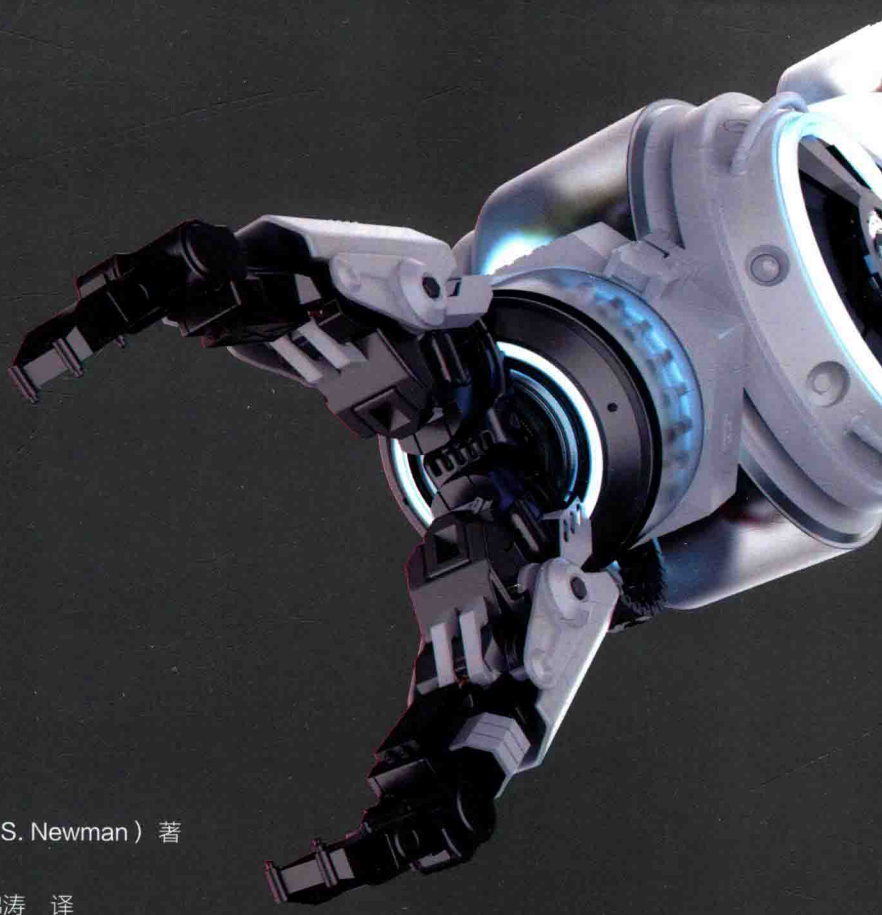
HZ BOOKS
华章IT

机器人学译丛

CRC Press
Taylor & Francis Group



[美] 怀亚特·S. 纽曼（Wyatt S. Newman）著

李笔锋 祝朝政 刘锦涛 译

何明 李静 张瑞雷 审

ROS机器人编程

原理与应用

A SYSTEMATIC APPROACH TO LEARNING ROBOT PROGRAMMING WITH ROS

机械工业出版社
China Machine Press

机器人学译丛

[美] 怀亚特·S. 纽曼（Wyatt S. Newman）著

李笔锋 祝朝政 刘锦涛 译

何明 李静 张瑞雷 审

ROS机器人编程
原理与应用

A SYSTEMATIC APPROACH TO LEARNING ROBOT PROGRAMMING WITH ROS

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

ROS 机器人编程：原理与应用 /（美）怀亚特·S. 纽曼（Wyatt S. Newman）著；李笔锋，祝朝政，刘锦涛译 . —北京：机械工业出版社，2019.4
（机器人学译丛）
书名原文：A Systematic Approach to Learning Robot Programming with ROS

