

# Robot URDF model

Note: The ROS\_DOMAIN\_ID of the Raspberry Pi and the microROS control board need to be consistent. You can check [MicroROS Control Board Parameter Configuration] to set the microROS control board ROS\_DOMAIN\_ID. Check the tutorial [Connect MicroROS Agent] to determine whether the IDs are consistent.

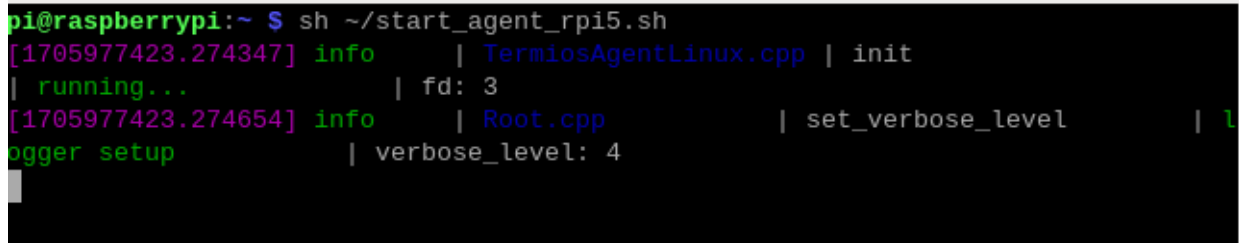
## 1. Program function description

The car connects to the agent, runs the program, and the URDF model will be displayed in rviz.

## 2. Start and connect to the agent

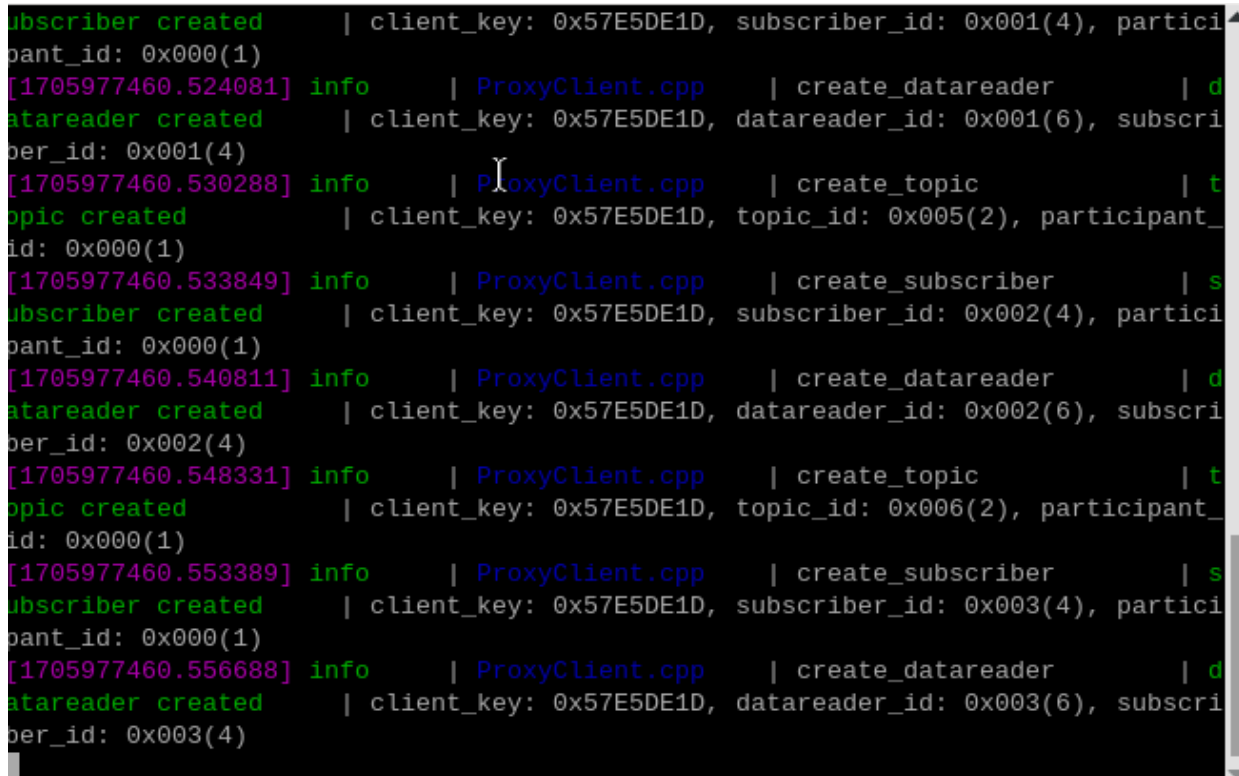
After successfully starting the Raspberry Pi, enter the following command to start the agent.

```
sh ~/start_agent_rpi5.sh
```



```
pi@raspberrypi:~ $ sh ~/start_agent_rpi5.sh
[1705977423.274347] info      | TermiosAgentLinux.cpp | init
| running...           | fd: 3
[1705977423.274654] info      | Root.cpp              | set_verbose_level      | 1
logger setup          | verbose_level: 4
```

