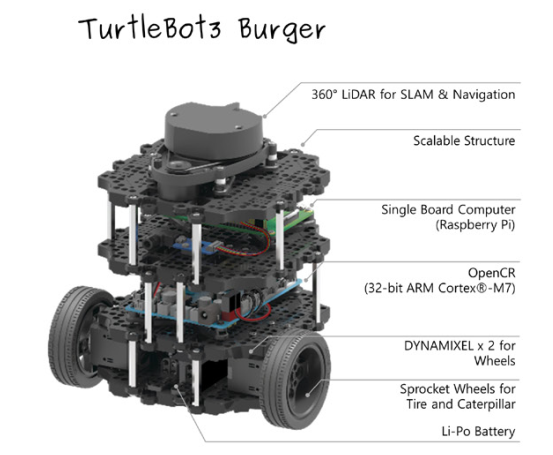
参考书籍：[1]ros官网文件《ROS机器人编程-从基本概念到机器人应用程序编程实践》，ROS\_Robot\_Programming\_CN，非常全可以作为查阅书籍

TurtleBot35有TurtleBot3 Burger、Waffle和Waffle Pi三种官方型号。如不特指，本书中将以TurtleBot3 Burger为例进行说明。重点看第10章

1. 自己总结

* 注意下载的turtlebot3、turtlebot3\_msgs是功能包，而不是整个工作空间
* release版和git直接下的不一样
* P99安装功能包
* catkin\_ws留着安装功能包用

1. 硬件结构如下图

* openCR：车上单片机，用于控制电机和采集一些传感器数据。OpenCR的固件还被称为turtlebot3\_core。固件将OpenCR作为中间控制器，读取TurtleBot3的驱动舵机Dynamixel的编码器值来估算机器人的位置，或者根据上位软件的命令来控制速度。另外，固件还从安装在OpenCR上的3轴加速度和3轴陀螺仪传感器获得加速度和角加速度，以此估计机器人的方向，此外还测量电池电压并将其以话题传输。
* raspberry Pi：文中称为turtlebot3控制实际机器人且搜集传感器信息，用于采集激光雷达和深度摄像头数据，并与远程PC通信
* 远程PC：运行远程控制、SLAM和导航功能包

1. 软件：TurtleBot3的软件由OpenCR控制板的固件（FW）和4个ROS功能包组成。其中，turtlebot3功能包包括TurtleBot3的机器人模型、SLAM和导航功能包、遥控功能包以及与行驶相关的bringup功能包。另外，TurtleBot3的消息文件的集合turtlebot3\_msgs、仿真功能包的集合turtlebot3\_simulations以及应用程序的集合turtlebot3\_applications构成了TurtleBot3的ROS功能包。两台PC都要安装Linux（Ubuntu 16.04兼容的Linux Mint和Ubuntu MATE）作为基本操作系统

节点:节点为椭圆，话题为方形

各个包的简介如：<http://wiki.ros.org/turtlebot3_bringup>

具体跟着书上步骤看，源代码可以在ROBOTIS-GIT远程库中找,以下为pkg-type(name)，pkg为元功能包或功能包名，type为定义的可执行文件名称，和rosrun参数相同，name会覆盖type。launch文件一般以name命名。有些包是turtlebot3私有的，有些是ros公用包

* remote PC：元功能包有

slam：

* turtlebot3\_teleop\_key（turtlebot3\_teleop\_keyboard）：可以接收键盘输入并控制机器人的节点。向turtlebot3\_core节点发送移动速度和旋转速度命令。
* robot\_state\_publisher（robot\_state\_publisher）：将两个轮子和每个关节的三维位置和方向信息以TF形式发布
* slam\_gmapping(turtlebot3\_slam\_gmapping):节点用于绘制地图，根据scan信息（由传感器测量的距离值）和tf值（传感器的位置值，见P362图11-13）来创建地图。
* map\_server-map\_server(map\_saver):创建一个地图。当移动机器人时，机器人会根据测位（odometry）、tf信息和传感器的扫描信息来创建地图。map\_server功能包中的map\_saver节点将利用这个地图的信息（/map话题）生成一个可保存的map.pgm文件和一个信息文件map.yaml。

