

东莞市微宏智能科技有限公司

淘宝店铺: minibalance.taobao.com

网址: www.wheeltec.net

ROS2 快速上手教程

推荐关注我们的公众号获取更新资料



版本说明:

版本	日期	内容说明
V1.0	2021/04/19	第一次发布

序言

本文档主要讲解了 ROS2 系统的使用方法，文档分为三个部分，包括：ROS2 概述、系统环境搭建、基于 wheeltec_robot 的 ROS2 功能包的使用。本文档目的是让用户能快速上手使用 ROS2 系统。

注意：本文档是基于 wheeltec_robot 机器人搭建的 ROS2 系统，用户拿到产品时机器人端 ROS 主控已经安装好 ROS2，无需再次进行安装；同时，我们也会提供已配置好 ROS1 与 ROS2 共存的虚拟机环境。关于 rviz2 的配置较为繁琐，若用户自行在虚拟机端配置了 ROS2 的开发环境，则请用户另外找我们的工作人员提供 rviz2 的配置文件替换即可。

目录

序言.....	2
1. ROS2 概述.....	4
1.1 ROS2 支持的操作系统.....	5
1.2 ROS2 新特性.....	5
2. 环境搭建.....	7
2.1 系统准备.....	7
2.2 运行测试例程.....	9
3. ROS2 中的常用命令.....	11
3.1 工作空间与功能包的创建.....	11
3.2 ROS2 功能包的使用.....	13

1. ROS2 概述

ROS2 的工作原理与 ROS1 类似。ROS2 是独立发行版，并不是 ROS1 的一部分，但是，它又可以使用工具嵌入 ROS1 软件包并与之协同工作。ROS1 的特征栈是用 C++ 编写的，客户端库是使用 C++ 和 Python 编写的而 ROS2 的组件是用 C 语言编写的，ROS2 中有一个用 C 语言编写的独立栈，用来连接到 ROS2 的客户端库，客户端库主要包括 rclcpp、rclpy 和 rcljava 等。

ROS2 的设计目标明确，旨在改进可用于实时系统和产品阶段解决方案的通信网络框架，其目标是：

- 支持涉及不可靠网络的多机器人系统
- 支持实时通信控制
- 跨系统平台支持
- 直接在硬件层面提供 ROS 层
- 铲除原型和最终产品之间的鸿沟

1.1 ROS2 支持的操作系统

ROS 2 支持在 Linux、Windows、macOS 以及实时操作系统（RTOS）OS 层，ROS1 只支持 Linux 和 macOS 层。ROS2 发行版及支持的操作系统如表 1-1-1 所示。

图 1-1-1 ROS2 发行版及对应的操作系统

ROS2 发行版	支持的操作系统
Foxy	Ubuntu20.04; macOS 10.14(sierra); Windows10-需配置 visual studio 2019
Eloquent	Ubuntu18.04(Bionic-arm64 和 amd64) , Ubuntu18.04(Bionic-arm32); macOS 10.12(sierra) ; Windows10-需配置 visual studio 2019
Dashing	Ubuntu18.04(Bionic-arm64 和 amd64) , Ubuntu18.04(Bionic-arm32); macOS 10.12(sierra) ; Windows10-需配置 visual studio 2019
Crystal	Ubuntu18.04(Bionic); Ubuntu 16.04(Xenial)-源码安装; macOS 10.12(sierra); Windows10
Bouncy	Ubuntu18.04(Bionic); Ubuntu 16.04(Xenial)-源码安装; macOS 10.12(sierra); Windows10-需配置 visual studio 2017
Ardent	Ubuntu 16.04(Xenial)、macOS 10.12(sierra)、Windows10