Then turn on the car switch and wait for the car to connect to the agent. The connection is successful as shown in the figure below.



```
Subscriber created      | client_key: 0x57E5DE1D, subscriber_id: 0x001(4), partici
pant_id: 0x000(1)
[1705977460.524081] info      | ProxyClient.cpp       | create_datareader      | d
atareader created      | client_key: 0x57E5DE1D, datareader_id: 0x001(6), subscri
ber_id: 0x001(4)
[1705977460.530288] info      | ProxyClient.cpp       | create_topic           | t
opic created           | client_key: 0x57E5DE1D, topic_id: 0x005(2), participant_
id: 0x000(1)
[1705977460.533849] info      | ProxyClient.cpp       | create_subscriber      | s
Subscriber created      | client_key: 0x57E5DE1D, subscriber_id: 0x002(4), partici
pant_id: 0x000(1)
[1705977460.540811] info      | ProxyClient.cpp       | create_datareader      | d
atareader created      | client_key: 0x57E5DE1D, datareader_id: 0x002(6), subscri
ber_id: 0x002(4)
[1705977460.548331] info      | ProxyClient.cpp       | create_topic           | t
opic created           | client_key: 0x57E5DE1D, topic_id: 0x006(2), participant_
id: 0x000(1)
[1705977460.553389] info      | ProxyClient.cpp       | create_subscriber      | s
Subscriber created      | client_key: 0x57E5DE1D, subscriber_id: 0x003(4), partici
pant_id: 0x000(1)
[1705977460.556688] info      | ProxyClient.cpp       | create_datareader      | d
atareader created      | client_key: 0x57E5DE1D, datareader_id: 0x003(6), subscri
ber_id: 0x003(4)
```

### 3. Enter the car system docker

Open another terminal and enter the following command to enter docker.

```
sh ros2_humble.sh
```

When the following interface appears, you have successfully entered docker. Now you can control the car through commands.

```
pi@raspberrypi:~ $ ./ros2_humble.sh
access control disabled, clients can connect from any host
Successful
MY_DOMAIN_ID: 20
root@raspberrypi:/#
```

