

BÁO CÁO GIỮA KỲ ROS

Họ và tên: Phạm Văn Bách

Mã sinh viên: 22027539

I. Giới thiệu chung về robot

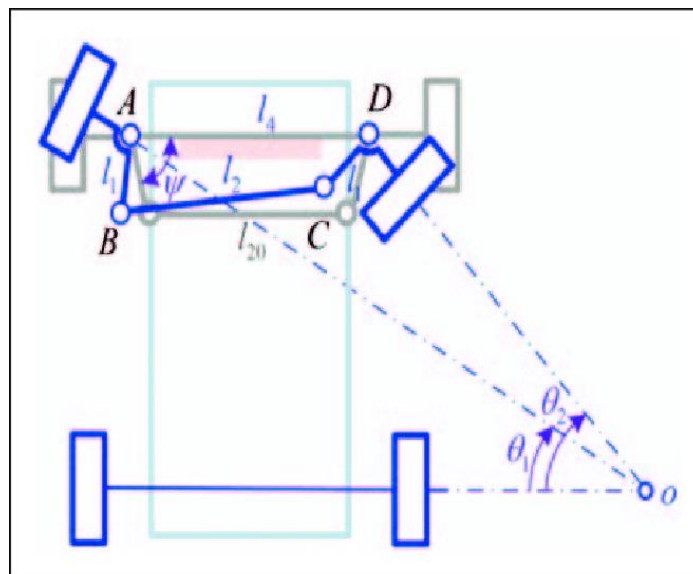
1. Dạng robot

Đề tài giữa kỳ được lựa chọn là robot di chuyển bằng bốn bánh theo thiết kế Ackermann Steering bên trên gắn tay máy hai khớp quay. Ba cảm biến được lựa chọn gồm có lidar, camera và encoder.

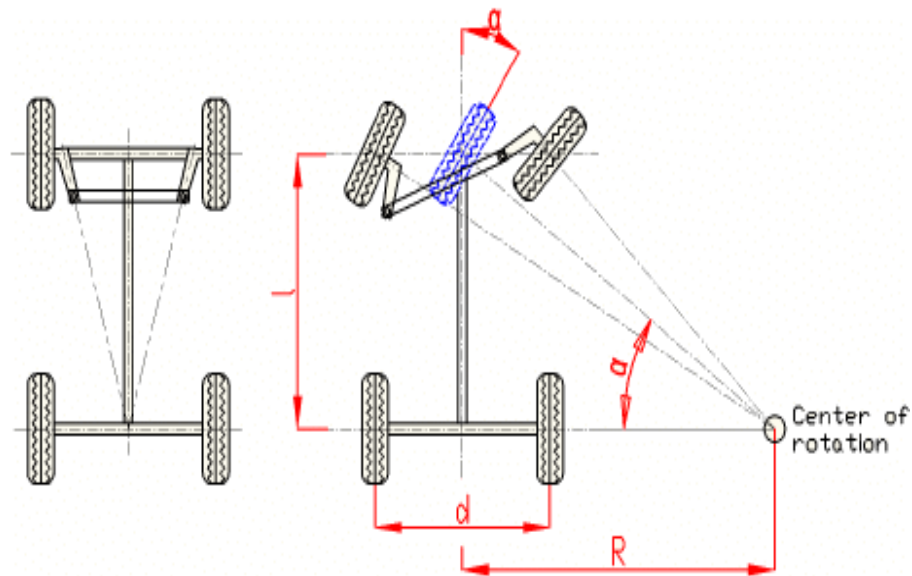
2. Động học

- Động học di chuyển: Ackermann Steering là một nguyên lý điều khiển dựa trên kết nối của các link trong xe. Mục đích của Ackermann Steering là để cho các bánh dẫn đường của xe có bán kính quay khác nhau khi vào dẫn cua nhằm tăng độ ổn định.

Hai bánh trước dẫn đường của Ackermann Steering khi di chuyển đều quay chung quanh một tâm. Hướng quay và bán kính quay của hai bánh là khác nhau. Trong điều khiển lý tưởng bánh trong gần tâm quay nhất sẽ có góc quay lớn hơn bánh ngoài. Ta có cơ chế chuyển động trong điều kiện lý tưởng

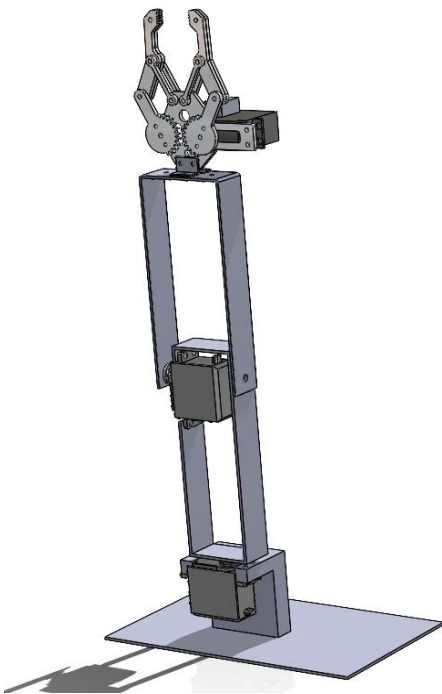


Đề điều khiển cho hai bánh dẫn đường quay được với góc quay khác nhau, hai bánh được nối với nhau bằng một kết cấu hình thang cân. Hai cạnh bên của hình thang nếu kéo dài ra sẽ gặp nhau tại trung điểm trục bánh xe sau.



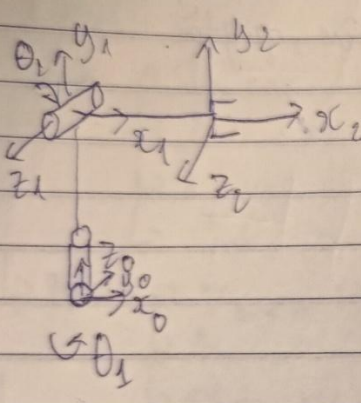
- Động học tay máy

Tay máy Rotation - Rotation là tay máy hai khớp quay với góc quay.



Đặt các trục cho tay máy như hình vẽ ta có thể xác định được bảng D-H cho tay máy như sau.
 Dựa trên bảng D-H đã xác định ta tính được động học thuận cho tay máy.

