

具有无线充电的变电站巡检机器人操作系统

目录

一、 机器人操作系统 ROS	1
1.1 ROS 简介	1
1.2 元操作系统	1
1.3 ROS 的组件	2
二、 搭建 ROS 开发环境	2
2.1 安装 ROS	2
2.2 搭建 ROS 开发环境	5
2.2.1 ROS 配置	5
2.2.2 加载 ROS 配置	5
2.2.3 配置 ROS 网络	5
2.2.4 集成开发环境 (IDE)	6
三、 ROS 的重要概念	8
3.1 ROS 术语	8
3.2 消息通信和消息	10
3.2.1 话题 (topic)	11
3.2.2 服务 (service)	12
3.2.3 动作 (action)	12
3.3 名称	13
3.4 客户端库	13
3.5 异构设备间的通信	13
3.6 文件系统	14
3.6.1 文件组织结构	14
3.6.2 安装目录	14
3.6.3 工作目录	15
3.7 构建系统	15
四、 ROS 命令	16
4.1 ROS 命令概述	16
4.2 ROS shell 命令	16
4.3 ROS 执行命令	16
4.4 ROS 信息命令	16
五、 ROS 工具	16
5.1 三维可视化工具	17
5.2 ROS GUI 开发工具	17
六、 机器人、传感器和电机	17
6.1 机器人功能包	17
6.2 传感器功能包	17
6.3 相机	17
6.4 激光距离传感器	18
七、 结束语	18

一、机器人操作系统 ROS

1.1 ROS 简介

ROS 是一个开放源代码的机器人元操作系统。它提供了我们对操作系统期望的服务，包括硬件抽象、低级设备控制、常用功能的实现、进程之间的消息传递以及功能包管理。它还提供了用于在多台计算机之间获取、构建、编写和运行代码的工具和库。

换句话说，ROS 包括一个类似于操作系统的硬件抽象，但它不是一个传统的操作系统，它具有可用于异构硬件的特性。此外，它是一个机器人软件平台，提供了专门为机器人开发应用程序的各种开发环境。

1.2 元操作系统

ROS 是一个元操作系统（Meta-Operating System）。元操作系统是一个利用应用程序和分布式计算资源之间的虚拟化层来运用分布式计算资源来执行调度、加载、监视、错误处理等任务的系统。

ROS 不是传统的操作系统，如 Windows、Linux 和 Android，反而是在利用现有的操作系统。使用 ROS 前需要先安装诸如 Ubuntu 的 Linux 发行版操作系统，之后再安装 ROS，以使用进程管理系统、文件系统、用户界面、程序实用程序（编译器、线程模型等）。此外，它还以库的形式提供了机器人应用程序所需的多数不同类型的硬件之间的数据传输/接收、调度和错误处理等功能。这个概念也被称为中间件（Middleware）或软件框架（Software framework）。

ROS 开发、管理和提供基于元操作系统的各种用途的应用功能包，并拥有一个负责分享用户所开发的功能包的生态系统（Ecosystem）。如图 1-1 所示。ROS 是在使用现有的传统操作系统的同时，通过使用硬件抽象概念来控制机器人应用程序所必需的机器人和传感器，同时也是开发用户的机器人应用程序的支持系统。



图 1.1 元操作系统的 ROS

ROS 数据通信可以在一个操作系统中进行，但也适用于使用多种硬件的机器人开发，因为可以在不同的操作系统、硬件和程序之间交换数据。

1.3 ROS 的组件

ROS 由支持多种编程语言的客户端库、用于控制硬件的硬件接口、数据通信通道、帮助编写各种机器人应用程序的机器人应用框架（Robotics Application Framework）、基于此框架的服务应用程序 Robotics Application、在虚拟空间中控制机器人的仿真（Simulation）工具和软件开发工具（Software Development Tool）等组成。ROS 组件如下图 1.2 所示。



图 1.2 ROS 组件

二、搭建 ROS 开发环境

2.1 安装 ROS

1) 设置网络时间协议 (NTP, Network Time Protocol)

为了缩小 PC 间通信中的 ROS Time 的误差，我们需要设置 NTP。设置方法是安装 chrony 之后用 ntpdate 命令指定 ntp 服务器即可。这样一来会表示服务器和当前计算机之间的时间误差，进而会调到服务器的时间。

```
$ sudo apt-get install -y chrony ntpdate
```

```
$ sudo ntpdate -q ntp.ubuntu.com
```

2) 添加代码列表

在 ros-latest.list 添加 ROS 版本库。打开新的终端窗口，输入如下命令：

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

3) 设置公钥 (Key)

为了从 ROS 存储库下载功能包，下面添加公钥。作为参考，以下公钥可以根据服务器的操作发生变更。

```
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key
421C365BD9FF1F717815A389552 3BAEEB01FA116
```

4) 更新软件包索引

```
$ sudo apt-get update && sudo apt-get upgrade -y
```

5) 安装 ROS Kinetic Kame

使用以下命令安装台式机的 ROS 功能包。这包括 ROS、rqt、RViz、机器人相关的库、仿真和导航等等。

```
$ sudo apt-get install ros-kinetic-desktop-full
```

上面的安装只包含基本的 rqt，需要安装所有额外的 rqt 相关的功能包。用下面的命令安装所有 rqt 相关的功能包，可以很方便地使用各种 rqt 插件。

```
$ sudo apt-get install ros-kinetic-rqt*
```

6) 初始化 rosdep

在使用 ROS 之前，必须初始化 rosdep。rosdep 是一个通过在使用或编译 ros 的核心组件时轻松安装依赖包来增强用户便利的功能。

```
$ sudo rosdep init
```

```
$ rosdep update
```