由于我们的产品使用的环境是 Ubuntu18.04，所以本次适配的 ROS2 版本为 Eloquent，即第五个发行版本。

1.2 ROS2 新特性

相比于 ROS1，ROS2 使用更先进的分布式架构，拥有更高的可靠性，对实时性与嵌入式设备也提供支持。二者架构如图 1-1-1 所示。

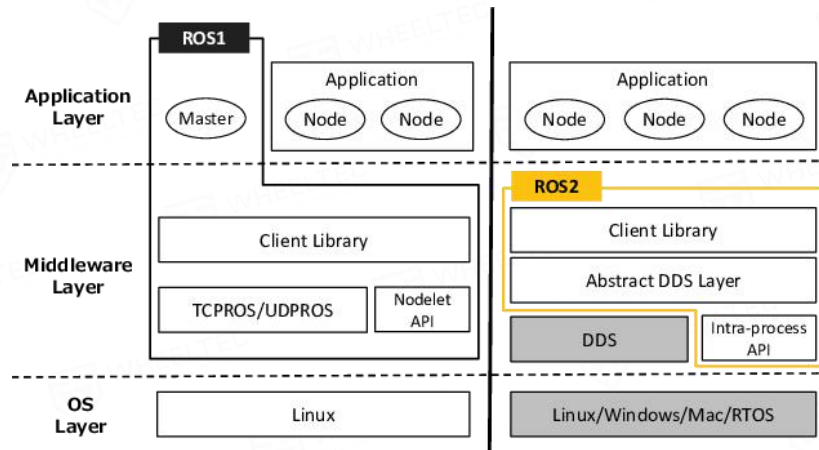


图 1-1-1 ROS1 与 ROS2 架构图

ROS 1 的通讯系统基于 TCPROS/UDPROS, 强依赖于 master 节点的处理, 而 ROS 2 的通讯系统是基于 DDS, 进而取消了 master, 同时在 ROS2 内部提供了 DDS 的抽象层与 ROS2 客户端库连接, 有了这个抽象层, 用户不需要去关注底层的 DDS API 的存在就可以与操作系统连接。ROS2 的 API 架构图如图 1-1-2 所示。

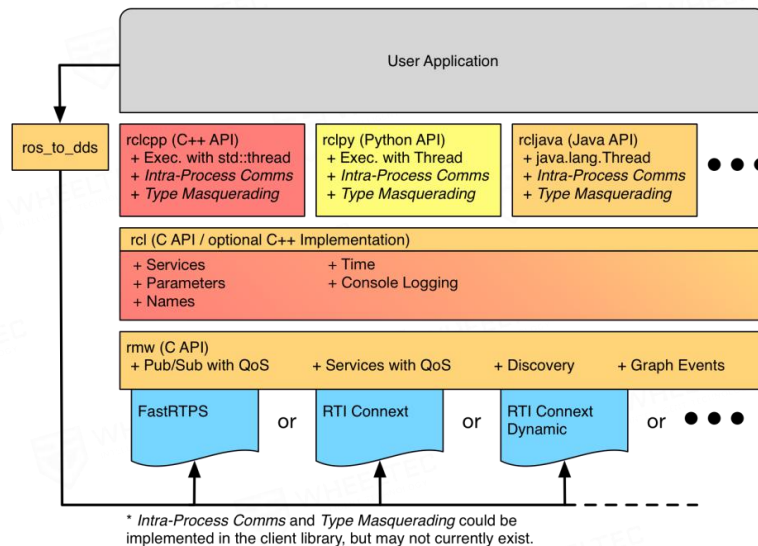


图 1-1-2 ROS2 的 API 架构图

除了以上提到的内容之外, ROS2 也推出了很多新特性, 主要有:

- 1) 跨系统平台支持: ROS 2 支持在 Ubuntu Xenial, OS X El Capitan 以及 Windows 10 这三种平台上使用。
- 2) ROS2 实现了分布式架构: 实现节点的分布式执行, 使用 ROS 2 选择的 DDS 中间件用于数据交换, 使这些组件可以在分布式环境中相互通信。
- 3) 支持实时控制。
- 4) 使用新版本的编程语言: ROS 2 广泛使用 C++ 11 标准, ROS2 的 Python 版本至少为 3.5。
- 5) 使用了新的编译系统 Ament。
- 6) ROS1 可以通过 ros_bridge 和 ROS 2 进行通信。
- 7) 使用托管启动: 用户可以指定节点启动顺序。
- 8) 取消了 nodelet 的概念, 支持多节点初始化。
- 9) launch 文件使用 python 编写, 相比于 xml 拓展了功能性。

