

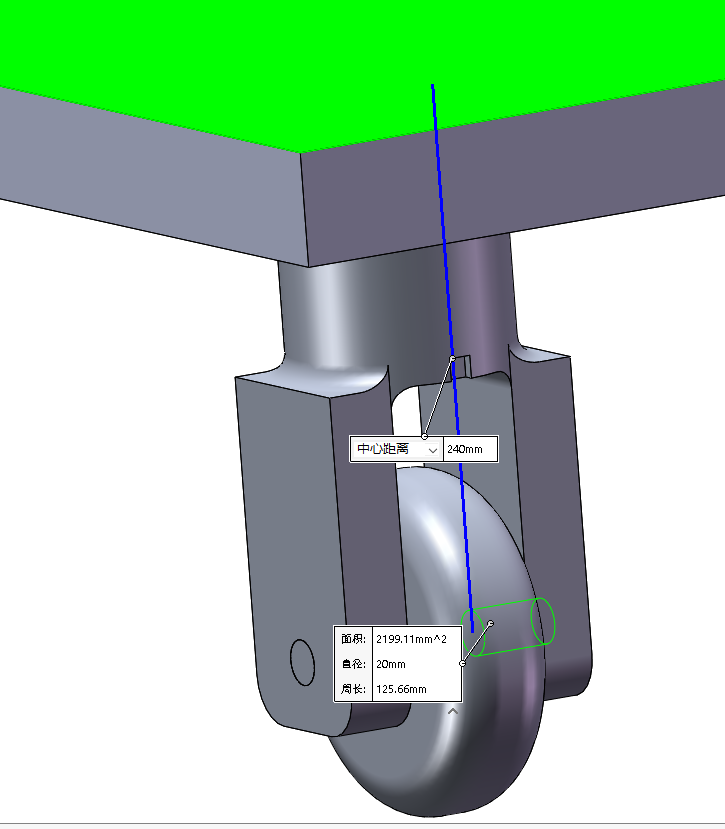
上图是一个麦克纳母轮的电机标号，我们模仿这种标号方式，右前方为第一个舵轮，逆时针为正，分别是第234个舵轮(俯视图)。底盘正方向为x方向