Date _____ No. _____



\Rightarrow D-H table

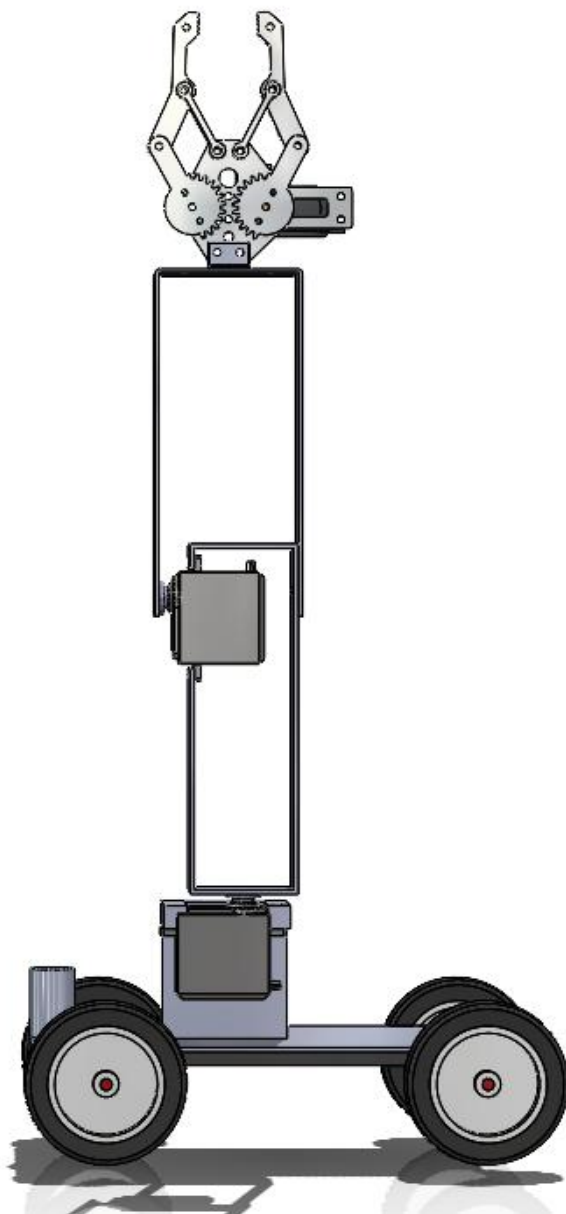
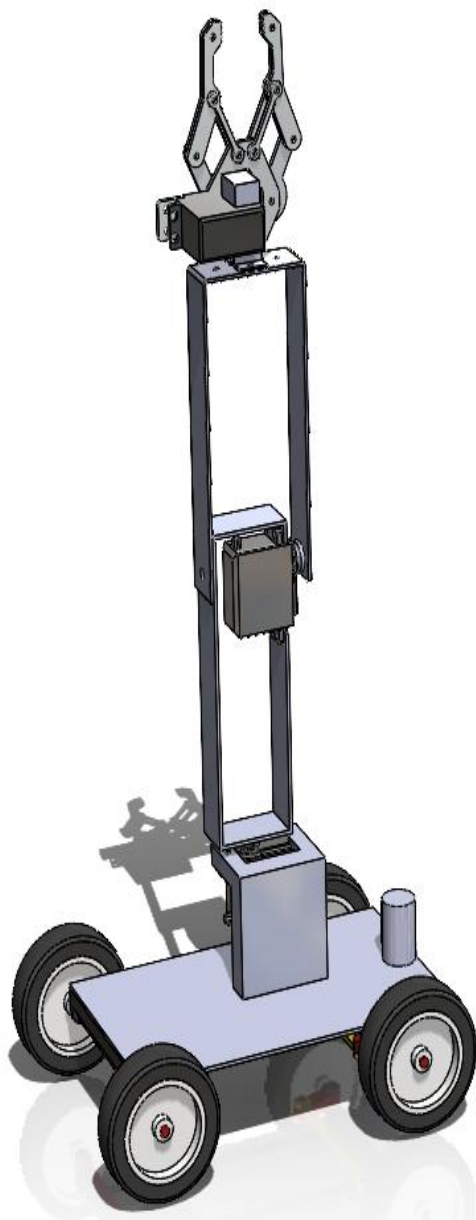
i	θ_i	α_i	a_i	d_i
1	θ_1	$\pi/2$	0	l_1
2	θ_2	0	l_2	0

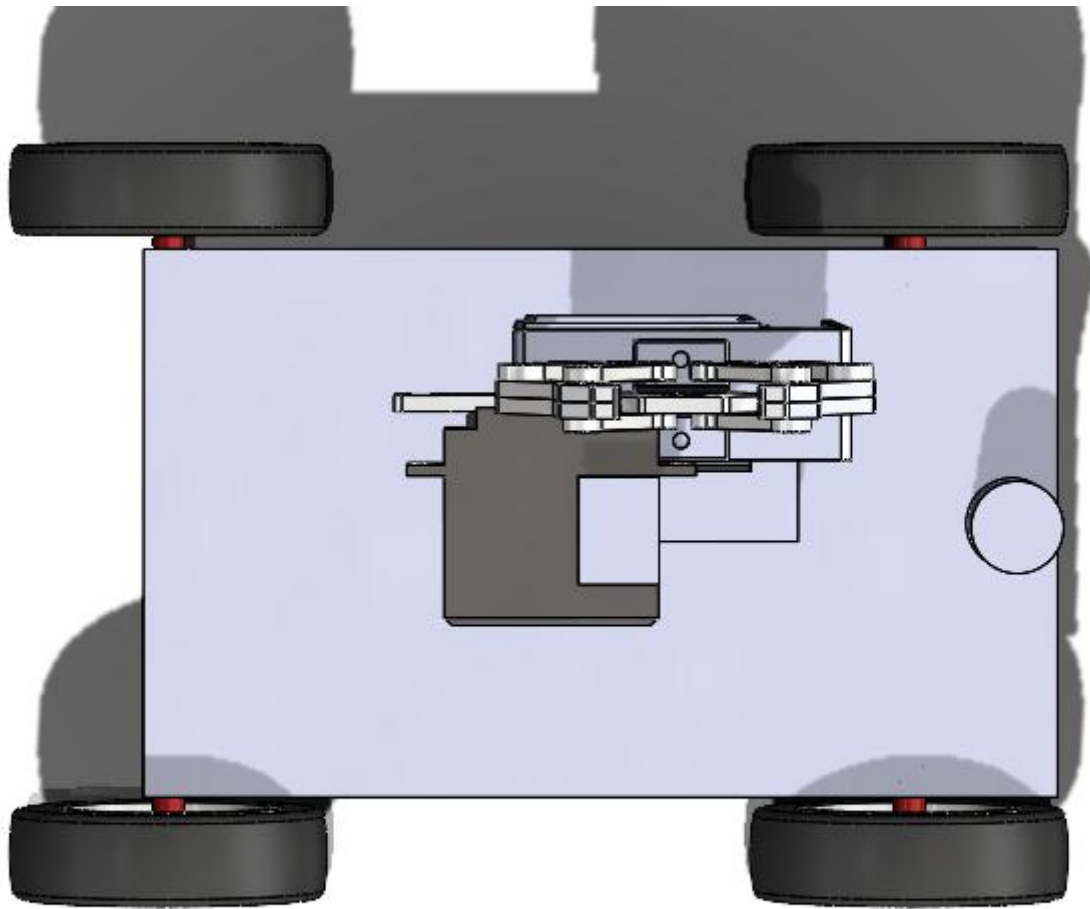
\Rightarrow Ta tính được tọa độ EF

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} l_2 \cos \theta_2 \cdot \cos \theta_1 \\ l_2 \cos \theta_2 \cdot \sin \theta_1 \\ l_1 + l_2 \sin \theta_2 \end{bmatrix}$$

II. Thiết kế

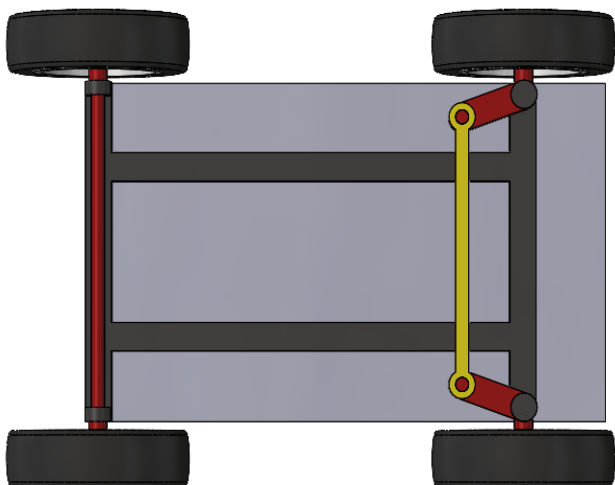
1. Bản vẽ 3D:



