teleop package 建立

建立工作环境及资源文件夹:

```
mkdir -p ~/plantbot/src
```

创建功能包

```
cd ~/plantbot/src
catkin_create_pkg teleop roscpp
```

创建节点

```
cd teleop
touch teleop.cpp
gedit teleop.cpp
```

编写 teleop.cpp 程序，然后保存退出

在 CmakeLists.txt 添加相关代码

编译

```
cd ~/plantbot
catkin_make
```

打开节点管理器:

```
roscore
```

打开新标签页，将工作空间添加到 ROS 工作环境中:

```
source ~/plantbot/devel/setup.bash
```

运行节点:

```
rosrun teleop keyboard_control
```

这时可以用 w s a d 字母键分别控制上下左右了（在此界面输入命令后还需要按 enter 键才能发布）

通过 rqt_graph 来查看当前系统的运行情况。在新的标签页运行如下命令:

```
rosrun rqt_graph rqt_graph
```

打开新标签页,显示发布的数据:

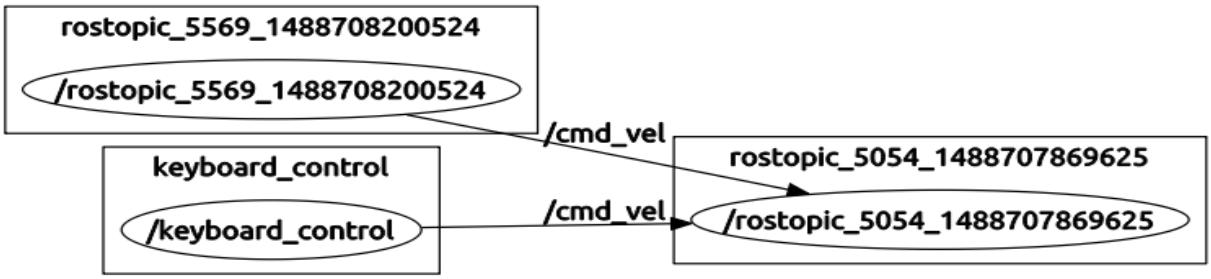
```
rostopic echo cmd_vel
```

打开新标签页,通过终端发布消息:

```
rostopic pub cmd_vel geometry_msgs/Twist -r 1 -- '[0.01, 0.0, 0.0]' '[0.0, 0.0, 0.2]'
```

此时再观看系统运行情况:

```
rosrun rqt_graph rqt_graph
```



由图可以看出，节点管理器会在连续的终端发布消息及终端显示节点的命令下，自动生成终端发布节点与终端显示节点。

激光雷达测试

复制 rplidar_ros 包到 src 文件夹

```
cd ~/plantbot/src
git clone https://github.com/ncnynl/rplidar_ros.git
```

编译

```
cd ~/plantbot
catkin_make
```

添加工作空间到 ROS 环境

```
source ~/plantbot/devel/setup.bash
```

插入激光雷达，检查对应的串口:

```
ls -l /dev |grep ttyUSB
```

结果为 ttyUSB0，增加写的权限，设置串口权限:

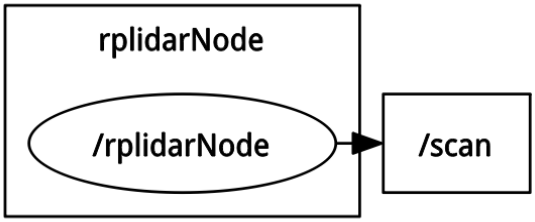
```
sudo chmod 666 /dev/ttyUSB0
```

运行 rplidar,打开 rviz 测试:

```
roslaunch rplidar_ros view_rplidar.launch
```

打开 rqt_graph:

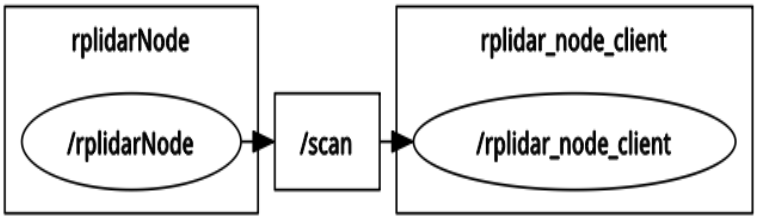
```
rosrun rqt_graph rqt_graph
```



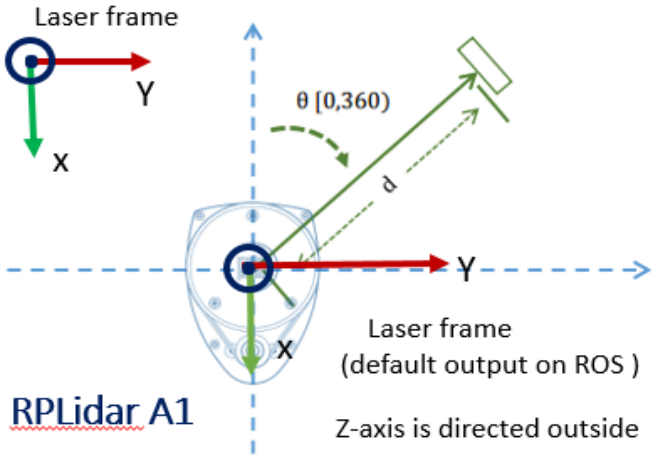
打开新标签页，测试一个节点：
`rosrun rplidar_ros rplidarNodeClient`
输出结果如下：

```
[ INFO] [1488705656.565746911]: : [30.000021, 1.299000]
[ INFO] [1488705656.565781357]: : [31.000015, 1.285000]
[ INFO] [1488705656.565814917]: : [32.000011, inf]
[ INFO] [1488705656.565847027]: : [33.000019, 1.277000]
[ INFO] [1488705656.565879168]: : [34.000011, 1.260000]
[ INFO] [1488705656.565906835]: : [35.000019, 1.252000]
[ INFO] [1488705656.565936018]: : [36.000015, 1.244000]
```