2. 环境搭建

2.1 系统准备

① Ubuntu18.04 安装 ROS2-Eloquent

在搭建 ROS2 环境之前首先需要确认系统已经将 ROS1 安装成功。

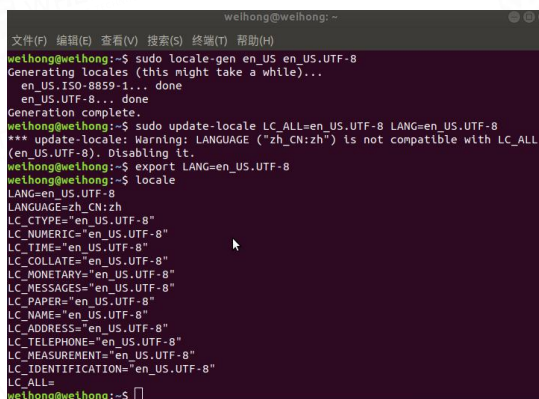
有两种方法安装 ROS2，一种是源码安装，另一种是二进制安装。本教程使用的是二进制安装。

② 安装步骤

- 1) 设置系统区域。首先需要确保安装环境支持 UTF-8 格式，使用一下指令设置：

```
sudo apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8
locale # verify settings
```

执行完之后终端为：



```
weihong@weihong: ~
weihong@weihong:~$ sudo locale-gen en_US en_US.UTF-8
Generating locales (this might take a while)...
  en_US.ISO-8859-1... done
  en_US.UTF-8... done
Generation complete.
weihong@weihong:~$ sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
*** update-locale: Warning: LANGUAGE ("zh_CN:zh") is not compatible with LC_ALL
(en_US.UTF-8). Disabling it.
weihong@weihong:~$ export LANG=en_US.UTF-8
weihong@weihong:~$ locale
LANG=en_US.UTF-8
LANGUAGE=zh_CN:zh
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_COLLATE="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
LC_PAPER="en_US.UTF-8"
LC_NAME="en_US.UTF-8"
LC_ADDRESS="en_US.UTF-8"
LC_TELEPHONE="en_US.UTF-8"
LC_MEASUREMENT="en_US.UTF-8"
LC_IDENTIFICATION="en_US.UTF-8"
LC_ALL=
weihong@weihong:~$
```

图 2-1-1 设置完系统区域的终端输出信息

- 2) 添加 ROS2 的代码仓库

使用一下指令授权密钥：

```
sudo apt update && sudo apt install curl gnupg2 lsb-release
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo
apt-key add -
```

将代码仓库添加到源列表：

```
sudo sh -c 'echo "deb [arch=$(dpkg --print-architecture)]
http://packages.ros.org/ros2/ubuntu $(lsb_release -cs) main" >
/etc/apt/sources.list.d/ros2-latest.list'
```


执行完后终端显示如图 2-1-2 所示。

```
weihong@weihong:~$ curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
OK
weihong@weihong:~$ sudo sh -c 'echo "deb [arch=$(dpkg --print-architecture)] http://packages.ros.org/ros2/ubuntu $(lsb_release -cs) main" > /etc/apt/sources.list.d/ros2-latest.list'
weihong@weihong:~$
```

图 2-1-2 授权密钥的终端输出信息

3) 更新列表:

```
sudo apt update
```

4) 安装 ROS2 桌面版, 包括 ROS, RViz, demos, tutorials。

```
sudo apt install ros-eloquent-desktop
```

5) 安装自动补全工具

```
sudo apt install -y python3-pip
pip3 install -U argcomplete
```

6) 安装编译工具

```
sudo apt install python3-colcon-common-extensions
```

7) 安装依赖和 ROS 工具

需要安装依赖项和工具, 执行以下指令:

```
sudo apt update && sudo apt install -y build-essential cmake git
python3-colcon-common-extensions python3-pip python-rosdep python3-vcstool wget
```