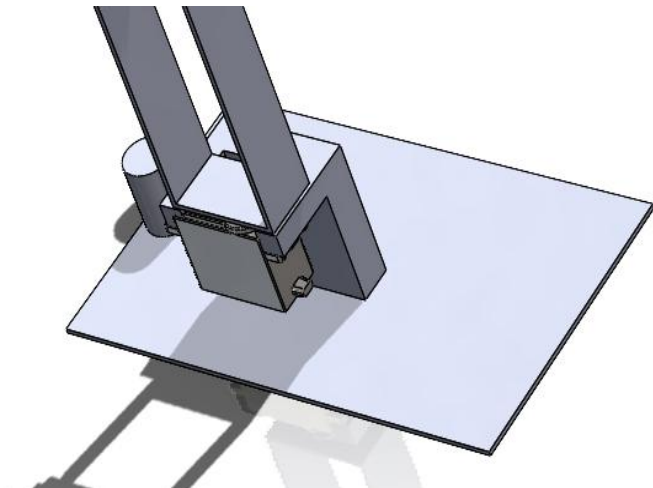


- Khung cơ cấu ackerman:

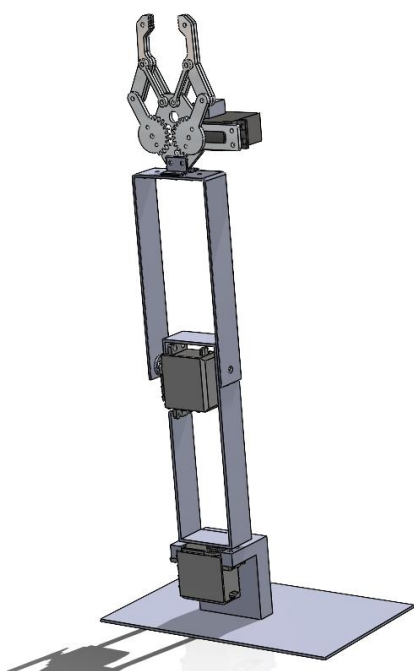
- Khoảng cách giữa trục bánh trước và trục bánh sau là 163mm
- Khoảng cách giữa 2 bánh cùng trục là 166mm
- Bánh xe có đường kính là 7cm



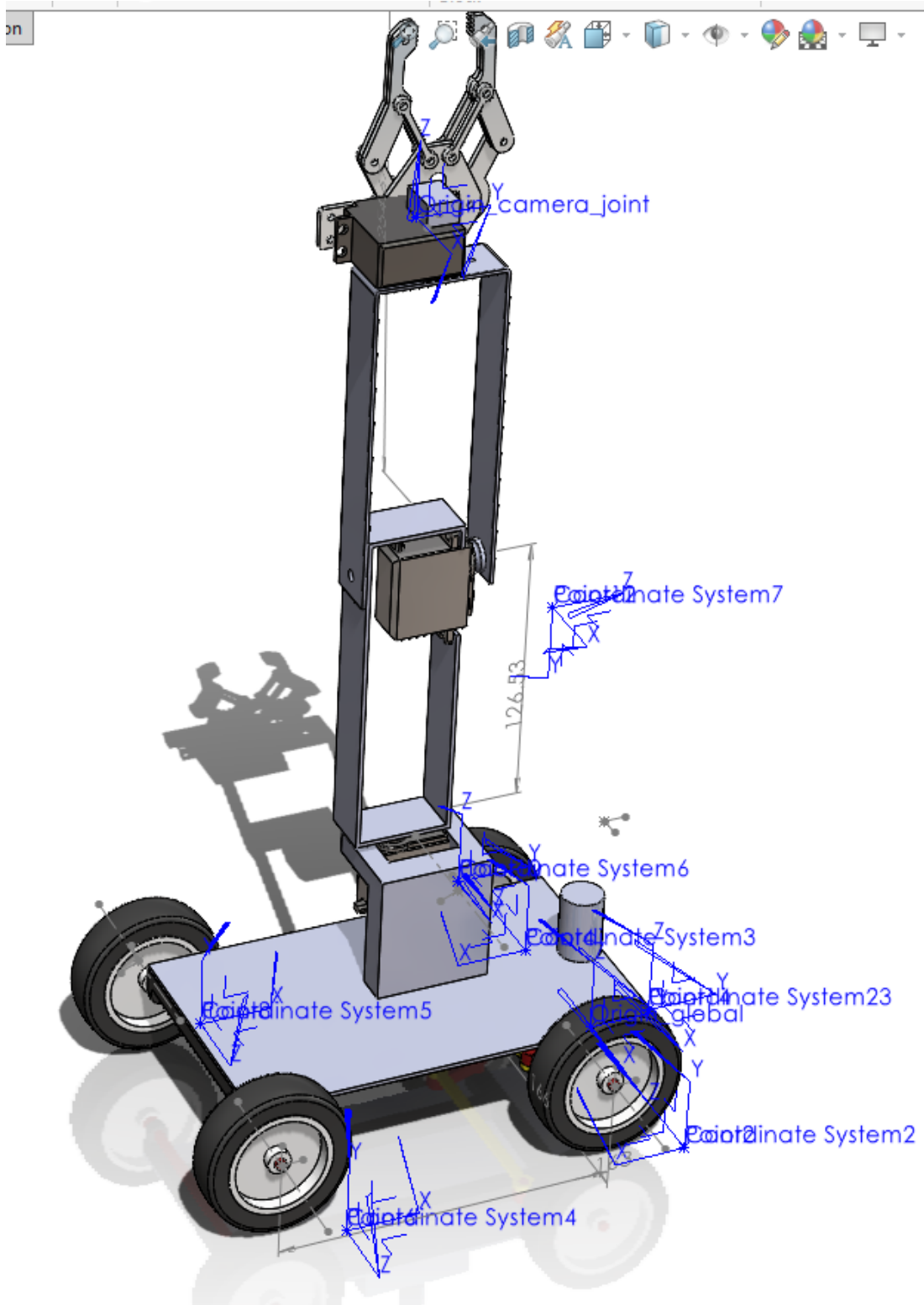
- Khung base_link: Nơi đặt lidar và bộ đồ tay máy
 - Kích cỡ base 200mm x 120mm



- Tay máy 2 góc quay gắn thêm camera tại EF:
 - Độ dài link 1 từ trục quay servo 1 đến trục quay servo 2 là: 126.5mm
 - Độ dài link 2 từ trục quay servo 2 đến điểm cuối của gripper là: 255mm
 - Camera là hình vuông, có size 1.5cm x 2cm x 1cm



2. Đặt trục để xuất file URDF như hình vẽ:



- **Hệ tọa độ gốc (base_link):** Đặt tại tâm đáy của thân xe robot. Trục X hướng về phía trước robot, trục Y hướng sang trái và trục Z hướng lên trên (theo quy tắc bàn tay phải).