轮子的直径记录下来150mm

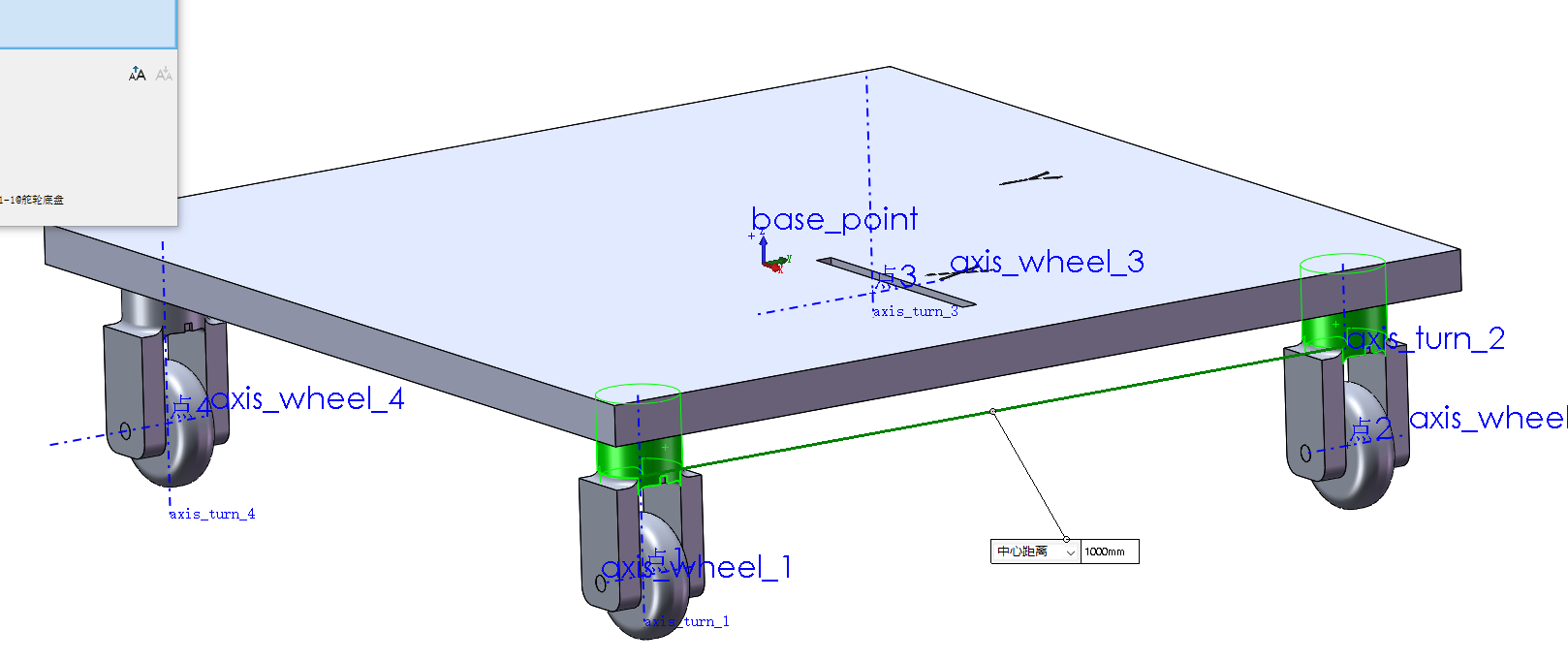


分别用不同的模型表示，复制粘贴，改改名字，之所以这样做，去看之前的文章。否则模型必炸。

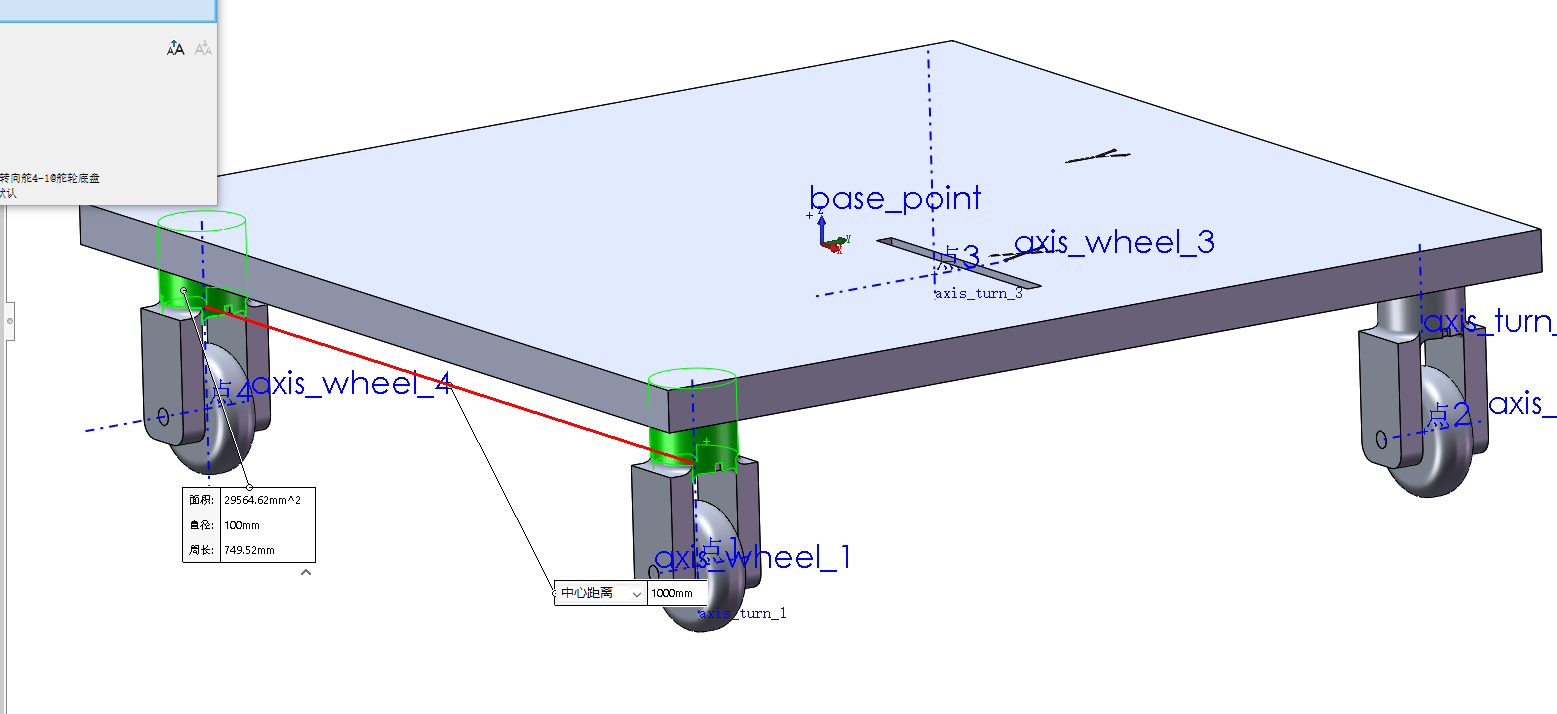
板子上面和轮轴相差240mm，加上半径75，一共是315mm



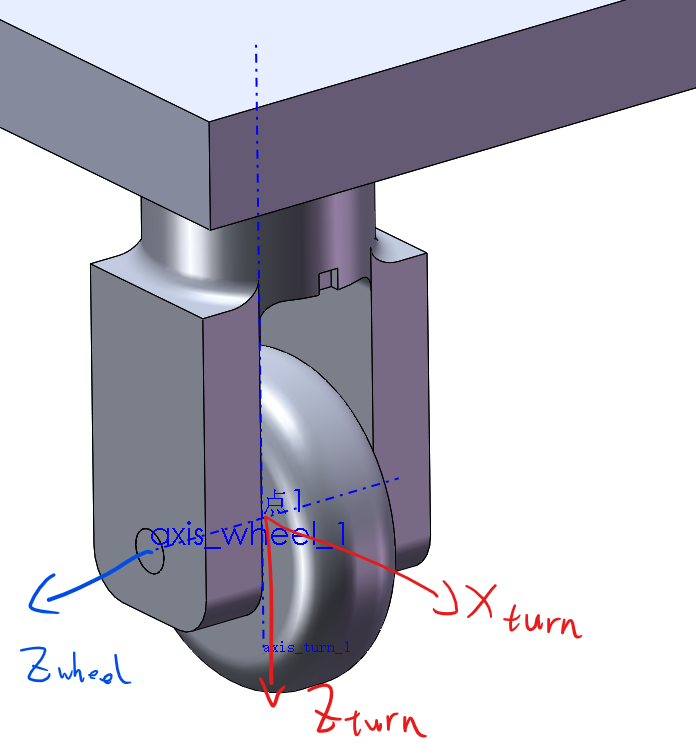
宽为1000mm



长为1000mm



因为转向电机是是向下安装的，所以我们仿真模型的Z轴应该朝下，x轴朝车辆的前方。行走驱动轮一般是向外安装，所以z轴朝外面

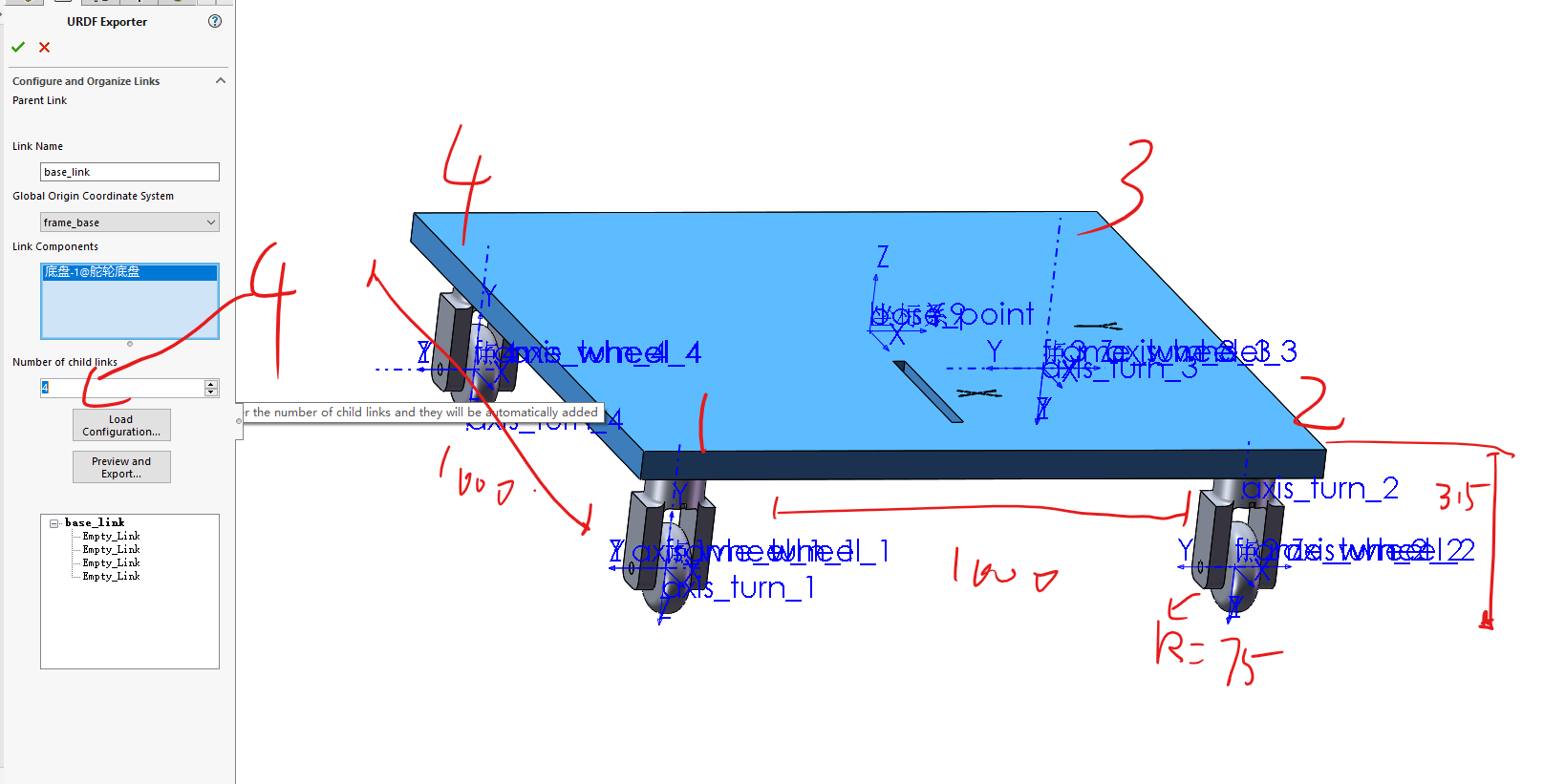


因为驱动轮是连续运动型的，xy轴随便

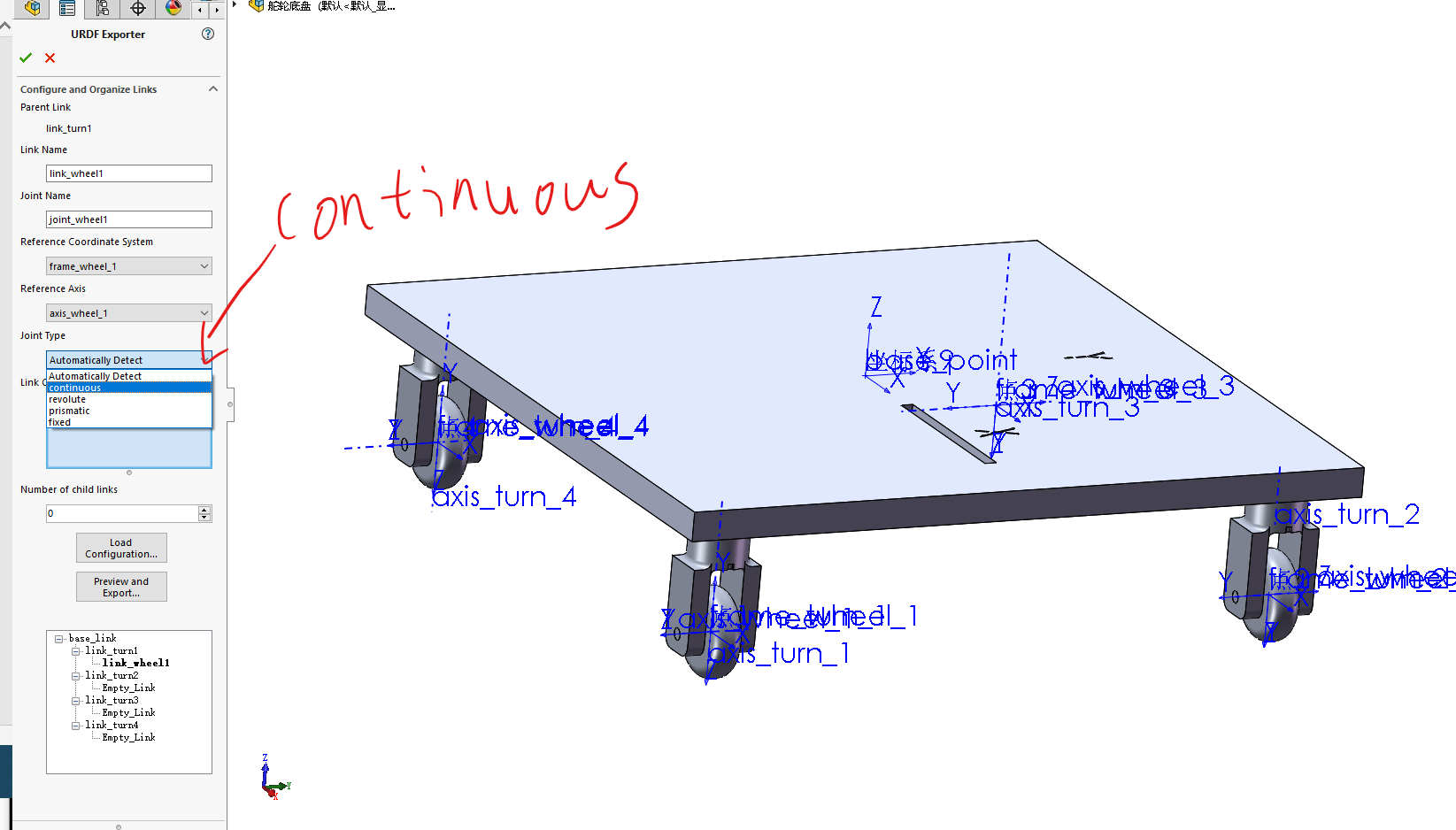
配置好了大概是这个样子



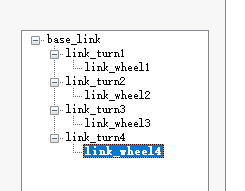
这里的child\_link就是4了



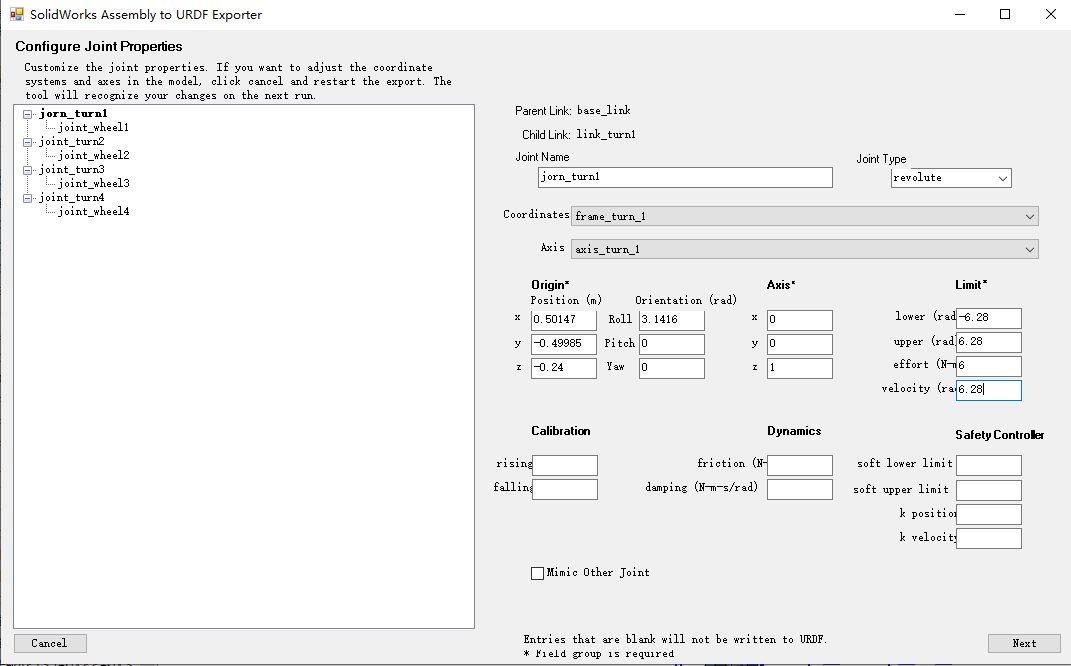
在选择wheel的joint类型的时候要选择continuous