使用 pip3 安装测试功能包, 执行以下指令:

```
python3 -m pip install -U argcomplete flake8 flake8-blind-except flake8-builtins
flake8-class-newline flake8-comprehensions flake8-deprecated flake8-docstrings
flake8-import-order flake8-quotes pytest-repeat pytest-rerunfailures pytest
pytest-cov pytest-runner setuptools
```

安装 FAST-RTPS 依赖项, 执行以下指令:

```
sudo apt install --no-install-recommends -y libasio-dev libtinyxml2-dev
```

安装 Cyclone DDS 依赖项, 执行以下指令:

```
sudo apt install --no-install-recommends -y libcunit1-dev
```


③ ROS2 的环境设置

到现在为止，我们使用的 Ubuntu 系统同时安装了 ROS1 和 ROS2，所以需要进行 ROS1 与 ROS2 共存环境的设置。为了避免用户每次打开终端都执行一次 source 命令，可以将相关指令写进环境配置文件，使用 alias 命令设置到 bash 脚本中。两种环境的共存设置步骤为：

- 1) 使用 nano 编辑器打开环境配置文件：

```
nano ~/.bashrc
```

- 2) 在文件对应位置添加以下两句指令：

```
alias initros1=" source /opt/ros/melodic/setup.bash"  
alias initros2=" source /opt/ros/eloquent/setup.bash"
```

修改完后文件如图所示

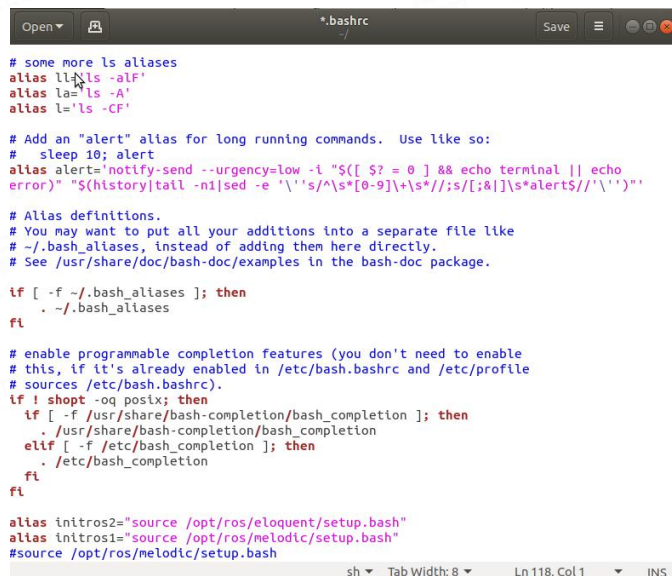


图 2-1-3 修改后的环境配置文件

- 3) 保存退出脚本文件。使用以下命令使修改生效：

```
source ~/.bashrc
```

这个修改是为了方便用户可以根据自己的需要，使用”initros1”或”initros2”命令调用系统环境。

2.2 运行测试例程

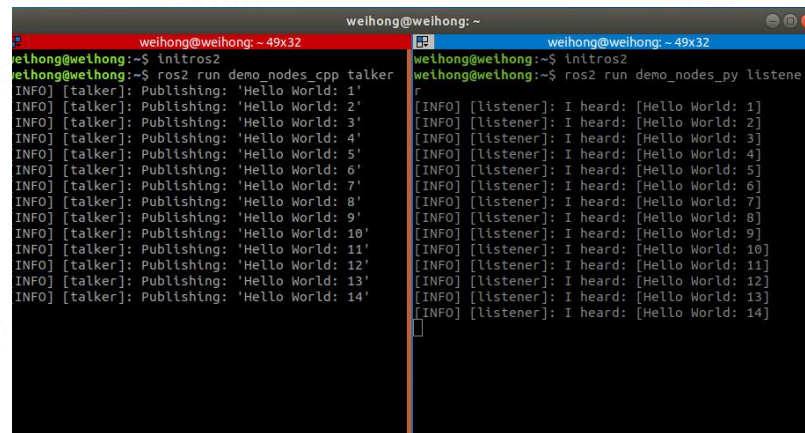
- 1) 使用 ctrl+alt+t 打开两个终端，输入以下命令初始化 ROS2 环境；

```
initros2
```