7) 安装 rosinstall

这是安装 ROS 的各种功能包的程序。它是被频繁使用的有用的工具，因此务必要安装它。

```
$ sudo apt-get install python-roinstall
```

8) 加载环境设置文件

下面加载环境设置文件。里面定义着 ROS_ROOT 和 ROS_PACKAGE_PATH 等环境变量。

```
$ source /opt/ros/kinetic/setup.bash
```

9) 创建并初始化工作目录

ROS 使用一个名为 catkin 的 ROS 专用构建系统。为了使用它，用户需要创建并初始化 catkin 工作目录，如下所示。除非用户创建新的工作目录，否则此设置只需设置一次。

```
$ mkdir -p ~/catkin_ws/src
```

```
$ cd ~/catkin_ws/src
```

```
$ catkin_init_workspace
```

如果用户已经创建了一个 catkin 工作目录，下面我们来进行构建。目前，只有 src 目录和 CMakeLists.txt 文件在 catkin 工作目录中，可以使用 catkin_make 命令来构建。

```
$ cd ~/catkin_ws/
```

```
$ catkin_make
```

当用户构建没有问题时，运行 ls 命令。除了自己创建的 src 目录之外，还出现了一个新的 build 和 devel 目录。catkin 的构建系统的相关文件保存在 build 目录中，构建后的可执行文件保存在 devel 目录中。

最后，我们加载与 catkin 构建系统相关的环境文件。

```
$ source ~/catkin_ws/devel/setup.bash
```

10) 测试安装结果

所有 ROS 安装已完成。要测试它是否正确安装，请关闭所有终端窗口并运行一个新的终端窗口。现在通过输入以下命令来运行 roscore。

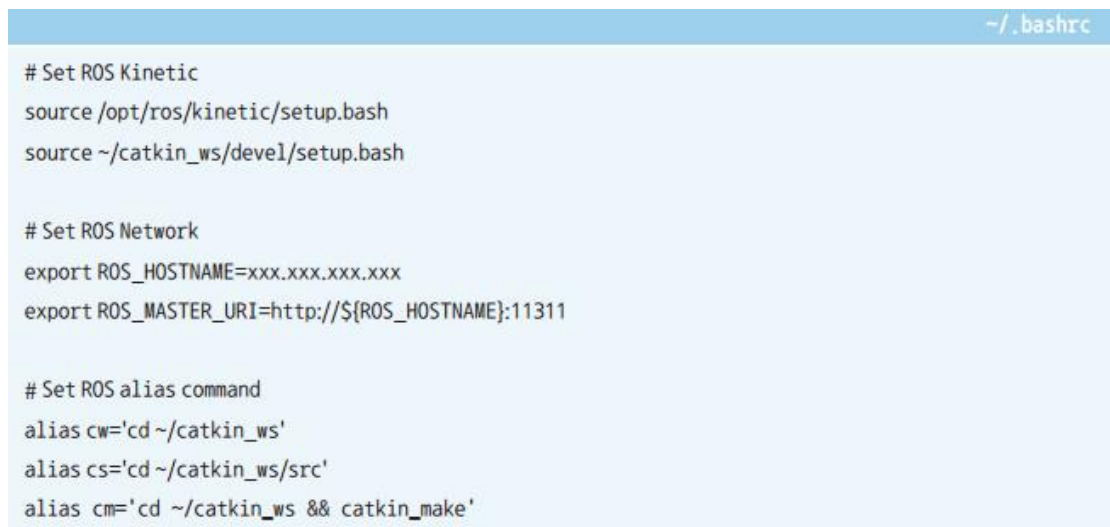
```
$ roscore
```

2.2 搭建 ROS 开发环境

2.2.1 ROS 配置

为了避免每次打开终端都需要运行加载配置文件，可以设置终端，使得每次打开新的终端窗口时，都读入配置文件。另外，配置 ROS 网络，还将常用的命令简化为快捷命令。\$ gedit ~/.bashrc

打开 bashrc 文件就可以看到已经有了很多设置。不要修改以前的设置，而是到 bashrc 文件的最底部添加以下内容。输入了所有内容之后，保存用户的更改并退出 gedit。更改后的 bashrc 如下图 2.1 所示。



```
~/.bashrc

# Set ROS Kinetic
source /opt/ros/kinetic/setup.bash
source ~/catkin_ws/devel/setup.bash

# Set ROS Network
export ROS_HOSTNAME=xxx.xxx.xxx.xxx
export ROS_MASTER_URI=http://${ROS_HOSTNAME}:11311

# Set ROS alias command
alias cw='cd ~/catkin_ws'
alias cs='cd ~/catkin_ws/src'
alias cm='cd ~/catkin_ws && catkin_make'
```

图 2.1 bashrc 文件

为了让修改了的 bashrc 文件发挥作用，输入如下命令。或者，如果用户关闭当前正在运行的终端窗口并运行新的终端窗口，用户也将得到相同的效果，因为用户在 bashrc 中所做的设置会适用于新的终端窗口。\$ source ~/.bashrc

2.2.2 加载 ROS 配置

```
# Set ROS Kinetic

source /opt/ros/kinetic/setup.bash

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

2.2.3 配置 ROS 网络

下面是 ROS_MASTER_URI 和 ROS_HOSTNAME 设置。此配置非常重要，因为 ROS 通过网络在节点之间传递消息。首先，两个项目必须输入自己的网络 IP。将来，

如果有专用于总机（MASTER PC）的 PC，并且机器人使用主机（HOST PC），则可以通过分别输入不同的 IP 地址进行通信。现在让我们输入他们的网络 IP。以下示例显示 IP 为 192.168.1.100 的情况的示例。用户可以在终端窗口中使用 ifconfig 命令检查用户的 IP 信息。