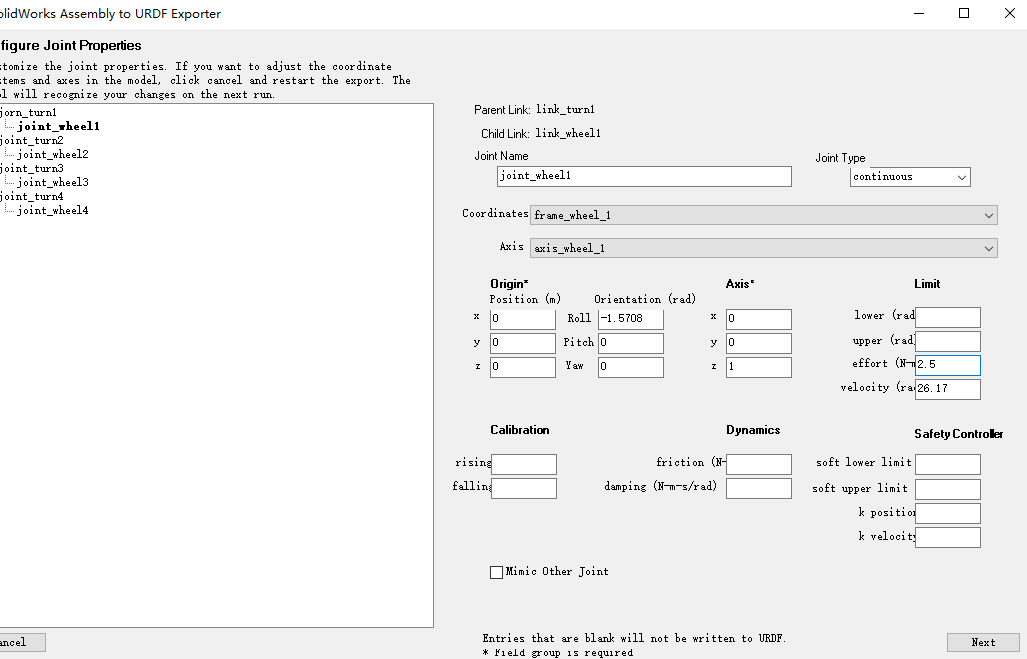


大概配置好了是这个树状图

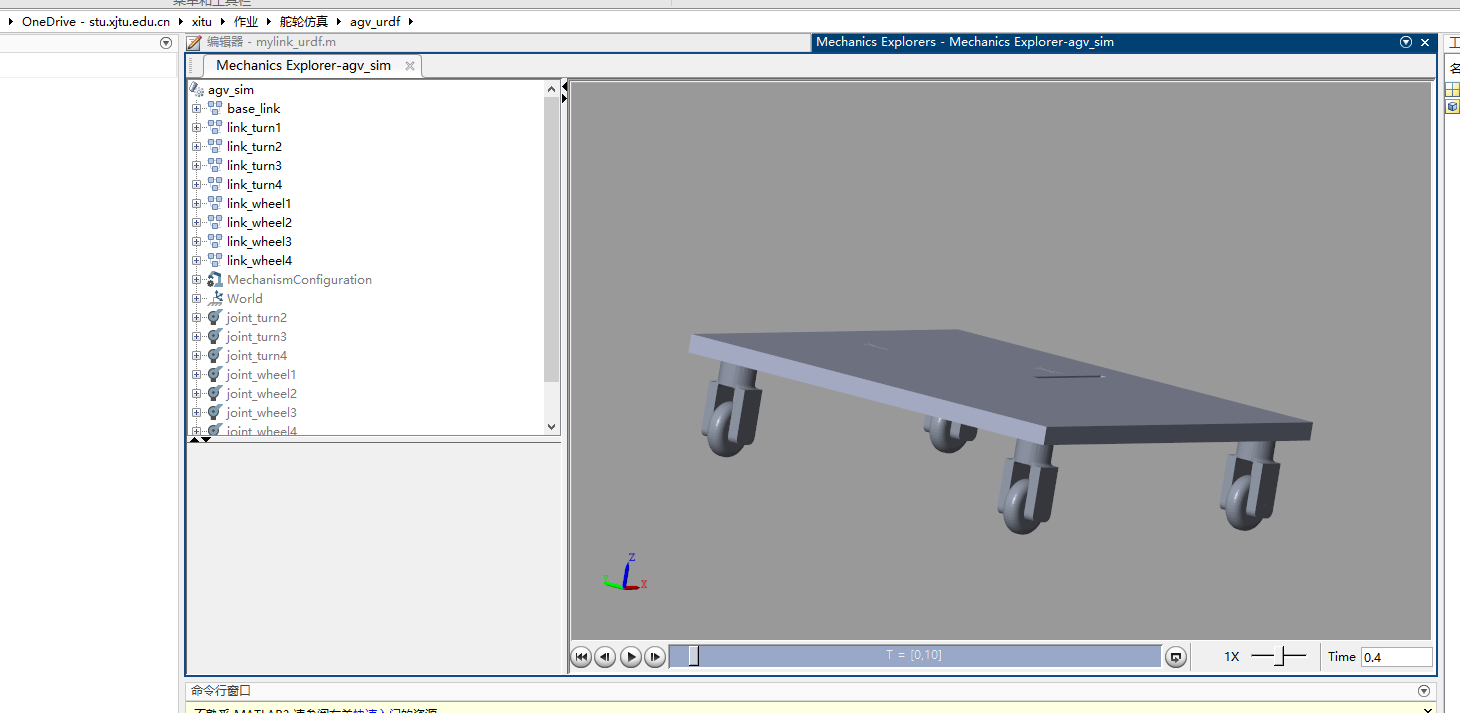


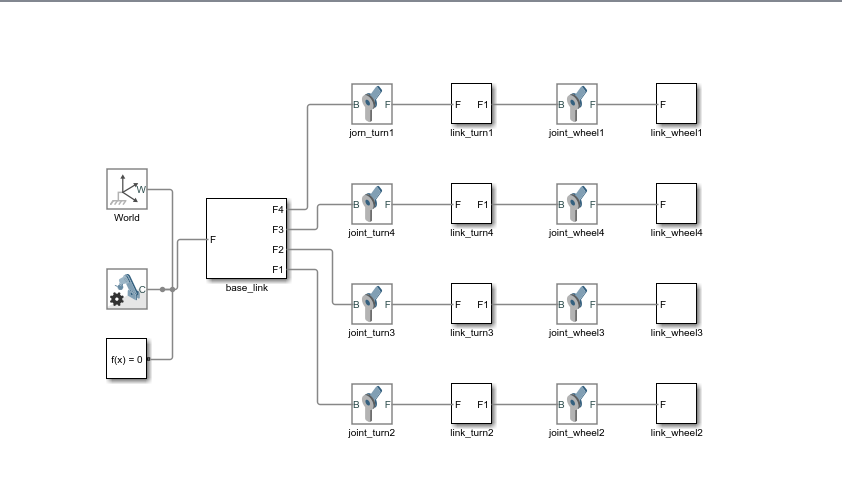
这里根据电机的情况设置





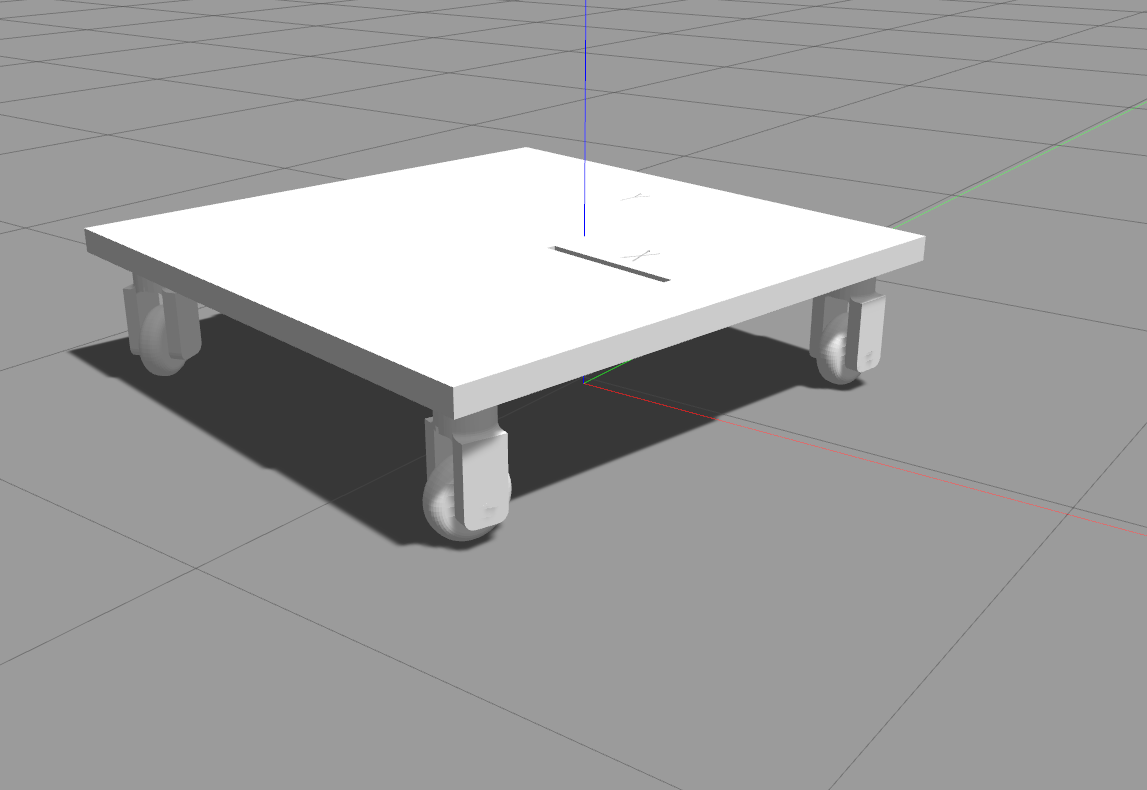
在matlab中显示





我们下一步转入ros中进行配置

我们用roslaunch进入gazebo



roslaunch agv\_sim gazebo.launch

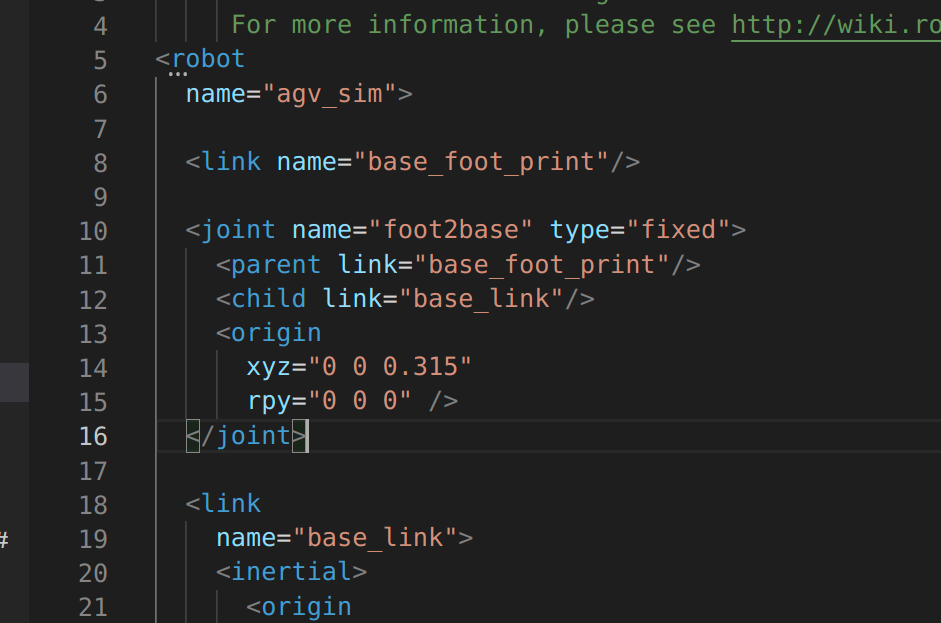
发现轮子晃来晃去,而且不再原点,这是因为gazebo是物理仿真环境,默认它的初始link在原点,还记得我们一开始配置的base\_link么?在机器人的上顶板,也就是说我们把它的初始位置按在了地板里...机器人不堪受辱弹了出来.解决这个问题的方法很简单,就是修改URDF模型...前提是你学习了urdf格式文件的编写…

软件插件配置不是万能的,必须学习最基础的操作.

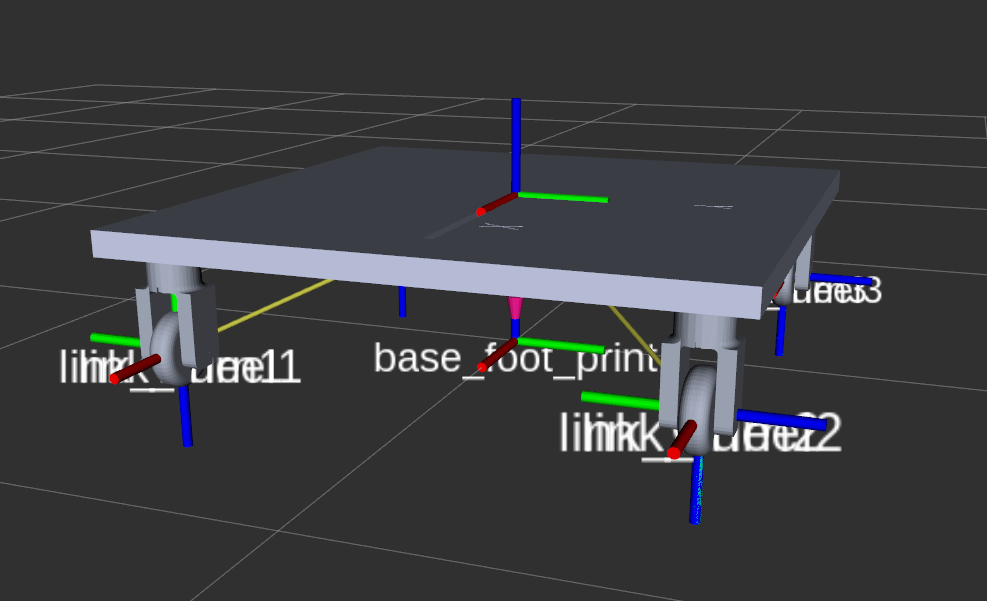
还记得之前那个315mm的高度么?