刷新 `rqt_graph`：



参考：https://github.com/robopeak/rplidar_ros/wiki
激光雷达在 plantbot 上的安装：



安装 Energia

下载最新版本的 energia：**energia-1.6.10E18-linux64.tar.xz** 到 Software 文件夹，解压：

```
cd ~/Software
tar -xJf Download/energia-1.6.10E18-linux64.tar.xz
```

运行：

```
cd ~/Software/energia-1.6.10E18
./energia &
```

energia 语法参考：http://energia.nu/reference/
&标识符表示命令到后台运行，这样继续输入命令时就不用打开新的终端或标签页，在打开节点管理器时也可直接输入 `roscore &` ，配置结束后可以直接在同一界面输入命令
在打开程序后一直出现烧不进去程序的问题，错误提示：
`error: CORTEX_M4_0: Error connecting to the target: Frequency is out of range.`

但同样的条件下在 Windows 下却可以正常烧录，应该是操作系统的问题，但一直找不到问题所在，最后重新查阅官网安装指导 Setup Energia on Linux，发现确实少了步骤

For all Boards

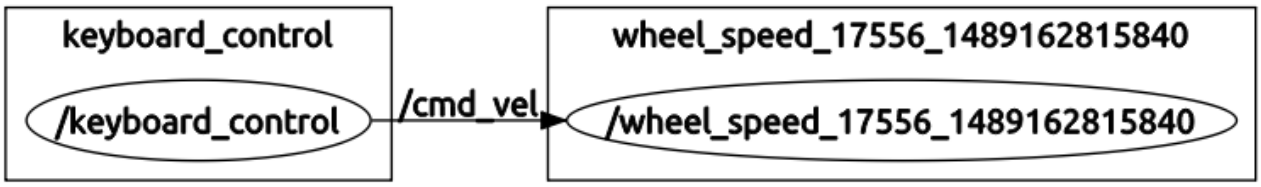
All Boards need a udev rule to be able to program as a regular user (not root). To install these udev rules, follow the steps below:

1. Download the udev rules: TI udev rules
2. Open a terminal and execute the following command: sudo mv /71-ti-permissions.rules /etc/udev /rules.d/

安装 udev rule:
gedit 71-ti-permissions.rules
拷贝 udev rules 到 71-ti-permissions.rules
移动到相应目录下:
sudo mv /71-ti-permissions.rules /etc/udev/rules.d/
此时再烧录程序，成功。

wheel_speed 节点编程

cd ~/plantbot/src/teleop
mkdir scripts
cd scripts
gedit wheel_speed.py
编程结束后保存
增加权限，使其成为可执行的脚本文件:
chmod 755 wheel_speed.py
#也可写成 chmod +x wheel_speed.py
cd ~/plantbot
catkin_make
(在编译结果中没有发现 wheel_speed 节点，因为之前在利用 catkin_create_pkg 命令创建 teleop 功能包时没有添加 rospy 依赖，配置 teleop 文件夹中的 package.xml，在依赖项标签中添加 <build_depend>rospy</build_depend> <run_depend>rospy</run_depend> 两句后继续编译，仍然没有生成，可能 python 语言写成的节点不会在编译结果中出现？)
roscore &
rosrun teleop keyboard_control
打开新标签页:
source ~/plantbot/devel/setup.bash
rosrun teleop wheel_speed.py
可以看到源源输出的左右轮速度信息，在运行 keyboard_control 节点的标签页输入 w /a /s /d 命令后返回可以看到输出的速度信息发生相应变化。
观察运行关系:
rosrun rqt_graph rqt_graph
如图所示:



pid_control 节点编程

cd ~/plantbot/src/teleop/scripts
gedit pid_control.py
编写 python 程序，保存后修改权限:
chmod +x pid_control.py
将路径添加到初始化配置文件
echo "source ~/plantbot/devel/setup.bash" >> ~/.bashrc
roscore

source 初始化配置文件
```
source ~/.bashrc
```
运行：
```
rosrun teleop pid_control.py
```

创建 launch 文件

```
roscd teleop
mkdir launch
cd launch
gedit teleop.launch
```
编辑 launch 文件
运行 launch 文件：
```
roslaunch teleop teleop.launch
```
总是提示
```
ERROR: cannot launch node of type [teleop/teleop_node]: can't locate node [teleop_node] in package [teleop]
```
在打开 teleop_node 时出错。在 launch 文件中对应语句为：
```
<node pkg="teleop" name="keyboard_control" type="teleop_node" launch-prefix="xterm -e"/>
```
单独运行这个节点：
rosrun teleop teleop_node 时也提示在 teleop package 下找不到节点 teleop_node。查询 CmakeLists.txt：
```
## Declare a C++ executable
add_executable(keyboard_control teleop_node.cpp)
```

