**实验项目七 ROS机器人运动控制**

1. **实验目的**

1、 掌握机器人仿真模型的传动装置设置；

2、 掌握机器人仿真模型的控制器设计方法；

3、 掌握Gazebo软件的使用方法；

4、 xacro文件的介绍与使用。

1. **实验设备**

1、 雷达模块一台；

2、 无线键鼠一套；

3、 显示器一套。

1. **实验步骤和内容**

**1、Rviz中控制机器人模型运动**

通过URDF结合rviz可以创建并显示机器人模型，不过，当前实现的只是静态模型，如何控制模型的运动呢？在此，可以调用Arbotix实现此功能。

**简介：**

Arbotix：Arbotix是一款控制电机、舵机的控制板，并提供相应的ros 功能包，这个功能包的功能不仅可以驱动真实的Arbotix控制板，它还提供一个差速控制器，通过接受速度控制指令更新机器人的joint状态，从而帮助我们实现机器人在rviz中的运动。

这个差速控制器在arbotix\_python程序包中，完整的arbotix程序包还包括多种控制器，分别对应dynamixel电机、多关节机械臂以及不同形状的夹持器。

接下来，通过一个案例演示arbotix的使用。

**需求描述：**

控制机器人模型在rviz中做圆周运动。

**实现流程：**

1. 安装Arbotix；
2. 创建新功能包，准备机器人urdf、xacro文件；
3. 添加Arbotix配置文件；
4. 编写launch文件配置Arbotix；
5. 启动launch文件并控制机器人模型运动。

**（1）安装Arbotix**

方式1：命令行调用

$ sudo apt-get install ros-<<VersionName()>>-arbotix

将<<VsersionName()>>替换成当前ROS版本名称，机器人里ROS版本是melodic，如果提示功能包无法定位，请采用方式2。

方式2：源码安装

先从github下载源码，然后调用catkin\_make编译：

$ git clone https://github.com/vanadiumlabs/arbotix\_ros.git

如果以上两种方法都不可用，可以在浏览器搜索ros arbotix，进入官网查看官方提供的网址，更换网址后重复方式2。

**（2）创建新功能包，准备机器人urdf、xacro**

urdf和xacro调用之前实现即可。

**（3）添加arbotix所需的配置文件**

添加arbotix所需配置文件,详情见test\_02功能包中config文件夹里的control.yaml文件：

# 该文件是控制器配置,一个机器人模型可能有多个控制器，比如: 底盘、机械臂、夹持器(机械手)....

# 因此，根 name 是 controller

controllers: {

# 单控制器设置

base\_controller: {

#类型: 差速控制器

type: diff\_controller,

#参考坐标

base\_frame\_id: base\_footprint,

#两个轮子之间的间距

base\_width: 0.2,

#控制频率

ticks\_meter: 2000,

#PID控制参数，使机器人车轮快速达到预期速度

Kp: 12,

Kd: 12,

Ki: 0,

Ko: 50,

#加速限制

accel\_limit: 1.0

}

}

**另请参考：**[http://wiki.ros.org/arbotix\_python/diff\_controller](http://wiki.ros.org/arbotix_python/diff_controller" \t "/home/ha/文档x/_blank)

**（4）launch文件中配置arbotix节点**

部分launch示例代码，test\_02中launch文件下的control.launch文件下方，是已经配置好的arbotix节点，在运行该launch文件时，会产生错误[dviver-5]，原因是arbotix运行的是python3而ROS运行的是python2，解决方法是进入arbotix\_python功能包，进入bin，进入arbotix\_driver将开头的python3改为python2.7：

<node name="arbotix" pkg="arbotix\_python" type="arbotix\_driver" output="screen">

<rosparam command="load" file="$(find test\_02)/config/control.yaml" />

<param name="sim" value="true" />

</node>

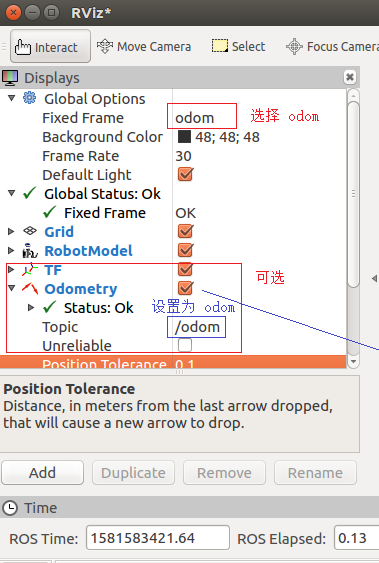
代码解释：

* <node>调用了arbotix\_python功能包下的arbotix\_driver节点；
* <rosparam>arbotix驱动机器人运行时，需要获取机器人信息，可以通过file加载配置文件；
* <param>在仿真环境下，需要配置sim为true。