这里要派上用场了

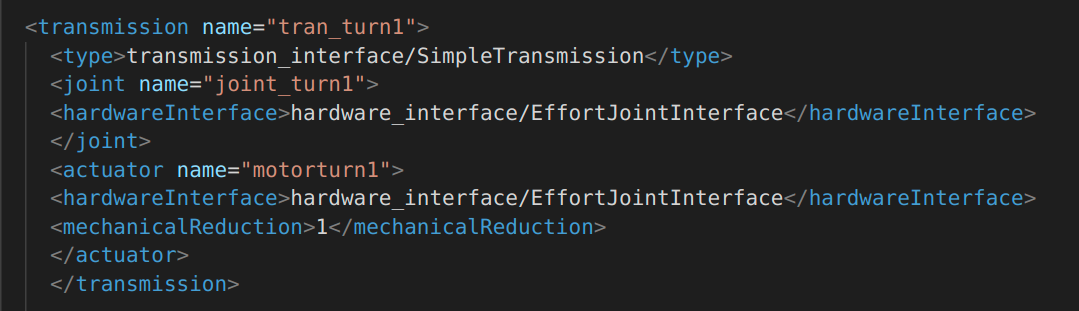
我们在urdf文件中base\_link前面添加一个这个玩意,link的名字随便取

我们通过rviz显示一下看看是否正常

roslaunch agv\_sim display.launch

我们可以看到一切正常

添加传动

<transmission name="tran\_turn1">

<type>transmission\_interface/SimpleTransmission</type>

<joint name="joint\_turn1">

<hardwareInterface>hardware\_interface/EffortJointInterface</hardwareInterface>

</joint>

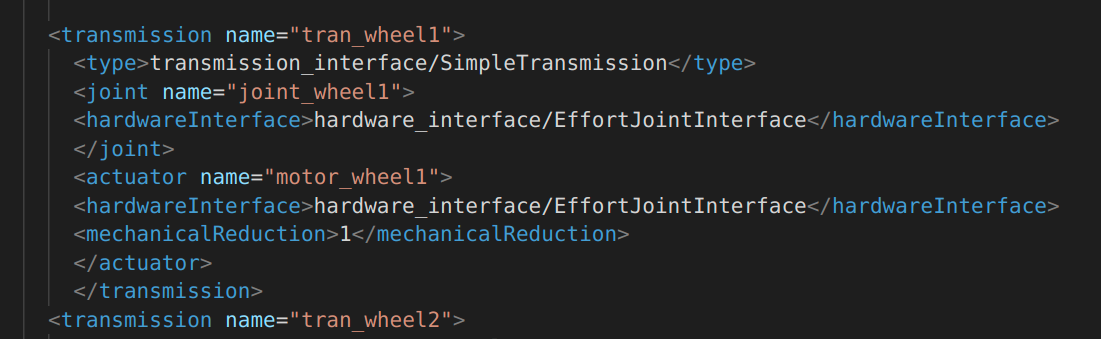
<actuator name="motorturn1">

<hardwareInterface>hardware\_interface/EffortJointInterface</hardwareInterface>

<mechanicalReduction>1</mechanicalReduction>

</actuator>

</transmission>

<transmission name="tran\_wheel1">

<type>transmission\_interface/SimpleTransmission</type>

<joint name="joint\_wheel1">

<hardwareInterface>hardware\_interface/EffortJointInterface</hardwareInterface>

</joint>

<actuator name="motor\_wheel1">

<hardwareInterface>hardware\_interface/EffortJointInterface</hardwareInterface>

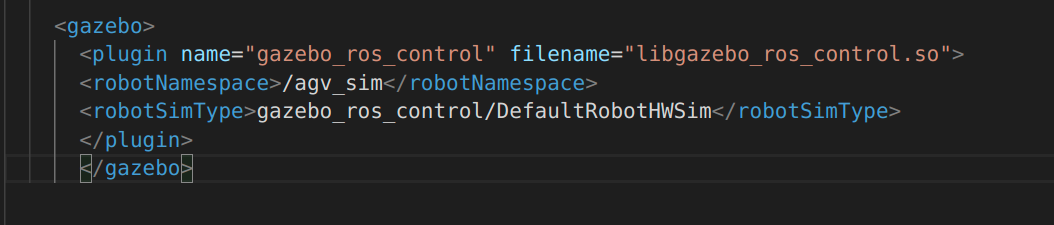
<mechanicalReduction>1</mechanicalReduction>

</actuator>

</transmission>

其他的同理,注意拼写错误.

添加gazebo标签:

<gazebo>

<plugin name="gazebo\_ros\_control" filename="libgazebo\_ros\_control.so">

<robotNamespace>/agv\_sim</robotNamespace>

<robotSimType>gazebo\_ros\_control/DefaultRobotHWSim</robotSimType>

</plugin>

</gazebo>

同样记住命名空间/agv\_sim

将机器人的描述参数加载到参数服务器中

有一个语句可加可不加

<gazebo>

<plugin name="joint\_state\_publisher" filename="libgazebo\_ros\_joint\_state\_publisher.so">

<jointName>joint\_turn1, joint\_turn2, joint\_turn3, joint\_turn4, joint\_wheel1,joint\_wheel2,joint\_wheel3,joint\_wheel4</jointName>

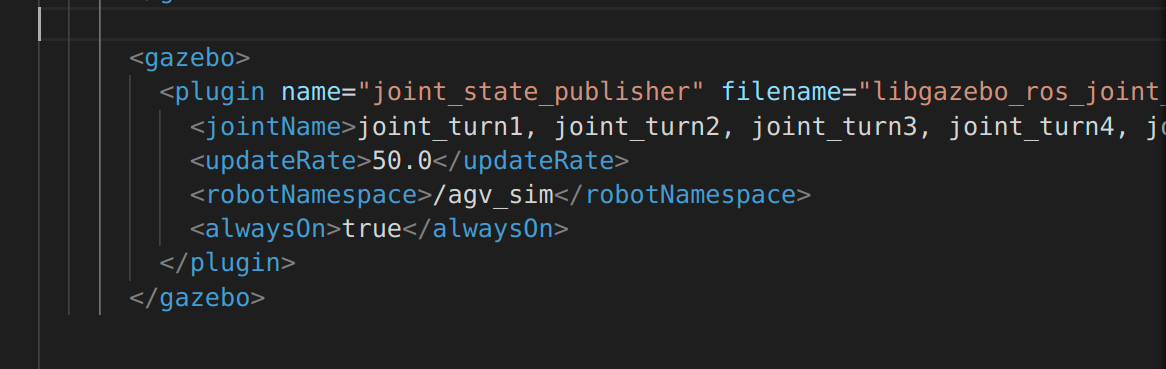
<updateRate>50.0</updateRate>

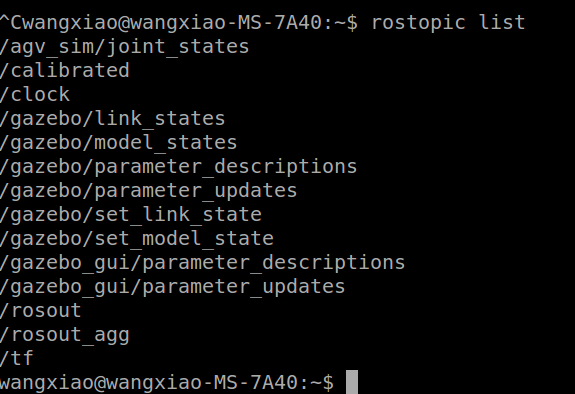
<robotNamespace>/agv\_sim</robotNamespace>

<alwaysOn>true</alwaysOn>

</plugin>

</gazebo>

如果加上,那么在启动gazebo的时候就开始发布关节状态,如果不加,在启动控制器的时候才会开始发布关节状态

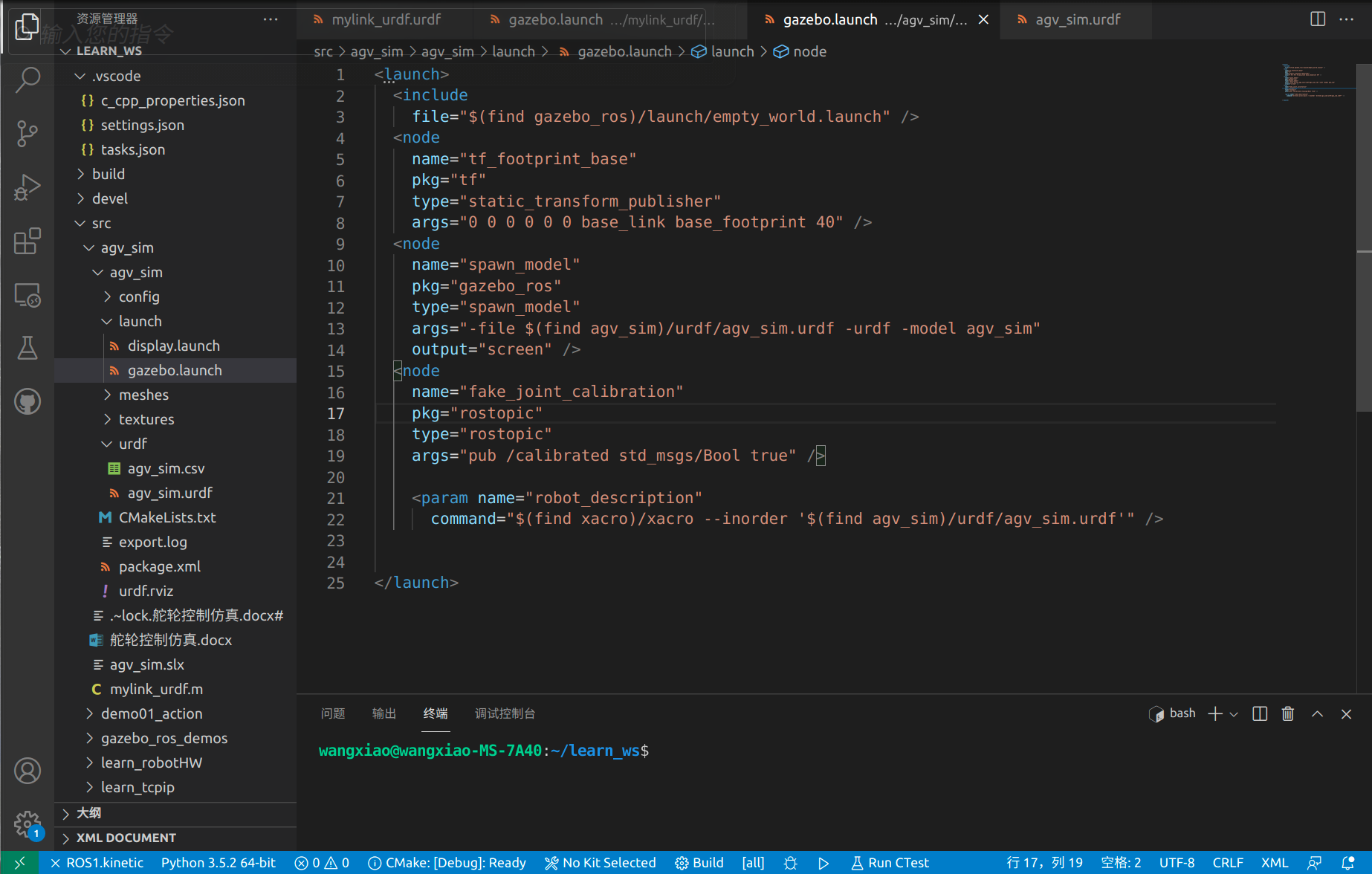
不加的话启动gazebo之后就没有第一行那个话题/agv\_sim/joint\_states

但是在你启动控制器的时候,这个话题就会自动生成...

