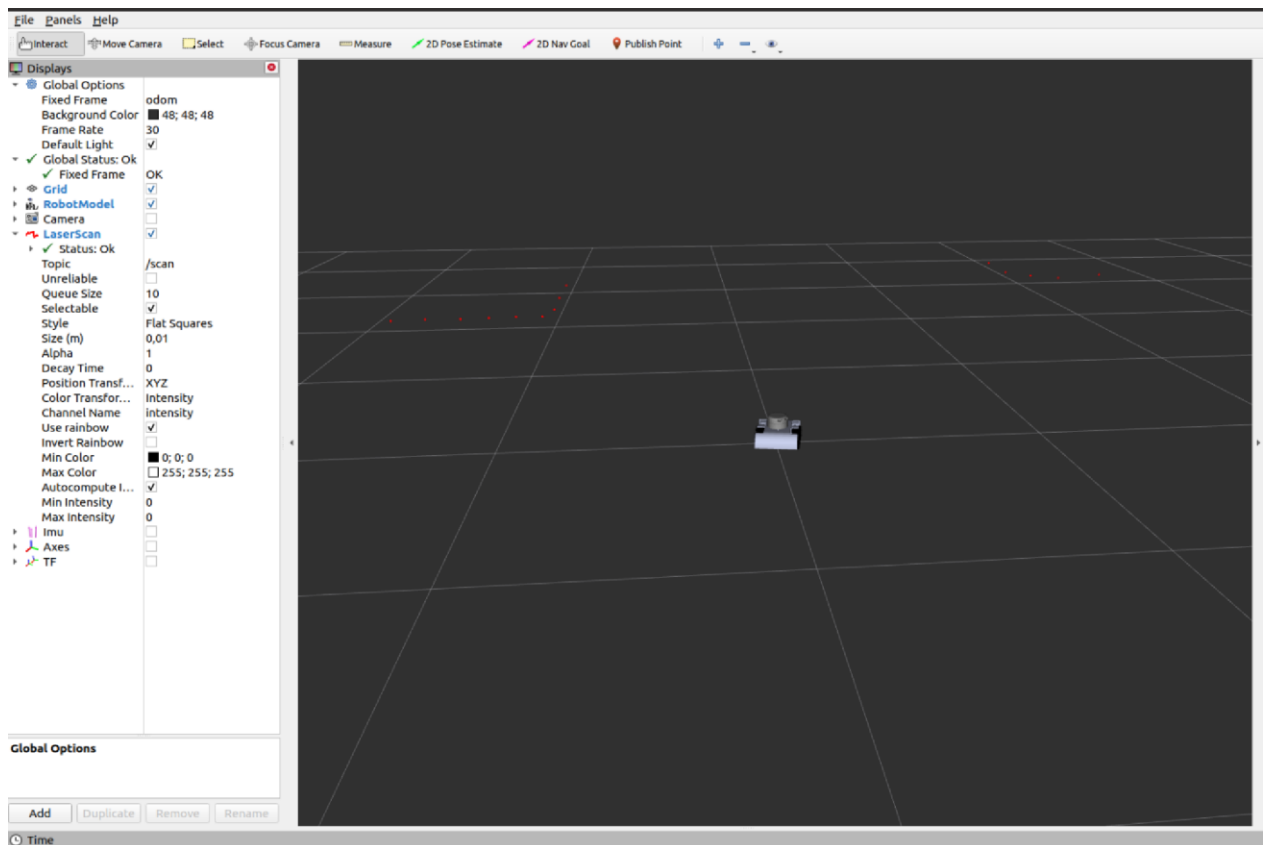


Hình 4.2: Hình ảnh thu được từ Camera

A terminal window titled 'hung@hung-VirtualBox: ~/catkin\_ws/src/daoduyhung/omnifl...' displays the output of the command 'rostopic echo /imu/data'. The output shows a series of sensor data fields including header information (seq, stamp, frame\_id), orientation (x, y, z, w), orientation covariance, angular velocity (x, y, z), angular velocity covariance, linear acceleration (x, y, z), and linear acceleration covariance. The terminal has a dark background with light-colored text.

```
hung@hung-VirtualBox:~/catkin_ws/src/daoduyhung/omnifl...$ rostopic echo /imu/data
header:
  seq: 0
  stamp:
    secs: 44
    nsecs: 853000000
  frame_id: "base_footprint"
orientation:
  x: -0.0018579914002304744
  y: -0.012196302482448103
  z: -0.10014275473667715
  w: 0.9948965909819287
orientation_covariance: [1e-06, 0.0, 0.0, 0.0, 1e-06, 0.0, 0.0, 0.0, 1e-06]
angular_velocity:
  x: 0.592813261394826
  y: 1.39661506231819
  z: 0.0036492251586322847
angular_velocity_covariance: [1e-06, 0.0, 0.0, 0.0, 1e-06, 0.0, 0.0, 0.0, 1e-06]
linear_acceleration:
  x: 0.00048631724159070165
  y: 4.001359995637457e-05
  z: -0.0009594223274679547
linear_acceleration_covariance: [1e-06, 0.0, 0.0, 0.0, 1e-06, 0.0, 0.0, 0.0, 1e-
```