- 2) 在两个终端里分别运行系统例程的话题发布与订阅节点，命令为：

```
ros2 run demo_nodes_cpp talker  
ros2 run demo_nodes_py listener
```

这里需要注意命令中的 `ros2` 与 `run` 之间的空格, 执行完指令后终端输出信息如下:



```
weihong@weihong: ~ - 49x32
weihong@weihong:~$ initros2
weihong@weihong:~$ ros2 run demo_nodes_cpp talker
[INFO] [talker]: Publishing: 'Hello World: 1'
[INFO] [talker]: Publishing: 'Hello World: 2'
[INFO] [talker]: Publishing: 'Hello World: 3'
[INFO] [talker]: Publishing: 'Hello World: 4'
[INFO] [talker]: Publishing: 'Hello World: 5'
[INFO] [talker]: Publishing: 'Hello World: 6'
[INFO] [talker]: Publishing: 'Hello World: 7'
[INFO] [talker]: Publishing: 'Hello World: 8'
[INFO] [talker]: Publishing: 'Hello World: 9'
[INFO] [talker]: Publishing: 'Hello World: 10'
[INFO] [talker]: Publishing: 'Hello World: 11'
[INFO] [talker]: Publishing: 'Hello World: 12'
[INFO] [talker]: Publishing: 'Hello World: 13'
[INFO] [talker]: Publishing: 'Hello World: 14'

weihong@weihong: ~ - 49x32
weihong@weihong:~$ initros2
weihong@weihong:~$ ros2 run demo_nodes_py listener
[INFO] [listener]: I heard: [Hello World: 1]
[INFO] [listener]: I heard: [Hello World: 2]
[INFO] [listener]: I heard: [Hello World: 3]
[INFO] [listener]: I heard: [Hello World: 4]
[INFO] [listener]: I heard: [Hello World: 5]
[INFO] [listener]: I heard: [Hello World: 6]
[INFO] [listener]: I heard: [Hello World: 7]
[INFO] [listener]: I heard: [Hello World: 8]
[INFO] [listener]: I heard: [Hello World: 9]
[INFO] [listener]: I heard: [Hello World: 10]
[INFO] [listener]: I heard: [Hello World: 11]
[INFO] [listener]: I heard: [Hello World: 12]
[INFO] [listener]: I heard: [Hello World: 13]
[INFO] [listener]: I heard: [Hello World: 14]
```

图 2-2-1 ROS2 中的发布者(左)与订阅者(右)的输出

其实 ROS1 和 ROS2 运行节点的方式没有太大区别。上述指令可以理解为: 功能包的名字是 `demo_nodes_cpp`, 节点名是 `talker` 和 `listener`, 同时, 这也可以验证 C++ 和 Python API 是否正常工作。

3. ROS2 中的常用命令

3.1 工作空间与功能包的创建

① ROS2 工作空间的创建

工作空间是一个包含 ROS 2 功能包的目录。创建步骤为:

- 1) 输入命令”initros2”调出 ROS2 环境
- 2) 创建一个名为 wheeltec_robot_ros2 的文件夹作为工作空间:

```
mkdir -p ~/wheeltec_robot_ros2/src  
cd ~/wheeltec_robot_ros2/src
```

在 src 中可以直接拷贝放置用户的功能包,也可以在线使用 git clone 命令下载 github 中的开源代码。

② ROS2 功能包的创建

ROS 2 中使用 Ament 作为功能包的构建系统, colcon 作为它的构建工具。在 ROS2 中有 C++ 和 python 两种功能包,其创建指令不一样,以下教程以 C++ 功能包为示例。具体步骤为:

- 1) 输入命令”initros2”调出 ROS2 环境
- 2) 进入工作空间的 src 文件夹:

```
cd ~/wheeltec_robot_ros2/src
```