**（5）启动launch文件并控制机器人模型运动**

启动launch：roslaunch test\_02 control.launch

配置rviz：



控制小车运动：

此时调用rostopic list会发现一个熟悉的话题：/cmd\_vel



也就说我们可以发布cmd\_vel话题消息控制小车运动了，该实现策略有多种，可以另行编写节点，或者更简单些可以直接通过如下命令发布消息,下列代码在输入完Twist之后，按两下Tab键，改完参数后再enter，改参数时，只能用左右键移动：

$ rostopic pub -r 10 /cmd\_vel geometry\_msgs/Twist "linear:

x: 1.0

y: 0.0

z: 0.0

angular:

x: 0.0

y: 0.0

z: 1.0"

现在，小车就可以运动起来了。

**另请参考：**<http://wiki.ros.org/arbotix>

1. **控制MV-AIBOT机器人运动**

**（1）MV-AIBOT机器人与上位机通信机制**

MV-AIBOT机器人与上位机通过有线网络连接，可以利用socket进行两者之间的通信，硬件设备有了，要想两个设备互通，就需要有固定的通信协议，两个设备按照固定的协议来互通数据，参考协议如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **功能编码2B** | **数据段0~6B** | **描述** | **功能描述** |
| 0x1100 | int Speed\_L; 4Bytes  int Speed\_R; 4Bytes | 运动控制指令、遥控器失能控制指令。 | Speed指示左右轮运动速度，取值为真实速度\*100，例如：100表示1m/s，250表示2.5m/s。 |
| 0xcdab | short int Speed;  2 Bytes  short int A\_Speed;  2 Bytes  short int Angle;  2 Bytes | 速度状态反馈、  角度反馈指令 | Speed指示运动速度，A\_Speed指示运动角速度，Angle为当前角度。 |
| 0x1111 | 无 | 关闭速度、角速度和角度反馈。 | （默认开启底盘速度、角速度、角度数据反馈，下发指令后关闭。） |
| 0xaabb | 无 | 打开底盘角度闭环控制、失能遥控器控制。 | （使用视觉模块时首先发送该命令，再进行角度和速度控制，否则角度控制无效。） |
| 0xaacc | int Speed; 4Bytes  int Angle; 4Bytes | 运动控制指令。 | （使用视觉模块时使用该命令控制底盘前进、后退和转向。） |

**（2）在上位机编写MV-AIBOT机器人与上位机通信程序**

根据上述协议，编写MV-AIBOT机器人与上位机通信程序，最终实现通过键盘来控制MV-AIBOT机器人的移动。参考代码如下，或者参考car\_teleop.py。

#!/usr/bin/env python

#coding=utf-8

import roslib; roslib.load\_manifest('mbot\_bringup')

import rospy

from geometry\_msgs.msg import Twist

import socket

def callback(msg):

cmd\_twist\_rotation = msg.angular.z

cmd\_twist\_x = msg.linear.x

wheelspeed = odom\_to\_speed(cmd\_twist\_x, cmd\_twist\_rotation)

if int(wheelspeed[0]) < 0:

L\_wheelspeed = int(wheelspeed[0]) & 0xff

L\_wheelspeed = str(hex(L\_wheelspeed))

Lfuhao = 0

else:

L\_wheelspeed = str(hex(int(wheelspeed[0])))

Lfuhao = 1

if int(wheelspeed[1]) < 0:

R\_wheelspeed = int(wheelspeed[1]) & 0xff

R\_wheelspeed = str(hex(R\_wheelspeed))

Rfuhao = 0

else:

R\_wheelspeed = str(hex(int(wheelspeed[1])))

Rfuhao = 1

num\_L = len(L\_wheelspeed)

num\_R = len(R\_wheelspeed)

if num\_L < 4:

send\_L = (L\_wheelspeed[-1:])