ISBN 978-7-111-62576-6

I. R… II. ①怀… ②李… ③祝… ④刘… III. 机器人 – 程序设计 IV. TP242

中国版本图书馆 CIP 数据核字（2019）第 074214 号

本书版权登记号：图字 01-2018-4598

ROS 已在学术界、工业界和研究机构中广泛使用。本书系统化地介绍了 ROS、ROS 包、ROS 工具的组织，以及如何将现有 ROS 包纳入新的应用中，并开发新的 ROS 包。本书分为六部分，共 18 章。第一部分介绍如何编写 ROS 节点和 ROS 工具，也覆盖了消息、类和服务器。第二部分介绍如何用 ROS 进行仿真和可视化，其中包括坐标变换。第三部分讨论 ROS 的感知过程。第四部分介绍 ROS 中的移动机器人控制和导航。第五部分介绍 ROS 机械臂的相关知识。第六部分介绍系统集成和更高级别的控制，包括基于感知的移动操作。本书既可作为 ROS 课程的教材，也可作为机器人研究人员、工程师及爱好者的参考书。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：冯秀泳　　责任校对：殷　虹
印　　刷：中国电影出版社印刷厂　　版　　次：2019 年 5 月第 1 版第 1 次印刷
开　　本：185mm × 260mm　1/16　　印　　张：29
书　　号：ISBN 978-7-111-62576-6　　定　　价：199.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88379833　　投稿热线：（010）88379604
购书热线：（010）68326294　　读者信箱：hzjsj@hzbook.com

THE TRANSLATOR'S WORDS

译者序

易科机器人实验室联手机械工业出版社华章公司，为传播最新的机器人技术以及方便国内读者学习，引进出版了一系列 ROS 入门与实践的书籍，如侧重于入门基础的《ROS 机器人程序设计（原书第 2 版）》(ISBN：978-7-111-55105-8）、《ROS 机器人高效编程（原书第 3 版）》(ISBN：978-7-111-57846-8），以及体现当前 ROS 开发应用最新成果的《ROS 机器人项目开发 11 例》(ISBN：978-7-111-59817-6）、《ROS 机器人开发：实用案例分析》(ISBN：978-7-111-59372-0）。除此之外，近年来国内亦有许多专家贡献了大量的作品，所涵盖的内容已经非常丰富了。但翻遍所有已出版的 ROS 图书，心中不免又有一点遗憾，总感觉似乎还缺少这么一本书——它在宽度上，能够概述机器人学相关基础；在深度上，又能挖掘 ROS 底层原理。为了弥补此缺憾，我们团队成员以及相关专家甚至为此开过数次研讨会，甚至都拟定了初步的提纲，差点就要撸起袖子来自己干了！

2018 年，Wyatt S. Newman 教授出版了《A Systematic Approach to Learning Robot Programming with ROS》，我们发现此书不同于以往图书只是注重实践操作，它对 ROS 的底层原理做了深入的解释，对一些机器人学基础知识也做了必要的介绍，这对于机器人入门学习是非常有帮助的，正是符合我们之前设想的一本难得的好书！因而决定翻译引进。在书名翻译上，我们也是几经周折，先后调整了三四次。为了凸显此书特色，我们再三斟酌决定增加副书名“原理与应用”。为了言简意赅，华章公司的老师也建议不必直译书名，而是关注国内读者阅读习惯，最后本书命名为《ROS 机器人编程：原理与应用》。

关于 ROS 诸多优势在此不过多赘述，接下来补充介绍此书不同于已有图书的特点。我们认为，它是迄今为止已出版 ROS 图书中涉及领域最为系统全面的一本，除了系统讲解 ROS 内部的工作机制外，还着重介绍了移动机器人与机械臂

的基本原理，以及系统集成、高级控制等方面的应用，能够让读者充分了解 ROS 原理以及在机器人主要领域的实践应用。但也正如作者所言：ROS 涵盖了机器人研究的众多领域，每一个具体领域都需要专业的知识，甚至可以作为一个科研方向进行研究。希望本书能够帮助你为未来的研究与工作奠定良好的基础，找到自己真正感兴趣的方向，深入下去！