我们要修改gazebo.launch文件,加入以下语句

<param name="robot\_description"

command="$(find xacro)/xacro --inorder '$(find agv\_sim)/urdf/agv\_sim.urdf'" />



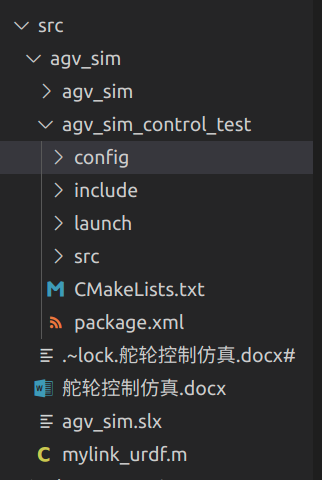
运行一下gazebo.launch,没有错误..

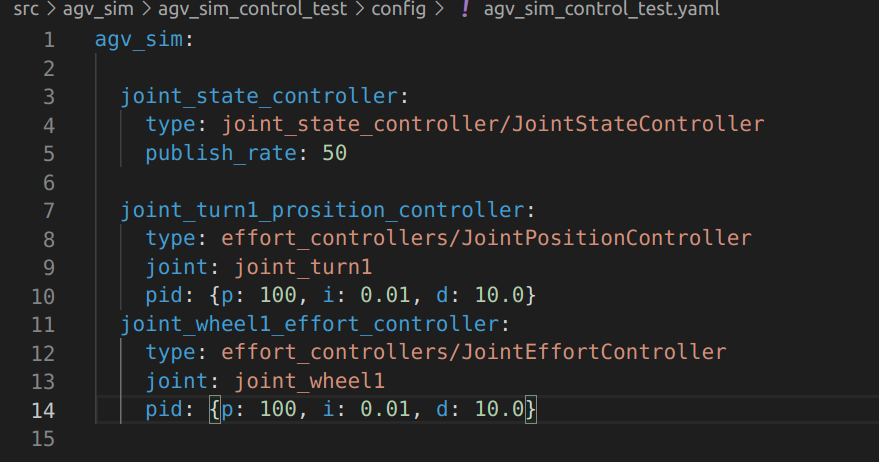
接下来编写控制器,因为这次做个小测试,所以只加入一个turn的控制器和一个行走电机的控制器做下测试

创建功能包,功能包名字随便取,导入依赖

controller\_manager joint\_state\_controller robot\_state\_publisher roscpp rospy

同样,新建config和launch文件夹

新建config文件



先这样写着,待会看有没有bug

编写launch文件

<launch>

<!-- Load joint controller configurations from YAML file to parameter server -->

<rosparam file="$(find agv\_sim\_control\_test)/config/agv\_sim\_control\_test.yaml" command="load"/>

<!-- load the controllers -->

<node name="controller\_spawner" pkg="controller\_manager" type="spawner" respawn="false"

output="screen" ns="/agv\_sim" args="joint\_turn1\_prosition\_controller

joint\_wheel1\_effort\_controller

joint\_state\_controller"/>

<!-- convert joint states to TF transforms for rviz, etc -->

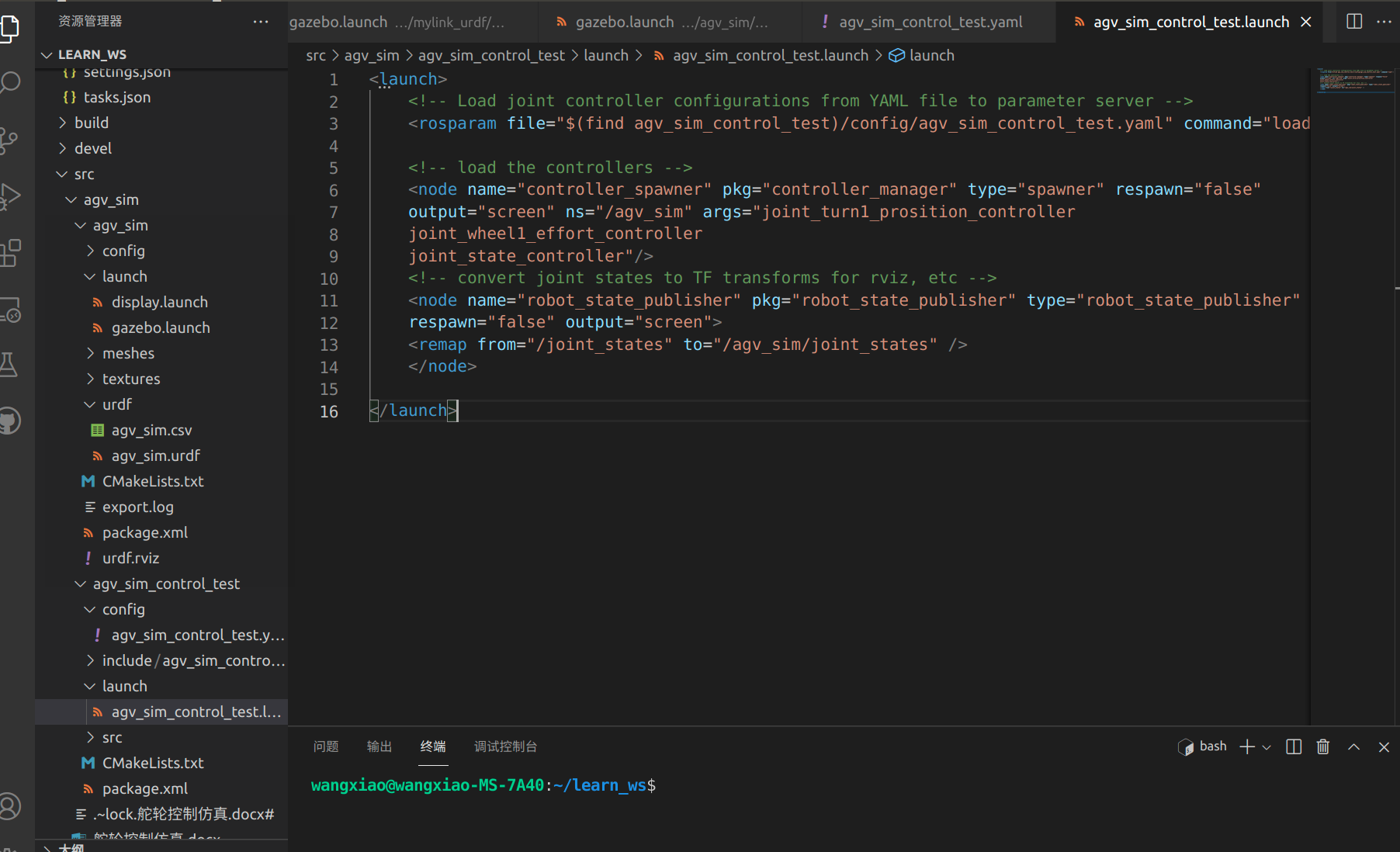
<node name="robot\_state\_publisher" pkg="robot\_state\_publisher" type="robot\_state\_publisher"

respawn="false" output="screen">

<remap from="/joint\_states" to="/agv\_sim/joint\_states" />

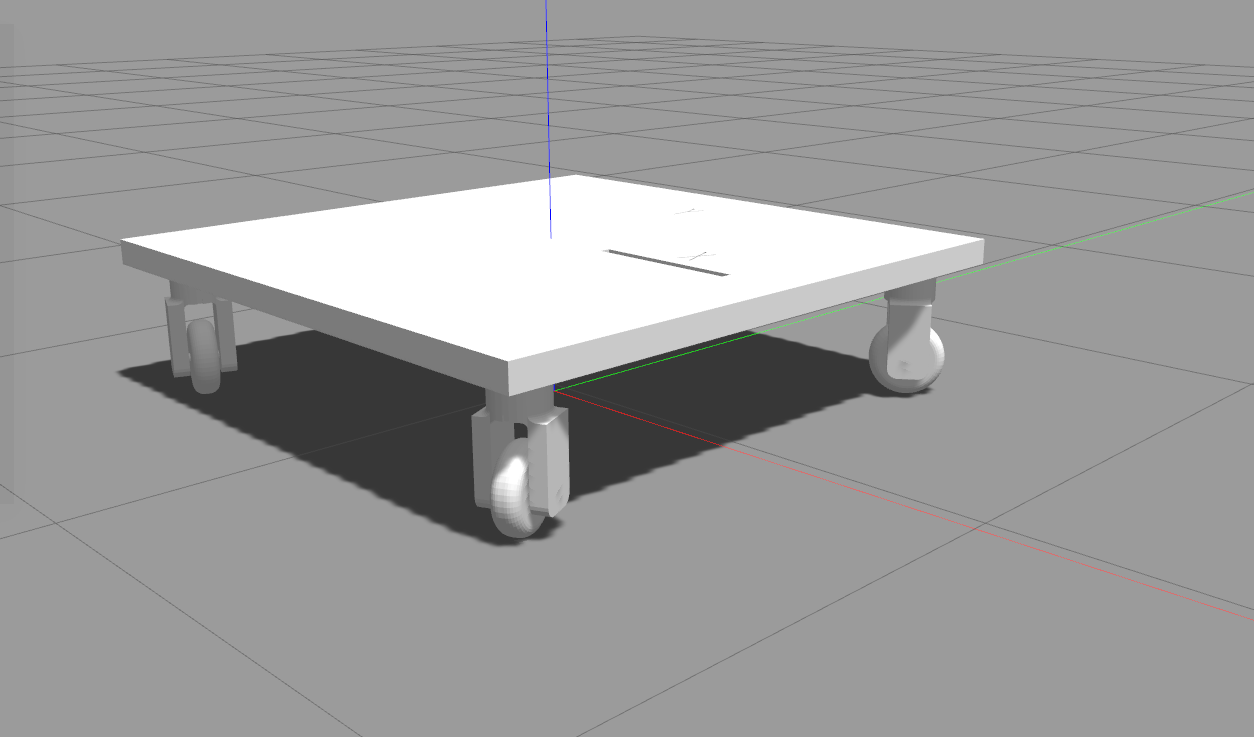
</node>

</launch>



我们启动gazebo验证一下,见证奇迹的时刻就要到了

roslaunch agv\_sim gazebo.launch



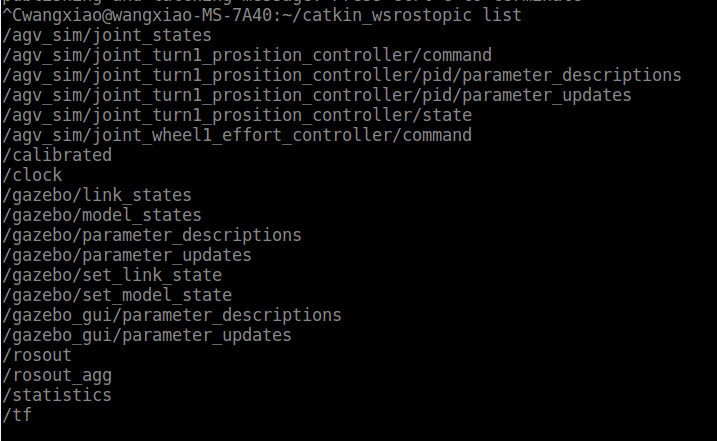
机器人正常显示

运行控制器:

roslaunch agv\_sim\_control\_test agv\_sim\_control\_test.launch

完全没有报错

rostopic list查看一下当前的topic

发现那个带command的就是我们可以发布控制命令的接口

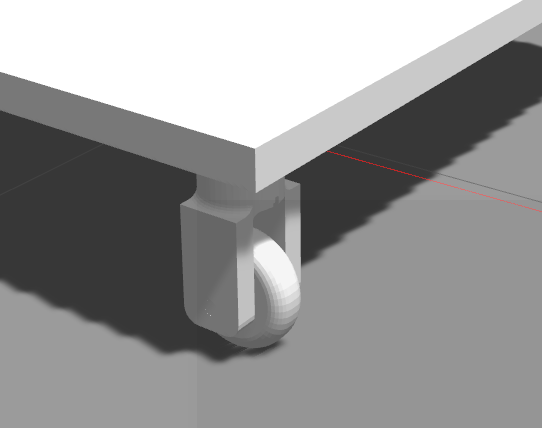
我们用rostopic pub 发送命令

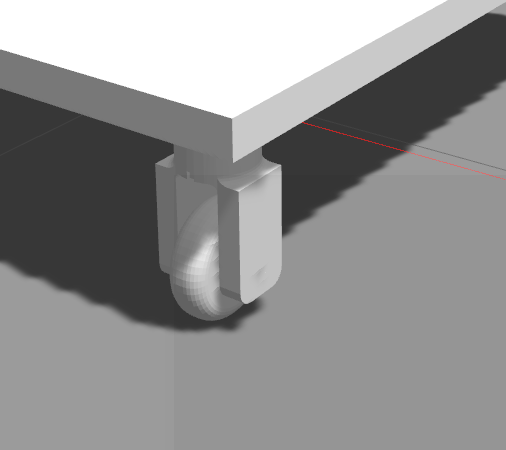
rostopic pub /agv\_sim/joint\_turn1\_prosition\_controller/command std\_msgs/Float64 0