L\_wheelspeed = '0' + send\_L

else:

send\_L = (L\_wheelspeed[-2:])

L\_wheelspeed = send\_L

if num\_R < 4:

send\_R = (R\_wheelspeed[-1:])

R\_wheelspeed = '0' + send\_R

else:

send\_R = (R\_wheelspeed[-2:])

R\_wheelspeed = send\_R

L\_wheelspeed = L\_wheelspeed.decode('hex')

R\_wheelspeed = R\_wheelspeed.decode('hex')

if Lfuhao == 1 and Rfuhao == 1:

send\_wheelspeed = ('\x00\x11' + L\_wheelspeed + '\x00\x00\x00' + R\_wheelspeed + '\x00\x00\x00')

if Lfuhao == 1 and Rfuhao == 0:

send\_wheelspeed = ('\x00\x11' + L\_wheelspeed + '\x00\x00\x00' + R\_wheelspeed + '\xff\xff\xff')

if Lfuhao == 0 and Rfuhao == 1:

send\_wheelspeed = ('\x00\x11' + L\_wheelspeed + '\xff\xff\xff' + R\_wheelspeed + '\x00\x00\x00')

if Lfuhao == 0 and Rfuhao == 0:

send\_wheelspeed = ('\x00\x11' + L\_wheelspeed + '\xff\xff\xff' + R\_wheelspeed + '\xff\xff\xff')

if send\_wheelspeed == ('\x00\x11\x00\x00\x00\x00\x00\x00\x00\x00'):

mysocket.send(send\_wheelspeed)

mysocket.send(send\_wheelspeed)

else:

mysocket.send(send\_wheelspeed)

def odom\_to\_speed(cmd\_twist\_x = 0, cmd\_twist\_rotation = 0):

cent\_speed = cmd\_twist\_x

yaw\_speed = cmd\_twist\_rotation

Lwheelspeed = cent\_speed - 0.49\*yaw\_speed/2

Rwheelspeed = cent\_speed + 0.49\*yaw\_speed/2

CM\_Lwheelspeed = Lwheelspeed \* 100

CM\_Rwheelspeed = Rwheelspeed \* 100

return CM\_Lwheelspeed, CM\_Rwheelspeed

def listener():

rospy.init\_node('cmd\_vel\_listener',anonymous=True)

rospy.Subscriber("/cmd\_vel", Twist, callback)

rospy.spin()

if \_\_name\_\_ == '\_\_main\_\_':

mysocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

host = '192.168.0.10'

port = 8800

try:

mysocket.connect((host, port))

print('connect\_success')

except:

print('connect\_unsuccess')

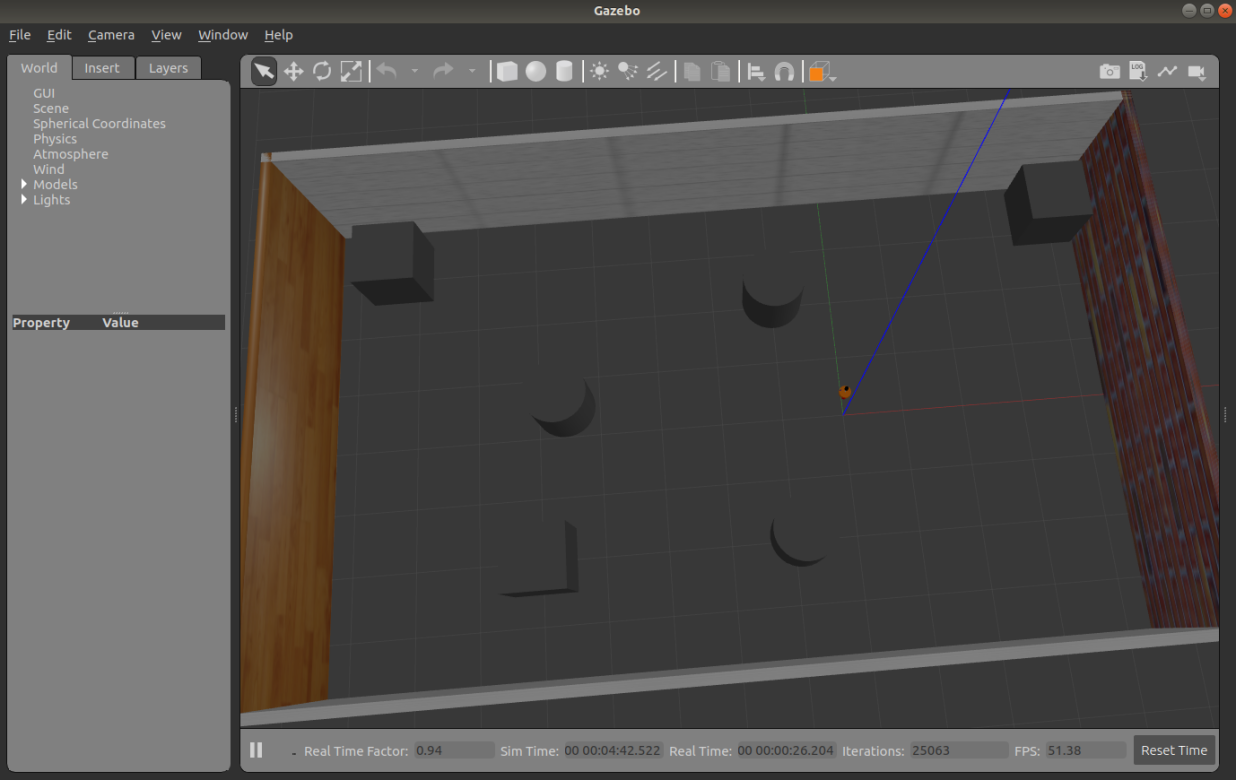
exit(1)