```
# Set ROS Network
```

```
export ROS_HOSTNAME=192.168.1.100
```

```
export ROS_MASTER_URI=http://{ROS_HOSTNAME}:11311
```

如果用户在一台 PC 上运行所有 ROS 功能包，则可以指定 localhost 而不是指定特定的 IP。

```
# Set ROS Network
```

```
export ROS_HOSTNAME=localhost
```

```
export ROS_MASTER_URI=http://localhost:11311
```

2.2.4 集成开发环境（IDE）

集成开发环境（IDE）是一种软件，在这个软件中完成与程序开发相关的所有任务：如编写代码、调试、编译和发布等。以下是使用 QtCreator 的 ROS 开发环境的描述。

1) 安装 QtCreator

```
$ sudo apt-get install qtcreator
```

2) 运行 QtCreator

以双击 QtCreator 图标的方式运行 QtCreator 也不影响其运行。但为了将记录到 ~/.bashrc 的 ROS 路径等设置应用于 QtCreator，需要打开新的终端窗口，并运行如下命令。这样才能用到 ~/.bashrc 里的配置。

```
$ qtcreator
```

3) 加载 ROS 功能包

如前所述，QtCreator 使用 CMakeLists.txt，而 ROS 功能包也基于 CMakeLists.txt，如图 2.2 所示，点击 OpenProject 键，选择相应的 ROS 功能包的 CMakeLists.txt，可以方便地打开工程。可以利用构建快捷库 Ctrl+b 运行 catkin_make。但是，与构建相关的文件是在与功能包相同的位置的新目录中创建的。例如，当用户编译 tms_rp_action 功能包时，所有与构建相关的文件都被

放置在 build-tms_rp_action-Desktop-Default 目录中。最初，要保存在 \sim / \sim /catkin_ws/build 和 \sim /catkin_ws/devel 中的文件是被分开编译的，并放置在一个新的位置，所以用户需要在终端窗口中再次执行 catkin_make。并非每次都需要反复这个过程，在开发过程中，在 QtCreator 开发和调试之后在运行时分别执行 catkin_make 即可。QtCreator 工程编程界面如图 2.3 所示。

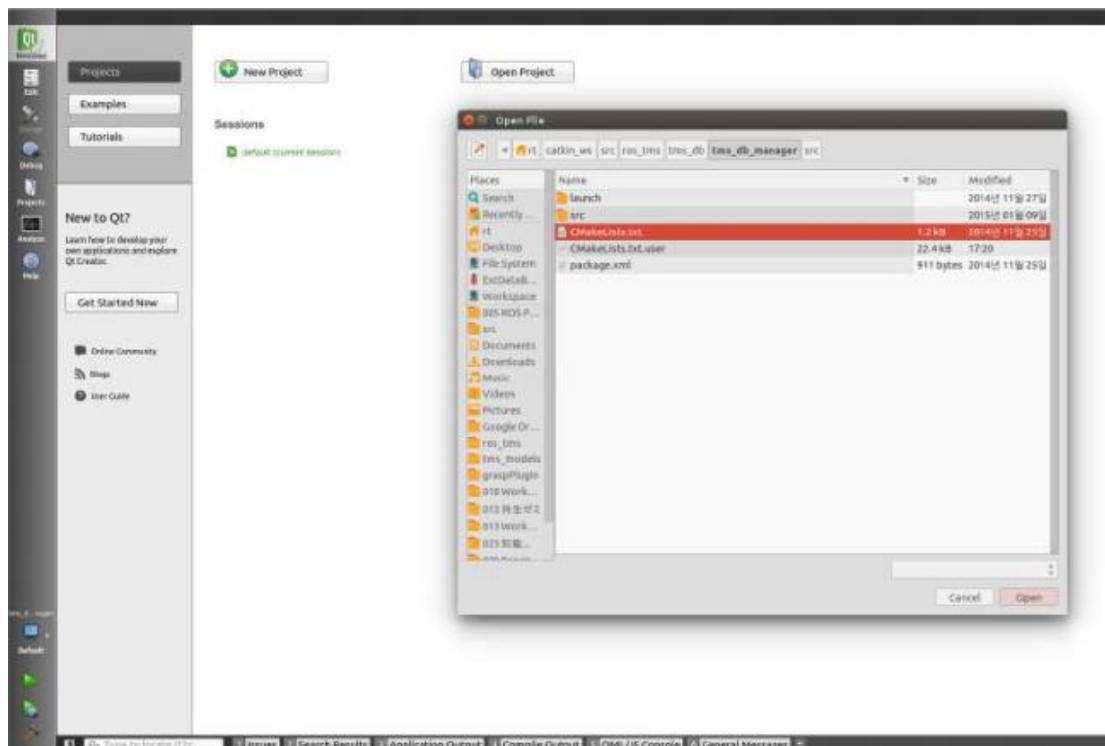


图 2.2 QtCreator 工程打开界面

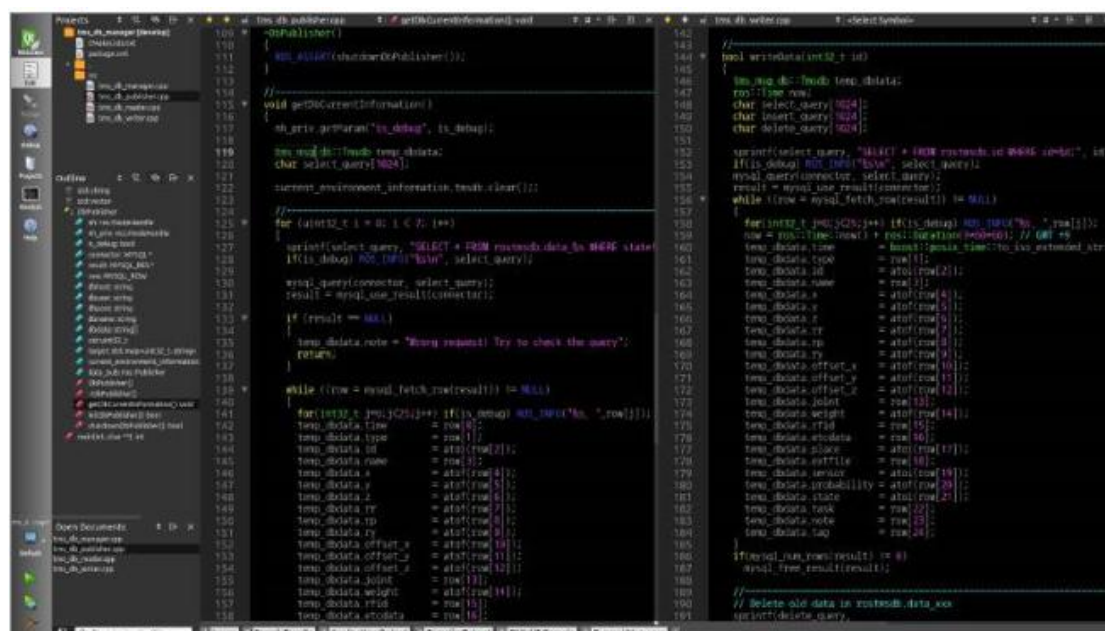


图 2.3 QtCreator 工程编程界面

三、ROS 的重要概念

3.1 ROS 术语

1) 节点

节点 (node) 是指在 ROS 中运行的最小处理器单元。可以把它看作一个可执行程序。在 ROS 中, 建议为一个目的创建一个节点, 建议设计时注重可重用性。节点在运行的同时, 向主节点注册节点的名称, 并且还注册发布者 (publisher)、订阅者 (subscriber)、服务服务器 (service server)、服务客户端 (service client) 的名称, 且注册消息形式、URI 地址和端口。基于这些信息, 每个节点可以使用话题和服务与其他节点交换消息。