究其原因，是 teleop_node 是文件名，但并非生成的可执行文件名，对应的可执行文件名是 keyboard_control,在 launch 文件中改正即可。

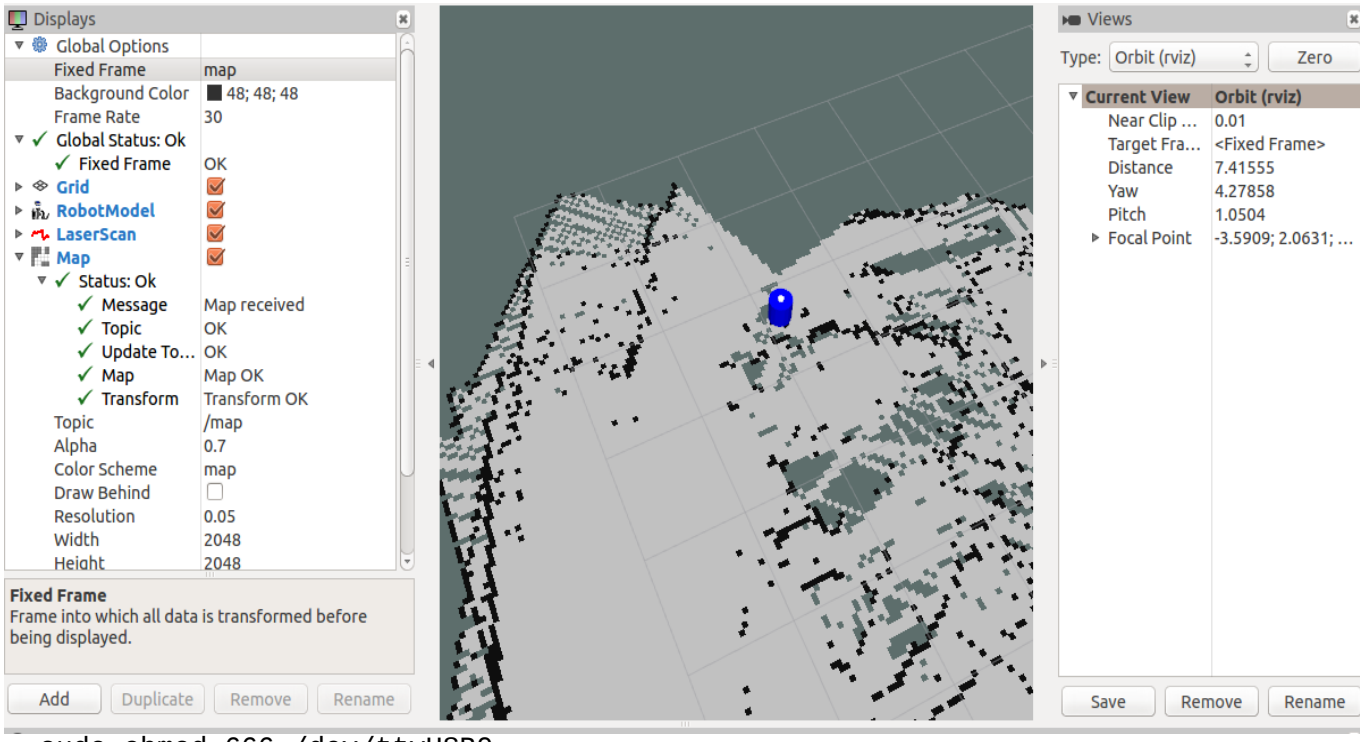
display 节点编程

```
cd ~/plantbot/src/teleop/scripts
gedit display.py
```
编写 python 程序，保存后修改权限：
```
chmod +x display.py
```
编译：
```
cd ~/plantbot
catkin_make
```
source 初始化配置文件
```
source ~/.bashrc
```

激光雷达运行 hector_mapping 绘图

安装 hector_slam：
sudo apt-get install ros-indigo-hector-slam
下载编译 rplidar 的驱动：
cd ~/plantbot/src
git clone https://github.com/robopeak/rplidar_ros.git
在 rplidar_ros/launch 下新建 hector_rplidar.launch
编译：
```
cd ..
catkin_make
```
插入激光雷达，检查对应的串口：
```
ls -l /dev |grep ttyUSB
```

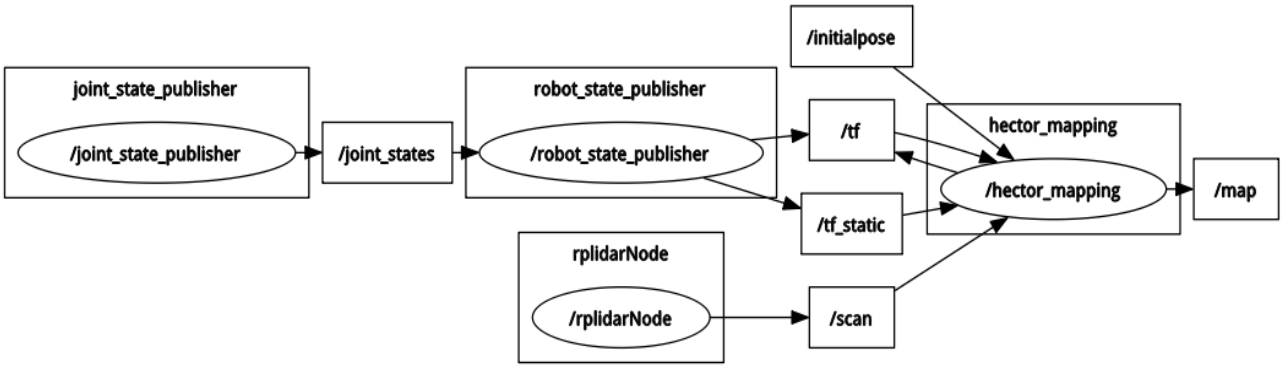
结果为 ttyUSB0，增加写的权限，设置串口权限：




```
sudo chmod 666 /dev/ttyUSB0
```