此书适合本科高年级学生、研究生和工程技术人员。此书内容系统且深入，附有大量示例代码，尤其适合需要自学的朋友。愿此书能够陪伴你开启美好的学习之旅！

最后感谢机械工业出版社华章公司的领导及张梦玲编辑对此书的大力支持！感谢易科机器人实验室的张瑞雷博士、林远山博士和吴中红博士审阅此书并提出宝贵的修改意见。感谢河海大学国防生机器人运动与视觉实验室的吕泽宇、乔睿哲和刘官明同学参与了文字审校工作。感谢华东师范大学机器人运动与视觉实验室负责人张新宇副教授的帮助与支持！

PREFACE
前 言

ROS（Robot Operating System，机器人操作系统）正在成为现代机器人学的实际标准编程方法。ROS wiki（https://www.ros.org/history/）写道：

ROS 生态系统现在由全世界数以万计的用户组成，覆盖了从桌面爱好项目到大型工业自动化系统。

为什么是 ROS？ 在 1956 年，Joseph Engelberger 创立了 Unimation 公司，世界上第一个机器人公司[7]。然而，在过去的半个世纪里，机器人技术的进步令人失望。世界范围内的机器人学研究也仅限于实验室里的演示和探奇。这一领域的新生研究人员通常一无所有，从头开始构建新型机器人，解决执行器和传感器接口的问题，构建底层的伺服控制，并且通常在实现更高级的机器人能力之前就已经精疲力竭了，而这些自定义的机器人和软件很少被复用于后续工作。

人们认识到采用构建巴比塔的模式是徒劳的，构建更智能的机器人的任务需要持续的、协作的努力，并建立在能够不断达到更高层能力的基础上。在 1984 年，Vincent Hayward 和 Richard Paul 引入了机器人控制 C 库（Robot Control C Library, RCCL）[15] 作为解决这一长期问题的方法。不幸的是，RCCL 没有获得机器人研究人员足够的认可。National Instruments[24] 和 Microsoft[39-40] 均引入了试图使机器人编程标准化的产品。然而，研究人员发现这些方法烦琐而昂贵。

ROS 于 2007 年由斯坦福人工智能实验室发起[26]，它试图统一零碎的谷歌所采用的机器人学方法，且于 2008 年至 2013 年得到 Willow Garage 的支持[12]，随后自 2013 年至今得到谷歌开源机器人基金会（Open Source Robotics Foundation, OSRF）的支持[10]。ROS 方法遵循了开源软件和分布式协作的现代方法。此外，它桥接和提升了其他并行的开源工作，包括 OpenCV[28]、PointCloudLibrary[21]、

Open Dynamics Engine[8]、Gazebo[19]和 Eigen[20]。对于研究人员而言，ROS 在开放性和易用性方面可能与 RCCL 相似，而谷歌持续七年的支持是 ROS 存活的关键。

ROS 现在在学术界、工业界和研究机构中得到了全世界的广泛使用。开发人员提供了数以千计的软件包，包括来自一些世界领先的专家在相关领域的解决方案。新的机器人公司在它们的产品上提供了 ROS 接口，并且已建立的工业机器人公司也引入了 ROS 接口。随着广泛采用 ROS 作为机器人编程实际标准的做法，人们对提高机器人的能力有了新的希望。在最近的 DARPA 机器人挑战赛中，大多数入围的团队使用了 ROS。新型自动驾驶汽车的开发商正在开发 ROS。新的机器人公司正在崛起，这部分是由 ROS 资源驱动的。鉴于 ROS 的势头和功绩，显而易见，当今的机器人工程师必须精通 ROS 编程。

什么是 ROS？将其称为“机器人操作系统”并不全面。简洁地定义 ROS 很困难，因为它包含了很多方面，包括：编程风格（特别是依赖于松散耦合的分布式节点），节点间通信的接口定义和范例，库和包合并的接口定义，可视化、调试、数据记录和系统诊断的工具集合，共享源代码的存储仓库，桥接多个有用的、独立的开源库的桥梁。因此，ROS 是机器人程序员的一种生活方式，而不只是一种简单的操作系统。ROS 的定义可以参考 ROS wiki（https://wiki.ros.org/ROS/Introduction）：

ROS 是一个针对机器人的开源、元级操作系统。它提供了用户在操作系统上所期望的服务，包括硬件抽象、低层设备控制、常用功能的实现、进程之间的消息传递以及包管理。它还提供了在多台计算机上获取、生成、编写以及运行代码的工具和库。