导航：

* robot\_state\_publisher（robot\_state\_publisher）：将两个轮子和每个关节的三维位置和方向信息以TF形式发布
* rviz（rviz）：运行命令如下，先运行ROS的可视化工具RViz，以便在导航中可以直观地确认目标指定命令和结果。还可以用绿色箭头显示位姿。

$ rosrun rviz rviz -d `rospack find turtlebot3\_navigation`/rviz/turtlebot3\_nav.rviz

* map\_server(map\_server)：用于加载先前创建的地图
* amcl（amcl）：自适应蒙特卡罗定位，Adaptive Monte Carlo Localization），实现二维地图中的机器人定位
* move\_base-move\_base(move\_base)：实现机器人导航中的最优路径规划。包括全局路径规划（global\_planner）和本地实时规划（local\_planner）,前者使用Dijkstra或A\*算法进行全局路径规划。在实际情况中，机器人往往无法严格按照全局路线行驶，所以需要针对地图信息和机器人附近随时可能出现的障碍物规划机器人每个周期内应该行驶的路线，使之尽量符合全局最优路径，后者使用DWA(Dynamic Window Approaches)算法搜索躲避和行进的多条路径，综合各平均标准（是否会撞击障碍物，所需时间等）选取最优路径，并且计算行驶周期内的线速度和角速度，避免与动态出现的障碍物发生碰撞。
* Turtlebot3(树莓派)：书上称呼
* serial\_node(turtlebot3\_core): 负责与TurtleBot3的控制器OpenCR的通信节点接收用户的命令并移动机器人。此时在内部发送测得的机器人自己的位置odom信息，且还会以

odom（里程计模型）→base\_footprint（机器人的中心位置）→base\_link（传感器位姿）→base\_scan（扫描点位置）的顺序将odom的相对坐标变换信息以tf形式发布。

* hls\_lfcd\_lds\_driver（turtlebot3\_lds）：负责运行360度距离传感器LDS。

turtlebot3\_lds节点运行LDS传感器，并将SLAM所需的scan信息发送到slam\_gmapping节点。

* turtlebot3\_diagnostics(turtlebot3\_diagnostics)
* 1

1. Launch文件，元功能包为src下，可在在ros官网Wiki中package搜索

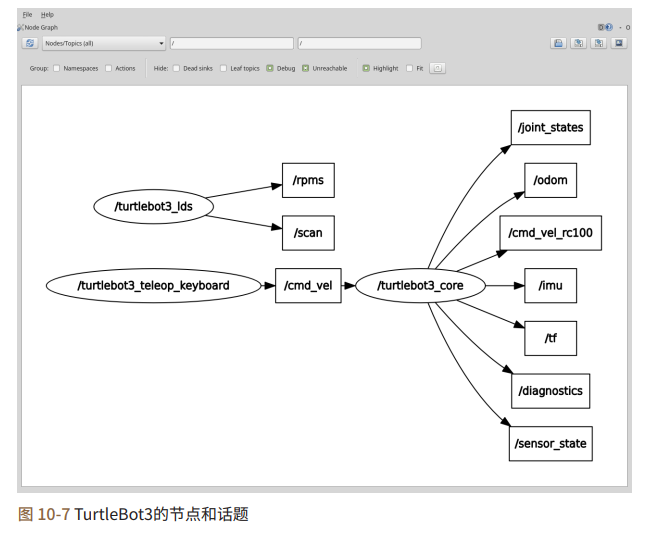
远程PC

* 元功能包turtlebot3
* Turtlebot3\_bringup
* Turtlebot3\_remote.launch:启动desceription.launch、 robot\_state\_publisher
* Turtlebot3\_slam
* turtlebot3\_gmapping.launch:启动turtlebot3\_slam\_gmapping
* turtlebot3\_slam.launch: Turtlebot3\_remote.launch、turtlebot3\_gmapping.launch
* turtlebot3\_navigation
* turtlebot3\_navigation.launch: 启动Turtlebot3\_remote.launch、map\_server、acml.launch、move\_base.launch、rviz
* acml.launch:启动acml（自适应蒙特卡罗定位，Adaptive Monte Carlo Localization）
* move\_base.launch:启动movebase

树莓派

* 元功能包turtlebot3
* Turtlebot3\_bringup
* turtlebot3\_robot.launch：启动节点turtlebot3\_core、turtlebot3\_lds、turtlebot3\_diagnostics