运行测试：
roslaunch rplidar_ros hector_rplidar.launch model:='$(find teleop)/urdf/plantbot.urdf'
在 rviz 中选择 Fixed Frame 为 map,选择 Map 的 Topic 为/map
打开 rqt_graph：

```
rosrun rqt_graph rqt_graph
```



绘图完成后，可以使用 map_server 保存地图：
roscd rplidar_ros
cd maps

```
rosrun map_server map_saver -f <map_name>
```

参考：http://rosclub.cn/post-602.html

将 plantbot 上传到 GitHub 上

注册 GitHub 帐号 marooncn，创建新的 respository 取名 plantbot，复制此仓库的 git 地址：

git@github.com:marooncn/plantbot.git
本地 Git 仓库和 GitHub 库之间的传输是通过 SSH 加密的，首先在目标目录下创建 SSH key:
cd ~/plantbot
ssh-keygen -t rsa -C "marooncn@163.com"
然后一路回车，使用默认值即可
在用户主目录下显示所有文件（包括隐藏文件）：
cd ..
ls -ah
如果一切顺利的话，可以在用户主目录里找到.ssh 目录，里面有 id_rsa 和 id_rsa.pub 两个文件，这两个就是 SSH Key 的秘钥对，id_rsa 是私钥，不能泄露出去，id_rsa.pub 是公钥，可以放心地告诉任何人。
查看公匙:
cat ~/.ssh/id_rsa.pub
复制内容，打开 GitHub->Personal Setting->SSH and GPG keys->New SSH key,填写 Title 名为 plantbot,在 Key 文本框里粘贴 id_rsa.pub 文件的内容 -> Add Key
这时候就可以把新建的 plantbot 库与本地的仓库关联，继而把本地仓库的内容推送到 GitHub 仓库
创建关联:
git remote add origin https://github.com/marooncn/plantbot.git
添加后，远程库的名字就是 origin，这是 Git 默认的叫法，也可以改成别的，但是 origin 这个名字一看就知道是远程库。
(出现错误
fatal: Not a git repository (or any parent up to mount parent /home/kineam)
Stopping at filesystem boundary (GIT_DISCOVERY_ACROSS_FILESYSTEM not set).
解决方法：git init
原因：找不到 git 目录了～ 初始化一下就可以了～)
下一步，就可以把本地库的所有内容推送到远程库上:
git push -u origin master
把本地库的内容推送到远程，用 git push 命令，实际上是把当前分支 master 推送到远程。
由于远程库是空的，第一次推送 master 分支时，加上了-u 参数，Git 不但会把本地的 master 分支内容推送的远程新的 master 分支，还会把本地的 master 分支和远程的 master 分支关联起来，在以后的推送或者拉取时就可以简化命令。
若上句遇到报错，则强制执行：git push -f origin master
推送成功后，可以立刻在 GitHub 页面中看到远程库的内容已经和本地一模一样。
自此，只要本地作了提交，就可以通过命令:
git push origin master
把本地 master 分支的最新修改推送至 GitHub。
参考：廖雪峰 Git 教程

上位机与下位机远程通讯

在上下位机同时安装 chrony 包来保证时间同步：
sudo apt-get install chrony
安装后 chrony 守护进程会自动将机器与所连接的网络服务器同步
使用 **Zeroconf**（Zero-configuration networking）实现 ROS 联网
Zeroconf 是允许同一子网的机器使用本地的用户名而非 IP 地址连接的一种工具，查看上、下位机的本地用户名：
hostname
本地主机名后加上“.local"就是 Zeroconf 联网名称，如上位机是 freeman.local，下位机是 pc-desktop.local
检查连接：
使用 ping 命令检查两台机器是否已经连接，在上位机运行：
ping pc-desktop.local
在下位机运行：
ping freeman.local