ROS 的主要目标是支持机器人研究和开发中的代码复用。ROS 是一个分布式的进程（也称节点）框架，它能使可执行的文件单独设计以及在运行时松散耦合。这些进程可以打包成易于共享和分发的包。ROS 还支持一个代码库的联合系统，能够同时分发协作。从文件系统级到社区级的这个设计实现了开发和部署的独立决策，但所有这些都可以与 ROS 的基础底层工具一起使用。

Brian Gerkey 在网上的评论（https://answers.ros.org/question/12230/what-is-ros-exactly-middleware-framework-operating-system/）如下。

我是这样解释 ROS 的：

1. 管道：ROS 提供了发布 - 订阅消息传递基础结构，旨在支持分布式计算系统的快速、轻松构建。

2. 工具：ROS 提供了一套广泛用于配置、启动、反思、调试、可视化、记录、测试和停止的分布式计算系统的工具。

3. 功能：聚焦于移动性、操作性和感知性，ROS 提供了实现机器人有用的功能的广泛库集。

4. 生态系统：ROS 拥有规模庞大的社区支持，并通过着力聚焦于集成和文档而不断改进。ros.org 是一个一站式的站点，在这里可以查找和了解来自世界各地开发者的成千上万个可用 ROS 包。

来自参考文献［13］对 ROS 的解释如下：

ROS（发音 Ross［rɔs］）的主要目标是提供一个统一的开源编程框架，用于在各种真实世界和仿真环境中控制机器人。

来自参考文献［13］中的 ROS 管道：

ROS 中的核心实体称为节点。节点通常是用 Python 或 C++ 编写的小程序，用于执行一些相对简单的任务或过程。节点可以相互独立地启动和停止，并通过传递消息进行通信。节点可以在某些主题上发布消息或向其他节点提供服务。

例如，发布器节点可能会报告从连接到机器人微控制器的传感器传来的数据。/head-sonar 主题上数值为 0.5 的消息意味着传感器当前检测的物体有 0.5 m 远。任何想从这个传感器知晓读数的节点只需订阅 /head-sonar 主题即可。为了使用这些值，订阅器节点定义了一个回调函数，每当新消息到达订阅的主题时它就会执行。这种情况发生的频率取决于发布器节点更新其消息的速率。

节点还可以定义一个或多个服务。当从另一个节点发送请求时，ROS 服务会产生某个行为或发送应答。一个简单的例子就是打开或关闭 LED 的服务。一个更复杂的例子是，当给定一个目标位置和机器人的初始位姿时，返回一个移动机器人导航规划的服务。

学习 ROS 的方法： ROS 有很多功能、工具、风格和习惯。ROS 的学习曲线陡峭，因为在富有成效地使用它之前需要掌握很多细节。ROS wiki 有文档和一系列教程的链接。然而，这些对于 ROS 的初学者而言可能很难遵循，因为定义是分散的，并且所呈现的细节水平千差万别，从未经说明的示例到面向复杂用户的解释。本书的目的是从简单的代码示例以及相应的操作理论层面向读者介绍 ROS 的基本组件。这种介绍只会触及表面，但应该能让读者开始建立有用的 ROS 节点，并使教程变得更可读。

ROS 代码可以用 C++ 或 Python 编写。本书仅使用 C++。对于 Python，读者可参考《ROS 机器人编程实践》(中文版已出版，ISBN：978-7-111-58529-9)[34]。

本书配套的代码示例假定采用 Linux Ubuntu 14.04 和 ROS Indigo。如果你使用 PC 运行 Windows 或使用 Mac 运行 OSX，则一个选择是安装 VirtualBox 来设置虚拟 Linux 计算机，以便在不影响原操作系统的情况下运行。关于安装 VirtualBox、Ubuntu、ROS 以及附带的代码示例和工具，包括在 https://github.com/wsnewman/learning_ros 的子目录 additional_documents 中。(该目录还包括使用 git 的入门指南。)