Hình 4.3: Dữ liệu thu được từ IMU



Hình 4.4: Kết quả quét của LiDAR

Với cảm biến LiDAR, tại chỗ có các vật cản trong Gazebo, trong Rviz ta sẽ thấy chúng được đánh dấu bằng các dấu chấm màu đỏ. Do LiDAR đang sử dụng chỉ quét trên một mặt phẳng hai chiều.

## 4.2. Xe di chuyển

Với việc điều khiển xe di chuyển ta sẽ điều khiển bằng các phím nhận vào trên bàn phím. Sử dụng plugin Planar move kết hợp với viết code điều khiển.

```
<!-- planar plugin-->
<gazebo>
  <plugin name="object_controller" filename="libgazebo_ros_planar_move.so">
    <commandTopic>cmd_vel</commandTopic>
    <odometryTopic>odom</odometryTopic>
    <odometryFrame>odom</odometryFrame>
    <odometryRate>20.0</odometryRate>
    <robotBaseFrame>base_footprint</robotBaseFrame>
  </plugin>
</gazebo>
```

Hình 4.5: Code plugin Planar move file .gazebo

Code điều khiển xe [keyboard\\_teleop.cpp](#), khi nhận được tín hiệu nút bấm từ bàn phím, xe sẽ di chuyển theo code trong file.

```
// Map for movement keys
std::map<char, std::vector<float>> moveBindings
{
    {'i', {1, 0, 0, 0}}, // Tiến theo trục x
    {'j', {-1, 0, 0, 0}}, // Lùi theo trục x
    {'k', {0, 1, 0, 0}}, // Ngang trái theo trục y
    {'l', {0, -1, 0, 0}}, // Ngang phải theo trục y
    {'u', {0, 0, 0, 1}}, // Xoay trái
    {'o', {0, 0, 0, -1}}, // Xoay phải
    {'k', {0, 0, 0, 0}}, // Dừng lại
};

// Map for speed keys
std::map<char, std::vector<float>> speedBindings
{
    {'q', {1.1, 1.1}},
    {'z', {0.9, 0.9}}
};

// Map for duration adjustment keys
std::map<char, float> durationBindings
{
    {'w', 1.0}, // Tăng thời gian
    {'x', -1.0} // Giảm thời gian
};
```

Đoạn code trên định nghĩa một bảng ánh xạ với mỗi nút điều khiển. Mỗi phím sẽ tương ứng với 4 thông tin liên quan đến vận tốc trên các trục x, y, z, yaw. Trong đó x, y, z cho biết vận tốc tịnh tiến trên các trục x, y, z và yaw cho biết vận tốc góc trên trục yaw.

```

key = getch();
if (moveBindings.count(key) == 1)
{
    x = moveBindings[key][0];
    y = moveBindings[key][1];
    z = moveBindings[key][2];
    th = moveBindings[key][3];
    printf("\rCurrent: speed %f m/s\tturn %f rad/s | Last command: %c ", speed, turn, key);

    twist.linear.x = x * speed;
    twist.linear.y = y * speed;
    twist.linear.z = 0;
    twist.angular.x = 0;
    twist.angular.y = 0;
    twist.angular.z = th * turn;
}

```

Mỗi khi nhận được tín hiệu từ một nút bấm, từ vector được ánh xạ mà gán các giá trị của vector đó lần lượt cho các biến x, y, z, th tương ứng với vận tốc tịnh tiến trên 3 trục x, y, z và vận tốc góc trên trục yaw. Và từ đó tiến hành điều khiển bằng twist.linear và twist.angular.

Với các nút q, z, w và x sẽ định nghĩa các cài đặt điều chỉnh vận tốc và thời gian xe chạy mỗi khi bấm phím.