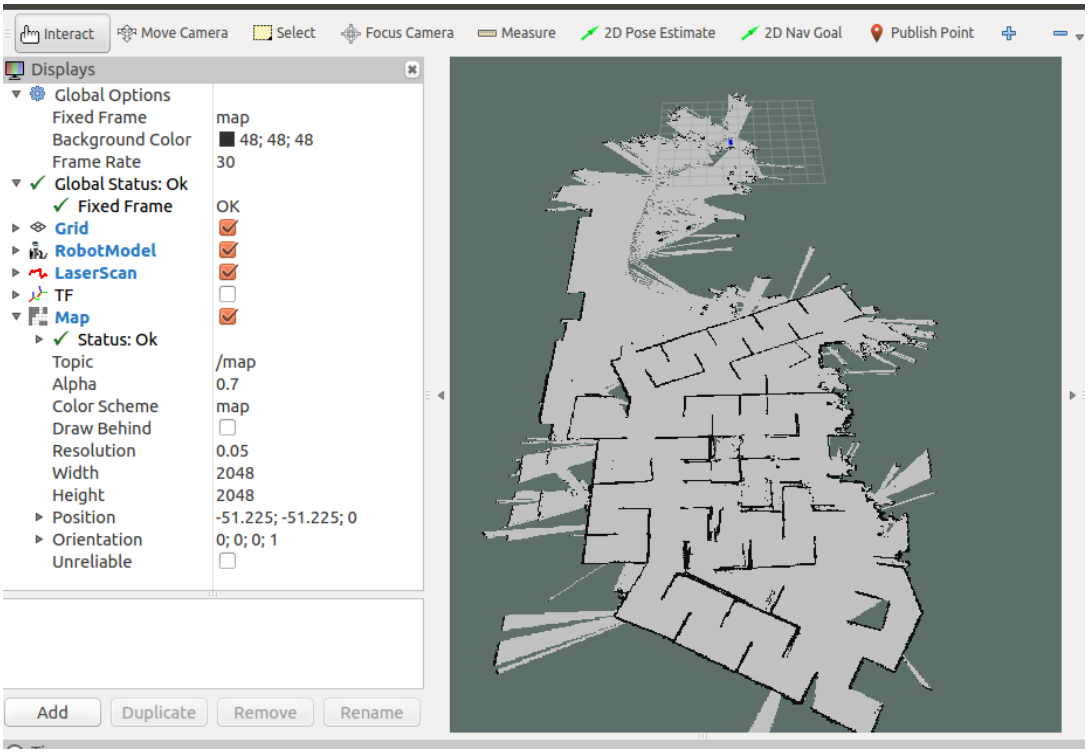
ping 测试出现“unknown host”错误时，可以尝试重启 avahi-daemon 进程：
sudo service avahi-daemon restart
在 ROS 网络中，一台机器被指定为 ROS master，运行 roscore 进程，另一台机器必须设置 ROS_MASTER_URI 环境变量来指向 master host。两台机器需要设置 ROS_HOSTNAME 为其 Zeroconf 用户名。
将下位机指定为 ROS master
在上位机运行：
export ROS_HOSTNAME=freeman.local
export ROS_MASTER_URI=http://pc-desktop.local:11311
在下位机运行：
export ROS_HOSTNAME=pc-desktop.local
roscore
保证下位机与上位机同步，在下位机上运行：
sudo ntpdate -b freeman.local
rostopic list
如果一切正常，就可以看到/rosout /rosout_agg
同时注意在打开的新标签页时 ROS_HOSTNAME、 ROS_MASTER_URI 得重新申明，当然更方便的做法是添加到~/.bashrc 文件中：
echo 'export ROS_HOSTNAME=pc-desktop.local' >> ~/.bashrc
在上位机终端：
echo 'export ROS_HOSTNAME=freeman.local' >> ~/.bashrc
echo 'export ROS_MASTER_URI=http://pc-desktop.local:11311' >> ~/.bashrc
由于下位机在实际工作的时候，没有鼠标与显示终端，因此用 ssh 进行通信，由上位机操作下位机
在上位机运行：
ssh pc-desktop.local
[问题：出错，提示 ssh: connect to host localhost port 22: Connection refused
ssh 的默认端口是 22，也就是说，登录请求会送进远程主机的 22 端口。出现这个问题是因为 Ubuntu 默认没有安装 openssh-server，可以通过一个命令来查看：
ps -e | grep ssh
没有任何输出，即没有安装 openssh-server，可通过下面命令安装：
sudo apt-get install openssh-server
安装后重新输入，让输入用户密码，但是当输入正确密码后显示：Permission denied, please try again.
问题出在 ssh 默认情况下 root 用户不能执行远程操作，修改 ssh 的配置文件/etc/ssh/sshd_config，由于配置文件是只读文件，因此先修改文件权限，a 表示全部，w 表示写权限：
sudo chmod a+w /etc/ssh/sshd_config
然后进入配置文件：
vim /etc/ssh/sshd_config
找到 PermitRootLogin without-password
改为：
PermitRootLogin yes 即 root 权限可远程登录
保存退出。
一切完成后重新输入，发现仍然提示：Permission denied, please try again，根据 http://blog.csdn.net/weiwei_pig/article/details/50954334 这篇文章，运行：
su -
提示认证失败，根据 http://blog.csdn.net/henren555/article/details/7546508 的说法，root 用户被默认锁定了，因此按照这篇文章做法，重新设置密码后登录 root 用户，命令行提示符由$变为#后，重新运行就可以了。]
roscore &

继续通过 ssh 打开下位机的 rviz:
rosrun rviz rviz
[问题：
rviz: can not connect X server
原因在于 ssh 不支持图形界面 gui，显性申明使用 X 即可：
ssh -X pc-desktop.local -l pc
-l 后也申明了对应用户名，以便加载正确的环境空间
参考：ssh rviz: cannot connect to X server]
由此下位机的 rviz 顺利由上位机打开并显示。注意在打开新标签页需要继续控制下位机需要重新输入：
ssh -X pc-desktop.local -l pc

参考：*ros_by_example_indigo_1 P.15*
ROS 官网相关知识：NetworkSetup

geotiff_mapper.launch 文件编程

1. 根据已有 bag 文件绘图
创建、编辑 launch 文件:
roscd rplidar_ros
gedit launch/geotiff_mapper.launch
复制、粘贴文件内容后保存。
下载已有的 bag 文件:
```
mkdir bags
cd bags
```
下载 `Team_Hector_MappingBox_RoboCup_2011_Rescue_Arena.bag` 文件，保存到新建的 bag 文件夹下
打开 launch 文件：
```
roslaunch rplidar_ros geotiff_mapper.launch
```
在新终端打开数据包：
```
rosbag play Team_Hector_MappingBox_RoboCup_2011_Rescue_Arena.bag  --clock
```
可以看到小车在 `rviz` 中的绘图过程



