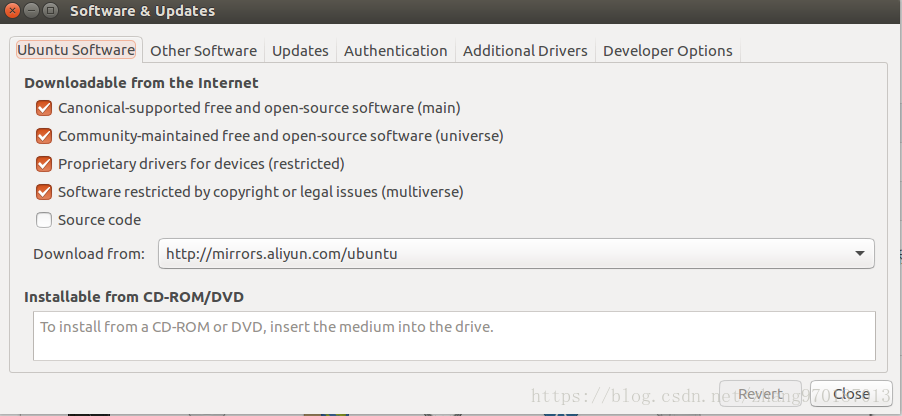
操作系统：Ubuntu 16.04

ROS kinetic

1.安装ROS

软件中心配置：



安装源

sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb\_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

增加key

sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 0xB01FA116

更新

sudo apt-get update

安装桌面完整版（包含ROS、rqt、rviz、通用机器人函数库、2D/3D仿真器、导航以及2D/3D感知功能）：

sudo apt-get install ros-indigo-desktop-full

在开始使用ROS之前你还需要初始化rosdep。rosdep可以方便在你需要编译某些源码的时候为其安装一些系统依赖，同时也是某些ROS核心功能组件所必需用到的工具。

sudo rosdep init

rosdep update

到了这一步要使用ROS的一些命令比如roscd、roslaunch还需要source一下setup.bash文件：

$ source /opt/ros/indigo/setup.bash

如果不想每次用ROS都先运行一句上面那条source命令而是希望每次打开一个新的终端时ROS环境变量都能够自动配置好（即添加到bash会话中），那将会方便得多：

echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc

source ~/.bashrc

安装rosinstall,便利的工具

sudo apt-get install python-rosinstall

**卸载Indigo**

用apt-get方式安装indigo的，卸载使用如下命令：

$ sudo apt-get remove ros-indigo-\*

#### **2.7创建工作空间**

下面是一个创建工作空间的例子

$ mkdir -p ~/catkin\_ws/src

$ cd ~/catkin\_ws/src

$ catkin\_init\_workspace

$ cd ~/catkin\_ws/

$ catkin\_make

$ source devel/setup.bash

在catkin\_ws的devel文件里有一些setup.bash文件，source这些文件可以将当前工作空间设置为catkin工作环境最顶层。

source devel/setup.bash

为了方便以后使用，直接将其加入环境变量

echo "source ~/catkin\_ws/devel/setup.bash" >> ~/.bashrc

source ~/.bashrc

验证安装，输入

roscore

ROS安装成功。

## **安装turtlebot包**

安装前的准备：

$ sudo apt-get install python-rosdep python-wstool ros-indigo-ros

$ sudo rosdep init

$ rosdep update

安装rocon、kobuki、turtlebot

这三个是组合空间的关系，必须按照顺序安装：

$ mkdir ~/rocon

$ cd ~/rocon

$ wstool init -j5 src https://raw.github.com/robotics-in-concert/rocon/indigo/rocon.rosinstall

$ source /opt/ros/indigo/setup.bash

$ rosdep install --from-paths src -i -y

$ catkin\_make

$ mkdir ~/kobuki

$ cd ~/kobuki

$ wstool init src -j5 https://raw.github.com/yujinrobot/yujin\_tools/master/rosinstalls/indigo/kobuki.rosinstall

$ source ~/rocon/devel/setup.bash

$ rosdep install --from-paths src -i -y

$ catkin\_make

$ mkdir ~/turtlebot

$ cd ~/turtlebot

$ wstool init src -j5 https://raw.github.com/yujinrobot/yujin\_tools/master/rosinstalls/indigo/turtlebot.rosinstall