节点使用 XMLRPC 与主站进行通信, 并使用 TCP/IP 通信系列的 XMLRPC 或 TCPRoS5 进行节点之间的通信。节点之间的连接请求和响应使用 XMLRPC, 而消息通信使用 TCPRoS, 因为它是节点和节点之间的直接通信, 与主节点无关。URI 地址和端口则使用存储于运行当前节点的计算机上的名为 ROS_HOSTNAME 的环境变量作为 URI 地址, 并将端口设置为任意的固有值。

2) 主节点

主节点 (master) 负责节点到节点的连接和消息通信, 类似于名称服务器 (Name Server)。roscore 是它的运行命令, 当运行主节点时, 可以注册每个节点的名字, 并根据需要获取信息。没有主节点, 就不能在节点之间建立访问和消息交流 (如话题和服务)。主节点使用 XML 远程过程调用 (XMLRPC, XML-Remote Procedure Call) 与节点进行通信。XMLRPC 是一种基于 HTTP 的协议, 主节点不与连接到主节点的节点保持连接。换句话说, 节点只有在需要注册自己的信息或向其他节点发送请求信息时才能访问主节点并获取信息。通常情况下, 不检查彼此的连接状态。由于这些特点, ROS 可用于非常大而复杂的环境。XMLRPC 也非常轻便, 支持多种编程语言, 使其非常适合支持各种硬件和语言的 ROS。

当启动 ROS 时, 主节点将获取用户设置的 ROS_MASTER_URI 变量中列出的 URI 地址和端口。除非另外设置, 默认情况下, URI 地址使用当前的本地 IP, 端口使用 11311。

3) 功能包

功能包 (package) 是构成 ROS 的基本单元。ROS 应用程序是以功能包为单位开发的。功能包包括至少一个以上的节点或拥有用于运行其他功能包的节点的

配置文件。它还包含功能包所需的所有文件，如用于运行各种进程的 ROS 依赖库、数据集和配置文件等。

4) 元功能包

元功能包 (metapackage) 是一个具有共同目的的功能包的集合。例如，导航元功能包包含 AMCL、DWA、EKF 和 map_server 等 10 余个功能包。

5) 消息

节点之间通过消息 (message) 来发送和接收数据。消息是诸如 integer、floating point 和 boolean 等类型的变量。用户还可以使用诸如消息里包括消息的简单数据结构或列举消息的消息数组的结构。使用消息的通信方法包括 TCPROS, UDPROS 等，根据情况使用单向消息发送/接收方式的话题 (topic) 和双向消息请求 (request) /响应 (response) 方式的服务 (service)。

6) 话题

话题 (topic) 就是“故事”。在发布者 (publisher) 节点关于故事向主节点注册之后，它以消息形式发布关于该故事的广告。希望接收该故事的订阅者 (subscriber) 节点获得在主节点中以这个话题注册的那个发布者节点的信息。基于这个信息，订阅者节点直接连接到发布者节点，用话题发送和接收消息。

7) 发布与发布者

发布 (publish) 是指以与话题的内容对应的消息的形式发送数据。为了执行发布，发布者 (publisher) 节点在主节点上注册自己的话题等多种信息，并向希望订阅的订阅者节点发送消息。发布者在节点中声明自己是执行发布的个体。单个节点可以成为多个发布者。

8) 订阅与订阅者

订阅是指以与话题内容对应的消息的形式接收数据。为了执行订阅，订阅者节点在主节点上注册自己的话题等多种信息，并从主节点接收那些发布此节点要订阅的话题的发布者节点的信息。基于这个信息，订阅者节点直接联系发布者节点来接收消息。订阅者在节点中声明自己执行订阅的个体。单个节点可以成为多个订阅者。

