

ROS By Example (Volume 2)

Packages and Programs for Advanced
Robot Behaviors

ROS 进阶实例

[美] R. 帕特里克·戈贝尔 (R. Patrick Goebel) © 著

[墨] J. 罗哈斯 (Juan Rojas)

刘振东 罗 盛 林晨宇 杨浦城 陈雅雪 黄镇杰 郭力维 李子然 © 译



中山大學出版社
SUN YAT-SEN UNIVERSITY PRESS

ROS By Example (Volume 2)

Packages and Programs for Advanced Robot Behaviors

ROS 进阶实例

[美] R. 帕特里克·戈贝尔 (R. Patrick Goebel) © 著

[墨] J. 罗哈斯 (Juan Rojas)

刘振东 罗 盛 林晨宇 杨浦城 陈雅雪 黄镇杰 郭力维 李子然 © 译



中山大学出版社
SUN YAT-SEN UNIVERSITY PRESS

· 广州 ·

版权所有 翻印必究

图书在版编目 (CIP) 数据

ROS 进阶实例 / (美) R. 帕特里克·戈贝尔 (R. Patrick Goebel) 著; (墨) J. 罗
哈斯 (Juan Rojas) 等译. —广州: 中山大学出版社, 2017. 4
ISBN 978 - 7 - 306 - 06030 - 3

I. ①R… II. ①R… ②J… III. ①机器人—程序设计 IV. ①TP242

中国版本图书馆 CIP 数据核字 (2017) 第 068188 号

出 版 人: 徐 劲

策划编辑: 周建华 黄浩佳

责任编辑: 黄浩佳

封面设计: 曾 斌

责任校对: 李艳清

责任技编: 何雅涛

出版发行: 中山大学出版社

电 话: 编辑部 020 - 84110283, 84111996, 84111997, 84113349

发行部 020 - 84111998, 84111981, 84111160

地 址: 广州市新港西路 135 号

邮 编: 510275 传 真: 020 - 84036565

网 址: [http:// www. zsup. com. cn](http://www.zsup.com.cn)

E - mail: [zdcbs@mail. sysu. edu. cn](mailto:zdcbs@mail.sysu.edu.cn)

印 刷 者: 湛江日报社印刷厂

规 格: 880mm × 1230mm 1/16 28.875 印张 670 千字

版次印次: 2017 年 4 月第 1 版 2017 年 4 月第 1 次印刷

定 价: 98.00 元

如发现本书因印装质量影响阅读, 请与出版社发行部联系调换。

目 录

1 本卷范围	1
2 安装 ROS-BY-EXAMPLE 代码	3
3 使用 ROS 执行任务	6
3.1 一个虚拟电池模拟器	7
3.2 运行案例所需的一套常规设定	8
3.3 ROS 动作的简短回顾	9
3.4 一个巡逻机器人的案例	10
3.5 使用标准脚本实现的巡逻机器人	11
3.6 脚本方法存在的问题	14
3.7 SMACH 还是行为树?	14
3.8 SMACH: 将任务作为状态机	15
3.8.1 SMACH 回顾	16
3.8.2 使用 SMACH 巡逻一个正方形区域	17
3.8.3 在 ArbotiX 模拟器中测试 SMACH 导航	21
3.8.4 从 SimpleActionState 中获取结果	23
3.8.5 SMACH 迭代器	24
3.8.6 在每个转移上执行命令	27
3.8.7 与 ROS 主题和服务进行交互	28
3.8.8 回调函数与自省	33
3.8.9 并发任务: 将电池检查加入巡逻程序中	34
3.8.10 对进行电池检查的巡逻机器人的注解	41
3.8.11 在状态与状态机之间传递用户数据	42
3.8.12 子任务与分层状态机	46
3.8.13 为房屋清洁机器人添加电池检查	52
3.8.14 状态机的缺点	52
3.9 行为树	53
3.9.1 行为树与分层状态机的对比	53
3.9.2 行为树的关键属性	54
3.9.3 建立一棵行为树	55