在回放绘图过程中打开新标签页运行：

```
cd ..
mkdir maps
cd maps
rostopic pub syscommand std_msgs/String "savegeotiff"
```

可以保存 geotiff 文件，结束或按 Ctrl-C 中断后：

```
ls
```

会看到绘制出来的包含地图信息的.tif 文件和包含环境信息的.tfw 文件。
也可以绘图时提到的 map_server 保存地图.pgm 图片格式：

```
rosrun map_server map_saver -f <map_name>
```

2.创建自己的 bag 文件
编辑 linux 启动的 rules，设置激光雷达连接后的端口状态：
sudo vim /etc/udev/rules.d/50-usb-seial.rules
加入：
SUBSYSTEMS=="usb",KERNEL=="ttyUSB0",MODE="0666"
此后插上激光雷达后不再需要手动设置端口权限了。
运行激光雷达绘图：
roslaunch rplidar_ros hector_rplidar.launch model:='$(find teleop)/urdf/plantbot.urdf'
打开新的标签页，运行：
roscd rplidar_ros
cd bags

```
rosbag record -a
```

运行一段时间后在两个运行窗口中按 Ctrl-C 退出命令，查看创建的.bag 文件：

```
ls
```

然后根据上面的步骤根据自己创建的.bag 文件进行回放数据绘图。
参考：How to build a Map Using Logged Data
Recording and playing back data

mpu6050_serial_to_imu

复制 mpu6050_serial_to_imu 包到 src 文件夹：

```
cd ~/plantbot/src
git clone git@github.com:fsteinhardt/mpu6050_serial_to_imu.git
```

程序中使用了 ros 的 serial 库，因此在编译之前要安装库：

```
sudo apt-get install ros-indigo-serial
```

编译

```
cd ~/plantbot
catkin_make
```

[ros 结点中使用了下述库

```
#include <geometry_msgs/Quaternion.h>
#include <ros/ros.h>
#include <serial/serial.h>
#include <sensor_msgs/Imu.h>
#include <sensor_msgs/Temperature.h>
#include <std_msgs/String.h>
#include <std_srvs/Empty.h>
#include <string>
#include <tf/transform_broadcaster.h>
#include <tf/transform_datatypes.h>
```

对应要保证 CmakeLists.txt 中添加了对应依赖项，这样在编译时才会找到库文件，在使用 catkin_create_pkg 命令创建包时在后边加上依赖项会自动添加，否则需要手动添加。

```
find_package(catkin REQUIRED COMPONENTS
  roscpp
  geometry_msgs
  sensor_msgs
  serial
  std_msgs
  std_srvs
  tf
)
```

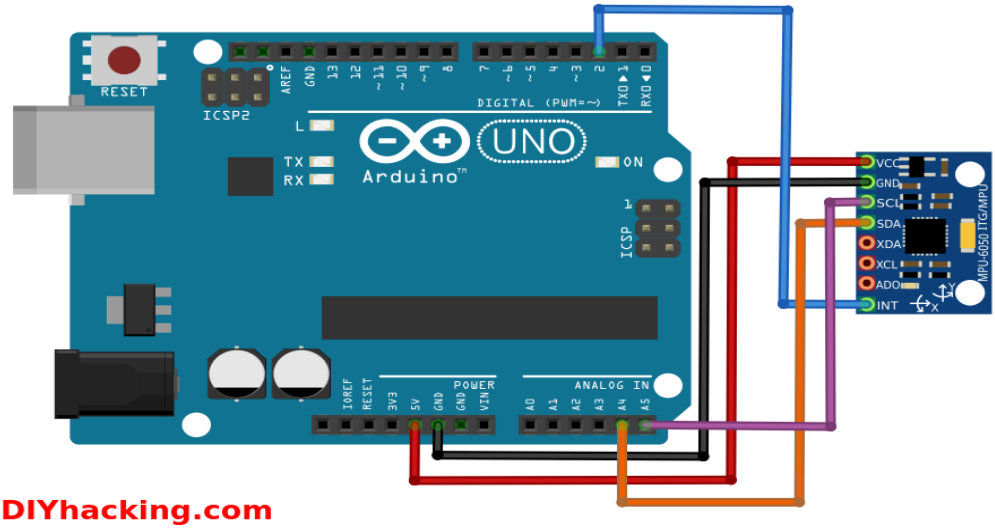
package.xml 中的 build_depend run_depend 项同理。
不过对于 roscpp geometry_msgs sensor_msgs 等最基本的依赖项并非那么严格，注释掉后编译仍然通过，但是当注释掉 serial 后编译出错，提示相关引用未定义。]
安装 arduino:
在官网 https://www.arduino.cc/en/Main/Donate 下载最新的 arduino Linux 版本 arduino-1.8.2-linux64.tar.xz 到 Software 文件夹:

```
cd ~/Software
tar -xJf arduino-1.8.2-linux64.tar.xz
```