发布和订阅概念中的话题是异步的，这是一种根据需要发送和接收数据的好方法。另外，由于它通过一次的连接，发送和接收连续的消息，所以它经常被用于必须连续发送消息的传感器数据。然而，在某些情况下，需要一种共同使用请求和响应的同步消息交换方案。因此，ROS 提供叫做服务 (service) 的消息同步

方法。服务分为响应请求的服务服务器和请求后接收响应的服务客户端。与话题不同，服务是一次性的消息通信。当服务的请求和响应完成时，两个节点的连接被断开。

9) 服务

服务 (service) 消息通信是服务客户端 (service client) 与服务服务器 (service server) 之间的同步双向消息通信。其中服务客户端请求对应于特定目的任务的服务，而服务服务器则负责服务响应。

10) 动作

动作 (action) 是在需要像服务那样的双向请求的情况下使用的消息通信方式，不同点是在处理请求之后需要很长的响应，并且需要中途反馈值。动作文件也非常类似于服务，目标 (goal) 和结果 (result) 对应于请求和响应。此外，还添加了对应于中途的反馈 (feedback)。它由一个设置动作目标 (goal) 的动作客户端 (action client) 和一个动作服务器 (action server)，动作服务器根据目标执行动作，并发送反馈和结果。

3.2 消息通信和消息

为了最大化用户的可重用性，ROS 是以节点的形式开发的，而节点是根据其目的细分的可执行程序的最小单位。节点则通过消息 (message) 与其他的节点交换数据，最终成为一个大型的程序。这里的关键概念是节点之间的消息通信，它分为三种。单向消息发送/接收方式的话题 (topic)；双向消息请求/响应方式的服务 (service)；双向消息目标 (goal) /结果 (result) /反馈 (feedback) 方式的动作 (action)。另外，节点中使用的参数可以从外部进行修改。这在大的框架中也可以被看作消息通信。消息通信如图 3.1 所示。

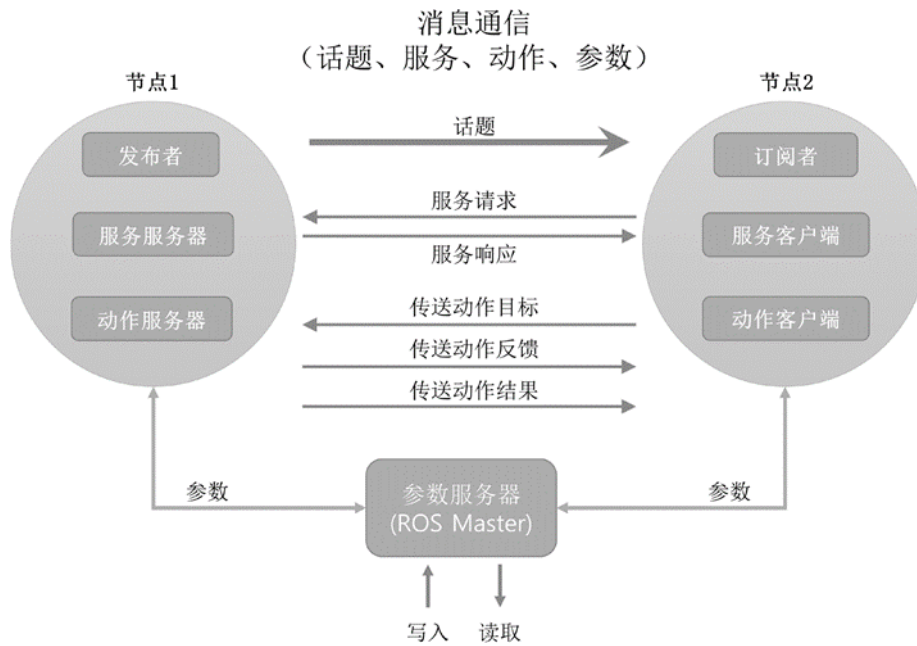
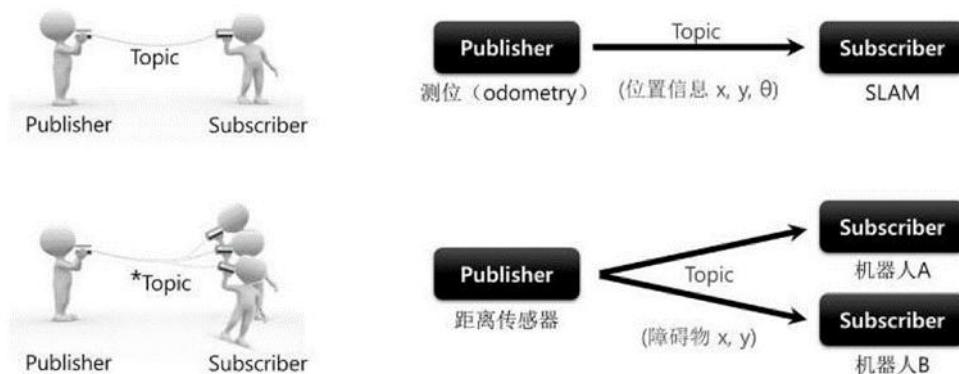


图 3.1 节点间的消息通信

3.2.1 话题 (topic)

如图 3.2 所示，话题消息通信是指发送信息的发布者和接收信息的订阅者以话题消息的形式发送和接收信息。希望接收话题的订阅者节点接收的是与在主节点中注册的话题名称对应的发布者节点的信息。基于这个信息，订阅者节点直接连接到发布者节点来发送和接收消息。例如，通过计算移动机器人的两个车轮的编码器值生成可以描述机器人当前位置的测位 (odometry) 信息，并以话题信息 (x, y, i) 传达，以此实现异步单向的连续消息传输。话题是单向的，适用于需要连续发送消息的传感器数据，因为它们通过一次连接连续发送和接收消息。另外，单个发布者可以与多个订阅者进行通信，相反，一个订阅者可以在单个话题上与多个发布者进行通信。当然，这两家发布者都可以和多个订阅者进行通信。



*利用Topic可以实现1:1的Publisher、Subscriber通信，也可以根据目的实现1:N、N:1和N:N通信。

图 3.2 话题消息通信

3.2.2 服务 (service)

服务消息通信是指请求服务的服务客户端与负责服务响应的服务服务器之间的同步双向服务消息通信，如图 3.3 所示。前述的发布和订阅概念的话题通信方法是一种异步方法，是根据需要传输和接收给定数据的一种非常好的方法。然而，在某些情况下，需要一种同时使用请求和响应的同步消息交换方案。因此，ROS 提供叫做服务的消息同步方法。