- **Hệ tọa độ khớp:**

- **Khớp bánh xe (fl_joint, fr_joint, bl_joint, br_joint):** Trục Z trùng với trục quay của bánh xe.
- **Khớp tay máy (base_servo_joint, end_servo_joint):** Trục Z trùng với trục quay của khớp, tuân theo quy ước D-H (Denavit-Hartenberg) nếu áp dụng

III. File URDF

1. **File URDF:** (Unified Robot Description Format) là file XML mô tả robot được export thông qua add on trên Solidwork. Cấu trúc chính tf tree bao gồm:

Odom -> base_link -> lidar_link

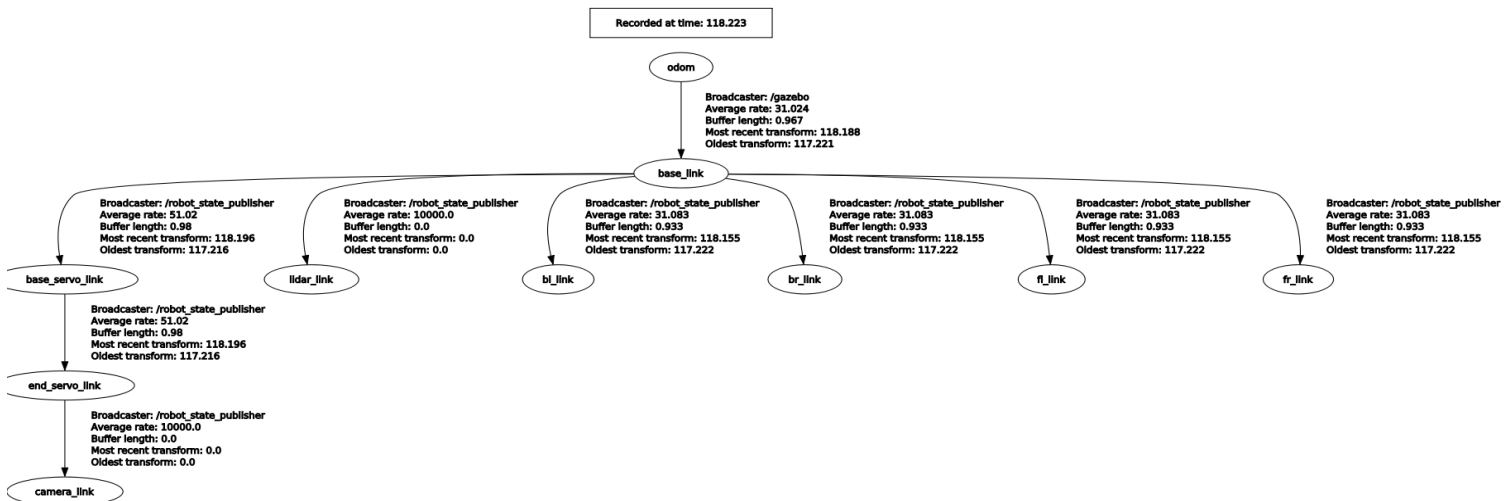
-> base_servo_link -> end_servo_link -> camera_link

-> fr_link

-> fl_link

-> br_link

-> bl_link



- `<robot name="tk8">`: Khai báo robot với tên "tk8".
- `<link name="...">`: Định nghĩa các liên kết cứng của robot: base_link, lidar_link, camera_link, base_servo_link, end_servo_link, fr_link, fl_link, br_link, bl_link. Mỗi link có các thẻ con:
 - `<inertial>`: Mô tả thuộc tính quán tính (khối lượng, tâm khối lượng, ma trận quán tính).
 - `<visual>`: Mô tả hình dạng hiển thị (geometry - mesh file STL, material, origin).
 - `<collision>`: Mô tả hình dạng va chạm (geometry - mesh file STL, origin).

Ví dụ với base_link

```
<link name="base_link">
  <inertial>
    <origin xyz="-0.00557264023375181 -0.0960602615045691 0.0594169936927835"
rpy="0 0 0" />
    <mass value="0.190548593408447" />
    <inertia ixx="0.000412726597714719" ixy="2.90219947110223E-06"
ixz="9.47643877840415E-06" iyy="0.000171360224234566" iyz="-1.29717645091752E-05"
izz="0.000480531579802367" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry><mesh filename="package://tk8/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 filename="package://tk8/meshes/base_link.STL" /></geometry>
  </collision>
</link>
```

- <joint name="..." type="...">: Định nghĩa các khớp (joint) kết nối các link: lidar_joint, base_servo_joint, end_servo_joint, camera_joint, fr_joint, fl_joint, br_joint, bl_joint.
- 4 bánh sử dụng khớp continuous, lidar và camera sử dụng khớp fixed, 2 servo điều khiển tay máy sử dụng khớp revolute

Mỗi joint có các thẻ con:

- <origin>: Vị trí và hướng của khớp so với link cha.
- <parent link="...">: Link cha.
- <child link="...">: Link con.
- <axis xyz="...">: Trục quay của khớp (nếu là khớp quay).
- <limit>: Giới hạn góc và vận tốc của khớp (nếu là khớp quay có giới hạn).
- <dynamics>: Thuộc tính động lực học (ví dụ: friction)

Ví dụ với joint bánh phải trước

```
<joint name="fr_joint" type="continuous">
  <origin xyz="0.0833 -0.032 0.03179" rpy="1.5708 0 -1.5708" />
  <parent link="base_link" />
  <child link="fr_link" />
  <axis xyz="0 0 1" />
  <dynamics friction="0.4" />
</joint>
```

- <transmission>: Mô tả cơ cấu truyền động cho các khớp điều khiển
base_servo_transmission, end_servo_transmission liên kết khớp với actuator và hardware interface.
- 2 joint servo đều sử dụng PositionJointInterface

```
<transmission name="base_servo_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="base_servo_joint">

<hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
  </joint>
  <actuator
name="base_servo_motor"><mechanicalReduction>1</mechanicalReduction></actuator>
</transmission>

  <transmission name="end_servo_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="end_servo_joint">

<hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
  </joint>
  <actuator
name="end_servo_motor"><mechanicalReduction>1</mechanicalReduction></actuator>
</transmission>
```

- `<gazebo reference="...">`: Định nghĩa các plugin Gazebo cho các link, đặc biệt là cảm biến và hệ thống điều khiển. Ở đây ta sử dụng plugin cho camera, lidar, 2 plugin `diff_drive` cho 2 cặp bánh trước và sau, `ros_control` cho 2 servo điều khiển tay máy
- Camera plugin có tần số cập nhật, hướng quay, kích cỡ hình ảnh thu được, các bộ lọc nhiễu và plugin điều khiển cùng các config liên quan ở code dưới đây

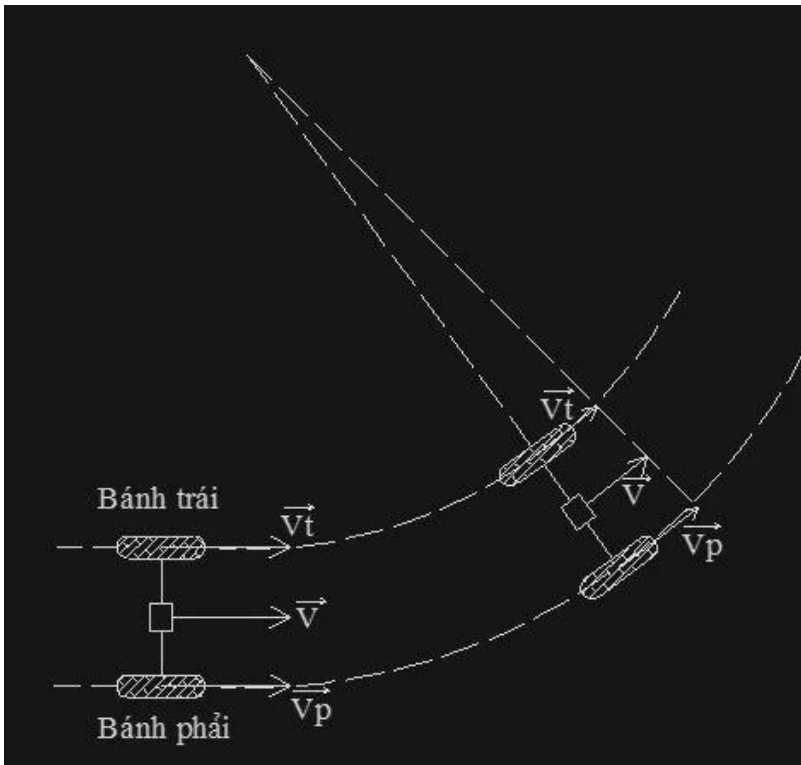
```
<!-- Camera Plugin -->
<gazebo reference="camera_link">
  <sensor type="camera" name="camera1">
    <pose>0 0 0 3.14 -1.5787 0</pose>
    <update_rate>30.0</update_rate>
    <camera name="head">
      <horizontal_fov>1.3962634</horizontal_fov>
      <image>
        <width>800</width>
        <height>800</height>
        <format>R8G8B8</format>
      </image>
      <clip>
        <near>0.02</near>
        <far>300</far>
      </clip>
      <noise>
        <type>gaussian</type>
        <mean>0.0</mean>
        <stddev>0.007</stddev>
      </noise>
    </camera>
    <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
      <alwaysOn>true</alwaysOn>
      <updateRate>0.0</updateRate>
      <cameraName>rrbot/camera1</cameraName>
      <imageTopicName>image_raw</imageTopicName>
      <cameraInfoTopicName>camera_info</cameraInfoTopicName>
      <frameName>camera_link</frameName>
      <hackBaseline>0.07</hackBaseline>
      <distortionK1>0.0</distortionK1>
      <distortionK2>0.0</distortionK2>
      <distortionK3>0.0</distortionK3>
      <distortionT1>0.0</distortionT1>
      <distortionT2>0.0</distortionT2>
    </plugin>
  </sensor>
</gazebo>
```

