一：构建网络。网络中有三个参数需要指定 ROS\_MASTER\_URI ROS\_IP ROS\_HOSTNAME

其中 ROS\_MASTER\_URI 最为重要，指向整个网络中唯一一个节点。因为M3、M2有双网卡，导致 电脑端，指定了ROS\_MASTER\_URI 能看见话题，就是收不到话题数据，两份了两天。

二：bringup.launch文件，第一部分

<node name="arduino\_serial\_node" pkg="rosserial\_python" type="serial\_node.py" output="screen">

<param name="port" value="/dev/rikibase" />

<param name="baud" value="115200" />

</node>

启动了一个串口的模块，这样下位机就能够和上位机 通过串口建立连接了。下位机可以直接订阅或发布话题，上位机就能直接看到toipc。

树莓派：roslaunch rikirobot bringup.launch

查看imu信息:roscd rikirobot

Rostopic echo /imu/data

校准： cd param/imu

rosrun imu\_calib do\_calib

键盘控制：

rosrun teleop\_twist\_keyboard teleop\_twist\_keyboard.py

在树莓派，/catkin\_ws/src/rikirobot\_project/rikirobot/param/imu下，执行 rosrun imu\_calib do\_calib程序。

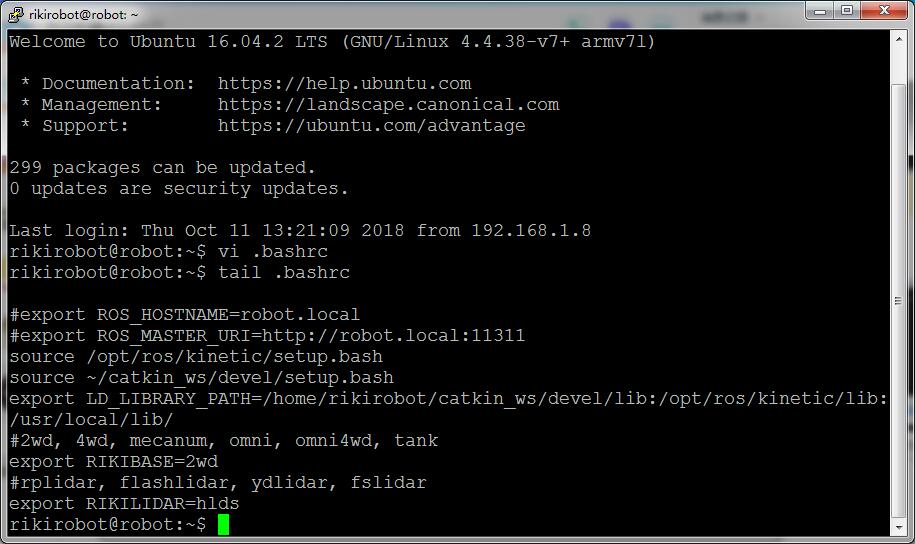
线速度测试 树莓派端：rosrun rikirobot\_nav calibrate\_linear.py

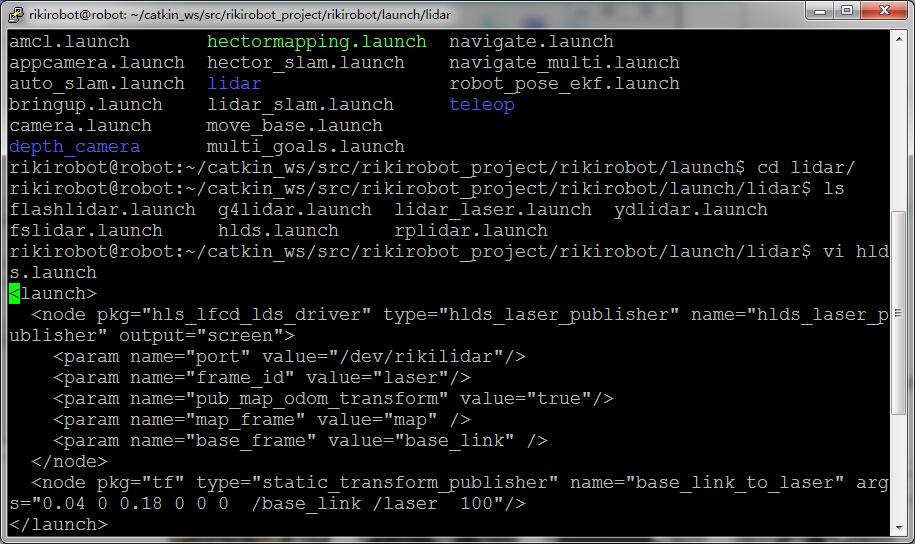
虚拟机端:rosrun rqt\_reconfigure rqt\_recondigure