Ta mong muốn mỗi khi bấm nút xe sẽ chạy với một khoảng thời gian mong muốn, do đó ta sẽ cài đặt sao cho xe dừng sau khoảng khoảng thời gian bằng với giá trị của biến move\_duration.

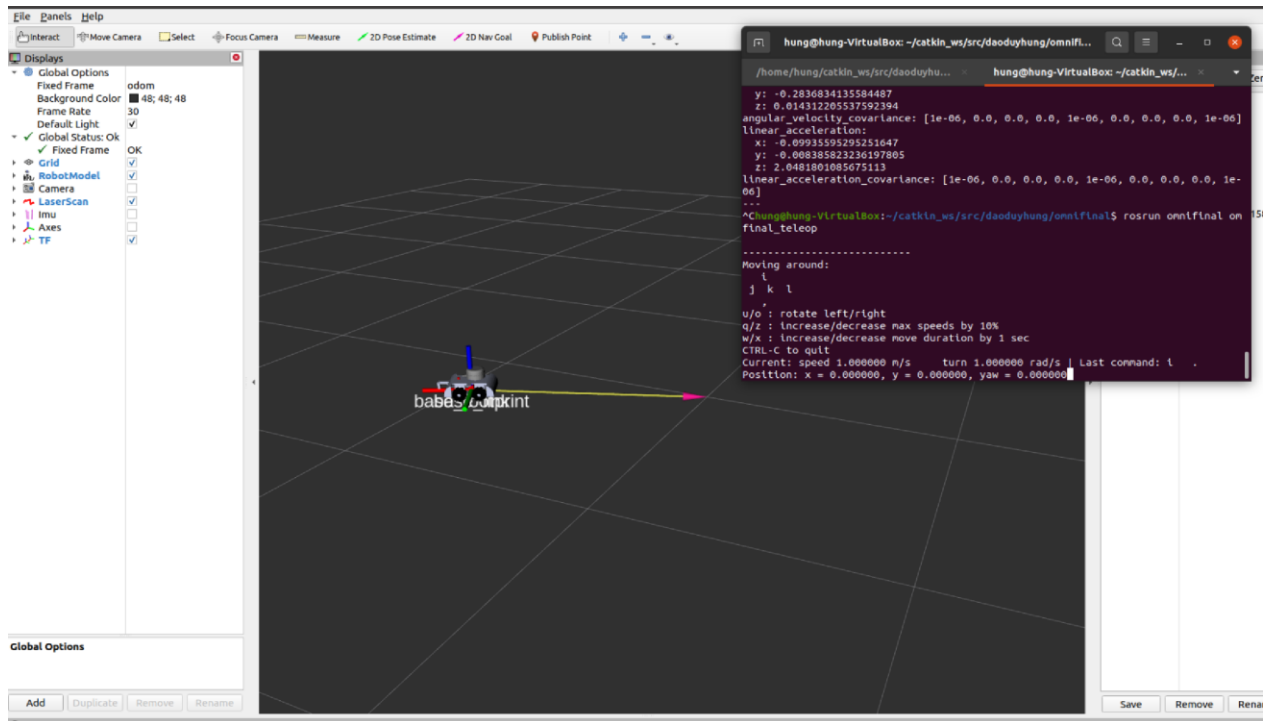
```

pub.publish(twist);
ros::Duration(move_duration).sleep(); // Dừng sau khoảng thời gian di chuyển
twist.linear.x = 0;
twist.linear.y = 0;
twist.angular.z = 0;
pub.publish(twist);

```

Kết quả điều khiển:

Xe di chuyển với tốc độ 1m/s trong khoảng thời gian 1 giây.



Hình 4.6: Điều khiển robot

## 5. Cấu trúc của dự án

Package của dự án tên là omnifinal chứa các folder sau:

1. config: chứa file cấu hình cho Rviz ...
2. launch: chứa các file .launch để chạy trình mô phỏng
3. mesh: chứa các file .STL, .DAE cấu hình mô hình 3D của robot
4. src: chứa các file code điều khiển
5. urdf: chứa file .xacro và file .gazebo

## 6. Các vấn đề và kết luận

Về mặt thiết kế robot còn những điểm trừ như vị trí đặt tay máy chưa thực sự tối ưu, vẫn còn sự hạn chế đến tầm hoạt động của Camera. Về mặt mô phỏng, điều khiển chưa thực sự đúng về mặt vật lý, khi điều khiển robot đi thẳng với khoảng cách dài có thiên hướng lệch sang phải.