3.9.4	选择器与序列器	56
3.9.5	使用修饰器（元行为）对行为进行自定义	58
3.10	为 ROS 和行为树编程	59
3.10.1	安装 pi_trees 程序库	59
3.10.2	pi_trees 程序库的基本部件	59
3.10.3	ROS 专用行为树类	67
3.10.4	一个使用行为树的 Patrol Bot 例子	73
3.10.5	一个使用行为树的居家清洁机器人	82
3.10.6	并行任务	88
3.10.7	添加和移除任务	90
4	为你的机器人创建一个 URDF 模型	92
4.1	从底座和轮子开始	92
4.1.1	robot_state_publisher 和 joint_state_publisher 节点	94
4.1.2	底座的 URDF/Xacro 文件	95
4.1.3	替代/base_footprint 坐标系的方法	100
4.1.4	把底座加到机器人模型中	101
4.1.5	查看机器人的变换树（tf 树）	102
4.1.6	使用网格模型创建底座	103
4.2	简化你的网格模型	108
4.3	添加一个躯干	108
4.3.1	为躯干建模	109
4.3.2	把躯干附着到底座上	110
4.3.3	使用网格模型为躯干建模	112
4.3.4	把网格躯干附着到网格底座上	113
4.4	测量、计算和调整	115
4.5	添加一个相机	115
4.5.1	相机的位置	116
4.5.2	为相机建模	116
4.5.3	在躯干和底座上添加相机	120
4.5.4	查看有躯干和相机情况下的变换树	122
4.5.5	使用网格模型为相机建模	123
4.5.6	使用 Asus Xtion Pro 替换 Kinect	124
4.6	添加一个激光扫描仪（或其他传感器）	124
4.6.1	为激光扫描仪建模	125

4.6.2	添加一个激光扫描仪（或其他传感器）到一个网格底座	126
4.6.3	配置激光节点的启动文件	127
4.7	添加一个可摇移和倾斜的头部	128
4.7.1	用 Asus Xtion Pro 取代 Kinect	130
4.7.2	为摇移和倾斜头建模	130
4.7.3	理解旋转轴	133
4.7.4	使用网格模型的 Pi 机器人云台头	134
4.7.5	在 Pi 机器人上使用 Asus Xtion Pro 网格取代 Kinect	135
4.8	添加一条或两条机械臂	136
4.8.1	机械臂的放置	136
4.8.2	机械臂的建模	136
4.8.3	为计划增加一个夹持器坐标系	139
4.8.4	增加第二条机械臂	140
4.8.5	使用机械臂电机和支架的网格	142
4.9	为 Box 机器人增加一个可伸缩躯干	144
4.10	为 Pi 机器人增加一个可伸缩躯干	144
4.11	一个桌面上的单臂 Pi 机器人	145
4.12	用 ArbotiX 仿真器来测试你的模型	146
4.12.1	虚拟 Box 机器人	147
4.12.2	一个虚拟的 Pi 机器人	149
4.13	创建你的机器人的描述程序包	149
4.13.1	从 rbx2_description 程序包中复制文件	150
4.13.2	创建一个测试启动文件	150
5	控制伺服电机：再一次	152
5.1	安装 ArbotiX 程序包	152
5.2	启动 ArbotiX 节点	152
5.3	ArbotiX 配置文件	158
5.4	在模拟模式中测试 ArbotiX 关节控制器	163
5.5	在真实伺服电机中测试 ArbotiX 关节控制器	165
5.6	松弛所有的伺服电机	167
5.7	启用或禁用所有的伺服电机	171
6	机器诊断	172
6.1	DiagnosticStatus 信息	173
6.2	分析器配置文件	174

6.3	监控 Dynamixel 伺服电机温度	175
6.3.1	为云台监控伺服电机	175
6.3.2	查看在/diagnostics 主题上的信息	177
6.3.3	通过监控/diagnostics 主题来保护伺服电机	180
6.4	监控笔记本的电池	185
6.5	创造属于你自己的诊断信息	186
6.6	监控其他硬件状态	193
7	动态重置	195
7.1	添加动态参数到你的节点	196
7.1.1	创建 .cfg 文件	196
7.1.2	让 .cfg 文件可执行	197
7.1.3	配置 CMakeLists.txt 文件	197
7.1.4	构建程序包	198
7.2	将动态重置容量加入电池仿真器节点中	198
7.3	添加动态重配置客户端支持到 ROS 节点	204
7.4	从命令行动态重配置	207
8	多话题 with mux & yocs	209
8.1	为 mux 话题设置启动文件	209
8.2	用虚拟 TurtleBot 机器人测试 mux	211
8.3	使用 mux 服务切换输入	212
8.4	优先处理 mux 输入的 ROS 节点	212
8.5	Yujin 机器人的 YOCS 控制器	217
9	3D 世界中的头部追踪	221
9.1	追踪虚构的 3D 目标	221
9.2	在机器人上追踪一个点	223
9.3	3D 头部追踪节点	226
9.3.1	真实的以及虚拟的头部追踪	226
9.3.2	将目标映射到摄像头平面	227
9.4	用真实的伺服电机做头部追踪	231
9.4.1	真实伺服电机以及虚拟目标	231
9.4.2	真实的伺服电机，真实的目标	232
9.4.3	节点和启动文件	234