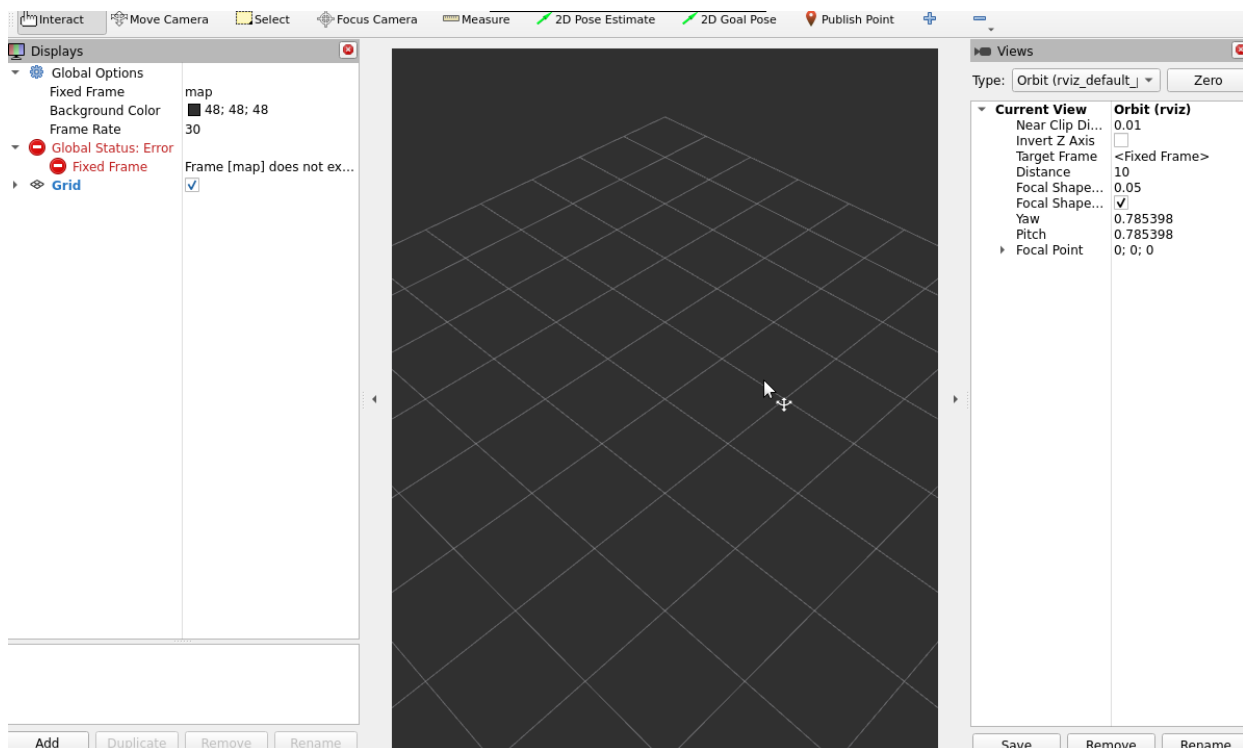
### 4. Program startup

Load URDF and generate a simulation controller, terminal input.