listener()

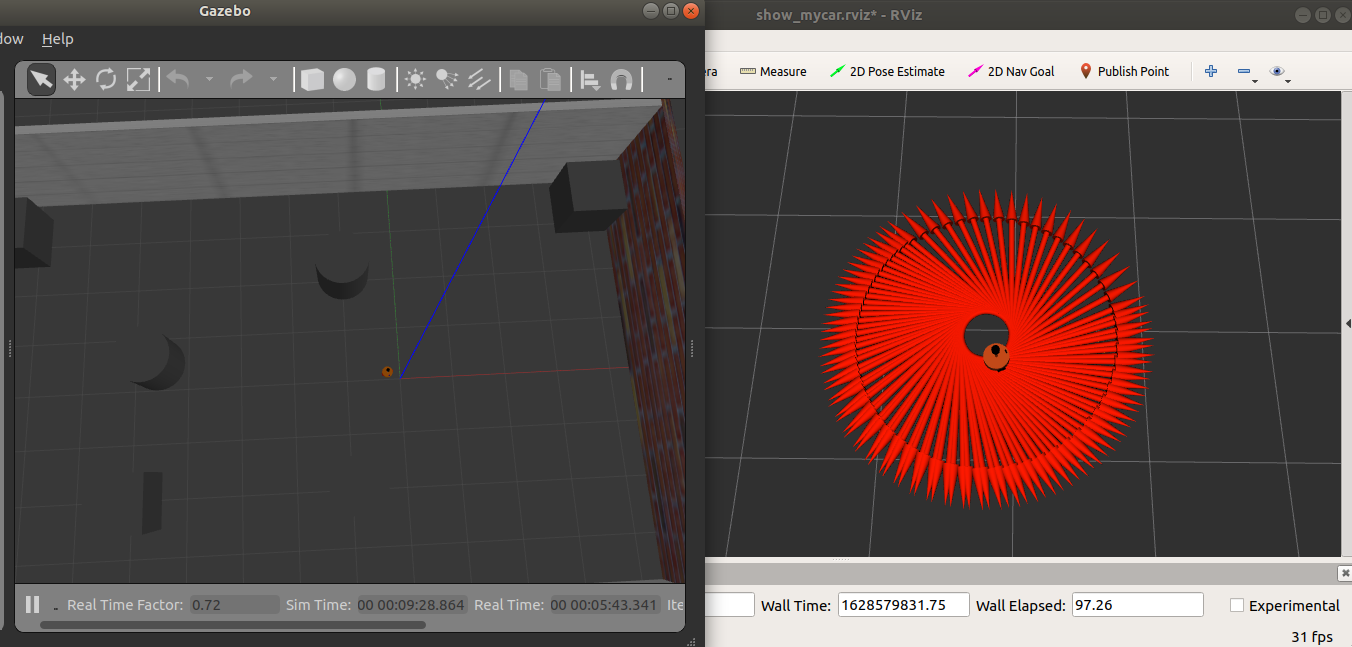
在运行上述文件前，先打开提前准备好的launch文件，test\_03功能包下，launch文件夹中的mycar.launch文件，再打开键盘控制节点mbot\_teleop.launch和小车通信节点car\_teleop.py,就可以用键盘控制节点控制小车运动了。