配置计算机来使用 ROS 可能是一个挑战。可参考《机器人操作系统（ROS）浅析》（中文版已出版）[27] 以进一步阐明和协助 ROS 的安装，并获得 ROS 组织和通信的更多细节和幕后解释。（关于 ROS 的其他书籍列于：https://wiki.ros.org/Books。）

ROS 安装的在线描述链接是：https://wiki.ros.org/indigo/Installation/Ubuntu。

安装 ROS 时，用户有命名 ROS 工作区的选择权。在本书中，假定该目录位于用户的主目录下，并命令为 ros_ws。如果你为 ROS 工作区选择另外一个名称，请将在本书中所有地方的 ros_ws 替换为该名称。

本书的代码示例可以在以下网址找到：https://github.com/wsnewman/learning_ros。与此代码一起使用的一些附加包位于：https://github.com/wsnewman/learning_ros_external_packages。应将两个软件库都复制到子目录 ~/ros_ws/src 中的用户 ROS 工作区，以便能够编译示例。要想手动安装这些软件，请在设置 ROS 环境后，cd 至 ~/ros_ws/src 并输入：

```
git clone https://github.com/wsnewman/learning_ros.git
```

和

```
git clone https://github.com/wsnewman/learning_ros_external_packages.git
```

这将使此处引用的所有示例代码都显示在这两个子目录中。

或者（推荐此方法），使用软件库 https://github.com/wsnewman/learning_ros_setup_scripts 中包含的自动安装 ROS 的脚本，来安装本书的示例代码、安装其他有用的工具以及设置 ROS 工作区。网站在线提供了获取和运行这些脚本的说明。这些说明适用于本地 Ubuntu-14.04 安装或 Ubuntu-14.04 的 VirtualBox 安装。（注意，当运行计算密集型或图形密集型代码时，VirtualBox 可能会卡顿。本机 Linux 安装和兼容的 GPU 处理器更可取。）

本书内容无法面面俱到。感兴趣的学生、研究者、自动化工程师或机器人爱好者可以自行探索数以千计的 ROS 包。此外，还有在线教程有更详细的细节和扩展内容。本书的目的是提供一个概览，使读者能够理解 ROS、ROS 包和 ROS 工具的组织，将现有 ROS 包纳入新的应用中，并开发新的包用于机器人和自动化系统。本书使读者能够更好地了解现有的在线文档以便进一步学习。

本书内容分为六部分：

- ROS 基础
- ROS 中的仿真和可视化
- ROS 中的感知处理
- ROS 中的移动机器人

- ROS 中的机械臂
- 系统集成与高级控制

每个主题都覆盖了广泛的领域，包含了大量专业研究成果。本书无法一一在这些领域指导读者。然而，机器人系统需要集成的元素横跨硬件集成、人机界面、控制理论、运动学和动力学、操作规划、运动规划、图像处理、场景解释和人工智能等一系列主题。机器人工程师必须是通才，因此至少需要了解这些领域的基本实践。ROS 生态系统的一个目的就是让工程师可以导入以上每个领域现有的包，并将它们集成到一个定制的系统中，而不必成为每个领域的专家。因此，了解每个领域的 ROS 接口和 ROS 方法对系统集成商来说极具价值，可以充分利用世界各地的机器人研究者、计算机科学家和软件工程师所贡献的专业知识。

致谢

在学术界，工作乐趣之一就是经常接触聪明且富有激情的学生们。感谢我的前顾问 Chad Rockey 和 Eric Perko，他们于 2010 年将我带入 ROS 的大门。从此，我从一个 ROS 质疑者变成了传播者。感谢这一路相伴的学生们，包括 Tony Yanick、Bill Kulp、Ed Venator、Der-Lin Chow、Devin Schwab、Neal Aungst、Tom Shkurti、TJ Pech 和 Luc Bettaieb，他们帮我实现了转变并学习了新的 ROS 技巧。