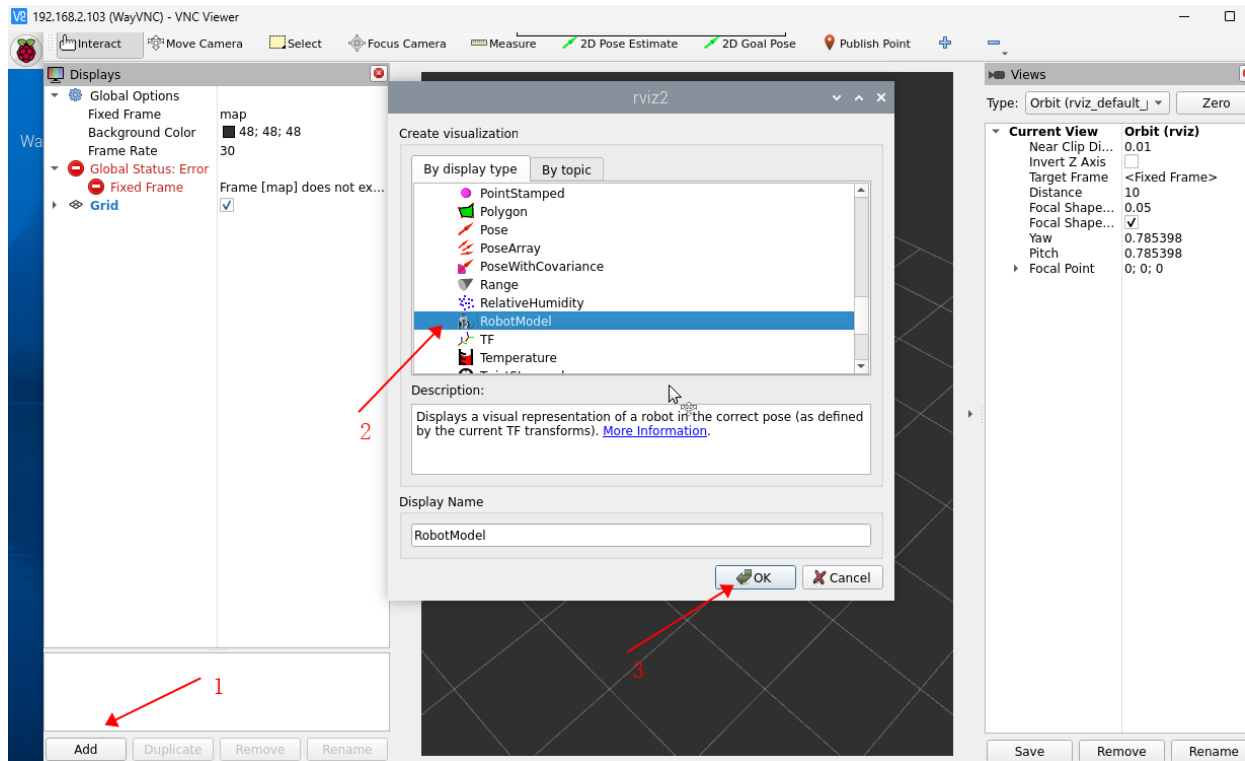
```
ros2 launch yahboomcar_description display_launch.py
```

Open the rviz display model and enter in the terminal (note that multiple terminals need to enter the same docker container. For details, see the [Enter the robot's docker container] tutorial in [Docker Development Environment])