一个服务被分成服务服务器和服务客户端，其中服务服务器只在有请求 (request) 的时候才响应 (response)，而服务客户端会在发送请求后接收响应。与话题不同，服务是一次性消息通信。因此，当服务的请求和响应完成时，两个连接的节点将被断开。该服务通常被用作请求机器人执行特定操作时使用的命令，或者用于根据特定条件需要产生事件的节点。由于它是一次性的通信方式，又因为它在网络上的负载很小，所以它也被用作代替话题的手段，因此是一种非常有用的通信手段。例如，如在图 3.3 中，当客户端向服务器请求当前时间时，服务器将确认时间并作出响应。

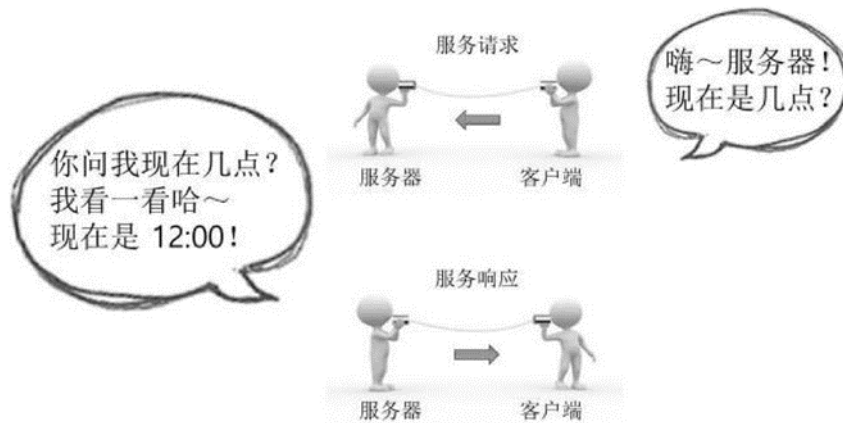


图 3.3 服务消息通信

3.2.3 动作 (action)

动作消息通信是在如下情况使用的消息通信方式：服务器收到请求后直到响应所需的时间较长，且需要中途反馈值。这与服务非常相似，服务具有与请求和响应分别对应的目标 (goal) 和结果 (result)。除此之外动作中还多了反馈 (feedback)。收到请求后需要很长时间才能响应，又需要中间值时，使用这个反

反馈发送相关的数据。消息传输方案本身与异步方式的话题（topic）相同。反馈在动作客户端（action client）和动作服务器（action server）之间执行异步双向消息通信，其中动作客户端设置动作目标（goal），而动作服务器根据目标执行指定的工作，并将动作反馈和动作结果发送给动作客户端。

3.3 名称

ROS 有一个称为图（graph）的抽象数据类型作为其基本概念。这显示了每个节点的连接关系以及通过箭头表达发送和接收消息（数据）的关系。为此，服务中使用的节点、话题、消息以及 ROS 中使用的参数都具有唯一的名称（name）。话题名称分为相对的方法、全局方法和私有方法

3.4 客户端库

由于 ROS 需要支持具有多重目的特点的机器人，因此 ROS 提供面向多种语言的便利接口。具体来说，各个节点可以用各自的语言来编写，而节点间通过消息通信交换信息。其中，允许用各自的语言编写的软件模块就是客户端库（client library）。常用且具有代表性的库有支持 C++ 的 roscpp、支持 Python 的 rospy、支持 LISP 的 roslisp、支持 Java 的 rosjava。此外还有 roscs、roseus、rosgo、roshask、roscpp、roslisp、rosnodejs、RobotOS.jl、roslua、PhaROS、rosR、rosruby 和 Unreal-Ros-Plugin 等。不仅如此，各个客户端库此时此刻也在为丰富 ROS 的语言多样性而被开发和改进中。

3.5 异构设备间的通信

ROS 基本上支持异构设备间的通信如图 3.4 所示。节点间的通信不受各节点所在的 ROS 底层的操作系统的种类的影响，也不受编程语言的影响，因此可以很容易地进行通信。例如，即使机器人安装了 Linux 的发行版之一的 Ubuntu，开发者可以在 MacOS 上监测机器人的状态，同时，用户可以通过基于 Android 的应用程序（APP）来向机器人发送指令。

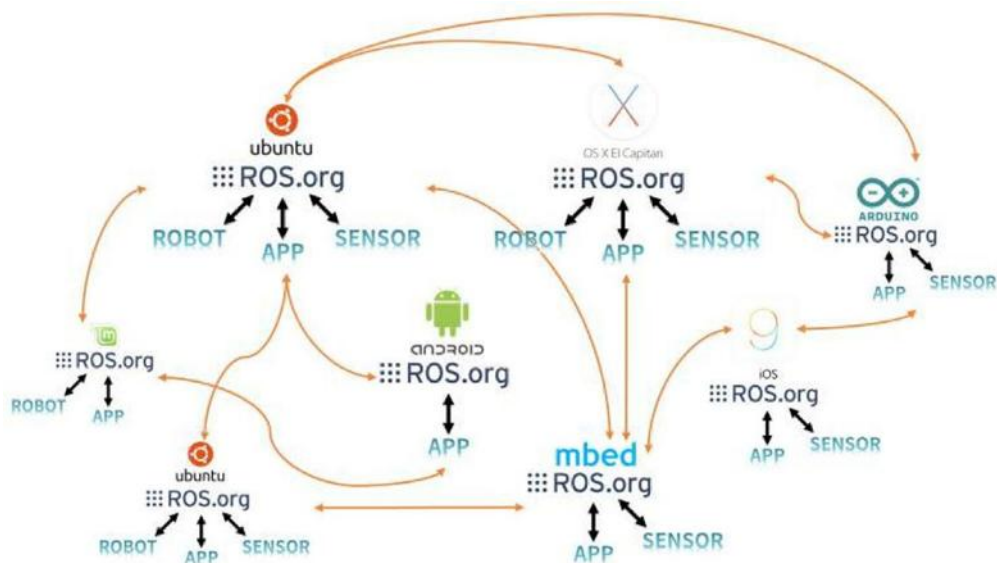


图 3.4 异构设备间的通信

3.6 文件系统

3.6.1 文件组织结构

在 ROS 中，组成软件的基本单位是功能包（package），因此 ROS 应用程序是以功能包为单位开发的。功能包包含一个以上的节点或包含用于运行其他节点的配置文件。每个功能包都包含一个名为 `package.xml` 的文件，该文件是一个包含功能包信息的 XML 文件，包括其名称，作者，许可证和依赖包。此外，ROS 构建系统 catkin 基本上使用 CMake，并在功能包目录中的 `CMakeLists.txt` 文件中描述构建环境。另外，它由节点的 源代码和消息文件组成，用于节点之间的消息通信。