1. 启动机器人上节点如下，分别为树莓派发布话题和订阅话题。发布话题为turtlebot3将传感器等数据传给远程计算机，订阅话题主要为远程计算机对机器人的控制信号。P295,话题列表,下列为名称（形式），话题查找方法，ros-wiki-package-kinetic-nav\_msgs(ros1)，按如下方法一层一层查找的话题



* sensor\_state（turtlebot3\_msgs/SensorState）：这是一个可以查看安装在TurtleBot3上的传感器值。Turtlebot3自定义

<https://github.com/ROBOTIS-GIT/turtlebot3_msgs/blob/master/msg/SensorState.msg>

* battery\_state(sensor\_msgs/BatteryState)可以获得诸如电池电压等状态值。<https://github.com/ros/common_msgs/blob/jade-devel/sensor_msgs/msg/BatteryState.msg>
* rpms（std\_msgs/uint16）：雷达旋转速度
* Scan（sensor\_msgs/LaserScan）：这是一个可以查看安装在TurtleBot3上的雷达的扫描值的话题。

<https://github.com/ros/common_msgs/blob/jade-devel/sensor_msgs/msg/LaserScan.msg>

Header header //一次扫描第一束波束的时间

float32 angle\_min # start angle of the scan [rad]

float32 angle\_max # end angle of the scan [rad]

float32 angle\_increment# angular distance between measurements [rad]

//time between measurements [seconds] - if your scanner ismoving, //this will be used in interpolating position of 3d points

float32 time\_increment

float32 scan\_time # time between scans [seconds]

float32 range\_min # minimum range value [m]

float32 range\_max # maximum range value [m]

float32[] ranges # range data [m] (Note: values < range\_min or > range\_max should be discarded)

# intensity data [device-specific units]. If your device does not provide intensities, please leave the array empty.

float32[] intensities

* Imu（sensor\_msgs/Imu）：这个话题包含机器人的方向值，此方向值由加速度

传感器和陀螺仪传感器值计算而得。

<https://github.com/ros/common_msgs/blob/jade-devel/sensor_msgs/msg/Imu.msg>

# If you have no estimate for one of the data elements (e.g. your IMU doesn't produce an orientation

# estimate), please set element 0 of the associated covariance matrix to -1

# If you are interpreting this message, please check for a value of -1 in the first element of each

# covariance matrix, and disregard the associated estimate.

Header header

geometry\_msgs/Quaternion orientation

float64[9] orientation\_covariance # Row major about x, y, z axes

geometry\_msgs/Vector3 angular\_velocity

float64[9] angular\_velocity\_covariance # Row major about x, y, z axes

geometry\_msgs/Vector3 linear\_acceleration

float64[9] linear\_acceleration\_covariance # Row major x, y z

* odom（nav\_msgs/Odometry）：根据编码器和IMU(Inertial Measurement Unit)信息，可以获得TurtleBot3的测位（odometry）信息。<https://index.ros.org/p/nav_msgs/github-ros-common_msgs/#kinetic-assets>

消息内容：

# This represents an estimate of a position and velocity in free space.

# The pose in this message should be specified in the coordinate frame given by header.frame\_id.

# The twist in this message should be specified in the coordinate frame given by the child\_frame\_id

//<https://github.com/ros/std_msgs/blob/kinetic-devel/msg/Header.msg>

Header header //frame\_id是消息中与数据相关联的参考坐标系id

string child\_frame\_id

<https://github.com/ros/common_msgs/blob/jade-devel/geometry_msgs/msg/PoseWithCovariance.msg>

<https://github.com/ros/common_msgs/blob/jade-devel/geometry_msgs/msg/Pose.msg>

<https://github.com/ros/common_msgs/blob/jade-devel/geometry_msgs/msg/Point.msg>

<https://github.com/ros/common_msgs/blob/jade-devel/geometry_msgs/msg/Quaternion.msg>

geometry\_msgs/PoseWithCovariance pose //可以将四元数当成欧拉角用

<https://github.com/ros/common_msgs/blob/jade-devel/geometry_msgs/msg/TwistWithCovariance.msg>