感谢 Sethu Vijayakumar 教授和苏格兰信息学与计算机科学联盟，感谢他们对我在爱丁堡大学开设 ROS 课程和本书基础课程时给予的支持。感谢爱丁堡大学的 Chris Swetenham、Vladimir Ivan 和 Michael Camilleri，我们在 DARPA 机器人挑战赛中一起开展 ROS 编程合作。在这个过程中，他们教会了我很多额外的 ROS 编程技巧。

感谢 Hung Hing Ying 家庭的支持，他们的基金会使得我成为香港大学的 Hung Hing Ying 客座教授，期间与香港大学 DARPA 机器人挑战赛团队一起组织并开展工作。这是一次宝贵的实践 ROS 的经历。感谢东京大学的 Kei Okada 及其学生对我们港大团队所做的贡献，包括 ROS 使用的宝贵意见和技巧。

感谢 Taylor and Francis 的资深策划编辑 Randi Cohen，她鼓励并指导了本书的出版。感谢为本书提出了宝贵建议的审稿者，他们是 NASA Goddard 空间飞行中心和马里兰大学的 Craig Carignan 博士和广东工业大学的 Juan Rojas 教授。

最后，感谢我的妻子 Peggy Gallagher、女儿 Clea 和 Alair Newman 的支持，以及不断的鼓励和帮助。

感谢谷歌和开源机器人基金、许多创建了有价值的 ROS 包和在线教程以及回答大量 ROS 问题的在线贡献者，正是有了他们的支持，ROS 才能成功。

作者简介

Wyatt S. Newman 是凯斯西储大学电气工程和计算机科学系的教授，自 1988 年开始执教。他的研究领域是机电一体化、机器人学和计算智能，拥有 12 项专利并发表了超过 150 篇学术出版物。他在哈佛大学获得了工程科学专业的学士学位，在麻省理工学院热流体科学系获得了机械工程专业的硕士学位，在哥伦比亚大学获得了控制理论和网络理论专业的电机工程理学硕士学位，在麻省理工学院设计与控制系获得了机械工程专业的博士学位。他是机器人学方面的 NSF 青年研究员，担任过以下职务：飞利浦实验室高级研究员、飞利浦 Natuurkundig 实验室的访问科学家、美国桑迪亚国家实验室智能系统和机器人中心的访问学者、NASA 格伦研究中心的 NASA 夏季教员、普林斯顿大学神经科学的访问学者、爱丁堡大学信息学院的杰出访问学者、香港大学的 Hung Hing Ying 杰出客座教授。他带领机器人团队参加了 2007 年 DARPA 城市挑战赛和 2015 年 DARPA 机器人挑战赛，并将继续致力于机器人的广泛应用。

CONTENTS

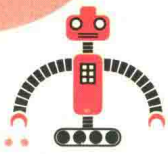
目 录

PART1

第一部分

ROS 基础

第一部分从基本概念、工具和结构开始，介绍机器人操作系统（Robot Operating System，ROS）的基础知识。演示ROS包、节点和工具的创建和使用。介绍ROS通信的基础，包括发布器和订阅器、服务和客户端、动作服务器和动作客户端、参数服务器。这些内容均是ROS的基础，也是讨论机器人学技术细节的前导知识。



CHAPTER1

第1章

概述：ROS 工具和节点

引言

这是介绍性的一章，将从最简单的例子开始，主要讲解 ROS 中节点的概念，以及一些 ROS 工具，以说明 ROS 节点的行为。在这里将会用到最简单的 ROS 通信方法——发布和订阅，可供选择的通信方法（service 和 actionserver）将会在第 2 章讲解。

1.1 ROS 基础概念

节点（node）间的通信是 ROS 的核心。节点是一种使用 ROS 的中间件进行通信的 ROS 程序。一个节点可以独立于其他节点启动，多个节点也可以以任何顺序启动。多个节点可以运行在同一台计算机中，或者分布于一个计算机网络中。但一个节点只有当与其他节点通信并最终与传感器和执行器连接时，它才是有用的。