- Plugin lidar gồm tần số cập nhật, cảm biến tia laser, chế độ hiển thị các tia, số lượng tia quét, góc quét, độ phân giải, khoảng cách tối thiểu và tối đa mà tia quét có thể phát hiện được vật và cuối cùng gói ros hỗ trợ chạy cảm biến.
- Ở đây ta cấu hình lidar chỉ quét 180 độ trước mặt

```
<!-- Lidar Plugin -->
<gazebo reference="lidar_link">
  <material>Gazebo/Black</material>
  <sensor type="ray" name="lidar_sensor">
    <pose>0 0 0 0 0 0</pose>
    <visualize>true</visualize>
    <update_rate>10</update_rate>
    <ray>
      <scan>
        <horizontal>
          <samples>360</samples>
          <resolution>1</resolution>
          <min_angle>0</min_angle>
          <max_angle>3.14</max_angle>
        </horizontal>
      </scan>
      <range>
        <min>0.1</min>
        <max>12.0</max>
        <resolution>0.01</resolution>
      </range>
      <noise>
        <type>gaussian</type>
        <mean>0.0</mean>
        <stddev>0.01</stddev>
      </noise>
    </ray>
    <plugin name="lidar_controller" filename="libgazebo_ros_laser.so">
      <topicName>scan</topicName>
      <frameName>lidar_link</frameName>
    </plugin>
  </sensor>
</gazebo>
```

- Plugin diff drive là plugin hỗ trợ chuyển động vi sai cho 2 bánh xe, nó chứa: joint 2 bánh, thông số vật lý của bánh xe, publish các trạng thái khớp,...
- Ở đây ta dùng 2 bộ vi sai cho 2 cặp bánh trước và sau của robot

```
<plugin name="gazebo_ros_diff_drive_front"
filename="libgazebo_ros_diff_drive.so">
  <commandTopic>cmd_vel</commandTopic>
  <odometryTopic>odom</odometryTopic>
  <odometryFrame>odom</odometryFrame>
  <odometrySource>world</odometrySource>
  <publishOdomTF>true</publishOdomTF>
  <publishWheelTF>false</publishWheelTF>
  <publishTf>true</publishTf>
  <publishWheelJointState>false</publishWheelJointState>
  <updateRate>30</updateRate>
  <leftJoint>fl_joint</leftJoint>
  <rightJoint>fr_joint</rightJoint>
  <wheelSeparation>0.166</wheelSeparation>
  <wheelDiameter>0.07</wheelDiameter>
  <robotBaseFrame>base_link</robotBaseFrame>
  <wheelAcceleration>1</wheelAcceleration>
  <wheelTorque>10</wheelTorque>
  <rosDebugLevel>na</rosDebugLevel>
</plugin>
</gazebo>
<gazebo>
  <plugin name="gazebo_ros_diff_drive_rear"
filename="libgazebo_ros_diff_drive.so">
    <commandTopic>cmd_vel</commandTopic>
    <odometryFrame>odom_rear_only</odometryFrame>
    <odometrySource>world</odometrySource>
    <publishOdomTF>false</publishOdomTF>
    <publishWheelTF>false</publishWheelTF>
    <publishTf>false</publishTf>
    <publishWheelJointState>false</publishWheelJointState>
    <legacyMode>false</legacyMode>
    <updateRate>30</updateRate>
    <leftJoint>bl_joint</leftJoint>
    <rightJoint>br_joint</rightJoint>
    <wheelSeparation>0.166</wheelSeparation>
    <wheelDiameter>0.07</wheelDiameter>
    <robotBaseFrame>base_link</robotBaseFrame>
    <wheelAcceleration>1</wheelAcceleration>
    <wheelTorque>10</wheelTorque>
    <rosDebugLevel>na</rosDebugLevel>
  </plugin>
</gazebo>
```



- Cơ chế của plugin điều khiển vi sai: Điều khiển vi sai là phương pháp thay đổi tốc độ của những bánh xe (trái, phải) lúc xe vào cua. Ta điều khiển những bánh xe di chuyển với tốc độ khác nhau, tạo thẳng bằng cho xe. Đặc trưng lúc vào cua, bánh xe phía ngoài sẽ di chuyển quãng đường dài hơn bánh xe phía trong nên cần vận tốc tức thời to hơn để dễ vào cua. Nếu không mang bộ vi sai, 2 bánh xe vẫn di chuyển cùng tốc độ. Lúc đó, xe dễ gặp tình trạng trượt quay bánh xe khi vào cua.
- Plugin publish trạng thái khớp gồm góc và vận tốc của 4 joint bánh

```
<gazebo>
  <plugin name="joint_state_publisher"
filename="libgazebo_ros_joint_state_publisher.so">
    <jointName>fl_joint, fr_joint, bl_joint, br_joint</jointName>
    <updateRate>30</updateRate>
  </plugin>
</gazebo>
```

- Plugin tích hợp ros_control để điều khiển tay máy

```
<gazebo>
  <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
    <robotSimType>gazebo_ros_control/DefaultRobotHWSim</robotSimType>
    <legacyModeNS>true</legacyModeNS>
  </plugin>
</gazebo>
```

2. Thiết kế điều khiển

- Điều khiển xe

Xe được điều khiển bằng tạo publisher với topic /cmd_vel với kiểu message gửi là Twist. Code điều khiển xe sử dụng gói package điều khiển xe teleop_twist_keyboard thực hiện được bằng 9 phím.

