URDF Model

1. Program function description

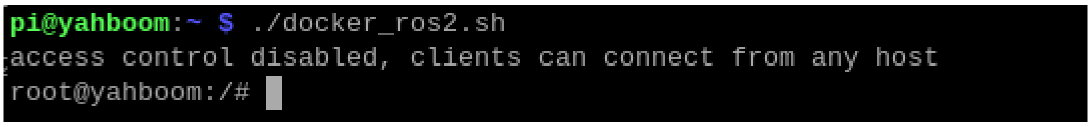
Start the car, run the program in the terminal, and the URDF model will be displayed in rviz.

2. Enter the car docker

Open the terminal and enter the following command to enter the docker,

```
./docker_ros2.sh
```

The following interface appears, which means that you have successfully entered Docker. Now you can control the car through commands.

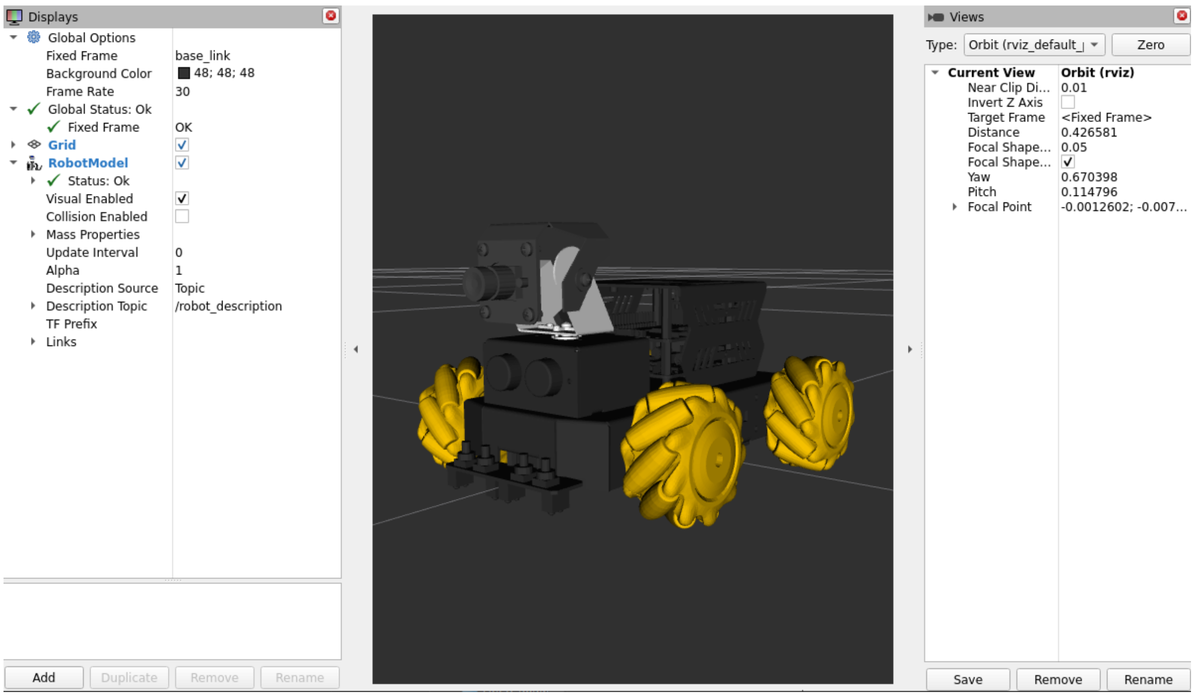


```
pi@yahboom:~ $ ./docker_ros2.sh
access control disabled, clients can connect from any host
root@yahboom:/#
```

4. Program startup

Load URDF and generate a simulation controller and start rviz, input in the terminal,