10	检测与跟踪 AR 标签	239
10.1	安装与测试 ar_track_alvar 程序包	239
10.1.1	创建你自己的 AR 标签	240
10.1.2	生成并打印 AR 标签	241
10.1.3	启动摄像头驱动和 ar_track_alvar 节点	242
10.1.4	测试标记检测	243
10.1.5	理解/ar_pose_marker 主题	244
10.1.6	观察在 RViz 中的标记	245
10.2	访问你的程序中的 AR 标签位姿	246
10.2.1	ar_tags_cog.py 脚本	246
10.2.2	用平移和倾斜头跟踪标签	251
10.3	使用标记束跟踪多个标签	251
10.4	移动机器人跟随一个 AR 标签	252
10.5	练习：使用 AR 标签定位	257
11	用 MoveIt! 做机械臂导航	258
11.1	我需要一台有机臂的实体机器人吗?	259
11.2	自由度	259
11.3	关节类型	260
11.4	关节轨迹和关节轨迹动作控制器	260
11.5	正向和逆向机械臂运动学	263
11.6	逆运动学的数值解法和分析解法	264
11.7	MoveIt! 的架构	264
11.8	安装 MoveIt!	265
11.9	为你的机器人创建静态 URDF 模型	266
11.10	运行 MoveIt! 设置助手	267
11.10.1	加载机器人的 URDF 模型	268
11.10.2	生成碰撞矩阵	269
11.10.3	添加 base_odom 虚拟关节	269
11.10.4	添加右臂规划组	270
11.10.5	添加右夹持器规划组	273
11.10.6	定义机器人位姿	275
11.10.7	定义末端执行器	277
11.10.8	定义被动关节	277
11.10.9	生成配置文件	277

11.11	用 MoveIt! 设置助手创建的配置文件	278
11.11.1	SRDF 文件 (<机器人名>.srdf)	278
11.11.2	fake_controllers.yaml 文件	280
11.11.3	joint_limits.yaml 文件	280
11.11.4	kinematics.yaml 文件	281
11.12	move_group 节点和启动文件	283
11.13	在演示模式下测试 MoveIt!	284
11.13.1	探索 Motion Planning 插件的额外功能	287
11.13.2	重新运行设置助手	288
11.14	用命令行测试 MoveIt!	288
11.15	确定关节配置和末端执行器位姿	291
11.16	使用 ArbotiX 关节轨迹动作控制器	293
11.16.1	在仿真环境里测试 ArbotiX 关节轨迹动作控制器	294
11.16.2	用真实的电机测试 ArbotiX 关节轨迹控制器	303
11.17	配置 MoveIt! 关节控制器	303
11.17.1	创建 controllers.yaml 文件	304
11.17.2	创建控制器管理器启动文件	307
11.18	MoveIt! 的 API	308
11.19	正向运动学: 在关节空间进行规划	309
11.20	逆向运动学: 在笛卡尔空间内的规划	318
11.21	把手指向或伸向一个视觉目标	327
11.22	为规划的轨迹设置限制	328
11.22.1	执行笛卡尔路径	329
11.22.2	设置其他路径限制	334
11.23	调整轨迹速度	337
11.24	为规划添加障碍	342
11.25	把物体与工具附在机器人上	352
11.26	拾取和放置	354
11.27	添加一个传感器控制器	368
11.28	在一个真实机械臂上运行 MoveIt!	371
11.28.1	创建机器人的启动文件和脚本	371
11.28.2	运行机器人的启动文件	372
11.28.3	真实机械臂上的正运动学	372
11.28.4	真实机械臂上的逆运动学	373
11.28.5	真实机械臂上的笛卡尔路径	374