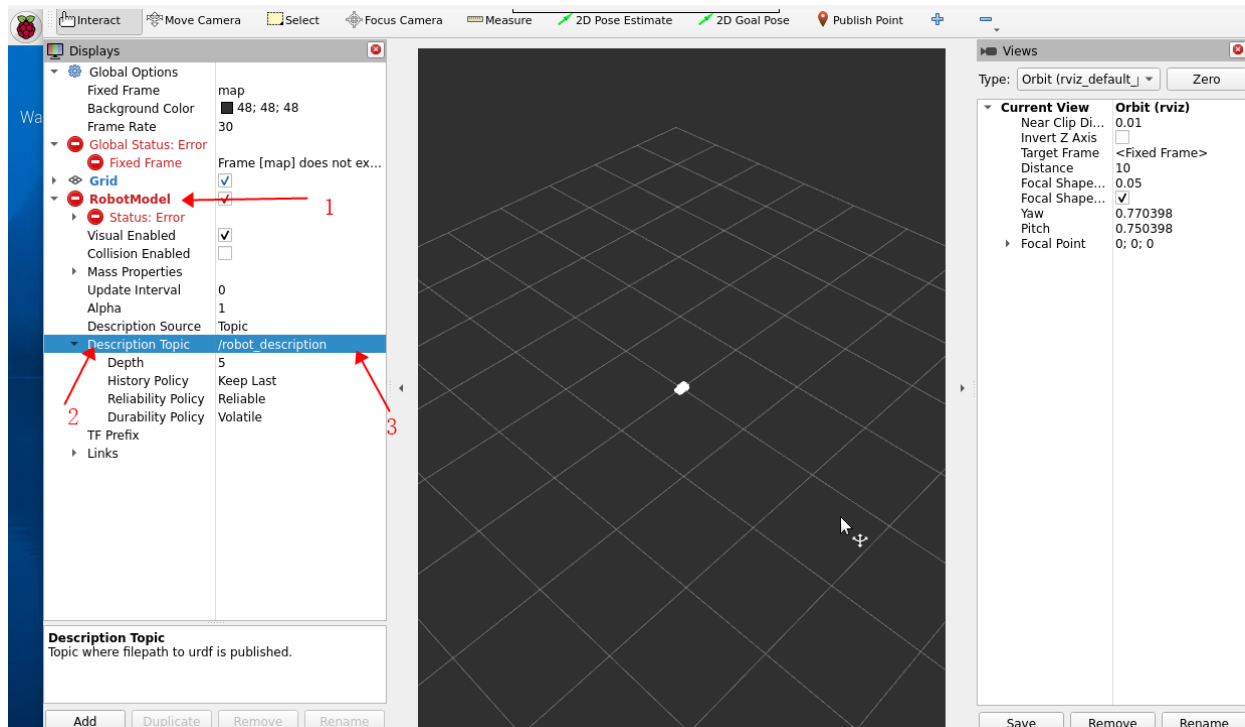
```
rviz2
```



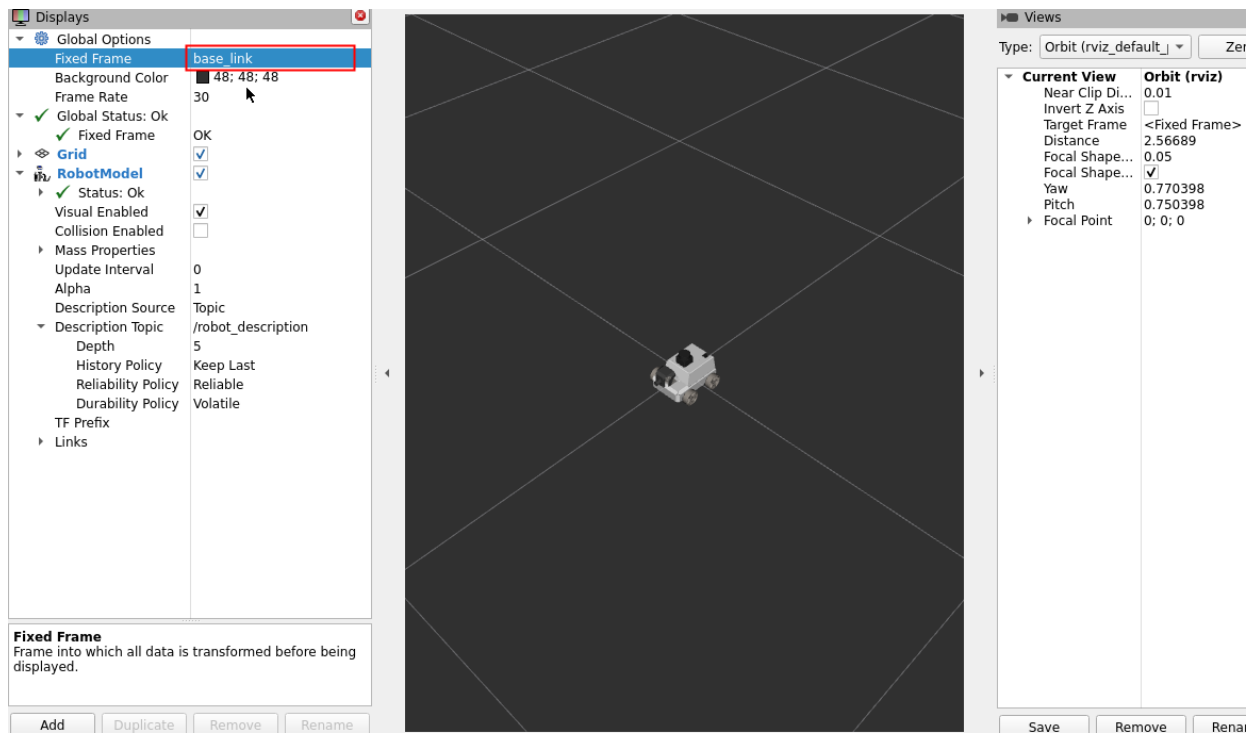
Click on the lower right corner to add model files, RobotModel