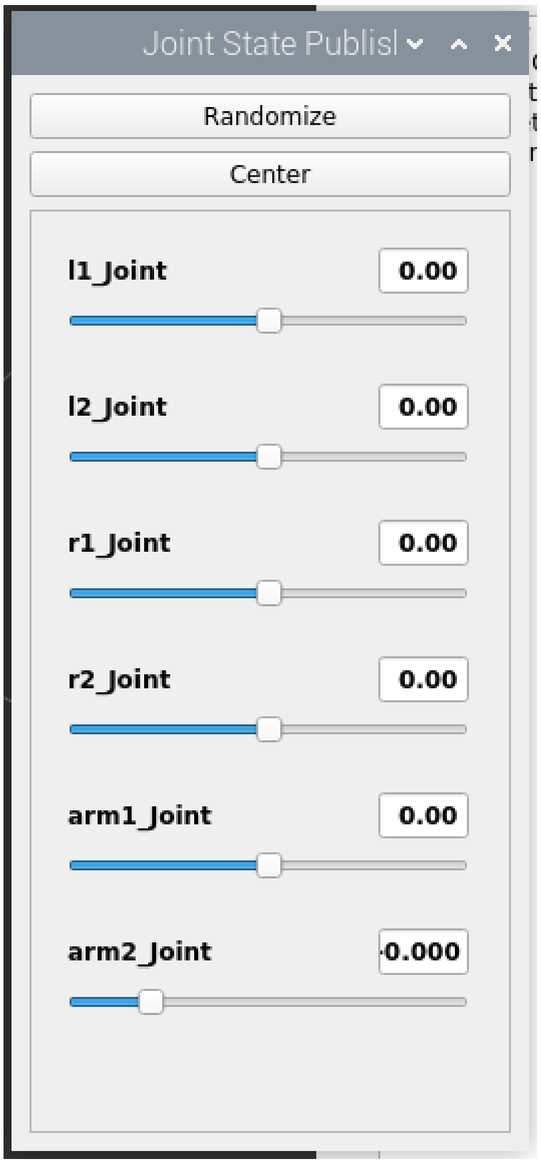
```
ros2 launch yahboomcar_description display_launch.py
```



Displays	
Global Options	
Fixed Frame	base_link
Background Color	48; 48; 48
Frame Rate	30
Global Status: Ok	
Fixed Frame	OK
Grid	✓
RobotModel	✓
Status: Ok	
Visual Enabled	✓
Collision Enabled	
Mass Properties	
Update Interval	0
Alpha	1
Description Source	Topic
Description Topic	/robot_description
TF Prefix	
Links	

Views	
Type:	Orbit (rviz_default_
Current View	Orbit (rviz)
Near Clip Di...	0.01
Invert Z Axis	
Target Frame	<Fixed Frame>
Distance	0.426581
Focal Shape...	0.05
Focal Shape...	✓
Yaw	0.670398
Pitch	0.114796
Focal Point	-0.0012602; -0.007...

Then, use the mouse to adjust the viewing angle and slide the simulated controller just created, and you can see the tires/camera of the car changing.



l1_Joint: Control the left front wheel

l2_Joint: Control the left rear wheel

r1_Joint: Control the right front wheel

r2_Joint: Control the right rear wheel

arm1_Joint: Control gimbal 1

arm2_Joint: Control gimbal 2

Randomize: Randomly publish values to each joint

Center: All joints return to the center

5. Code analysis

Code location,

```
/root/yahboomcar_ws/src/yahboomcar_description/launch
```

display_launch.py

```
from ament_index_python.packages import get_package_share_path
from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument
from launch.substitutions import Command, LaunchConfiguration

from launch_ros.actions import Node
from launch_ros.parameter_descriptions import ParameterValue

def generate_launch_description():
    urdf_tutorial_path = get_package_share_path('yahboomcar_description')
    default_model_path = urdf_tutorial_path / 'urdf/Raspbot-V2.urdf'
    default_rviz_config_path = urdf_tutorial_path / 'rviz/raspbotv2.rviz'

    model_arg = DeclareLaunchArgument(name='model',
default_value=str(default_model_path),
                                      description='Absolute path to robot urdf
file')
    robot_description = ParameterValue(Command(['xacro ',
LaunchConfiguration('model')]),
                                       value_type=str)
    rviz_arg = DeclareLaunchArgument(name='rvizconfig',
default_value=str(default_rviz_config_path),
                                     description='Absolute path to rviz config
file')

    robot_state_publisher_node = Node(
        package='robot_state_publisher',
        executable='robot_state_publisher',
        parameters=[{'robot_description': robot_description}]
    )

    joint_state_publisher_gui_node = Node(
        package='joint_state_publisher_gui',
        executable='joint_state_publisher_gui'
    )


    tf_base_footprint_to_base_link = Node(
        package='tf2_ros',
        executable='static_transform_publisher',
        arguments=['0', '0', '0.05', '0.0', '0.0', '0.0', 'base_footprint',
'base_link'],
    )

    rviz_node = Node(
        package='rviz2',
        executable='rviz2',
        name='rviz2',
        output='screen',
        arguments=['-d', LaunchConfiguration('rvizconfig')],
    )
    return LaunchDescription([
        model_arg,
        joint_state_publisher_gui_node,
        robot_state_publisher_node,
        tf_base_footprint_to_base_link,
        rviz_arg,
        rviz_node
    ])
```

- model_arg: load model parameters, the loaded model is Raspbot-V2.urdf, the location is `/root/yahboomcar_ws/src/yahboomcar_description/urdf/Raspbot-V2.urdf`
- joint_state_publisher_gui_node: publish sensor_msgs/JointState message
- robot_state_publisher_node: robot state publishing
- tf_base_footprint_to_base_link: publish static transformation from base_footprint to base_link

6. URDF model

URDF, the full name is Unified Robot Description Format, translated into Chinese as Unified Robot Description Format, is a robot model file described in XML format, similar to D-H parameters.

```
<?xml version="1.0" encoding="utf-8"?>
```

The first line is a required field for XML and describes the version information of XML.

```
<robot
  name="RaspbotV2">
</robot>
```

The second line describes the current robot name; all information about the current robot is included in the [robot] tag.