Ta sử dụng luôn rosrun teleop_twist_keyboard teleop_twist_keyboard.py có sẵn để điều khiển xe

- Điều khiển tay máy

```
# Publisher gửi lệnh quỹ đạo
pub = rospy.Publisher(CONTROLLER_TOPIC, JointTrajectory, queue_size=10)
```

- **Topic điều khiển tay máy:** Tay máy được điều khiển thông qua topic **/servo_controller/command**. Controller servo_controller (được cấu hình là JointTrajectoryController trong tk8_controller.yaml) đăng ký lắng nghe topic này để nhận lệnh điều khiển.
- **Message Type:** Message type được sử dụng để gửi lệnh đến tay máy là **JointTrajectory**. JointTrajectory là một message ROS tiêu chuẩn được sử dụng để điều khiển các khớp robot theo quỹ đạo thời gian. Mặc dù trong trường hợp điều khiển bằng bàn phím đơn giản, chúng ta thường chỉ gửi các điểm đích (goal points) đơn lẻ, nhưng việc sử dụng JointTrajectory vẫn tuân theo chuẩn ROS và dễ dàng mở rộng cho điều khiển quỹ đạo phức tạp hơn trong tương lai.
- Tương tự như code điều khiển xe, code điều khiển tay máy cũng có hàm hỗ trợ đọc đầu vào từ bàn phím. Ta sử dụng rosrun tk8 servo_teleop_keyboard.py để điều khiển tay máy

```
while not rospy.is_shutdown():
    key = get_key()
    moved = False # Cờ để kiểm tra xem có cần gửi lệnh không
    if key == 'a':
        current_base_pos -= BASE_SERVO_STEP
        moved = True
    elif key == 'd':
        current_base_pos += BASE_SERVO_STEP
        moved = True
    elif key == 'w':
        current_end_pos -= END_SERVO_STEP
        moved = True
    elif key == 's':
        current_end_pos += END_SERVO_STEP
        moved = True
    elif key == '1':
        print("quit")
        break
    elif key == '2':
```

```

        print("reset")
        current_base_pos = 0.0
        current_end_pos = 0.0
        moved = True
    else:
        # Không làm gì nếu nhấn phím khác
        continue

    # --- Áp dụng giới hạn khớp ---
    if current_base_pos > BASE_SERVO_MAX:
        current_base_pos = BASE_SERVO_MAX
    elif current_base_pos < BASE_SERVO_MIN:
        current_base_pos = BASE_SERVO_MIN

    if current_end_pos > END_SERVO_MAX:
        current_end_pos = END_SERVO_MAX
    elif current_end_pos < END_SERVO_MIN:
        current_end_pos = END_SERVO_MIN

    # --- Chỉ gửi lệnh nếu có sự thay đổi ---
    if moved:
        # Tạo điểm quỹ đạo mới
        point = JointTrajectoryPoint()
        point.positions = [current_base_pos, current_end_pos]
        point.time_from_start = rospy.Duration(TIME_TO_REACH_POINT)

        # Cập nhật và gửi message
        traj.points = [point] # Quỹ đạo chỉ có 1 điểm đích
        traj.header.stamp = rospy.Time.now() # Cập nhật thời gian
        pub.publish(traj)

    rospy.loginfo("Đang gửi: Base=%.2f rad, End=%.2f rad", current_base_pos,
current_end_pos)

```

- Cấu hình .yaml: Sử dụng joint_state_controller/JointStateController và position_controllers/JointTrajectoryController để điều khiển 2 servo với thông số như code

```

joint_state_controller:
  type: joint_state_controller/JointStateController
  publish_rate: 50
servo_controller:
  type: position_controllers/JointTrajectoryController
  joints:
    - base_servo_joint
    - end_servo_joint
  constraints:
    goal_time: 0.5
    stopped_velocity_tolerance: 0.05
  stop_trajectory_duration: 0.5
  state_publish_rate: 25

```

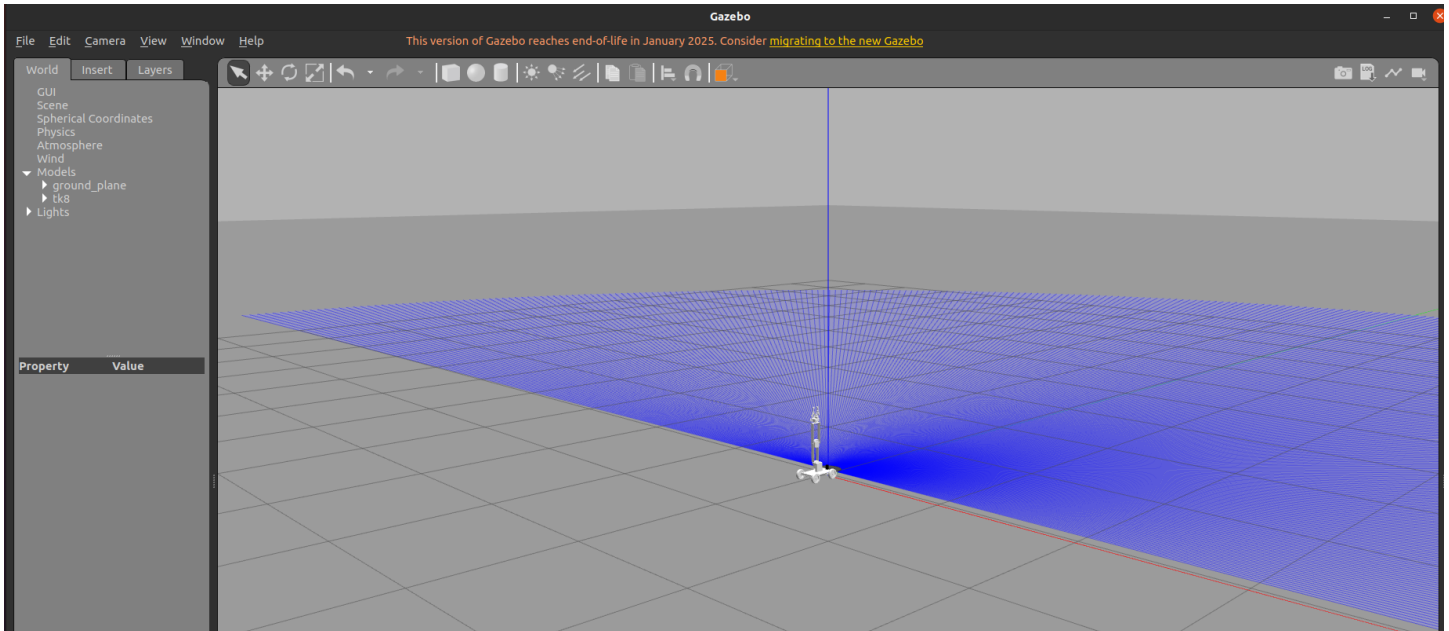

3. Cấu hình file launch

Để chạy robot trong Gazebo, yêu cầu phải tạo một file launch trong đó yêu cầu định sẵn các tham số sẵn của Gazebo, khởi tạo thành công một thế giới trống sau đó spawn được robot của mình vào thế giới, sau đó mở rviz đã có sẵn config của cảm biến và nạp các thông số cần thiết