11.28.6	真实机械臂上的拾取—放置	374
11.28.7	指向或伸向一个视觉目标	374
12	Gazebo: 模拟世界与机器人	376
12.1	安装 Gazebo	376
12.2	硬件图形加速	378
12.3	安装 ROS Gazebo 程序包	378
12.4	安装 Kobuki ROS 程序包	379
12.5	安装 Fetch Robot ROS 程序包	379
12.6	使用 Gazebo GUI	379
12.7	在 Gazebo 里测试 Kobuki 机器人	380
12.7.1	访问模拟传感器数据	383
12.7.2	为 Kobuki 添加安全控制	386
12.7.3	运行来自本书第一卷书的 nav_square.py 脚本	388
12.8	加载其他世界和物体	389
12.9	在 Gazebo 中测试 Fetch 机器人	390
12.9.1	Fetch 的关节轨迹	391
12.9.2	Fetch 和 MoveIt!	392
12.9.3	Fetch 的取和放	393
12.10	使用 simple_grasping 感知管道的真实取和放	394
12.10.1	深度摄像头的局限性	394
12.10.2	运行演示	395
12.10.3	理解 real_pick_and_place.py 脚本	398
12.11	运行 Gazebo Headless + RViz	402
13	ROSBRIDGE: 为你的机器人构建 WEB 图形用户界面	405
13.1	安装 rosbridge 程序包	405
13.2	安装 web_video_server 程序包	406
13.3	安装一个简单的 Web 服务器 (mini-httpd)	408
13.4	启动 mini-httpd、rosbridge 和 web_video_server	409
13.5	一个简单的 rosbridge HTML/Javascript 图形用户界面	411
13.6	在模拟 TurtleBot 中测试图形用户界面	413
13.7	在真实的机器人中测试图形用户界面	413
13.8	在网络中的另外一台设备上查看 Web 图形用户界面	413
13.9	使用浏览器调试控制台	414
13.10	理解简单的图形用户界面	415

13.10.1	HTML 布局: simple_gui.html	415
13.10.2	JavaScript 代码: simple_gui.js	421
13.11	一个使用 jQuery、jqWidgets 和 KineticJS 的更高级的图形用户界面	434
13.12	Rosbridge 总结	438
附录	ROS 的附件及使用 USB 设备: 创建 udev Rules	440
1	将你的账户加入 dialout 组	440
2	删除设备的序列号	440
3	UDEV 规则	441
4	测试一条 UDEV 规则	442
5	在 ROS 配置文件中 使用设备名	442

1 本卷范围

在第一卷（《ROS 入门实例》）中，我们学到了如何使用 ROS 的基本组件来对机器人进行编程，这些组件包括对可移动底座的控制、SLAM、机器人视觉（OpenCV，OpenNI 和一点 PCL）、语音识别和使用伺服电机来控制关节。在这一卷书中，我们将会面对更高级的 ROS 概念和一些对于真正的自动机器人行为编程有用的程序包，包括如下：

- 可执行任务管理器，例如：smach 和 behavior trees。
- 使用 URDF/Xacro 描述性语言来为你自己的机器人创建一个模型，这个模型包括一个可抬头摇头的头部、多关节机械臂和夹持器、可伸缩的躯干和安置像激光扫描仪那样的传感器。
- 配置 arbotix 程序包来控制伺服电机。
- 使用 ROS 诊断 程序包使你的机器人可以监测它自己的系统，例如电池水平和伺服电机温度。
- 使用 dynamic reconfigure 来即时改变机器人的参数。
- 使用 mux 和 yocs 来复用 ROS 主题，使控制输入可以被优先考虑，并且这些控制输入不会互相矛盾。
- 使用 AR 标签 用于目标检测和追踪。
- 跟踪三维物体，并感知物体和机器人的空间关系。
- 使用新的 MoveIt! 框架来控制多关节臂和夹持器，包括正向运动和反向运动的执行、碰撞避免、拾取、挑选和放置任务。
- 使用先进的 Gazebo 模拟器来与模拟机器人和环境工作。
- 使用 rosbridge，HTML 和 Javascript 为机器人创建一个基于 web 的图形用户界面。

注意：在 Gazebo 的那一章里，假定我们已经有一个像 Kobuki 或者 Fetch 那样的机器人模型，这些模型的开发者们提供了必要的 Gazebo 特性和插件来模拟机器人的物理特性。为一个新的机器人从头开始创建 worlds 或者 Gazebo 插件已经超出了这卷书的范围，但是这些在在线的 Gazebo 教程 中有介绍。