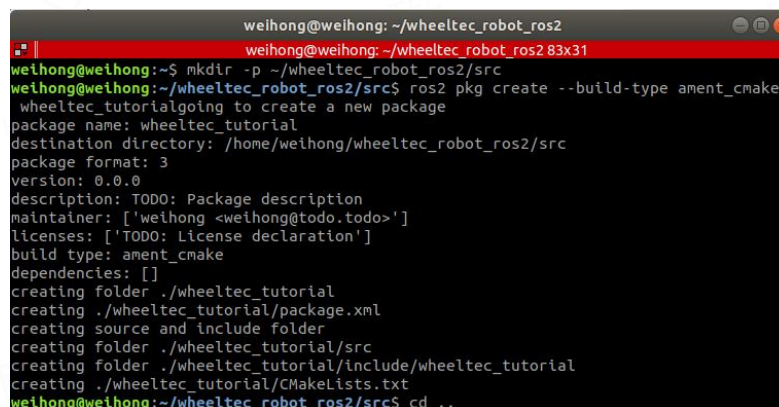
- 3) 创建一个名为 wheeltec_tutorials 的功能包:

```
ros2 pkg create --build-type ament_cmake wheeltec_tutorial
```

创建语法:

```
ros2 pkg create --build-type ament_cmake <package_name>
```

此时终端输出信息如图 3-1-1 所示。



```
weihong@weihong: ~/wheeltec_robot_ros2  
weihong@weihong: ~/wheeltec_robot_ros2 83x31  
weihong@weihong:~$ mkdir -p ~/wheeltec_robot_ros2/src  
weihong@weihong:~/wheeltec_robot_ros2/src$ ros2 pkg create --build-type ament_cmake  
wheeltec_tutorialgoing to create a new package  
package name: wheeltec_tutorial  
destination directory: /home/weihong/wheeltec_robot_ros2/src  
package format: 3  
version: 0.0.0  
description: TODO: Package description  
maintainer: ['weihong <weihong@todo.todo>']  
licenses: ['TODO: License declaration']  
build type: ament_cmake  
dependencies: []  
creating folder ./wheeltec_tutorial  
creating ./wheeltec_tutorial/package.xml  
creating source and include folder  
creating folder ./wheeltec_tutorial/src  
creating folder ./wheeltec_tutorial/include/wheeltec_tutorial  
creating ./wheeltec_tutorial/CMakeLists.txt  
weihong@weihong:~/wheeltec_robot_ros2/src$ cd ..
```

图 3-1-1 ROS2 创建一个功能包

4) 返回上一层目录后，输入以下指令进行编译。

```
colcon build#编译 src 目录中的全部功能包
colcon build --packages-select wheeltec_tutorial#编译所选择的功能包
```

```
weihong@weihong:~/wheeltec_robot_ros2/src$ cd ..
weihong@weihong:~/wheeltec_robot_ros2$ colcon build
Starting >>> wheeltec_tutorial
Finished <<< wheeltec_tutorial [2.07s]

Summary: 1 package finished [2.19s]
weihong@weihong:~/wheeltec_robot_ros2$ colcon build --packages-select wheeltec_tutorial
Starting >>> wheeltec_tutorial
Finished <<< wheeltec_tutorial [0.27s]

Summary: 1 package finished [0.38s]
weihong@weihong:~/wheeltec_robot_ros2$
```

图 3-1-2 编译功能包输出示例

5) 返回到 wheeltec_robot_ros2 目录下，运行以下命令以获取工作区：

```
. install/setup.bash
```

6) 创建完成的功能包如图 3-1-2 所示。

```
weihong@weihong:~/wheeltec_robot_ros2/src$ cd wheeltec_tutorial/
weihong@weihong:~/wheeltec_robot_ros2/src/wheeltec_tutorial$ tree
.
├── CMakeLists.txt
├── include
│   └── wheeltec_tutorial
├── package.xml
└── src

3 directories, 2 files
weihong@weihong:~/wheeltec_robot_ros2/src/wheeltec_tutorial$
```