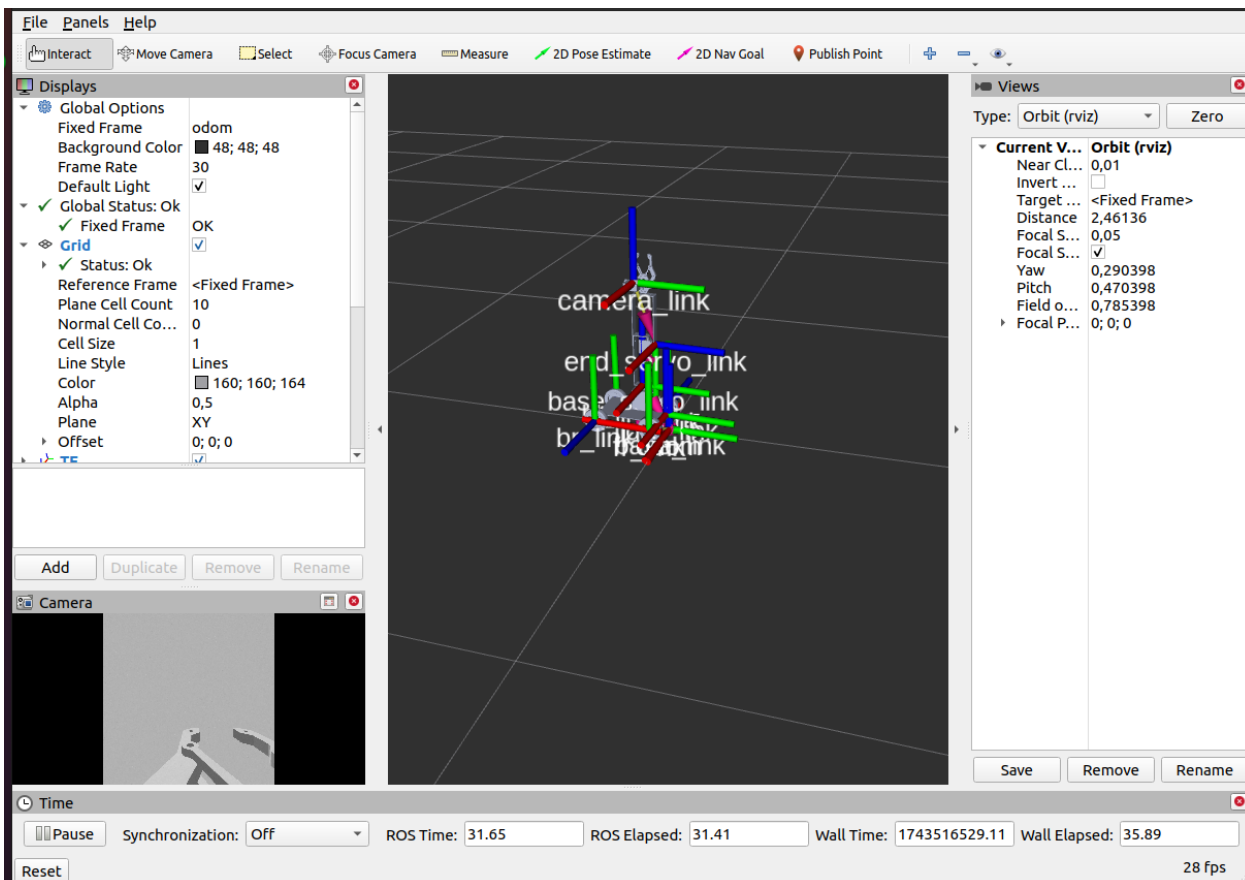
```
<rosparam file="$(find tk8)/config/tk8_controllers.yaml" command="load"/>
<param name="robot_description" command="$(find xacro)/xacro $(arg urdf_file)"
/>
<param name="/use_sim_time" value="$(arg use_sim_time)"/>
<include file="$(find gazebo_ros)/launch/empty_world.launch">
  <arg name="world_name" value="$(arg world_file)"/>
  <arg name="paused" value="$(arg paused)"/>
  <arg name="use_sim_time" value="$(arg use_sim_time)"/>
  <arg name="gui" value="$(arg gui)"/>
  <arg name="headless" value="$(arg headless)"/>
  <arg name="debug" value="$(arg debug)"/>
  <arg name="verbose" value="$(arg verbose)"/>
  <arg name="respawn_gazebo" value="$(arg respawn_gazebo)"/>
</include>
<node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false"
output="screen"
  args="-urdf -model $(arg model) -param robot_description
    -x $(arg x_pos) -y $(arg y_pos) -z $(arg z_pos)
    -R $(arg roll) -P $(arg pitch) -Y $(arg yaw)"/>
<node name="controller_spawner" pkg="controller_manager" type="spawner"
respawn="false"
  output="screen" args="joint_state_controller servo_controller"/>
<node name="robot_state_publisher" pkg="robot_state_publisher"
type="robot_state_publisher"
  respawn="false" output="screen"/>
<node name="servo_teleop_keyboard" pkg="tk8" type="teleop_robot.py"
  output="screen" launch-prefix="xterm -e" if="$(arg start_teleop)">
</node>
<!-- RViz configuration file path -->
<arg name="rviz_config_file" default="$(find tk8)/tk8_default.rviz" doc="Path to
the RViz config file"/>
<node pkg="rviz" type="rviz" name="rviz" args="-d $(arg rviz_config_file)"
if="$(arg rviz)" output="screen"/>
</launch>
```

IV. Kết quả

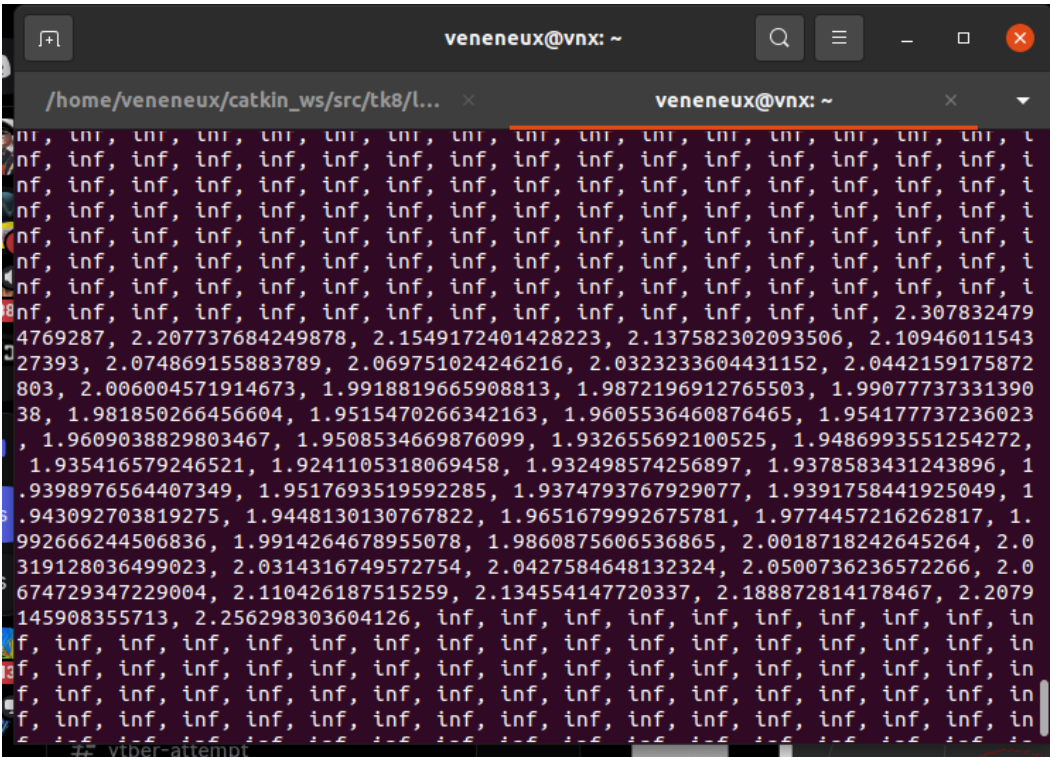
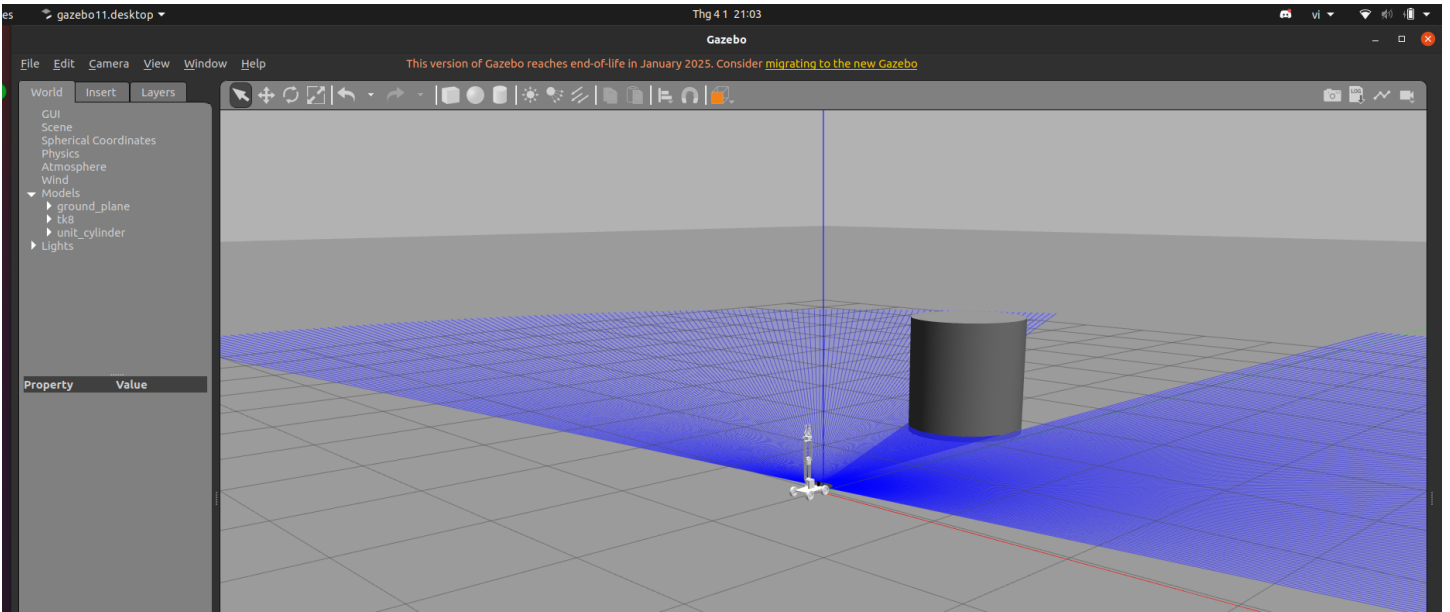
- Link video kết quả: [2025-04-01 11-43-54.mp4](#)
- Chạy file launch ta được kết quả sau
 - Gazebo khởi tạo thành công