运行:

```
cd ~/Software/arduino-1.8.2
./arduino &
```

arduino IDE 的编程语言与 energia IDE 相同，两者程序也很大程度上能够兼容，但并非全部，因此才使用 arduino 来处理 IMU 数据。在 IDE 中打开 mpu6050_serial_to_imu 中的 mpu6050.ino(路径为～/plantbot/src/mpu6050_serial_to_imu /arduino/）。
在验证程序之前需要添加用到的库 I2Cdev 和 MPU6050，将库文件复制到首选项提示的文件夹下 libraries 下即可。
接线：arduino 与 mpu6050 为 I2C 通信，需要调用相关 Wire 库，其文档（http://www.arduino.cc/en/Reference/Wire）中指出：在 UNO 板子上，SDA 接口对应的是 A4 引脚，SCL 对应的是 A5 引脚。MPU6050 需要 5V 的电源，可由 UNO 板直接供电。此外用到 MPU6050 的 DMP(Digital Motion Processing），需要接 INT pin，接线图如下说示:



程序设置好、接好线后按上传，将程序上传到板子上。一切没问题后，在终端输入:

```
roscore
rosrun mpu6050_serial_to_imu  mpu6050_serial_to_imu_node
rostopic list -v
```

显示运行的话题信息，[]中为话题类型，若有 sensor_msgs/Imu 则正常。回显话题数据:

```
rostopic echo /data
```

可以看到 IMU 的数据源源输出。

odom_node 节点编程

```
cd ~/plantbot/src
catkin_create_pkg plantbot_nav roscpp rospy tf std_msgs geometry_msgs nav_msgs sensor_msgs
cd plantbot_nav/src
gedit odom_node.cpp
```

编写节点程序，保存
打开 navigation 文件夹下 CmakeLists.txt，添加：
```
add_executable(odom_node src/odom_node.cpp)
target_link_libraries(odom_node
   ${catkin_LIBRARIES} )
```
然后保存，编译：
```
cd ~/plantbot
catkin_make
```
参考：http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom
http://answers.ros.org/question/231942/computing-odometry-from-two-velocities/

Arduino 与 ROS 交互

在 ROS 工作空间安装：
sudo apt-get install ros-indigo-rosserial-arduino
sudo apt-get install ros-indigo-rosserial
安装 ros_lib 到 Arduino 工作环境：
cd ~/Software/arduino-1.8.2/libraries
rm -rf ros_lib //若已安装先清除，否则下句会出错
rosrun rosserial_arduino make_libraries.py . //此句运行无须打开节点管理器，注意最后的”.”

重写 teleop 包的节点

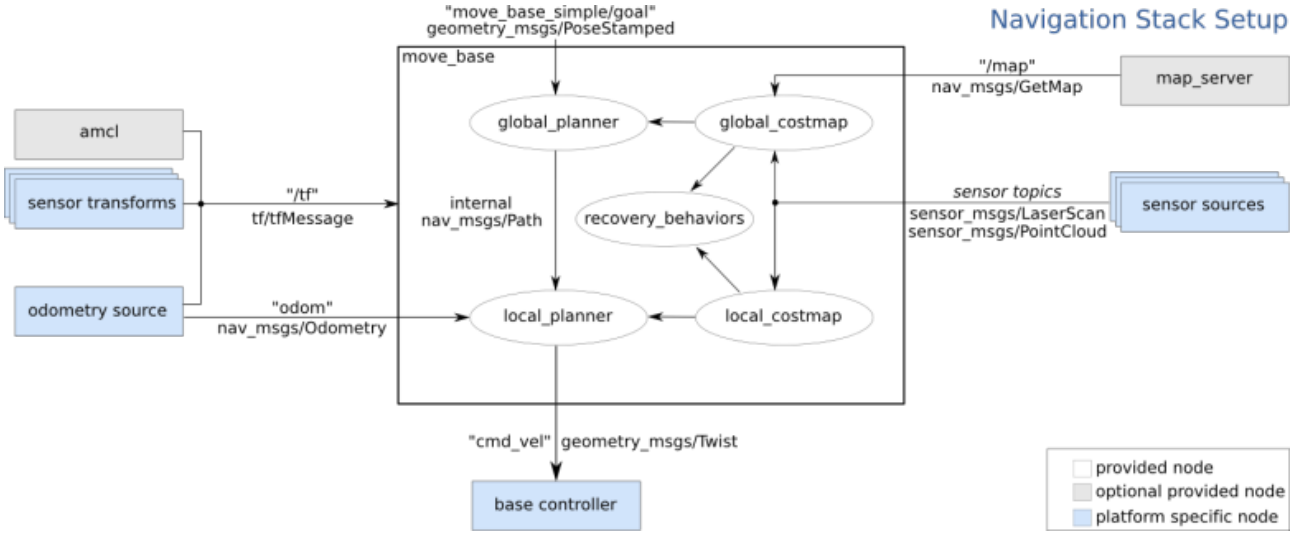
因为加了蓝牙命令控制，因此将原来的 teleop_node.cpp 重写，进行功能分割，由 keyboard_control 节点（keyboard_control.cpp 文件）发送字符控制话题 cmd_char，由 teleop_converter 节点（teleop_converter.cpp 文件）订阅字符控制话题 cmd_char，并转为相应的 cmd_vel 话题发布，这样 Arduino_node 发布的字符控制话题 cmd_char 也可以由 teleop_converter 节点订阅，实现键盘和蓝牙双重控制。对应的 teleop 包中 CmakeLists.txt 文件以及 lanch 文件夹下 teleop.launch 文件也要做相应修改。
参考：ros::spin() 和 ros::spinOnce() 区别及详解