节点之间的通信涉及了 message、topic、roscore、publisher 和 subscriber（service 同样有用，而且这些与发布器和订阅器紧密相关）。节点间所有通信都是序列化的网络通信。publisher 发布的 message 是一个数据包，需要查询对应的关键字进行解析。因为每条消息都以比特流形式接收，那么就需要查询关键字（消息类型描述）来知道怎么解析这些比特，并重构出对应的数据结构。一个简单的例子就是 Float64，它的定义在 ROS 内置程序包 std_msgs 中。此种消息类型帮助发布器把浮点数封装到所定义的 64 位比特流中，而且它同样也帮助订阅器把这些比特流解析为浮点数表示。

关于消息（message），更复杂一点的例子是 twist，它由描述 3 维平移和旋转速度的多个字段组成。有些消息还提供了可选的附加功能，比如时间戳和消息标识码。

当发布器（publisher）节点发布数据时，任何感兴趣的订阅器（subscriber）节点都可以使用它。订阅器节点必须能够和已发布的数据相连，发布的数据通常来自不同的节点，

原因是软件的开发会导致发布器发生变化，或者一些发布器节点在不同的环境中会与不同的节点联系。比如，负责控制关节速度的发布器节点可能是一个刚性的位置控制器，但在其他场景中则需要一个柔顺力控制器。通过改变发布速度命令的节点，便可实现交接。这也暴露了一个问题，即订阅器无法知道它的输入正由谁发布。但事实上，要知道哪个节点正在发布的需求会使大型系统的结构变得复杂化，这个问题由 topic 这个概念予以解决。

通过引入话题（topic），不同的发布器可以轮流发布至该话题。因此，一个订阅器只需要知道话题名，而不需要知道是哪个节点或哪些节点向该话题发布消息（message）。比如，用于控制速度的话题是 vel_cmd，然后机器人的底层控制器就会订阅该话题以接收速度命令。不同的发布器都可能负责向这个话题发布速度控制消息，不管是处于实验状态的节点还是随后替换进来的用于满足特定任务需求的可信节点。

尽管通过创建 topic 这一抽象概念可以起到一些作用，但发布器和订阅器都需要知道如何通过话题进行通信，这可以通过 ROS 中的通信中间件所提供的可执行节点 roscore 实现。roscore 像一个操作员一样来负责调整通信。尽管会有很多 ROS 节点分布在多网络计算机中，但只能有一个 roscore 实例可以运行，而运行 roscore 的机器则是这个系统的主计算机。

一个发布器节点通过通知 roscore 一个话题名（和相应的消息类型）来初始化该话题，这称为 advertising 话题。为了实现它，发布器实例化了类 ros::Publisher 的一个对象。这个类的定义属于 ROS 发行版的一部分，使用发布器对象避免了设计者自己编写通信代码。在实例化了发布器对象后，用户代码调用了成员函数 advertise，指定了消息类型，并声明了所需的话题名。此时，用户代码可以开始使用发布器成员函数 publish（它作为消息发布的一个参数），向已命名话题发送消息。

因为每个发布器节点都要和 roscore 通信，roscore 必须在所有 ROS 节点启动前就开始运行。要运行 roscore，打开 Linux 中的一个终端并输入 roscore。该指令的响应是一个确认："已启动核心服务"。它也会输出 ROS_MASTER_URI，它对通知运行在非主计算机上的节点如何连接 roscore 很有用。运行 roscore 的终端只能运行 roscore，无法执行其他任务。只要机器人仍处于有效控制状态（或希望接入机器人的传感器），roscore 节点就应持续运行。

在 roscore 已经启动后，就应该启动发布器节点了。发布器节点会发布它的话题，也可能会开始发送消息（以节点希望的速率发送，但频率受到系统性能的限制）。对于底层控制器和传感器数据，以 1kHz 的速率发布消息是很常见的。

把一个传感器接入一个 ROS 系统需要特定的代码（也许还需要特定的硬件）来与这个传感器通信。例如，LIDAR（激光雷达）传感器需要 RS488 通信，需要以特定格式接收指令，并以预定义格式开始流数据。必须用一个专用的微控制器（或主计算机内的一个节点）与 LIDAR 通信，接收数据，然后在一个 ROS 话题上用某种 ROS 消息类型发布数据。这些特定节点把每个特殊的传感器数据转化为通用的 ROS 通信格式。