用TAB键补全格式

给0角度

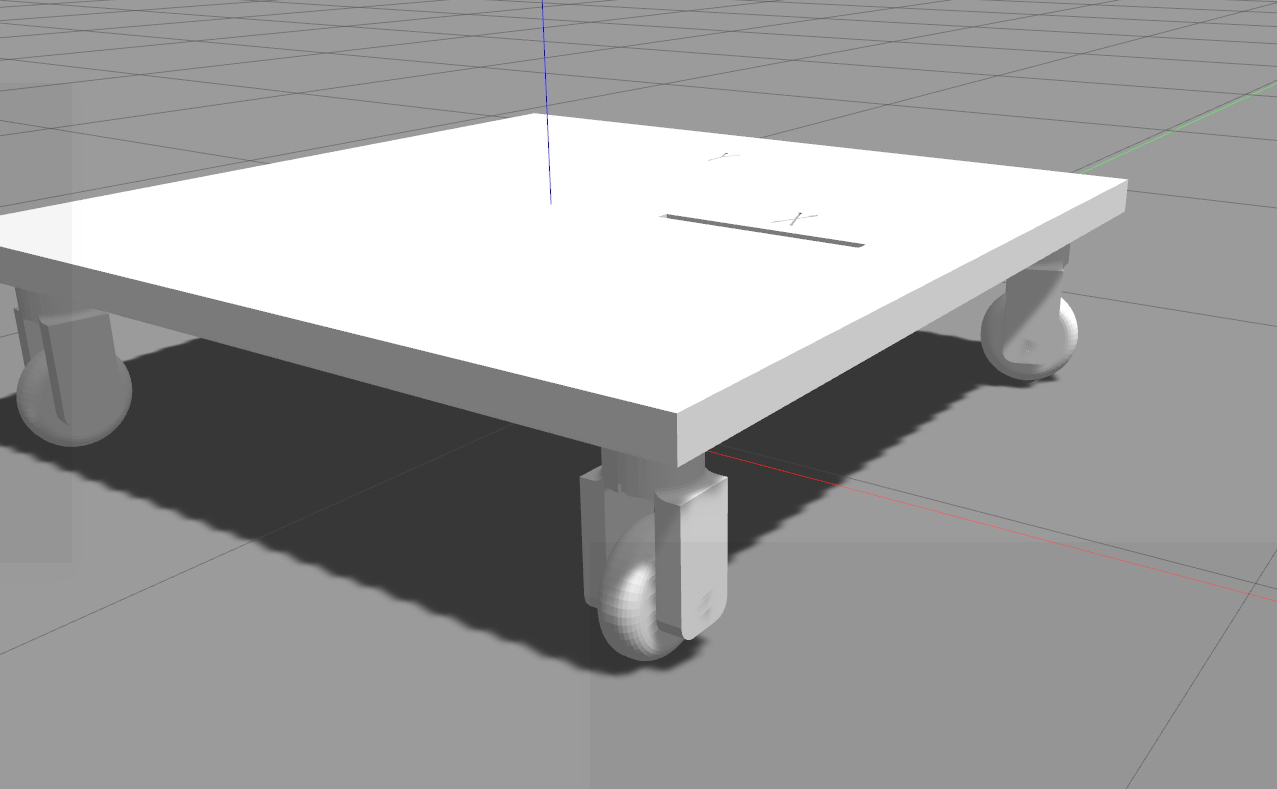
我们发现gazebo的角度归零

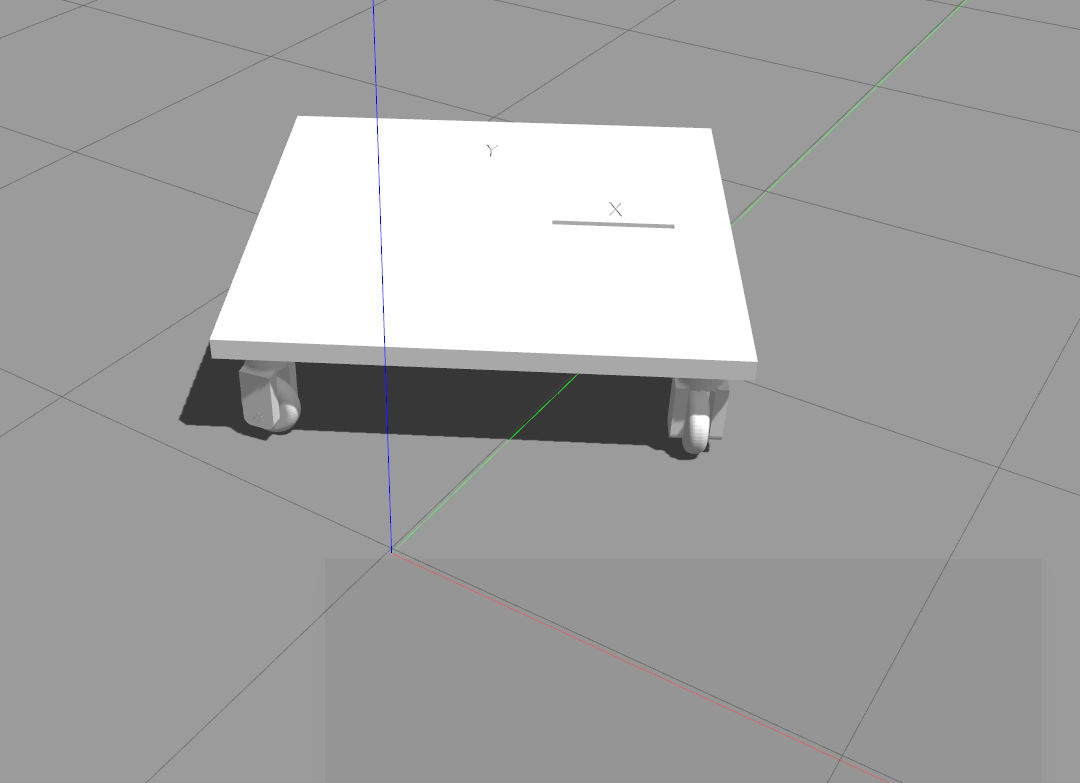
再发一个1.57,,发现转了90度

我们给轮子发布命令

rostopic pub /agv\_sim/joinwheel1\_effort\_controller/command std\_msgs/Float64 2.5

发现轮子转了起来,并且在缓慢移动

但是由于其它的轮子没有加入控制器,只有这一个轮子进行驱动,而且并没有设置地面摩擦参数,所以有打滑的现象.

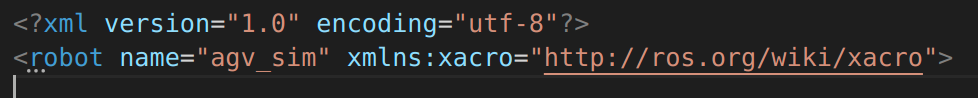
现在,我们要给小车提供一下摩擦参数等,

提示:最好是将urdf文件改为xacro

具体操作:

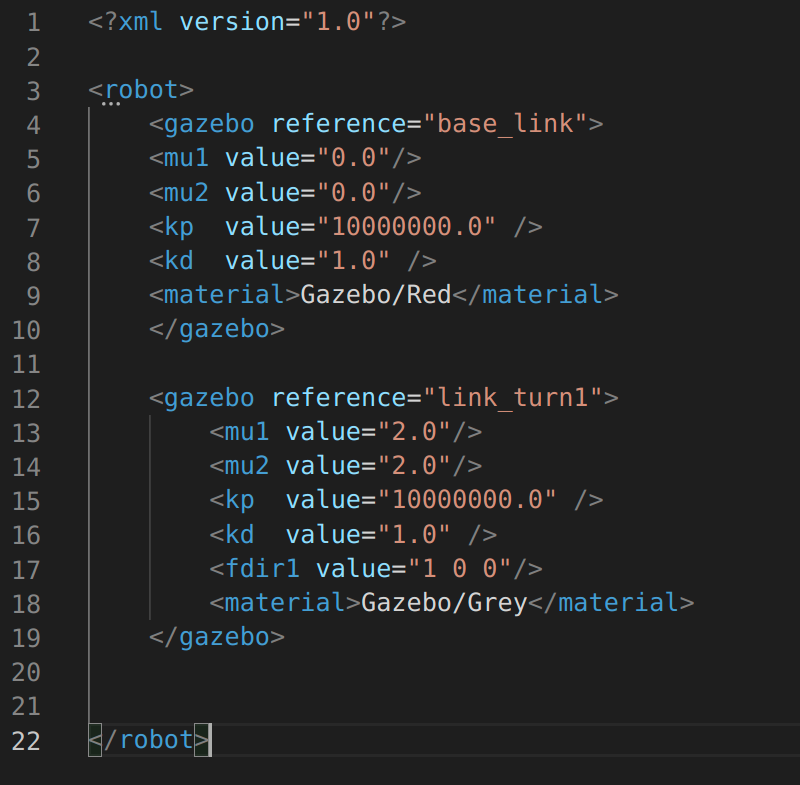
新建一个xacro文件 ,将urdf文件内容整个的复制过来

将robot改为:



<robot name="agv\_sim" xmlns:xacro="http://ros.org/wiki/xacro">

为我们新建一个xacro文件配置一些参数

<gazebo reference="base\_link">

<mu1 value="0.0"/>

<mu2 value="0.0"/>

<kp value="10000000.0" />

<kd value="1.0" />

<material>Gazebo/Red</material>

</gazebo>

其中:Kp为材料的刚度,kd 为材料的阻尼

<gazebo reference="link\_turn1">

<mu1 value="2.0"/>

<mu2 value="2.0"/>

<kp value="10000000.0" />

<kd value="1.0" />

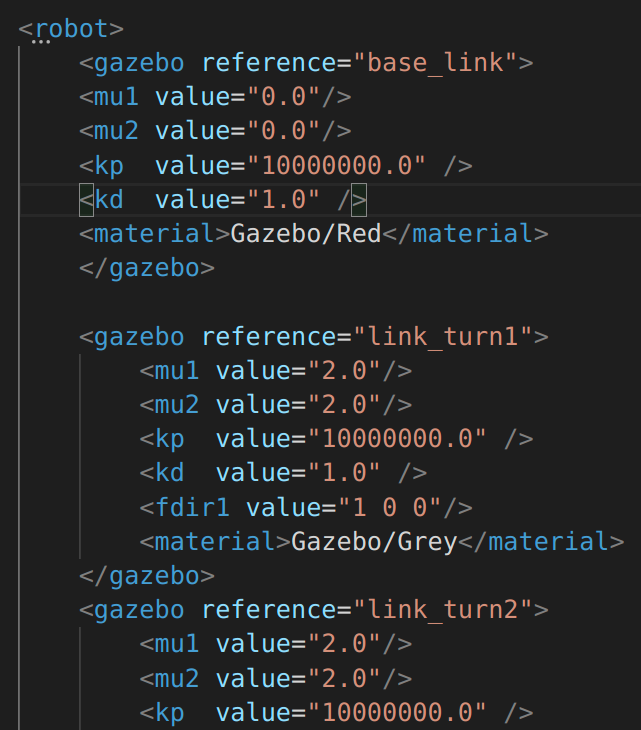
<fdir1 value="1 0 0"/>

<material>Gazebo/Grey</material>

</gazebo>

其中:mu1沿第一摩擦方向的摩擦系数

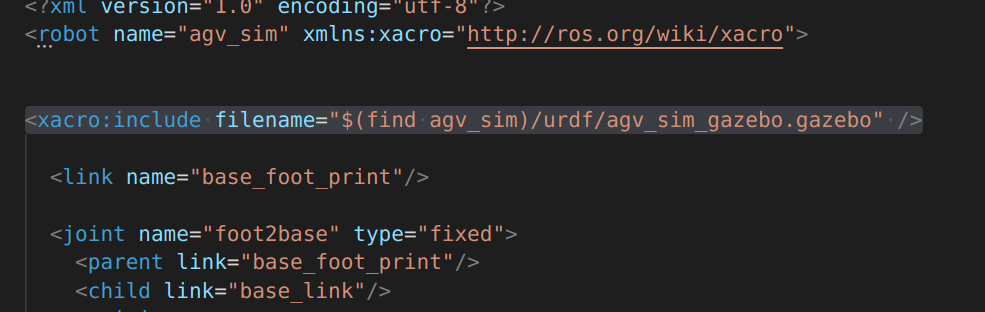
mu2沿第二摩擦方向的摩擦系数

类似于下图

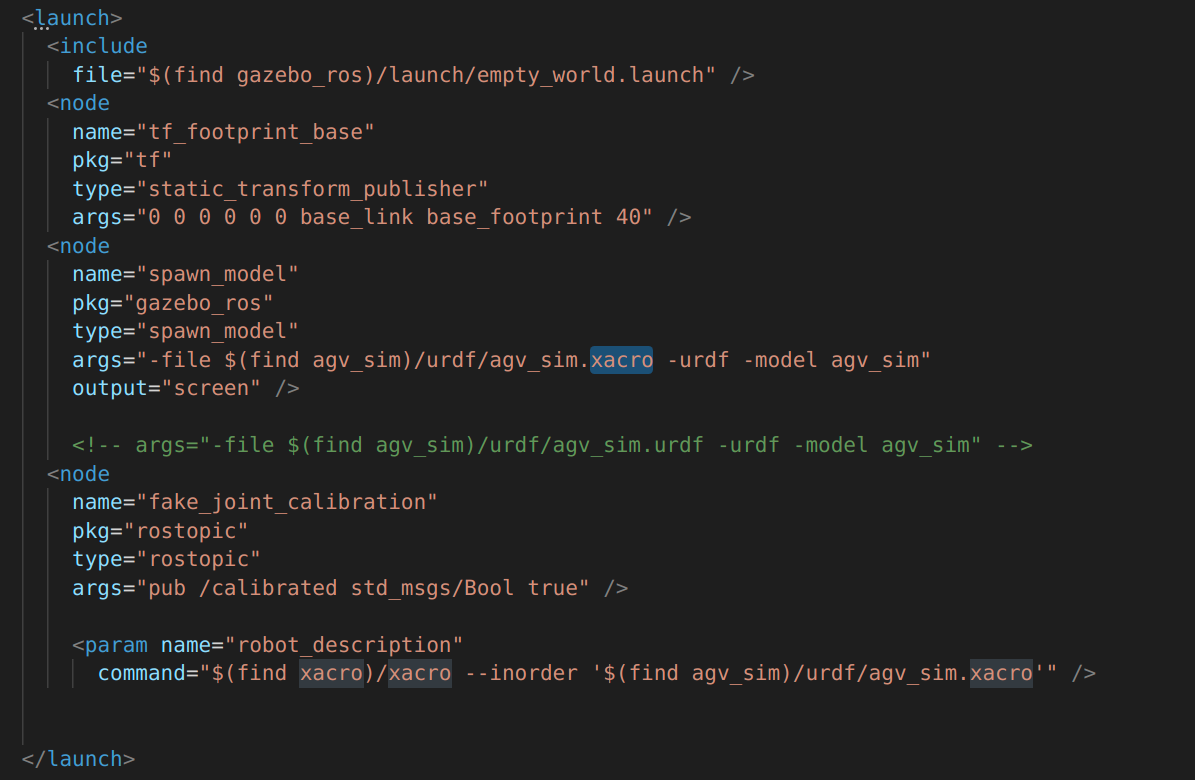
转到agv\_sim.xacro中

include刚才的文件

<xacro:include filename="$(find agv\_sim)/urdf/agv\_sim\_gazebo.gazebo" />

下面修改下launch文件

将urdf文件加载路径改为xacro的加载路径(高亮显示部分)

注意这里有两个地方需要改

启动gazebo没什么问题就可以了

我这里不知道为什么,include之后模型就不好用了,所以我还是改成了统一个文件

/////////////////////////////我是一调分割线

接下来我们要写一个完整的多论控制器

首先我们要明白舵轮的4个转向电机要工作在位置模式下,舵轮的行走电机,要工作在力矩模式下...所以我们先来定义接口

同样,新建一个cpp和一个.h文件,我们把.h文件中

,h中include文件为

#include <ros/node\_handle.h>

#include <ros/ros.h>

#include <urdf/model.h>

#include <control\_toolbox/pid.h>

#include <boost/scoped\_ptr.hpp>

#include <boost/thread/condition.hpp>

#include <realtime\_tools/realtime\_publisher.h>

#include <hardware\_interface/joint\_command\_interface.h>

#include <controller\_interface/controller.h>

#include <control\_msgs/JointControllerState.h>

#include <std\_msgs/Float64.h>

#include <control\_msgs/JointControllerState.h>

#include <realtime\_tools/realtime\_buffer.h>

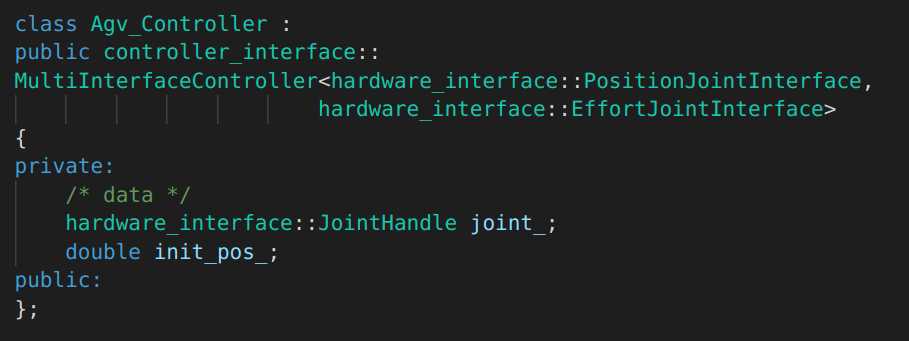
#include <controller\_interface/controller.h>

#include <hardware\_interface/joint\_command\_interface.h>

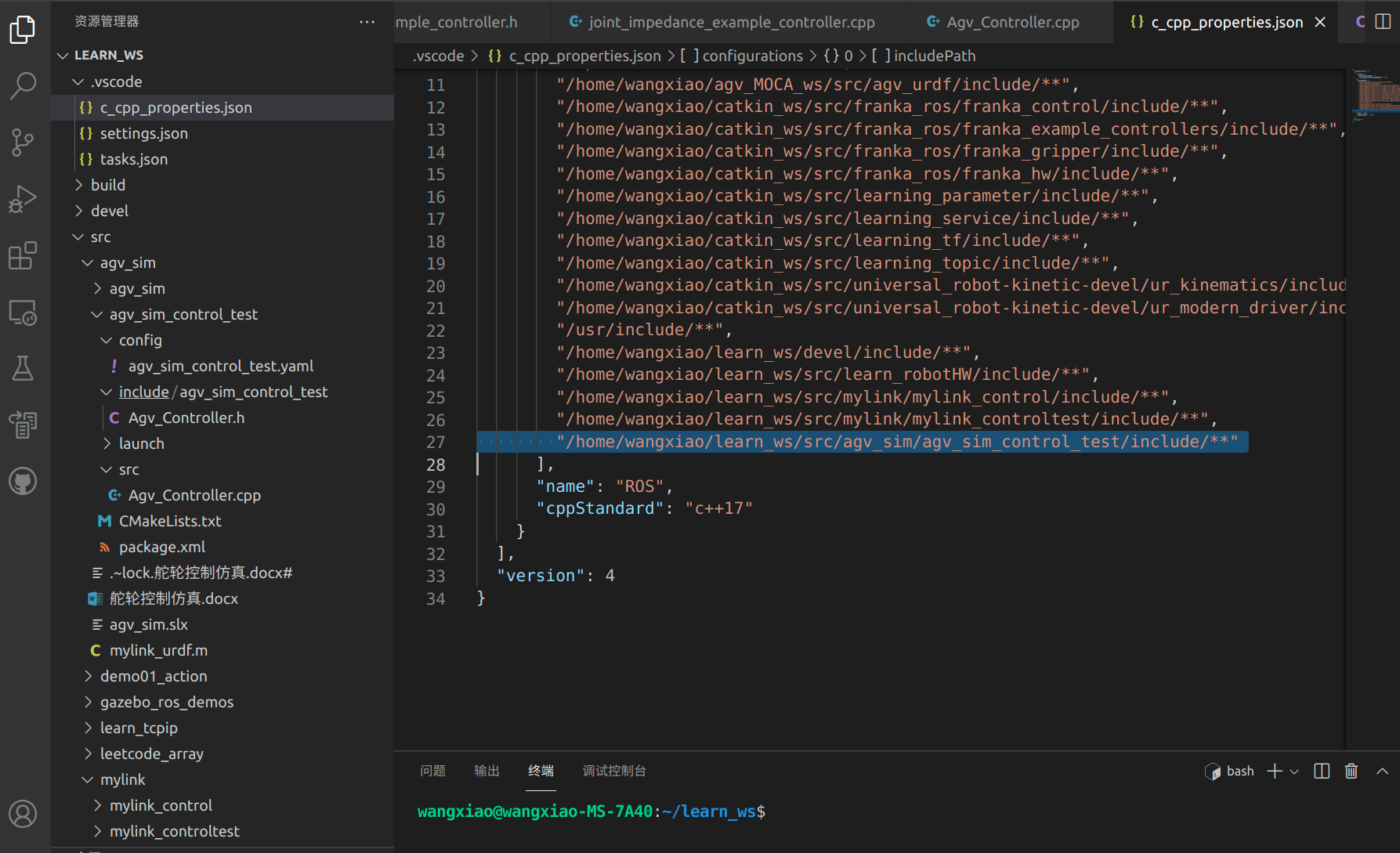
#include <controller\_interface/multi\_interface\_controller.h>

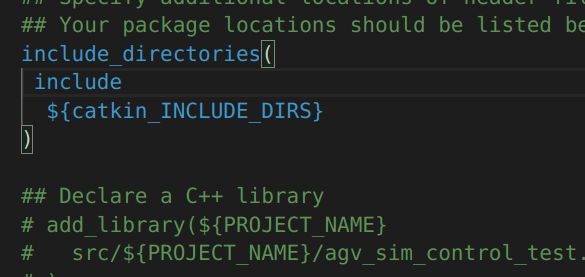
此时比以前多了一个文件,multi\_interface\_controller,这个文件是允许继承多个接口