$ source ~/kobuki/devel/setup.bash

$ rosdep install --from-paths src -i -y

$ catkin\_make

环境设置

# 设置udev规则使得电脑自动识别kobuki串口，无需每次用chmod赋权限

$ . ~/turtlebot/devel/setup.bash

$ rosrun kobuki\_ftdi create\_udev\_rules

# 如果使用Kinect作为3D传感器就设置一下TURTLEBOT\_3D\_SENSOR

$ echo "export TURTLEBOT\_3D\_SENSOR=kinect" >> ~/turtlebot/devel/setup.sh

$ source ~/turtlebot/devel/setup.bash

#将Turtlebot空间设置为默认空间，以后可以不用每次运行source命令

$ echo "source ~/turtlebot/devel/setup.bash" >> ~/.bashrc

电池变量：

# 运行turtlebot\_dashboard查看Turtlebot运行状态

$ roslaunch turtlebot\_bringup minimal.launch

$ roslaunch turtlebot\_dashboard turtlebot\_dashboard.launch

# 如果看到上面面板上的笔记本电池能够显示电量就不用专门设置下面的电池环境 变量了,如果无法正常显示电量的话需要设置以下变量

$ echo "export TURTLEBOT\_BATTERY=/proc/acpi/battery/BAT1" >> ~/turtlebot/devel/setup.sh

$ source ~/turtlebot/devel/setup.bash

修改~/.bashrc文件，这样每次启动都将自动设置这些环境变量，否则每次都要用export命令逐个添加这些环境变量。   
先打开.bashrc文件：

$ gedit ~/.bashrc

再在文件末尾添加如下语句：

# For a source installation

source ~/turtlebot/devel/setup.bash

# Export the turtlebot variables

export TURTLEBOT\_BASE=create

export TURTLEBOT\_STACKS=circles

export TURTLEBOT\_3D\_SENSOR=kinect

export TURTLEBOT\_SERIAL\_PORT=/dev/ttyUSB0

到这里就安装完成了。可以以最小模式运行turtlebot机器人测试一下是否安装成功，如果环境变量配置正确，下面命令中的单词输入一部分之后按下Tab键是可以自动补全该单词的。

$ roslaunch turtlebot\_bringup minimal.launch

ROS与navigation与配置激光雷达（RPLIDAR）

雷达配置：

建立工作空间（也可以利用现有的），编译包

$ mkdir -p ~/turtlebot\_ws/src

$ cd ~/turtlebot\_ws/src

## turtlebot建图依赖包

$ git clone https://github.com/turtlebot/turtlebot\_apps

#编译

$ cd ~/turtlebot\_ws

$ catkin\_make

添加环境变量，在〜/ .bashrc中最后添加一行：

$ source /home/ubu/turtlebot\_ws/devel/setup.bash

刷新配置

$ source ~/.bashrc

****创建激光雷达（Rplidar）的串口别名****

确认idVendor和idProduct，ID后面的部分idVendor：idProduct

$ lsusb

Bus 001 Device 006: ID 10c4:ea60

新建/etc/udev/rules.d/rplidar.rules文件，内容如下:(别名为rplidar，实际名称为：/ dev / rplidar）

KERNEL=="ttyUSB\*", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", MODE:="0666", GROUP:="dialout", SYMLINK+="rplidar"

增加当前用户对串口的默认访问权限：

$ sudo usermod -a -G dialout 用户名

使UDEV配置生效:(使串口的默认访问权限生效，需要重启机器）

$ sudo service udev reload

$ sudo service udev restart

****制作雷达驱动启动文件****

复制rplidar.launch到rplidar-laser.launch，并增加TF定义

$ roscd turtlebot\_navigation

$ mkdir -p laser/driver

$ sudo cp ~/turtlebot\_ws/src/rplidar\_ros/launch/rplidar.launch laser/driver/rplidar\_laser.launch

打开rplidar\_laser.launch，并修改

$ rosed turtlebot\_navigation rplidar\_laser.launch

检查frame\_id是否指定为激光

<param name="frame\_id" type="string" value="laser"/>

查看serial\_port是否指定正确端口，使用别名

检查端口：<param name="serial\_port" type="string" value="/dev/rplidar"/>，设置好别名，或直接端口/ dev / ttyUSB0

增加TF：

<node pkg="tf" type="static\_transform\_publisher" name="base\_to\_laser" args="0.0 0.0 0.18 0 0.0 0.0 base\_link laser 100"/>

修改为args =“0.0 0.0 0.18 0 0.0 0.0为自己的实际安装位置。

static\_transform\_publisher部分

static\_transform\_publisher xyz qx qy qz qw frame\_id child\_frame\_id period\_in\_ms

|  |
| --- |
| <launch>  <node name="rplidarNode" pkg="rplidar\_ros" type="rplidarNode" output="screen">  <param name="serial\_port" type="string" value="/dev/ttyUSB0"/>  <param name="serial\_baudrate" type="int" value="115200"/>  <param name="frame\_id" type="string" value="laser"/>  <param name="inverted" type="bool" value="false"/>  <param name="angle\_compensate" type="bool" value="true"/>  </node>  <node pkg="tf" type="static\_transform\_publisher" name="base\_to\_laser" args="0.0 0.0 0.18 0 0.0 0.0 base\_link laser 100"/>  </launch> |