图 3-1-3 新建功能包的树结构

package.xml 文件包含有关功能包的元信息的文件，CMakeLists.txt 文件描述如何在包中生成代码。

③ ROS2 功能包的编译

ROS2 中的编译构建工具变成了 colcon，工作空间的文件也与 ROS1 有所不同，ROS2 的工作空间没有 ROS1 中的 devel 文件夹；ROS2 编译工作空间下的功能包指令是：

```
colcon build
colcon build -h #查看编译子命令
```

colcon 不使用源代码构建，在编译完成后工作空间中生成的文件夹如下图所示 3-1-4 所示。

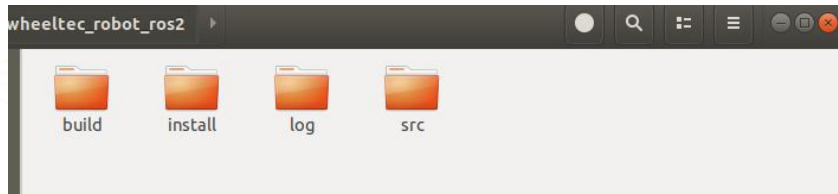


图 3-1-4 工作空间文件夹

build 文件夹：存储中间文件的位置。

install 文件夹：每个功能包的位置。

log 文件夹：所有日志信息可用的位置。

src 文件夹：源代码放置的位置。

3.2 ROS2 功能包的使用

本章是基于 wheeltec_robot 机器人的程序做关于使用的简单示例，方便用户迅速跑通 ROS2。关于功能包的详细信息在另一文档中有说明。如果用户要为自己的项目编写特定的功能包，推荐完整学习 ROS2 中的[用户手册](#)，其中包含更加全面的指导信息。

在运行以下程序之前需要先调出 ROS2 的环境，并声明工作空间的具体位置。终端依次执行以下指令：

```
initros2 #调出 ROS2 环境
cd ~/wheeltec_robot_ros2
. install/setup.bash
```

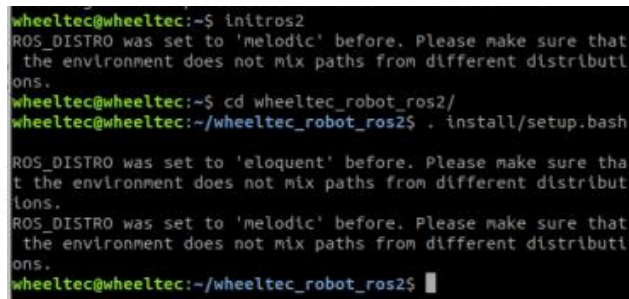


图 3-2-1 初始化工作环境

① 关于车型选择

若用户使用的是阿克曼小车，则需要在以下目录做出修改。

```
/wheeltec_robot_ros2/src/turn_on_wheeltec_robot/launch
```

将目录中的 `base_serial.launch.py` 文件中相应的位置改为“true”，非阿克曼小车的 `base_serial.launch.py` 文件如图所示。

注意：在 ROS2 中修改完 python 文件也是需要进行编译才能生效的，所以这里更改完之后需要执行以下指令编译一遍。

```
colcon build --packages-select turn_on_wheeltec_robot
```

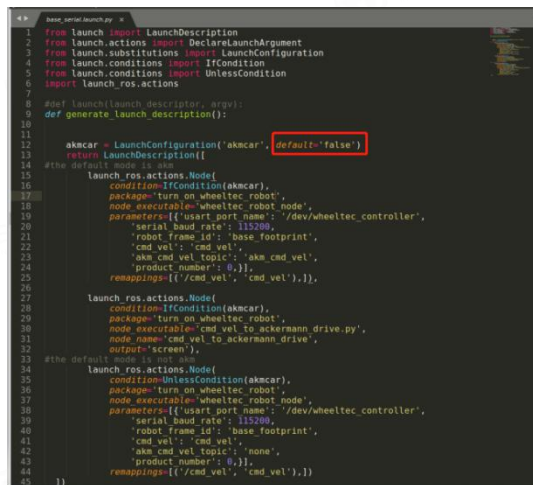


图 3-2-2 base_serial.launch.py 文件

② turn_on_wheeltec_robot

`turn_on_wheeltec_robot` 这个功能包主要作用是打开初始化底盘控制节点，运行指令为：

```
ros2 launch turn_on_wheeltec_robot turn_on_wheeltec_robot.launch.py
```