ROS 的文件系统分为安装目录和用户工作目录。安装 ROS desktop 版本后，在 `/opt` 目录中会自动生成名为 `ros` 的安装目录，里面会安装有 `roscore`、`rqt`、`RViz`、机器人相关库、仿真和导航等核心实用程序。用户很少需要修改这个区域的文件。如果要修改以二进制文件形式分发的功能包，请找到包含源代码的功能包存储库之后利用在“`~/catkin_ws/src`”目录下用“`git clone [存储库地址]`”命令直接复制源代码，而不是用“`sudo apt-get install ros-kinetic-xxx`”形式的功能包安装命令。

用户的工作目录可以在用户想要的位置创建。

3.6.2 安装目录

ROS 安装在“/opt/ros/[版本名称]”目录中。文件组织结构如图 3.5 所示，“/opt/ros/kinetic”目录下包含 bin、etc、include、lib、share 目录和一些配置文件。

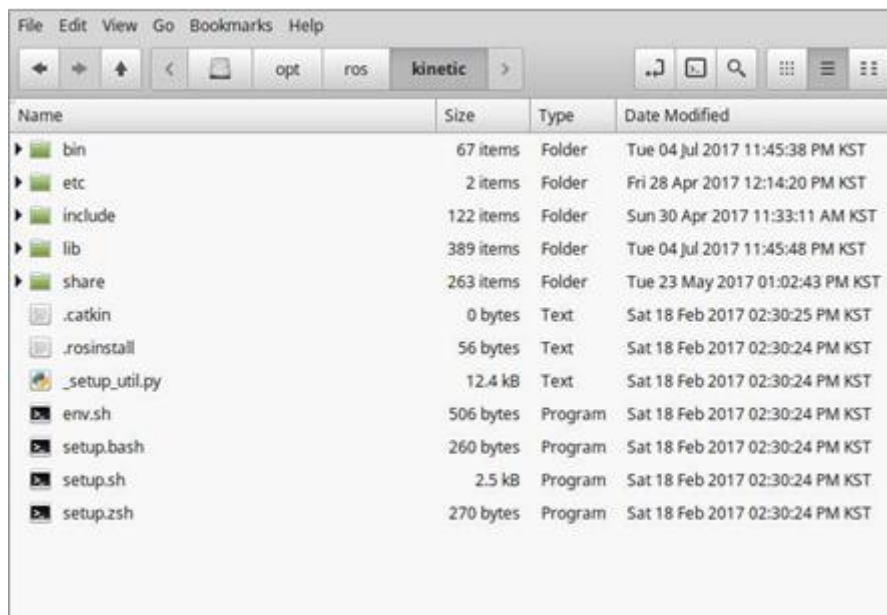


图 3.5 ROS 文件组织结构

3.6.3 工作目录

用户可以在任意位置创建工作目录，但是为了方便，本书中将 Linux 用户目录“~/catkin_ws/”用作工作目录。文件组织结构如图 3.6 所示，在“/home/用户名/”目录下有一个名为 catkin_ws 的目录，由目录 build、devel 和 src 组成。build 和 devel 目录是在 catkin_make 之后创建的。

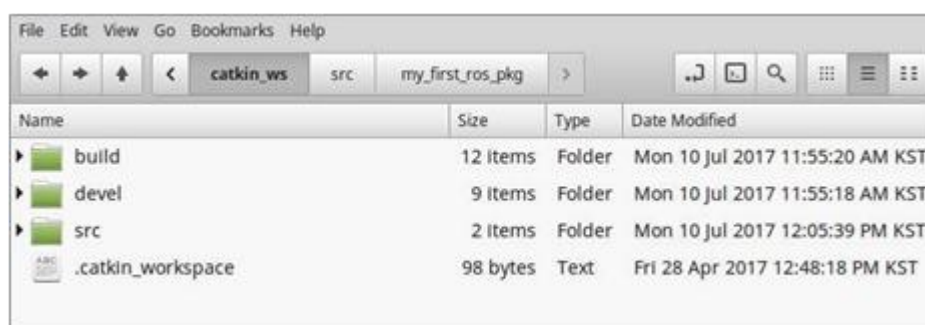


图 3.6 catkin workspace 的文件组织结构

目录“~/catkin_ws/src”是用户源代码的空间。在这个目录中，用户可以保存和建立自己的 ROS 功能包或其他开发者开发的功能包。

3.7 构建系统

ROS 的构建系统默认使用 CMake (Cross Platform Make), 其构建环境在功能包目录中的 CMakeLists.txt 文件中描述。在 ROS 中, CMake 被修改为适合于 ROS 的 “catkin” 构建系统。

在 ROS 中使用 CMake 的是为了在多个平台上构建 ROS 功能包。因为不同于只支持 Unix 系列的 Make, CMake 支持 Unix 类的 Linux、BSD 和 OS X 以外, 还支持 Windows 系列。并且, 它还支持 Microsoft Visual Studio, 也还可以轻松应用于 Qt 开发。此外, catkin 构建系统可以轻松使用与 ROS 相关的构建、功能包管理和功能包之间的依赖关系。

四、ROS 命令

4.1 ROS 命令概述

ROS 可以通过在 shell 环境中输入命令来进行文件系统的使用、源代码编辑、构建、调试和功能包管理等。为了正确使用 ROS, 除了基本的 Linux 命令之外, 还需要熟悉 ROS 专用命令。

4.2 ROS shell 命令

ROS shell 命令又被称为 rosbash1。这使我们可以在 ROS 开发环境中使用 Linux 中常用的 bash shell 命令。我们主要使用前缀是 ros 且带有多种后缀的命令, 例如 cd、pd、d、ls、ed、cp 和 run。

4.3 ROS 执行命令