1. **实验参考结果**

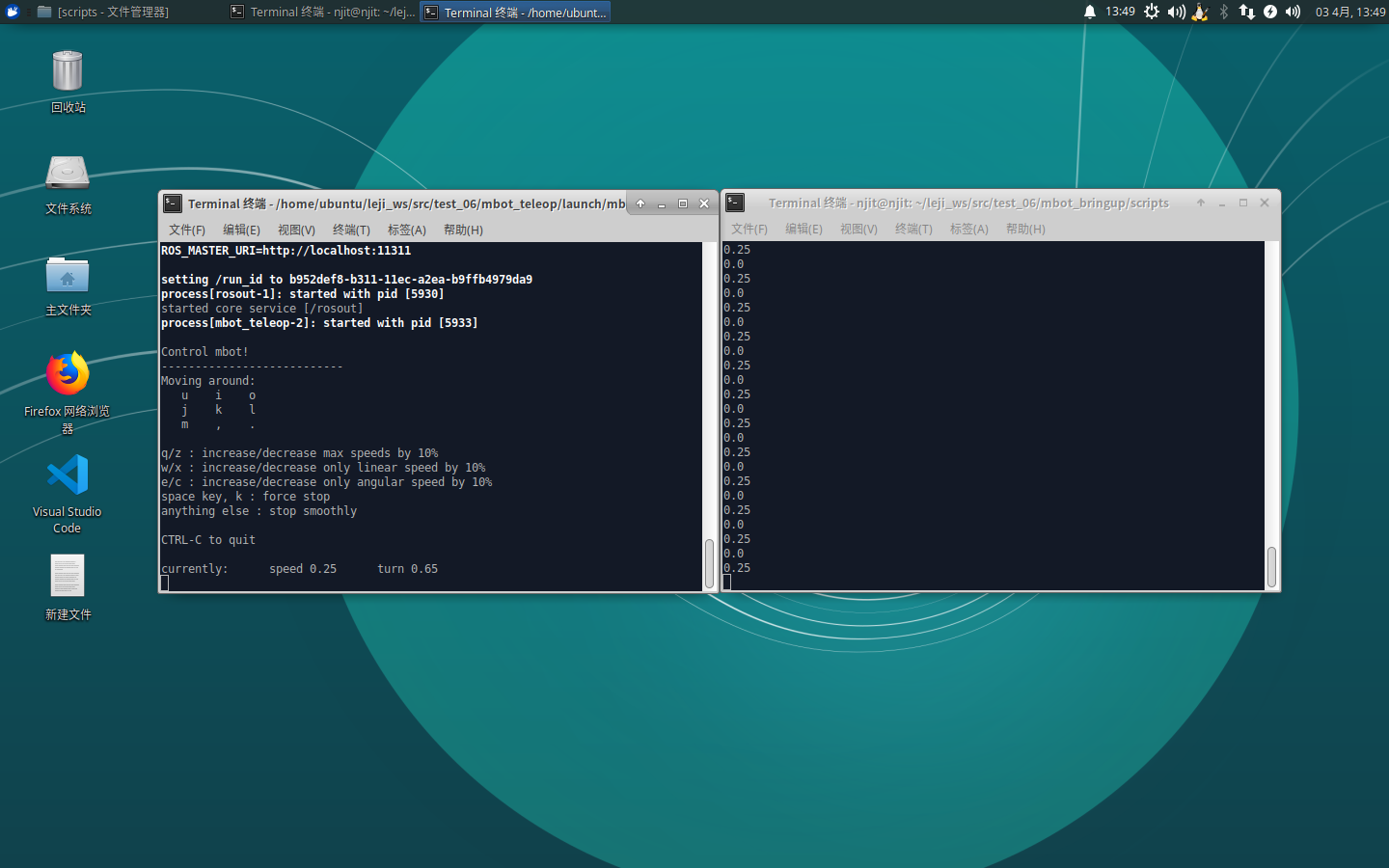
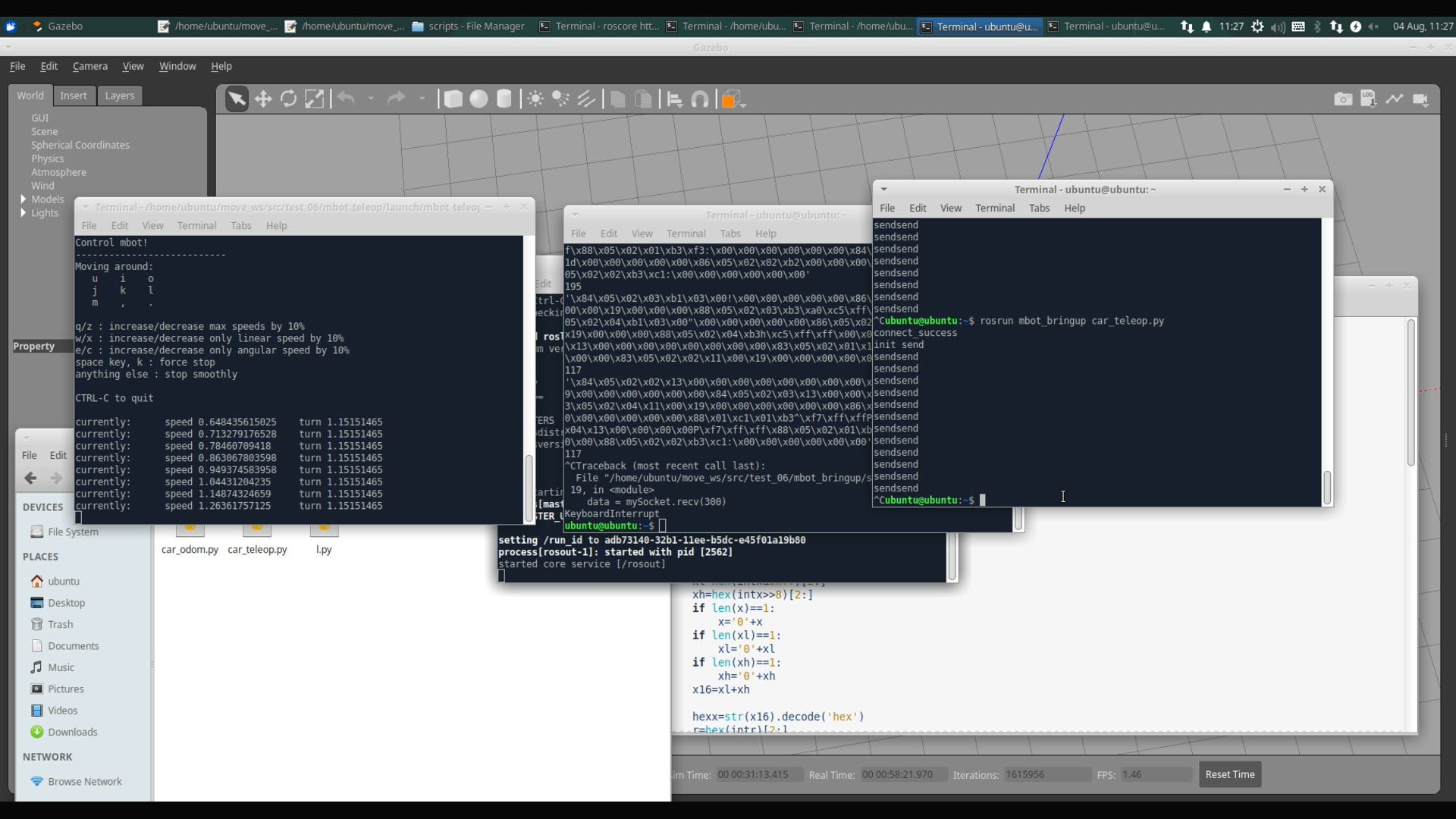
**1、小车在Gazebo中正常运行的结果如下图所示：**



**2、通过Rviz显示机器人的里程计信息以及运动朝向如下图所示：**



**3、MV-AIBOT机器人与上位机通信结果如下图所示：**

****

1. **相关的测试代码**

见代码包test\_02、test\_06。