<https://github.com/ros/common_msgs/blob/jade-devel/geometry_msgs/msg/Twist.msg>

<https://github.com/ros/common_msgs/blob/jade-devel/geometry_msgs/msg/Vector3.msg>

geometry\_msgs/TwistWithCovariance twist

* tf（tf2\_msgs/TFMessage）：它包含TurtleBot3的坐标转换值，例如base\_

footprint和odom等。<https://index.ros.org/p/tf2_msgs/github-ros-geometry2/#kinetic-assets>

//Header header

//string child\_frame\_id # the frame id of the child frame

//Transform transform

<https://github.com/ros/common_msgs/blob/jade-devel/geometry_msgs/msg/TransformStamped.msg>

<https://github.com/ros/common_msgs/blob/jade-devel/geometry_msgs/msg/Transform.msg>

# This represents the transform between two coordinate frames in free space.

//Vector3 translation

//Quaternion rotation

geometry\_msgs/TransformStamped[] transforms

//

* joint\_states(sensor\_msgs/JointState): 不同关节对应数组不同元素，两轮只需2个元素

<https://github.com/ros/common_msgs/blob/jade-devel/sensor_msgs/msg/JointState.msg>

Header header

string[] name //名称

float64[] position //位置，rad或m

float64[] velocity //速度，rad/s或m/s

float64[] effort //扭矩，Nm或N，也可以用电流表示

* Diagnostics（diagnostic\_msgs/DiagnosticArray）：可以获得自检信息。

<https://index.ros.org/p/diagnostic_msgs/github-ros-common_msgs/#kinetic-assets>

# This message is used to send diagnostic information about the //state of the robot

Header header #for timestamp

DiagnosticStatus[] status # an array of components being reported on

<https://github.com/ros/common_msgs/blob/jade-devel/diagnostic_msgs/msg/DiagnosticStatus.msg>

# This message holds the status of an individual component of the robot.

# Possible levels of operations

byte OK=0

byte WARN=1

byte ERROR=2

byte STALE=3

byte level # level of operation enumerated above

string name # a description of the test/component reporting

string message # a description of the status

string hardware\_id # a hardware unique string

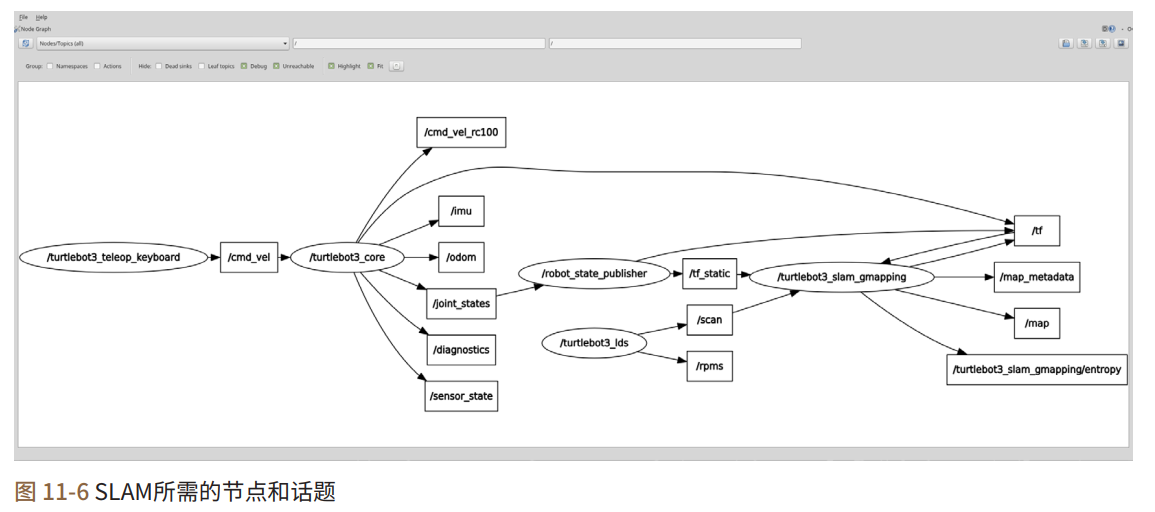
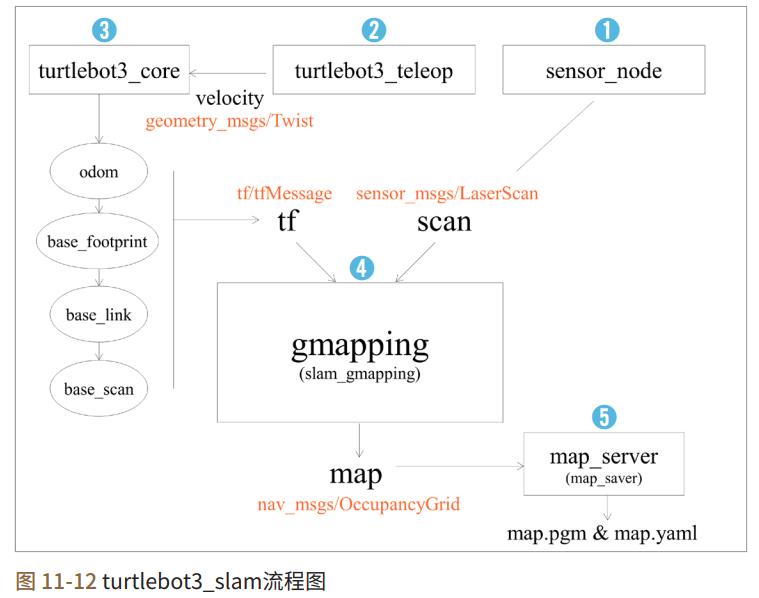
KeyValue[] values # an array of values associated with the status