在这本书中，我们将要介绍的一些主题如果单独拿出来可以用一整本书去介绍，所以我们将重点放在关键的概念介绍上。如果读者需要探索进一步的细节，我们也提

供了额外的参考。如在第一卷，所有的代码示例都是用 Python 写的。这种语言往往比 C++ 更容易入门。

一旦你掌握了本书中呈现过的概念，那你就能够为你的机器人创建一个 URDF 模型了。这个模型包括一个机械臂和一个可抬头或摇头的头部。然后编程使之进行一系列自动的任务和行为、监视它的子系统、确定控制输入的优先级以匹配当前状况、追踪三维物体、了解如何用碰撞感知逆运动学来为多关节臂和夹持器进行编程，并且使用网页浏览器编写 HTML/Javascript 接口来监视和控制你的机器人。

2 安装 ROS-BY-EXAMPLE 代码

请注意本书中的代码都是以 ROS Indigo 版本写的。虽然其中有一些代码能在更早的版本 ROS 中运行，但强烈建议读者使用 ROS Indigo 版本。特别是下面列出的依赖程序包和为机械臂导航的 MoveIt! 样例代码程序是只适用于 Indigo 的。

在安装第二卷代码本身之前，如果我们先安装那些大部分以后会用到的 ROS 程序包的话，后续可以节省一些时间。（在本书中如需安装某单个程序包，安装指导同样会给出。）简单地复制粘贴以下命令（不包括 \$ 符号）进终端来安装我们所需的 Debian 程序包。在每行命令末尾的 \ 字符让整个代码块对于 Linux 来说是作为一整行进行复制粘贴的：

```
$ sudo apt-get install ros-indigo-arbotix ros-indigo-openni-camera \
ros-indigo-dynamixel-motor ros-indigo-rosbridge-suite \
ros-indigo-mjpeg-server ros-indigo-rgbd-launch \
ros-indigo-moveit-full ros-indigo-moveit-ikfast \
ros-indigo-turtlebot-* ros-indigo-kobuki-* ros-indigo-moveit-python \
python-pygraphviz python-pygraphviz python-easygui \
mini-httpd ros-indigo-laser-pipeline ros-indigo-ar-track-alvar \
ros-indigo-laser-filters ros-indigo-hokuyo-node \
ros-indigo-depthimage-to-laserscan \
ros-indigo-gazebo-ros ros-indigo-gazebo-ros-pkgs \
ros-indigo-gazebo-msgs ros-indigo-gazebo-plugins \
ros-indigo-gazebo-ros-control ros-indigo-cmake-modules \
ros-indigo-kobuki-gazebo-plugins ros-indigo-kobuki-gazebo \
ros-indigo-smach ros-indigo-smach-ros ros-indigo-grasping-msgs \
ros-indigo-executive-smach ros-indigo-smach-viewer \
ros-indigo-robot-pose-publisher ros-indigo-tf2-web-republisher \
ros-indigo-move-base-msgs ros-indigo-fake-localization \
graphviz-dev libgraphviz-dev gv python-scipy liburdfdom-tools \
ros-indigo-laptop-battery-monitor ros-indigo-ar-track-alvar* \
ros-indigo-map-server ros-indigo-move-base* \
ros-indigo-simple-grasping
```

如果你阅读的是本书的纸质版，那复制粘贴可能行不通。不过你可以使用下面的命令来下载一个名为 rbx2-prereq.sh 的小的 shell 脚本，脚本将运行上面列出的 apt-get 命令。

```
$ cd ~
$ wget https://raw.githubusercontent.com/pirobot/rbx2/indigo-devel/\
rbx2-prereq.sh
$ sh rbx2-prereq.sh
```

我们还需要本书第一卷 (rbx1) 代码仓库中的代码 (即使你没有那本书)。运行下面的命令来安装 ROS Indigo 版的 rbx1 代码 (若你的电脑上还没有安装):

```
$ cd ~/catkin_ws/src
$ git clone -b indigo-devel https://github.com/pirobot/rbx1.git
$ cd ~/catkin_ws
$ catkin_make
$ source ~/catkin_ws/devel/setup.bash
```

要克隆以及编译第二卷 ROS Indigo 版本的代码仓库 (rbx2), 执行以下步骤:

```
$ cd ~/catkin_ws/src
$ git clone -b indigo-devel https://github.com/pirobot/rbx2.git
$ cd ~/catkin_ws
$ catkin_make
$ source ~/catkin_ws/devel/setup.bash
```