角速度测试 树莓派端：rosrun rikirobot\_nav calibrate\_angular.py

虚拟机端:rosrun rqt\_reconfigure rqt\_recondigure

雷达：

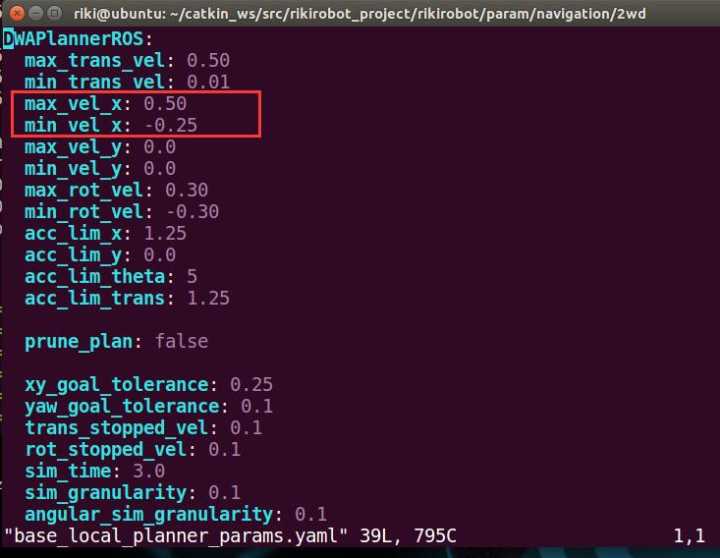




roslaunch rikirobot lidar\_slam.launch

roslaunch rikirobot camera.launch 只打开摄像头

导航速度修改：base\_local\_planner\_params.yaml 根据车型找路径改



关于缓冲区

在centos里面，swappiness的值的大小对如何使用swap分区是有着很大的联系的。swappiness=0的时候表示最大限度使用物理内存，然后才是 swap空间，swappiness＝100的时候表示积极的使用swap分区，并且把内存上的数据及时的搬运到swap空间里面。两个极端，对于 ubuntu的默认设置，这个值等于60，建议修改为10。具体这样做：  
1.查看你的系统里面的swappiness  
$ cat /proc/sys/vm/swappiness  
不出意外的话，你应该看到是 60  
2.修改swappiness值为10  
$ sudo sysctl vm.swappiness=10  
但是这只是临时性的修改，在你重启系统后会恢复默认的60，所以，还要做一步：  
$ gksudo gedit /etc/sysctl.conf  
在这个文档的最后加上这样一行:  
vm.swappiness=10  
然后保存，重启。ok，你的设置就生效了。

1.环境，ubuntu版本ubuntu-16.04.2-desktop-amd64，

2.安装完成后，设置swap；

3.sudo swapon -s，查看系统是否存在swap；

4.free -m，也可以查看内存情况；

5.df -h，查看磁盘使用情况；

6.sudo dd if=/dev/zero of=/swapfile bs=1G count=4。解释：创建swap文件，块大小bs=1G，块的个数为4个，如果内存为512M，那么可以设置为bs=256M，count=2，或者执行sudo dd if=/dev/zero of=/swapfile bs=1M count=6k (count=1k创建1G的swap,如果要创建6G则count=6k，这种指令执行的比较慢)，一般建议swap和内设置一样大，防止出现swap不够用，或者执行sudo fallocate -l 8G /swapfile，这条语句执行的比较快，本人电脑8G内存；