<https://github.com/ros/common_msgs/blob/jade-devel/diagnostic_msgs/msg/KeyValue.msg>

string key # what to label this value when viewing

string value # a value to track over time

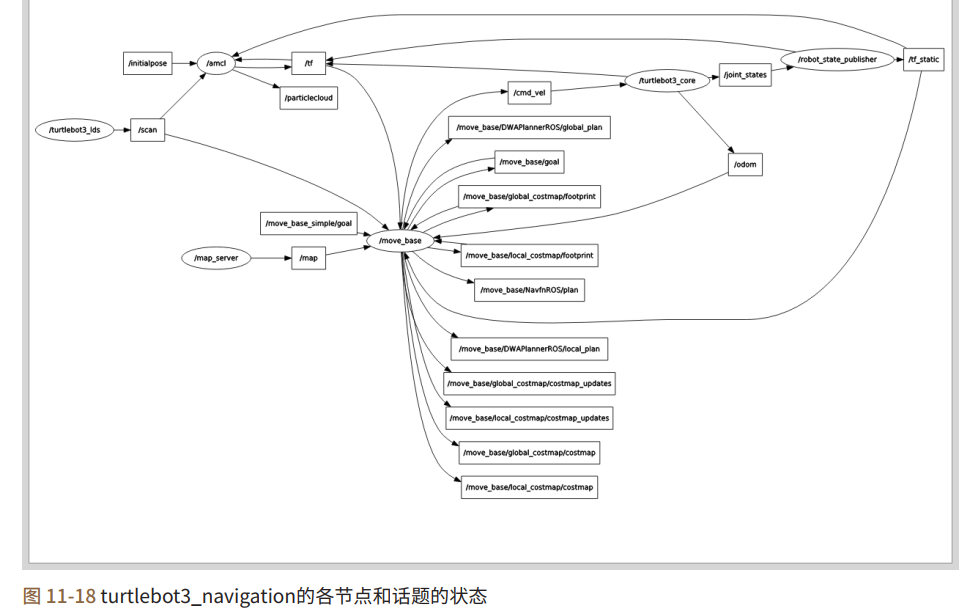
* version\_info（turtlebot3\_msgs/VersionInfo）：可以获得TurtleBot3的硬件、固件和软件等信息。<https://github.com/ROBOTIS-GIT/turtlebot3_msgs/blob/master/msg/VersionInfo.msg>
* cmd\_vel\_rc100（geometry\_msgs/Twist）：这是使用RC-100B（一种基于蓝牙的控制器）时使用的话题，用于移动机器人的速度控制并会订阅此话题。单位使用m/s和rad/s。