注意: 若你还没有这么做, 请把上面的最后一行加到 `~/ .bashrc` 文件的末尾。这将保证每次你打开一个新的命令行窗口时, `catkin` 程序包都被加到 `ROS_PACKAGE_PATH` 中。

如果 *ROS By Example* 的代码接下来更新了, 你可以把更新后的版本和本地保存的代码资源用以下命令合并:

```
$ cd ~/catkin_ws/src/rbx2
$ git pull
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
```

保持更新: 如果你想了解更多关于新书和本书中代码的更新情况, 请加入 [ros-by-example](#) 的 Google 群组。

所有的 *ROS By Example* 第二卷程序包都以 `rbx2` 开头。如果要列出所有程序包, 进入 `rbx2` 元程序包的父文件夹, 使用 Linux `ls` 命令:

```
$ roscd rbx2
$ cd ..
$ ls -F
```

我们应该会看到以下列出的信息：

```
pedestal_pi_no_gripper_moveit_config/  rbx2_bringup/      rbx2_msgs/
pedestal_pi_with_gripper_moveit_config/  rbx2_description/  rbx2_nav/
pi_robot_moveit_config/                 rbx2_diagnostics/  rbx2_tasks/
rbx2 /                                   rbx2_dynamixel/    rbx2_utils/
rbx2_arm_nav/                           rbx2_gazebo/       rbx2_vision/
rbx2_ar_tags/                           rbx2_gui/          README.md
```

贯穿本书，我们会使用 `roscd` 命令来从一个程序包进入另一个程序包。例如，要进入 `rbx2_dynamixel` 程序包，请使用下面的命令：

```
$ roscd rbx2_dynamixel
```

注意，你在任意目录下运行这个命令，ROS 都会找到这个程序包。

重要：如果你使用两台电脑来控制或监控你的机器人，例如有一个笔记本在机器人上，同时桌面上还有另一台电脑，请确保在两台机器上都复制和构建了 `rbx2` 和 `rbx1` 代码仓库的 Indigo 分支。

3 使用 ROS 执行任务

正如我们在第一卷中所发现的，给一个机器人编程以执行如脸部追踪、坐标之间导航或跟踪一个人等特定行为，是相对直接明了的。但是一个全自动机器人会被要求根据当前的任务以及当前的条件，从一个更大的行为集合中选取它自己的动作。

在本章中，我们将学习如何使用两种不同的启用 ROS 的任务执行框架：SMACH（使用了状态机，读作“smash”）和pi_trees（行为树）。这两种方法都有各自的优点和缺点，并且你的编程背景可能决定了你掌握其中一种方法的难易程度。但是，两种方法都提供了一个更为结构化的任务管理途径，而非简单地罗列一长串 if-then 语句。

全局任务控制器一般被称为任务执行器，多数此类执行器都至少包含以下关键特性：

- 任务优先级：若有一项高优先级的任务需要相同的资源（例如：驱动电机），则低优先级的任务应被抢占资源。
- 暂停与继续：当一项高优先级任务被给予了控制权，当前执行的一项（或多项）任务被暂停，而当执行器收回控制权，被抢占的一项（或多项）任务继续进行，这通常是一项可取的特性。例如，Neato 吸尘器就可以在充电之后返回之前中断的任务中。
- 任务层级：任务通常可以拆分成子任务，以进行细节处理。例如，一项被称为再充电的高级别任务可能由三项子任务组成：导航至充电站，将机器人置入充电站以及给电池充电。类似的，将机器人置入充电站的任务又可拆分成：对准指示灯，向前驱动，置入充电站后停止。
- 条件：传感器数据与内部编程变量都可对任务执行的时间与方式产生约束。在 ROS 中，这些变量通常以各种节点发布的信息为形式。例如，电池监控任务会订阅一个带有电池水平的诊断主题。当低电池水平被检测到时，检查电池的条件被激活，再充电的任务将会执行，其他任务则会暂停或中止。
- 并发性：多项任务可以并发进行。例如，导航任务（前往下一个路径点）与电池监控任务必须同时进行。

为了说得具体一些，我们将在这一整章中使用两个案例场景：一个是“巡逻机器人”，它必须在不让电池耗尽的同时有顺序地经过一系列的位置；另一个是“房屋清洁机器人”，它必须到达若干个房间并进行与每间房间相关的清洁任务。