- Rviz khởi tạo thành công



- Add thêm vật thể rostopic scan trả về kết quả đo được



- Hiện thị thành công line scan được trên rviz

- Joint_states

```
veneneux@vnx: ~/catkin_ws
/home/v... x venene... x venene... x veneneu... x veneneu... x
veneneux@vnx:~/catkin_ws$ rostopic echo /joint_states
header:
  seq: 13042
  stamp:
    secs: 163
    nsecs: 361000000
  frame_id: ''
name:
  - base_servo_joint
  - end_servo_joint
position: [-6.715124941436557e-06, 0.01363893838523822]
velocity: [0.0019158606372341503, 0.0008920960161679241]
effort: [0.0, 0.0]
---
header:
  seq: 13043
  stamp:
    secs: 163
    nsecs: 375000000
  frame_id: ''
name:
  - fl_joint
  - fr_joint
  - bl_joint
  - br_joint
position: [-0.013294558228145625, 0.04078822112469549, -0.16220676915115462, -0.05876399503781027]
velocity: [0.0004325714344617232, 0.0002570820142418946, 0.0026414491966272307, -0.002374655171571205]
effort: []
---
```

- Odom (cho encoder)

[illegible]

- cmd_vel (cho encoder)

```
veneneux@vnx: ~  
/home/veneneux/cat... x veneneux@vnx: ~ x veneneux@vnx: ~  
veneneux@vnx:~$ rostopic echo /cmd_vel  
linear:  
  x: 0.5  
  y: 0.0  
  z: 0.0  
angular:  
  x: 0.0  
  y: 0.0  
  z: 1.0  
---  
linear:  
  x: 0.0  
  y: 0.0  
  z: 0.0  
angular:  
  x: 0.0  
  y: 0.0  
  z: 0.0  
---
```

- camera_raw

```
veneneux@vnx: ~  
/home/veneneux/catkin_ws/src/tk8/l... x veneneux@vnx: ~  
78, 178, 177, 177, 177, 179, 179, 179, 179, 179, 179, 176, 176, 176, 178, 178, 1  
78, 180, 180, 180, 180, 180, 180, 177, 177, 177, 180, 180, 180, 178, 178, 178, 1  
76, 176, 176, 177, 177, 177, 177, 177, 177, 178, 178, 178, 178, 178, 176, 1  
76, 176, 176, 176, 176, 177, 177, 177, 178, 178, 178, 179, 179, 179, 181, 181, 1  
81, 176, 176, 176, 179, 179, 179, 180, 180, 180, 179, 179, 179, 177, 177, 177, 1  
79, 179, 179, 180, 180, 180, 178, 178, 178, 176, 176, 176, 180, 180, 180, 176, 1  
76, 176, 182, 182, 182, 180, 180, 180, 177, 177, 177, 176, 176, 176, 175, 175, 1  
75, 177, 177, 177, 178, 178, 178, 180, 180, 180, 175, 175, 175, 179, 179, 179, 1  
77, 177, 177, 177, 177, 177, 179, 179, 179, 178, 178, 178, 178, 178, 178, 178, 1  
78, 178, 177, 177, 177, 175, 175, 175, 177, 177, 177, 178, 178, 178, 181, 181, 1  
81, 177, 177, 177, 178, 178, 178, 180, 180, 180, 180, 180, 180, 177, 177, 177, 1  
78, 178, 178, 179, 179, 179, 178, 178, 178, 178, 178, 178, 179, 179, 179, 182, 1  
82, 182, 178, 178, 178, 179, 179, 179, 175, 175, 175, 180, 180, 180, 180, 180, 1  
80, 177, 177, 177, 175, 175, 175, 177, 177, 177, 178, 178, 178, 178, 178, 178, 1  
75, 175, 175, 178, 178, 178, 177, 177, 177, 179, 179, 179, 180, 180, 180, 179, 1  
79, 179, 178, 178, 178, 179, 179, 179, 186, 186, 186, 205, 205, 205, 201, 201, 2  
01, 200, 200, 200, 204, 204, 204, 201, 201, 201, 203, 203, 203, 203, 203, 203, 2  
01, 201, 201, 203, 203, 203, 201, 201, 201, 203, 203, 203, 202, 202, 202, 202, 2  
02, 202, 201, 201, 201, 201, 201, 201, 201, 201, 201, 202, 202, 202, 201, 201, 2  
01, 203, 203, 203, 203, 203, 203, 203, 203, 203, 200, 200, 200, 200, 200, 200, 2  
04, 204, 204, 199, 199, 199, 199, 199, 199, 204, 204, 204, 201, 201, 201, 202, 2  
02, 202, 201, 201, 201, 201, 201, 201, 204, 204, 204, 200, 200, 200, 204, 204, 2  
04, 200, 200, 200, 198, 198, 198, 201, 201, 201, 204, 204, 204, 202, 202, 202, 2  
03, 203, 203, 202, 202, 202, 199, 199, 199, 200, 200, 200, 203, 203, 203, 204, 2
```


V. Nhận xét và đánh giá

Robot đã có đầy đủ chi tiết và có thể thực hiện đầy đủ chức năng theo yêu cầu:

- Có khả năng điều di chuyển
- Có khả năng điều khiển 2 khớp tay máy
- 3 cảm biến đều hoạt động và trả về kết quả như mong muốn

Tuy nhiên robot vẫn có 1 vài bất cập trong thiết kế có thể kể đến:

- Link 2 của tay máy dài và to hơn link 1: Điều này có thể dẫn đến sai số điều khiển ngoài thực tế do khả năng chịu tải theo dạng link 2 nặng hơn link 1 không ổn định
- Xe hơi bé so với tay máy
- Trong gazebo có lúc xảy ra tình trạng trượt bánh rất nhỏ do thiết lập thông số ma sát và môi trường còn sai số