该平台为2019年最新的比赛平台，下面为使用该平台的教程：

第一步：安装ros(此处安装的版本为ubuntu16.04对应的Kinectic版本)

1. sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb\_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
2. sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
3. sudo apt-get update
4. sudo apt-get install ros-kinetic-desktop-full
5. sudo rosdep init
6. rosdep update
7. echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
8. source ~/.bashrc
9. sudo apt-get install ros-kinetic-turtlebot-gazebo

第二步，平台的使用

1. <https://github.com/zerowind168/SIMUROSOT-ROBOCHALLENGE>到该网址下载平台代码及资料
2. 进入home目录中，使用 Ctrl+H 显示隐藏文件，进入gazebo/models目录中。然后到下载的文件中复制models1/colorbox, models/newbox 和 models/newfield到上述文件中
3. 运行该平台有两种模式，第一种是服务端和客户端都在同一台机器上，第二种是客户端和服务端分别在两台不同的机器上。（平时训练可以使用第一种比较方便，比赛的时候用第二种模式）
4. 下面分别说明如何运行代码

4.1 客户端和服务端都在同一机器上

4.1.1 export LD\_LIBRARY\_PATH=~/ros\_scripts/plugin:$LD\_LIBRARY\_PATH

4.1.2 roslaunch turtlebot\_gazebo turtlebot\_world.launch world\_file:=/home/ username/ros\_scripts/ world/world1-1 （world\_file:=your world\_file's path）

4.1.3 python robot\_ser.py

4.1.4 python how\_to\_run.py

4.2 分别在两台机器上分别运行服务端和客户端

（首先要在clock\_cli.py和robot\_cli.py文件中修改ip地址，将该地址改为服务端的ip地址）

在服务端运行以下命令

4.2.1 export LD\_LIBRARY\_PATH=~/ros\_scripts/plugin:$LD\_LIBRARY\_PATH

4.3.2 roslaunch turtlebot\_gazebo turtlebot\_world.launch world\_file:=/home/ username/ros\_scripts/ world/world1-1 （world\_file:=your world\_file's path）

4.3.3 python clock\_ser.py

4.3.4 python robot\_ser.py

在客户端运行以下命令

4.3.5 roscore

4.3.6 python clock\_cli.py

4.3.7 python how\_to\_run.py

如果你能看到以下图片就说明你的运行时成功的

对工程中的文件的解释.

├── how\_to\_run.py<br>

├── robot.py<br>

├── robot\_ser.py<br>

├── robot\_cli.py<br>

├── clock\_ser.py<br>

├── clock\_cli.py<br>

├── judge.py<br>

└── world.world<br>

1. how\_to\_run.py

该文件主要来写你想要如何控制小车的算法，你可以在这里写下你的想法并且测试你的代码是否是理想的。

1. robot.py

该文件主要是描述一个机器人类，在此处指的是小车。他有转向和前进等各种功能，详细信息可见代码中的注释。

1. robot\_ser.py

机器人服务端，主要接受客户端传来的指令来与gazebo进行交互。

1. robot\_cli.py

机器人客户端，主要与服务端进行交互，从服务端获取状态等信息，也可以对服务端发送信息用以控制小车。

1. clock\_ser.py

服务端时间类，主要是记录比赛时间，与客户端时间统一。

1. clock\_cli.py

客户端时间类，主要是记录比赛时间，与服务端时间统一。

1. judge.py

裁判系统，主要记录比赛时间，机器人状态是否越界，是否成功完成比赛等

1. world.world

生成场景信息的文件

该比赛有两种不同的障碍物场景，分别如下图所示：默认是右图的场景，要改变场景的时候关掉gazebo，然后到.gazebo/models/materials/textures下将两个场景命名对换即可。