7.ls -lh /swapfile，查看swap信息；

8.sudo chmod 600 /swapfile，设置swap权限，root才有读写权限，如果不设置为root级别可能会出现漏洞；

9.ls -lh /swapfile，再次查看，swap信息发生变化；

10.sudo mkswap /swapfile，通知系统挂载swap；

11.sudo swapon /swapfile，通知系统启用swap；

12.sudo swapon -s，查看信息；

以上完成swap设置，如果想移除swap执行sudo swapoff /swapfile && sudo rm /swapfile，但是如果系统重启，再开机就不能自动使用swap了，接下来继续：

13.sudo gedit /etc/fstab，编辑脚本；

14.打开文件后，最后面添加一行/swapfile  none  swap  sw  0  0，然后保存；

15.重启电脑就好啦，此时的swap的使用就不会受重启的影响了。

调优部分，主要针对两个参数，但是这两个参数只是对系统的建议，具体系统怎么执行，还要看系统心情。一个参数是swappiness，另一个是vfs\_cache\_pressure

（1）swappiness：

的值是个参考值，是否会发生swap跟当前是哪种page reclaim及系统当前状态都有关系，所以设置了swappiness=0并不代表一定没有swap发生，同时设为0也确实会可能发生OOM。个人仍然认为线上环境设置swappiness=0是没有任何问题的。本人设置为30。

（2）vm.vfs\_cache\_pressure：

默认值：100。参数意义：控制内核回收再利用用于缓存目录与inode对象的内存的趁势。默认值设为100表示内核以平等的速度去考虑pagecache和swapcache的回收再利用。减小它，会触发内核保持目录与inodes的缓存内存。增大它，会触发内核回收再利用目录与inodes的缓存内存。本人设置为120。

这两参数设置的持久化话问题，大家直接百度就好了，不多说。个人感觉这两个参数没什么用。

这里留意，想编译源代码，内存消耗很大，会因为内存原因导致编译失败。解决办法，创建一个较大的swap区，然后把sysctl.conf 中的vm.swappiness值设大，然后再去catkin\_make。

2019/05/24:

使用了新做的电源板，但是还是出现了小车动起来，PI死机情况。还是怀疑电源问题。底盘运动起来也是有问题，非常抖。

计划：屏蔽ROS，在上位机上 使用串口与下位机通讯，记录小车动起来的所有状态信息。限制线速度 限制在0.25米以内。先搞好小车的运动，排查电池原因。

2019/8/12: 最近发现了个很麻烦的问题。就是电机的感抗问题。第一次做平衡车时，平衡小车之家购买的电机，应该是感抗很小的。用了(period=999 prescaler=3,计算频率为18000hz)效果很好。后来用了空心杯电机，效果也不错。但是购买了，大功率电机，发现了这个问题。频率高了，在pwm值低时，无法驱动电机。导致电机的动力与pwm值不成线性关系。再返回履带车，发现也存在这个问题，之前一直没能留意到。

现在是将电机驱动频率降低至(period=1999 prescaler=359,计算频率为100hz)才算是OK。但是噪音较大(100hz在耳朵可见的范围内了).

今天计划

1 校正小车。

2 鼠标构图：

把小车平放在你想要构建地图区域的地板上，打开资料里的虚拟机，打开一个终端，  
ssh 过去主控端启动 roslaunch rikirobot bringup.launch。

启动雷达：roslaunch rikirobot lidar\_slam.launch。

在虚拟机端打开一个终端，启动 rviz: rosrun rviz rviz

找到路径下~/catkin\_ws/src/rikirobot\_project/rikirobot/rviz/slam.rviz，的  
slam.rviz 文件。点 Open

用 2D Nav Goal 工具，在地图上设定目标点，小车到达目标点，完成该行走区域的地图扫描工作。这个就类似于键盘控制小车建地图一样。

当该区域内的地图扫描完毕后，在虚拟机端，再打开一个终端，ssh 过去主控端，进入到~/catkin\_ws/src/rikirobot\_project/rikirobot/maps 路径下，执行保存地图脚本，执行保存地图脚本前后可以通过 ll ,查看地图的大小时间等来初步判断地图有无保存成功。