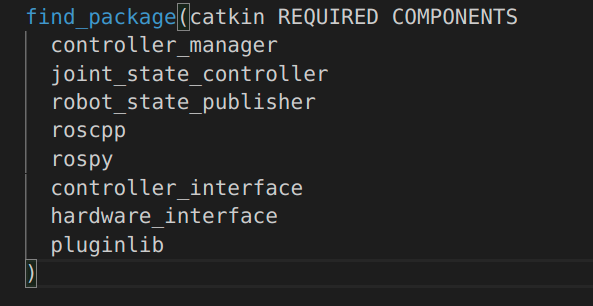
我们来编写继承类

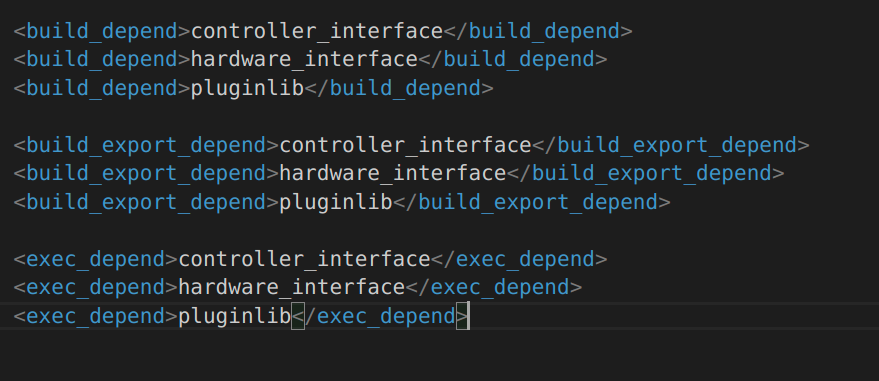


在c\_cpp…….配置文件里配置路径防止报错

cmakelist里面打开include

检查package.xml和cmakelist里面的包都包含了



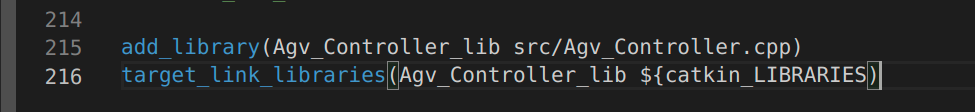
11

别忘了在cpp中

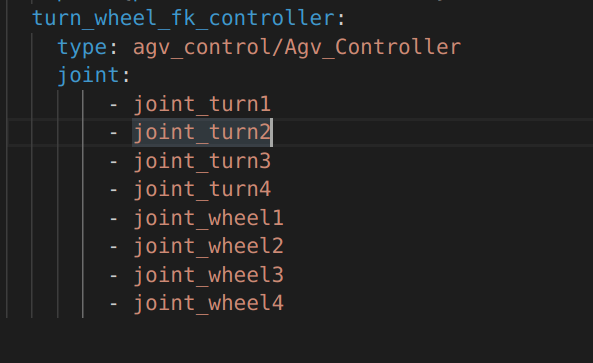
#include <pluginlib/class\_list\_macros.h>

否则插件会报错

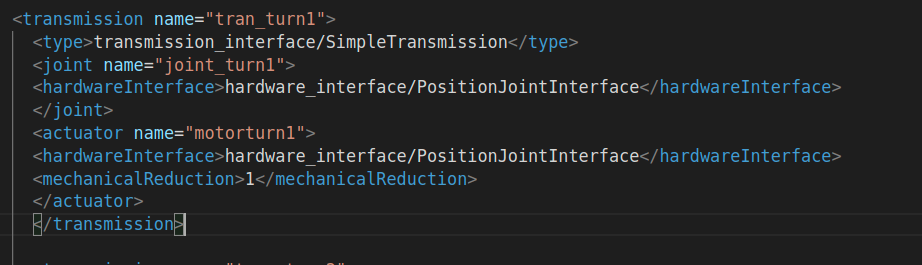
cmaklist编译生成动态连接库

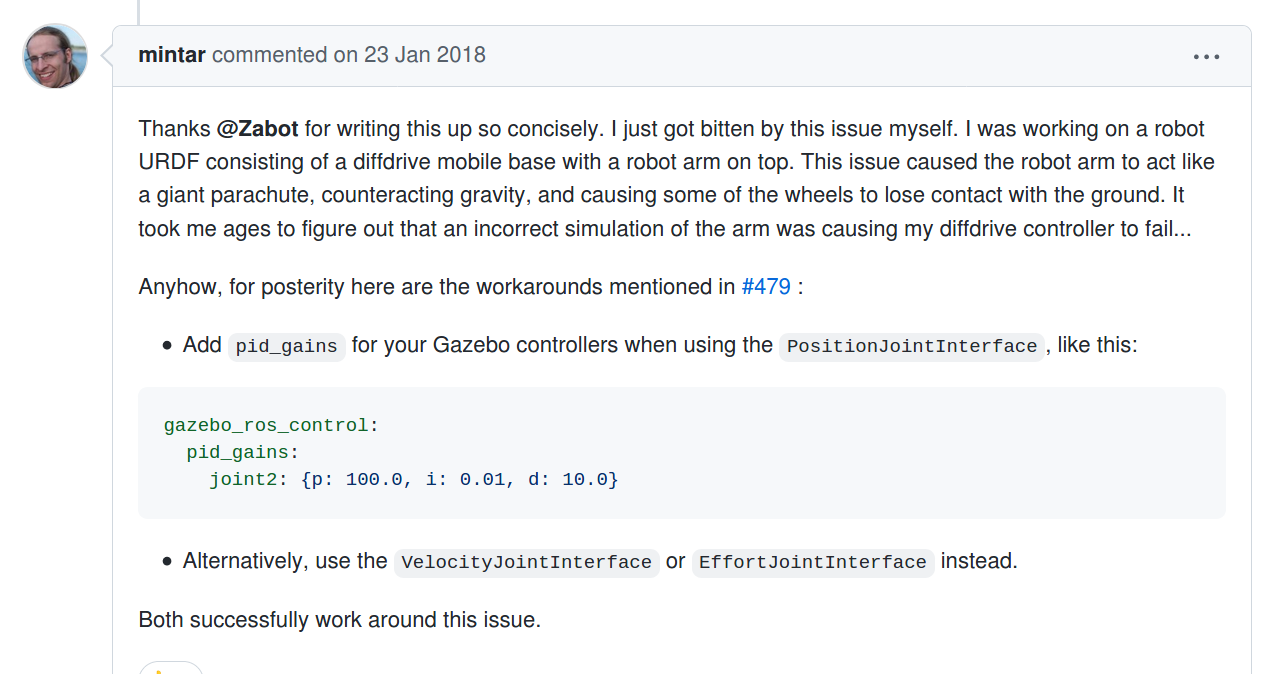
其他的跟上一节差不多,,我就不费精力去写了

我们有4个转向,一个行走电机,那么定义多个关节的时候就要用到这个



我们来更改urdf中gazebo标签信息,将先前设定的转向电机力矩接口设置为位置控制器接口

但是光这样还不行,如果你用的是gazebo9以下的版本,那么不设置pid参数就会报错,然后仿真的时候没有重力.这里给出一个老外的帖子

[](https://github.com/ros-simulation/gazebo_ros_pkgs/issues/612)https://github.com/ros-simulation/gazebo\_ros\_pkgs/issues/612

我们在config文件夹下新建一个yaml格式的文件,往参数服务器里面添加东西

gazebo\_ros\_control:

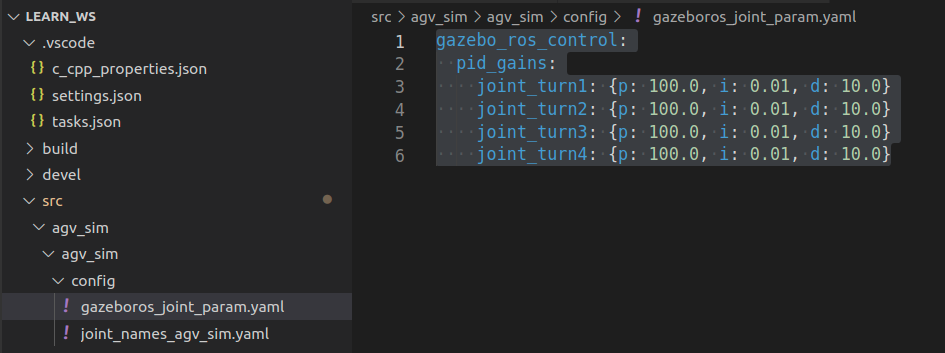
pid\_gains:

joint\_turn1: {p: 100.0, i: 0.01, d: 10.0}

joint\_turn2: {p: 100.0, i: 0.01, d: 10.0}

joint\_turn3: {p: 100.0, i: 0.01, d: 10.0}

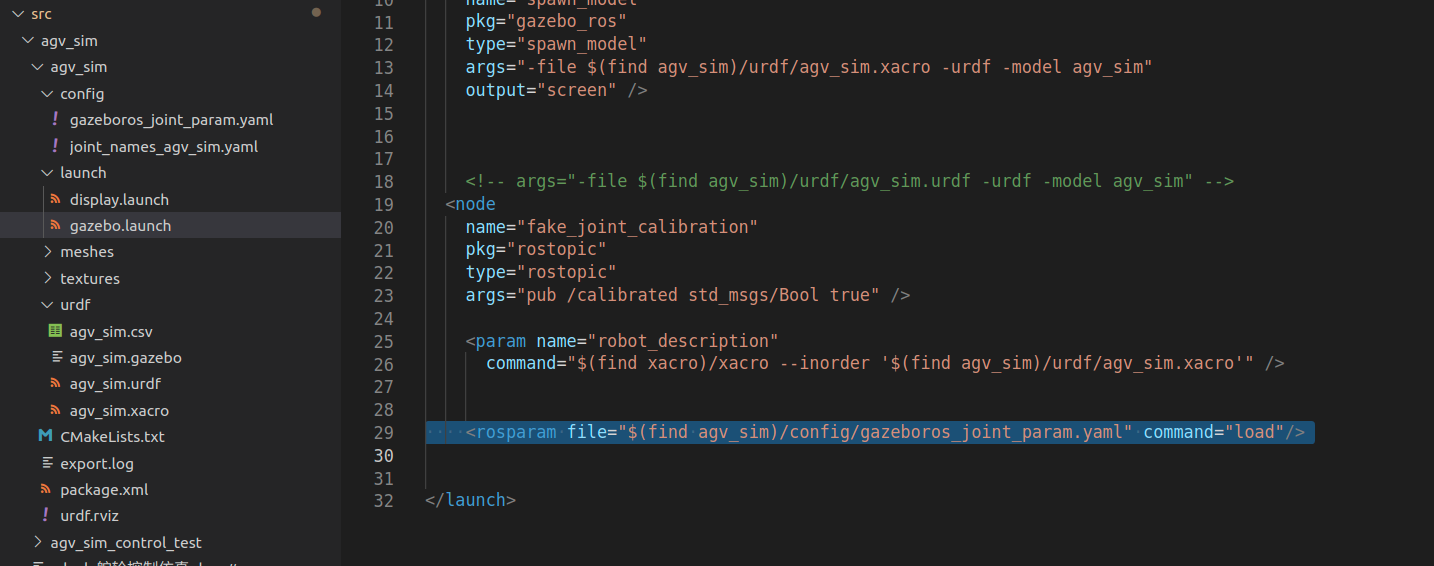
joint\_turn4: {p: 100.0, i: 0.01, d: 10.0}



然后在launch文件中添加这些参数(因为不会补全,所以要注意拼写)

<rosparam file="$(find agv\_sim)/config/gazeboros\_joint\_param.yaml" command="load"/>

gazeboros\_joint\_param.yaml这个文件就是我刚才新建的

launch文件运行一下没有报错就不截图了

之前总是冒进,这次准备稳扎稳打

现在编写init函数,init函数主要是负责将