6.1. Components

- Link, connecting rod, can be imagined as a human arm
- Joint, joint, can be imagined as a human elbow joint

Relationship between link and joint: two links are connected by joints, imagine that the arm has a forearm (link) and an upper arm (link) connected by an elbow joint (joint).

6.1.1. Link

1), Introduction

In the URDF descriptive language, link is used to describe physical properties,

- Describe visual display, label.
- Describe collision properties, label.
- Describe physical inertia, label is not commonly used.

Links can also describe the size of the connecting rod (size)\color (color)\shape (shape)\inertial matrix (inertial matrix)\collision parameters (collision properties), etc. Each Link will become a coordinate system.

2), Sample code

```
<link
    name="base_link">
    <inertial>
      <origin
        xyz="0.013209067968915 0.000305835286849597 0.034565647785585"
        rpy="0 0 0" />
      <mass
        value="0.315736522899678" />
      <inertia
        ixx="0.000126930677782624"
        ixy="-1.93570093947537E-07"
        ixz="-3.2159760519397E-08"
        iyy="0.000217028578454845"
        iyz="3.16386759318505E-07"
        izz="0.000309637617387382" />
    </inertial>
    <visual>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://yahboomcar_description/meshes/base_link.STL" />
      </geometry>
      <material
        name="">
        <color
          rgba="0.203921568627451 0.203921568627451 0.203921568627451 1" />
      </material>
    </visual>
    <collision>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://yahboomcar_description/meshes/base_link.STL" />
      </geometry>
    </collision>
  </link>
```

3. Tag introduction

- origin: describes the position information; the xyz attribute describes the coordinate position in the environment, and the rpy attribute describes its own posture.
- mess: describes the quality of the link.
- inertia: inertial reference system. Due to the symmetry of the rotational inertia matrix, only 6 upper triangular elements ixx, ixy, ixz, iyy, iyz, izz are required as attributes.
- geometry: the tag describes the shape; the main function of the mesh attribute is to load the texture file, and the filename attribute is the file address of the texture path
- material: the tag describes the material; the name attribute is a **required item**, which can be empty and can be repeated. The rgba attribute in the [color] tag is used to describe red, green, blue, and transparency, separated by spaces.

6.1.2, joints

1. Introduction

Describes the relationship between two joints, motion position and speed limits, kinematic and dynamic properties. There are several types of joints:

- fixed: fixed joints. No movement is allowed, it serves as a connection.
- continuous: revolute joint. It can rotate continuously, without rotation angle limit.
- revolute: revolute joint. Similar to continuous, with rotation angle limit.
- prismatic: sliding joint. It moves along a certain axis, with position limit.
- floating: floating joint. It has six degrees of freedom, 3T3R.
- planar: planar joint. It allows translation or rotation above the plane orthogonal to the plane.

2), sample code

```
<joint
   name="arm1_Joint"
   type="revolute">
   <limit effort="100" velocity="1" lower="-1.57" upper="1.57"/>
   <origin
     xyz="0.058 0 0.0575"
     rpy="0 0 0" />
   <parent
     link="base_link" />
   <child
     link="arm1_Link" />
   <axis
     xyz="0 0 1" />
   <limit
     lower="0"
     upper="0"
     effort="0"
     velocity="0" />
 </joint>
```

- In the [joint] tag, the name attribute is a **required item**, describing the name of the joint, and it is unique. In the [joint] tag, the type attribute is filled in with the six major joint types.

  3), Tag introduction

  - origin: sub-tag, refers to the relative position of the rotation joint in the parent coordinate system.
  - parent, child: parent, child sub-tags represent two links to be connected; parent is a reference object, and child rotates around the parent.
  - axis: sub-tag indicates which axis (xyz) the link corresponding to the child rotates around and the amount of rotation around the fixed axis.
  - limit: sub-tag is mainly used to limit the child. The lower attribute and upper attribute limit the range of rotation, and the effort attribute limits the force range during the rotation process. (positive and negative value, in Newton or N), the velocity attribute limits the speed of rotation, in meters/second or m/s.
  - mimic: describes the relationship between this joint and the existing joints.
  - safety_controller: describes the safety controller parameters. Protect the movement of the robot joints.