****检查turtlebot\_navigation包****

增加rplidar\_gmapping\_demo.launch文件，用于启动gmapping。

$ roscd turtlebot\_navigation

$ touch launch/rplidar\_gmapping\_demo.launch

$ rosed launch/rplidar\_gmapping\_demo.launch

输入内容：

|  |
| --- |
| <launch>  <!-- Define laser type-->  <arg name="laser\_type" default="rplidar" />  <!-- laser driver -->  <include file="$(find turtlebot\_navigation)/laser/driver/$(arg laser\_type)\_laser.launch" />  <!-- Gmapping -->  <arg name="custom\_gmapping\_launch\_file" default="$(find turtlebot\_navigation)/launch/includes/gmapping/$(arg laser\_type)\_gmapping.launch.xml"/>  <include file="$(arg custom\_gmapping\_launch\_file)"/>  <!-- Move base -->  <include file="$(find turtlebot\_navigation)/launch/includes/move\_base.launch.xml"/>  </launch> |

设置laser\_type为rplidar。

增加rplidar\_gmapping.launch.xml文件，执行gmapping建图

$ roscd turtlebot\_navigation

$ touch launch/includes/gmapping/rplidar\_gmapping.launch.xml

$ rosed launch/includes/gmapping/rplidar\_gmapping.launch.xml

输入内容：

|  |
| --- |
| <launch>  <arg name="scan\_topic" default="scan" />  <arg name="base\_frame" default="base\_footprint"/>  <arg name="odom\_frame" default="odom"/>  <node pkg="gmapping" type="slam\_gmapping" name="slam\_gmapping" output="screen">  <param name="base\_frame" value="$(arg base\_frame)"/>  <param name="odom\_frame" value="$(arg odom\_frame)"/>  <param name="map\_update\_interval" value="0.01"/>  <param name="maxUrange" value="4.0"/>  <param name="maxRange" value="5.0"/>  <param name="sigma" value="0.05"/>  <param name="kernelSize" value="3"/>  <param name="lstep" value="0.05"/>  <param name="astep" value="0.05"/>  <param name="iterations" value="5"/>  <param name="lsigma" value="0.075"/>  <param name="ogain" value="3.0"/>  <param name="lskip" value="0"/>  <param name="minimumScore" value="30"/>  <param name="srr" value="0.01"/>  <param name="srt" value="0.02"/>  <param name="str" value="0.01"/>  <param name="stt" value="0.02"/>  <param name="linearUpdate" value="0.05"/>  <param name="angularUpdate" value="0.0436"/>  <param name="temporalUpdate" value="-1.0"/>  <param name="resampleThreshold" value="0.5"/>  <param name="particles" value="8"/>  <!--  <param name="xmin" value="-50.0"/>  <param name="ymin" value="-50.0"/>  <param name="xmax" value="50.0"/>  <param name="ymax" value="50.0"/>  make the starting size small for the benefit of the Android client's memory...  -->  <param name="xmin" value="-1.0"/>  <param name="ymin" value="-1.0"/>  <param name="xmax" value="1.0"/>  <param name="ymax" value="1.0"/>  <param name="delta" value="0.05"/>  <param name="llsamplerange" value="0.01"/>  <param name="llsamplestep" value="0.01"/>  <param name="lasamplerange" value="0.005"/>  <param name="lasamplestep" value="0.005"/>  <remap from="scan" to="$(arg scan\_topic)"/>  </node>  </launch> |

测试激光雷达gmapping构建地图：

新开端口，打开roscore

$ roscore

新开端口，启动turtlebot

$ roslaunch turtlebot\_bringup minimal.launch

新开端口，启动gmapping，用于构建地图

$ roslaunch turtlebot\_navigation rplidar\_gmapping\_demo.launch

新开端口，启动rviz，实时查看建图情况

$ roslaunch turtlebot\_rviz\_launchers view\_navigation.launch

**构建地图结束保存地图**

新开端口，建立目录，保存地图

$ mkdir -p ~/map

$ rosrun map\_server map\_saver -f ~/map/rplidar\_gmapping

$ ls ~/map #查看内容，包含rplidar\_gmapping.pgm rplidar\_gmapping.yaml

查看地图，已经生成rplidar\_gmapping.pgm文件，可以用图像浏览器（gimp，eog，gthumb，等等）打开查看。

AMCL：

首先运行机器人节点：

roslaunch rbx1\_bringup fake\_turtlebot.launch

然后运行AMCL节点，使用测试地图：

roslaunch rbx1\_nav fake\_amcl.launch map:=test\_map.yaml

然后运行rviz：

rosrun rviz rviz -d `rospack find rbx1\_nav`/nav\_fuerte.vcg

这时在rviz中就应该显示出了地图和机器人，通过rviz在地图上选择目标位置了，然后就会看到机器人自动规划出一条全局路径，并且导航前进。