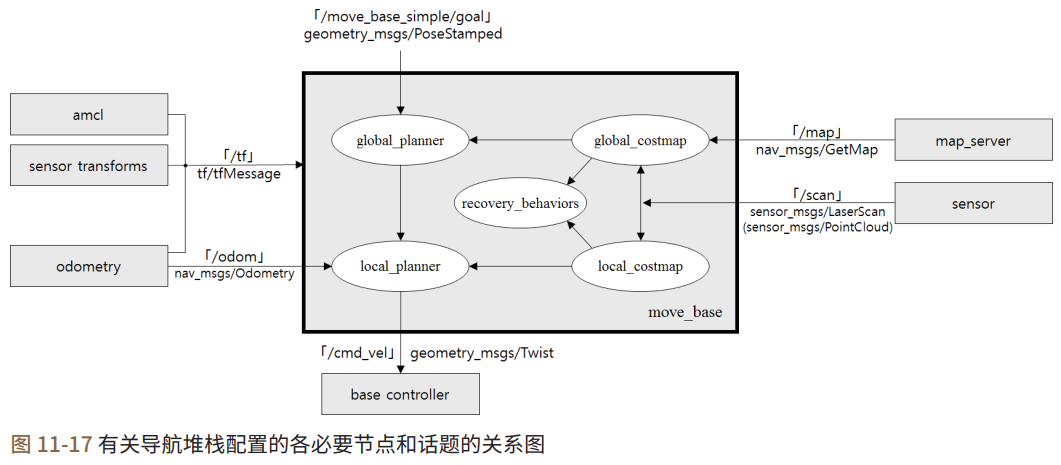
1. P352运行slam产生节点，在turtlebot3\_slam\_gmapping节点中，根据scan信息（由传感器测量的距离值）和tf值（传感器的位置值）来创建地图。

* map\_metadata (nav\_msgs/MapMetaData)：

<https://index.ros.org/p/nav_msgs/github-ros-common_msgs/#kinetic-assets>

* map ([nav\_msgs/OccupancyGrid](http://docs.ros.org/api/nav_msgs/html/msg/OccupancyGrid.html))

1. P371运行导航部分，导航只是用上面产生的地图，并不同时运行。

* tf（tf2\_msgs/TFMessage）：见上面
* odom（nav\_msgs/Odometry）：见上面。机器人的测位信息用于局部路径规划，利用接收到的机器人的当前速度等信息，产生局部移动路径或避开障碍物。
* scan(sensor\_msgs/LaserScan):见上面
* map (nav\_msgs/GetMap)：导航使用占用网格地图（occupancy grid map）。在本教程中，我们将使用map\_server功能包来发布我们之前编写的“map.pgm”和“map.yaml”文件。
* move\_base\_simple/goal（geometry\_msgs/PoseStamped）：目标坐标由用户直接指定。可以使用如平板电脑的设备创建和使用单独的目标坐标命令功能包，但在本教程中，将在ROS的可视化工具RViz中设置目标坐标。目标坐标由二维坐标（x，y）和姿态θ组成。
* cmd\_vel（geometry\_msgs/Twist）：根据最终规划的移动轨迹发布移动机器人的速度命令，而机器人根据该命令移动到目的地。
* 其他各话题待编程再完善

1. 文件类型

* 地图信息以\*.pgm文件格式（portable graymap format）存储和使

用。它还包含一个\*.yaml文件，它也包含地图信息。

1. 从0到255表达的灰度（gray scale）值表示。该值是通过贝叶斯定理的后验概率获得的，该概率代表占用状态的占用概率。由当占用概率超过占用阈值（occupied\_thresh）时表示为黑色的占用区域，而当占用概率小于自由阈值（free\_thresh）时表示为白色的自由区域（free area）
2. P12-15包含相关资源链接
3. 机器人功能包主要包括机器人驱动节
4. 点、获取安装上的传感器的数据的节点、应用安装上的传感器的数据的的节点，以及远程控制节点。如果机器人是多关节机器人，则还包括逆运动学节点，如果是移动机器人则包括导航节点。
5. USB摄像头意味着它是支持USB的视频录制设备。另一个名称是USB video device class（UVC）
6. 相机的接口并不只有USB。某些相机具有可连接到网络的功能。通常连接到局域网或WiFi，将视频数据以视频流形式传输到网络。这些相机应该被称为网络摄像头。此外，有些摄像机使用FireWire(IEEE 1394接口）进行高速传输，主要用于需要高速传输图像的研究目的。FireWire标准在大多数常见的电路板上无法找到，但它是由苹果公司开发的，因此主要用于苹果产品。
7. 作为使用该点云的API的集合，用的最多的是叫做PCL（Point Cloud Library） 12的库，功能包括滤波、分割、表面重构、用模型拟合或提取特征，等
8. P231安装功能包：二进制安装和源码安装
9. P271有关于imu的使用
10. P287软件的安装（开发环境的安装），可以通过这里了解其中详细内容
11. Ubuntu 16.04兼容的Linux Mint和Ubuntu MATE
12. 整体架构：作为参考，TurtleBot3将用户的个人台式机和笔记本电脑称为远程PC，这台PC将担任运行roscore的主节点，会负责远程控制、SLAM、导航等上层控制。与此PC配对的TurtleBot3配备了SBC（自己的树莓派），负责机器人行驶和传感器信息采集。以下远程控制设置示例是在远程PC上运行ROS Master时的示例。然后再是单片机板子
13. P293-SSH来远程访问
14. P294话题
15. 导航算法至少需要下面几种：