ROS 执行命令管理 ROS 节点的运行。最重要的是, roscore 被用作节点之间的名称服务器。执行命令是 rosrun 和 roslaunch。rosrun 运行一个节点, 当运行多个节点或设置各种选项时使用 roslaunch。rosclean 是删除节点执行时记录的日志的命令。

4.4 ROS 信息命令

ROS 信息命令用于识别话题、服务、节点和参数等信息。尤其是 rostopic、rosservice、roscall 和 rosparam 经常被使用, 并且 rosbag 是 ROS 的主要特征之一, 它具有记录数据和回放功能。

五、ROS 工具

5.1 三维可视化工具

RViz 是 ROS 的三维可视化工具。它的主要目的是以三维方式显示 ROS 消息，可以将数据进行可视化表达。例如，可以无需编程就能表达激光测距仪（LRF）传感器中的传感器到障碍物的距离，RealSense、Kinect 或 Xtion 等三维距离传感器的点云数据，从相机获取的图像值等。

5.2 ROS GUI 开发工具

除了三维可视化工具 RViz 之外，ROS 还为机器人开发提供各种 GUI 工具。从 ROS Fuerte 版本开始，这些 GUI 开发工具被称为 rqt，它集成了 30 多种工具，可以作为一个综合的 GUI 工具来使用。另外，RViz 也被集成到 rqt 的插件中，这使 rqt 成为 ROS 的一个不可缺少的 GUI 工具。

六、机器人、传感器和电机

6.1 机器人功能包

机器人主要分为硬件和软件。机械、电机、齿轮、电路和传感器被归类为硬件。直接驱动或控制机器人硬件的微控制器级别的固件，以及利用从传感器获得的信息进行识别、制图、导航和动作规划的应用软件均被分类为软件。ROS 属于应用软件的范畴，根据功能，ROS 的功能包被分类为机器人功能包 1、专为传感器的传感器功能包 2 和专为驱动部的电机功能包 3。

6.2 传感器功能包

传感器是与机器人无法分离的。有许多研究从传感器数据提供的无数环境信息中提取有意义的信息，或利用这些信息认识环境并传输给机器人。这样的环境信息有位置、空间、天气、声音、惯性、振动、气体、电流量、RFID，物体和外力识别等很多种。该信息被用作机器人执行实际任务的重要数据。

在制作机器人时，通过驱动轮或机器人手臂让机器人移动，且通过智能手机等进行遥控并不意味着开发完成。机器人只有在自己认识到周围环境，只提取有意义的信息，并能够计划和做出思考和判断，才能被视为机器人。这就是为什么传感器很重要。

6.3 相机

相机相当于机器人的眼睛。从相机获得的图像对于识别机器人周围的环境非常有用。例如，利用相机图像的对象识别和脸部识别；使用两台相机（立体相机）从两个不同图像之间的差异获得的距离值；利用距离值生成 3 维地图的 Visual-SLAM；单眼相机 Visual-SLAM；利用从彩色图像获得的颜色信息的颜色识别；跟踪特定对象的对象跟踪。

6.4 激光距离传感器

激光距离传感器（Laser Distance Sensor, LDS）有多种名称，比如激光雷达（LIDAR）、激光测距仪（Laser Range Finder, LRF）和激光扫描仪（Laser Scanner）。LDS 是利用激光光源来测量与物体的距离的传感器。LDS 传感器具有高性能、高速度和实时数据采集的优点，因此在距离测量方面有着广泛的应用。由于这些优点，它是在机器人领域被广泛使用的传感器，比如用于使用距离传感器的 SLAM（Simultaneous Localization and Mapping）或用于识别人或物体识别。由于其优越的实时性能，最近还被广泛用于无人驾驶车辆。

七、结束语

本文档是具有无线充电的变电站巡检机器人操作系统，是无线充电巡检机器人资料的第三册。文档不足之处恳请指正。