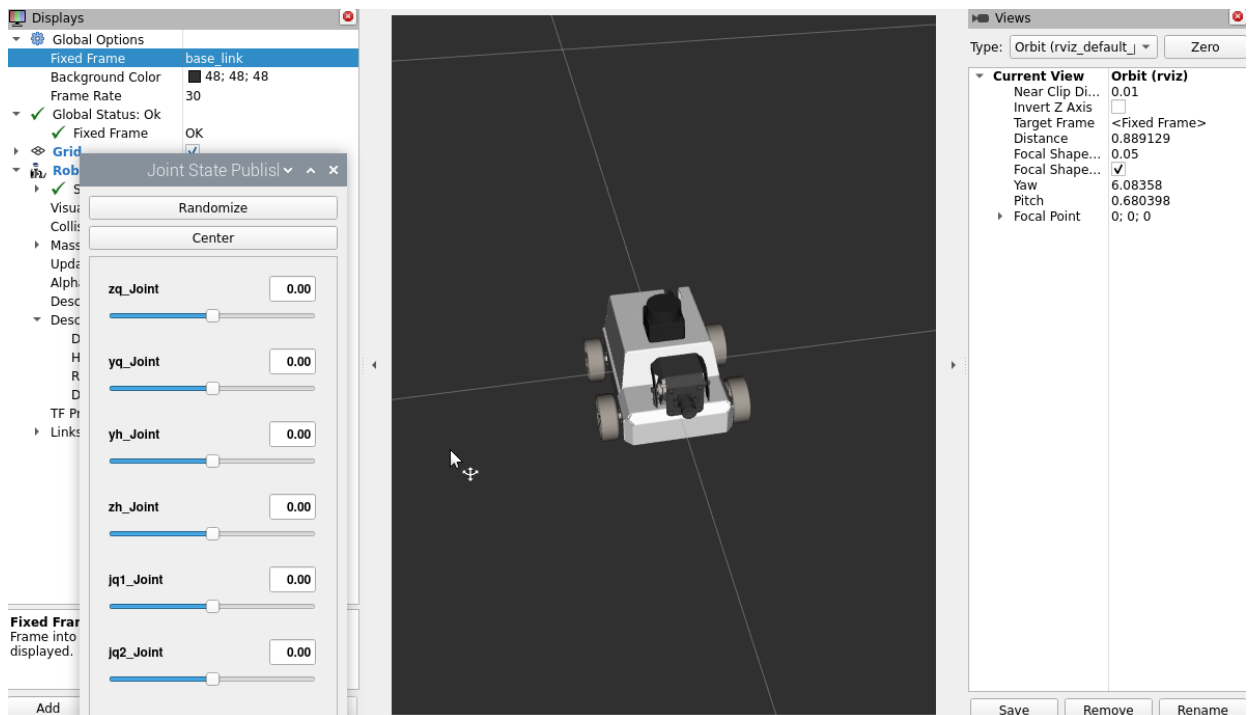
Click on the added RobotModel, click on the description topic, and select /robot\_description



Change [Fixed Frame] to base\_linke to display the car model.



Then use the mouse to adjust the viewing angle, slide the simulation controller just generated, and you can see that the tires/camera of the car are changing.



zq\_Joint: Control left front wheel

yq\_Joint: Control right front wheel

yh\_Joint: Control right rear wheel

zh\_Joint: Control left rear wheel

jq1\_Joint: Control PTZ 1

jq2\_Joint: Control PTZ 2

Randomize: Randomly publish values to each joint

Center: All Joints are centered

## 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/MicroROS.urdf'

    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)

    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'],
    )
```

```
return LaunchDescription([
    model_arg,
    joint_state_publisher_gui_node,
    robot_state_publisher_node,
    tf_base_footprint_to_base_link
])
```

- model\_arg: Load model parameters. The loaded model is MicroROS.urdf, located at `/home/yahboom/yahboomcar_ws/src/yahboomcar_description/urdf.`
- joint\_state\_publisher\_gui\_node: Publish sensor\_msgs/JointState message
- robot\_state\_publisher\_node: Robot status release
- tf\_base\_footprint\_to\_base\_link: Publish a static transformation of base\_footprint to base\_link

## 6、 URDF model

URDF: The full name is Unified Robot Description Format, which is translated into Chinese as Unified Robot Description Format. It 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 item for XML and describes the version information of XML.