* 地图（激光雷达或摄像头建图）
* 测量或估计机器人的姿态的功能（里程计或视觉slam）
* 识别障碍物，如墙壁和物体的功能（激光）
* 能够计算出最优路线并行驶的功能（A\*、D\*等）

1. 导航推测技术用机器人的车轮的旋转量来估计机器人本身的移动量。但车轮的旋转量具有不少的误差。因此还利用IMU传感器等获取惯性信息来补偿位置和方向值，以此减小误差。
2. P318、P337编码器+IMU计算位姿，通常用于位置估算的传感器有编码器（Encoder）和惯性测量单元（IMU）。编码器测量车轮的旋转量，并通过导航推测（dead reckoning）推算机器人的大致位置。在这种情况下会发生一定的误差，此时用惯性传感器测得的惯性信息补偿位置信息的误差。根据目的，位置也可以不用编码器，只用惯性传感器来估算。该位置估计根据通过在创建地图时使用的距离传感器或相机获得的周围环境的信息再次校正位置。这种位置估计方法包括卡尔曼滤波（Kalman filter）、马尔可夫定位（Markov localization）、利用粒子滤波（Particle filter）的蒙特卡罗定位（Monte

Carlo Localization）等等。

1. 称为路径搜索和规划的A\*算法8、势场算法9、粒子过滤算法10和RRT（Rapidly-exploring Random Tree）算法11等
2. 与SLAM相关的常用的功能包有gmapping12、cartographer13和rtabmap14。
3. P324本节中使用的SLAM相关的ROS功能包是turtlebot3元功能包、slam\_gmapping元功能包中的gmapping功能包，以及navigation元功能包中的map\_server功能包。Turtlebot3的大多数功能包已经集成到ros中，所以看不到源码
4. P325保存话题信息可以避免重复做实验
5. P328地图：ROS社区中常用的二维占用网格地图（ OGM，Occupancy Grid Map）。
6. 建图需要距离值和测量该距离值的位置，距离值在ROS中被称为scan，并且姿态（位置+方向）信息会根据相对坐标关系而改变，因此被称为tf（transform）。
7. 如果卡尔曼滤波器是一个分析方法，假设目标是一个线性系统，那么卡尔曼滤波器通过线性运动搜索参数，而粒子滤波器是一种基于尝试和错误（try-and-error）的方法通过仿真进行预测的技术。
8. p342移动机器人使用TurtleBot3，传感器使用LDS。测量环境也与SLAM相同。导航使用的与导航相关的ROS功能包括：turtlebot3元功能包；前一个SLAM课程中编写的turtlebot3元功能包；navigation元功能包中的move\_base、amcl和map\_server功能包。
9. P346避障：如果按照在运动规划中创建的移动轨迹向机器人发出速度命令，则机器人会根据移动轨迹移动到目的地。由于感应、位置估计和运动规划在移动时仍在被执行，因此使用动态窗口方法（Dynamic Window Approach，DWA）算法可避免突然出现的障碍物和移动物体。动态窗口方法（Dynamic Windows Approach，DWA）是在规划移动路径和躲避障碍物时常用的方法，具体是指在机器人的速度搜索空间（velocity search space）中选择适当的速度，以躲避可能碰撞的障碍物的同时能迅速到达目的地。曾广泛使用Trajectory planner，而最近由于DWA的性能优越，因此DWA在逐渐代替Trajectory planner。
10. 1
11. 