Arduino 编程

打开 Arduino IDE:
```
cd ~/Software/arduino-1.8.2
./arduino &
```
编程，保存为 Arduino_node.ino，传感器接好后上传到 Arduino 板子
测试运行，在终端输入：
```
roscore
rosrun rosserial_python serial_node.py _port:=/dev/ttyUSB0
rosrun teleop tele_converter
rostopic echo cmd_vel
```
手机蓝牙 app 与板子接的蓝牙匹配后（设置波特率保持一致，都为 9600），通过手机 app 发送字符'w'(前向加速),'s'（前向减速）,'a'（左转加速）,'d'（左转减速），观察运行 rostopic 的终端输出对应的速度则正常。
将此节点添加到 teleop.launch 文件。

参考：
rosserial_arduino Tutorials
直接使用 Arduino 提供的 RX（0）、TX（1）口与蓝牙通信：
http://blog.csdn.net/ling3ye/article/details/47067481?ref=myread
使用其他口时需要调用其他库：
http://www.instructables.com/id/Tutorial-Using-HC06-Bluetooth-to-Serial-Wireless-U/

导航



创建机器人 launch 文件：

在 plantbot_nav 文件夹下创建 launch 文件夹，新建 plantbot_configration.launch 文件

创建机器人配置文件：

在 plantbot_nav 文件夹下创建 param 文件夹，新建 costmap_common_params.yaml 、global_costmap_params.yaml、local_costmap_params.yaml、base_local_planner_params.yaml 配置文件

创建机器人 Navigation Stack 的 launch 文件：

在 launch 文件夹下创建 move_base.launch 文件

运行：

先运行 hector_mapping 绘制周围环境的地图信息，绘制完成后使用 map_server 在 rpliar_ros/maps 下保存为 my_map.pgm 文件（由 move_base.launch 文件规定）：

```
roslaunch rplidar_ros hector_rplidar.launch model:='$(find teleop)/urdf/plantbot.urdf'
```

roscd rplidar_ros

cd maps

```
rosrun map_server map_saver -f my_map
```

然后在可该环境中执行导航任务：

```
roscd plantbot_nav
cd launch
roslaunch plantbot_configuration.launch
roslaunch move_base.launch
```

设置目标点：

图形界面设置： _rviz and navigation tutorial

代码设置： _Sending Simple Navigation Goals

参考：

Setup and Configuration of the Navigation Stack on a Robot

拓展部分：

设计 GUI

使用 GUI 可以更为简便直接地控制机器人，将复杂专业的命令行代码简化为按钮等形式进行触发，达到同样控制机器人的效果

安装 Qt SDK：

sudo apt-get install qt-sdk

为了使 Qt 与 python 进行交互，安装 PyQt：

sudo apt-get install python-qt4 pyqt4-dev-tools

或 PySide：

sudo apt-get install python-pyside pyside-tools

打开 Qt Designer：

designer -qt4

gmapping 绘图与 acml 定位

1. slam_gmapping

Subscribed Topics:

tf(tf/Message） scan(sensor_msgs/LaserScan)

Published Topics:

map_metadata(nav_msgs/MapMetaData) map(nav_msgs/OccupancyGrid)

~entropy(std_msgs/Float64)

Required tf Transforms:

<the frame attached incoming scans> -> base_link base_link -> odom

Provided tf Transforms

map -> odom

运行：

rosrun gmapping slam_gmapping

2. acml

Both Gmapping and amcl publishing Map to Odom TF. You wouldn't typically run both at the same time. gmapping performs SLAM (builds a map and localizes simultaneously) while amcl performs localization on a pre-existing map.

利用官方库 rosserial_tivac 实现 launcpad node

安装 rosserial_tivac:

安装 rosserial 二进制文件:

sudo apt-get install ros-indigo-rosserial ros-indigo-rosserial-msgs ros-indigo-rosserial-client ros-indigo-rosserial-python

下载、编译 rosserial_tivac:

cd ~plantbot/src

git clone https://github.com/vmatos/rosserial_tivac.git

cd ..

catkin_make

catkin_make install

将 rosserial_tivac 添加到环境变量：

将 source ~/plantbot/install/setup.bash 添加到~/.bashrc，然后 source ～/.bashrc 下

【注意 plantbot install 文件夹下的 setup.bash 与 devel 文件夹下的 setup.bash 会将 roscd rplidar_ros 引向不同的位置，因此要将 source ~/plantbot/install/setup.bash 一句放在 source

~/plantbot/devel/setup.bash 之前】
改变目录到 energia 库下：
cd ~/Software/energia-1.6.10E18/libraries
rosrun rosserial_tivac make_libraries_energia .
打开 energia 检验：
cd ..
./energia &
在 Examples 下能找到 ros_lib 库及库里的示例程序，表明安装无误。
此时可以像 Arduino 与 ROS 交互那样直接在 Energia IDE 编写结点程序，在 rosserial_tivac Tutorials 更多地是介绍直接在 ROS 环境下编程。两者最后的执行程序都是：
rosrun rosserial_python serial_node.py _port:=/dev/ttyACM0
这样在运行 plantbot 时就需要运行两次 serial_node，虽然可以通过 port 区分开，但不知道同时运行两个相同的节点可不可行。