```
<robot name="name="micro4.0">
</robot>
```

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

### 6.1. Components

- link: The connecting rod can be imagined as a human arm
- joint: The joint can be imagined as a human elbow joint

The relationship between link and joint: Two links are connected through joints. Imagine that an arm has a forearm (link) and an upper arm (link) connected through an elbow joint (joint).

#### 6.1.1、 link

##### 1) 、 Introduction

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

- describe the visual display, `<visual>` Label.
- describe collision properties, `<collision>` Label.
- describe physical inertia, `<inertial>` Labels are not commonly used.

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

## 2) 、 sample code

```
<link
  name="base_link">
  <inertial>
    <origin
      xyz="-0.0038187037800903 -0.000532399212617988 -0.00668209865413972"
      rpy="0 0 0" />
    <mass
      value="0.222555109690442" />
    <inertia
      ixx="0.000160582675692647"
      ixy="-8.18530574494391E-07"
      ixz="-2.74575507729664E-06"
      iyy="0.000176217109527607"
      iyz="1.64721285063183E-07"
      izz="0.000302441932451338" />
  </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="1 1 1 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) 、 label introduction

- origin: Describes the pose information; `xyz` The attribute describes the coordinate position in the environment, `rpy` Attributes describe their own posture.
- mess: Describes the quality of the link.
- inertia: The inertial reference frame, due to the symmetry of the rotational inertia matrix, only needs 6 upper triangular elements `ixx`, `ixy`, `ixz`, `iyy`, `iyz`, `izz` as attributes.

- geometry: The label describes the shape; `mesh` The main function of the attribute is to load the texture file, `filename` The file address of the attribute texture path.
- material: The label describes the material; `name` Attributes are **required**, can be empty, and can be repeated. Through the [color] tag in `rgba` Attributes to describe red, green, blue, and transparency, separated by spaces.

### 6.1.2、 joints

#### 1) 、 Introduction

Describe the relationship between two joints, motion position and velocity limits, kinematic and dynamic properties.

Joint Type:

- fixed: Movement is not allowed and acts as a connection.
- continuous: Rotate the joint. It can be rotated continuously, and there is no limit to the rotation angle.
- revolute: Rotate the joint. Similar to continuous, there is a limit to the rotation angle
- prismatic: sliding joints. Move along a certain axis, there is a position limit.
- floating: floating joints. With six degrees of freedom, 3T3R.
- planar: Planar joints. Allows translation or rotation above the plane orthogonal.

#### 2) 、 sample code

```
<joint
  name="zq_Joint"
  type="continuous">
  <origin
    xyz="0.0455 0.0675 -0.02125"
    rpy="0 0 0" />
  <parent
    link="base_link" />
  <child
    link="zq_Link" />
  <axis
    xyz="0 1 0" />
</joint>
```

In the [joint] tag, the name attribute is **required**, which describes the name of the joint and is unique. In the type attribute of the [joint] tag, fill in the six joint types correspondingly.

#### 3) 、 label introduction

- origin: subtab, referring to the rotation joint in `parent` The relative position of the coordinate system.
- parent, child: The parent and child sub-labels represent two links to be connected; parent is the reference, and child rotates around the parent.



- **axis:** The child label indicates which axis(xyz) the corresponding link of the child rotates around and the amount of rotation around the fixed axis.
- **limit:** The child tag is mainly to limit the child. lowerproperties and upperThe property limits the radian range of rotation, effortThe property limits the force range during rotation. (positive and negative value, the unit is cattle or N) velocityThe property limits the speed at which it turns, in meters/second or m/s.
- **mimic:** Describe the relationship of this joint to existing joints.
- **safety\_controller:** Describes safety controller parameters. Protect the movement of robot joints.