在 `turn_on_wheeltec_robot.launch.py` 文件中，调用了 `wheeltec_robot` 机器人上的传感器与 `base_footprint` 的 `tf` 坐标转换、打开雷达节点、关于机器人描述的 `urdf` 文件以及融合滤波的节点。

③ simple_follower

在 `simple_follower` 这个功能包中有两个节点，一个是视觉巡线，另一个是雷达跟随。

运行视觉巡线指令为：

```
ros2 launch simple_follower_ros2 line.launch.py
```

这里需要注意 `ros2` 和 `launch` 之间的空格。

在 `line.launch.py` 文件中打开了 `usb` 摄像头和初始化底层控制节点，所以用户

不必另外再运行 `turn_on_wheeltec_robot.launch.py` 底盘启动文件。

运行雷达跟随指令为：

```
ros2 launch simple_follower_ros2 laser_follow.launch.py
```

这里需要注意 `ros2` 和 `launch` 之间的空格。

在 `line.launch.py` 文件中打开了雷达和初始化底层控制节点，所以用户不必另外再运行 `turn_on_wheeltec_robot.launch.py` 底盘启动文件。

④ wheeltec_robot_slam

`wheeltec_robot_slam` 这个文件夹里面包含了三个 `slam` 功能包，分别是 `slam_gmapping`、`slam_toolbox` 和 `cartographer` 的调用。`slam_gmapping` 软件包包含用于 OpenSlam 的 Gmapping 的 ROS 包装器它提供了基于激光的 SLAM（同时定位和映射），作为 `slam_gmapping` 的 ROS 节点。使用的是 Gmapping 的绘制地图算法。`slam_toolbox` 是 2D SLAM 构建的一组工具和功能，该软件包将允许用户完全序列化要重新加载的 SLAM 映射的数据和姿态图，以继续映射，本地化，合并或进行其他操作。`cartographer` 是在 ROS1 中也常用到的一种建图算法，这里就不再赘述。

建图步骤为：用户可以挑选以下三种建图方式的任意一种进行 2D 建图，搭配键盘控制节点使用。

使用 Gmapping 算法进行 2D 建图，指令如下。建图页面如图 3-2-3 所示。

```
ros2 launch slam_gmapping slam_gmapping.launch.py
```

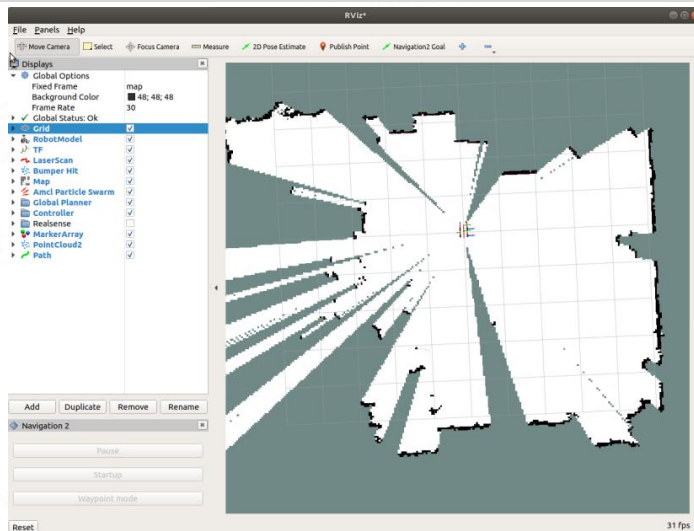


图 3-2-3 gmapping 算法建图页面

使用 `slam_toolbox` 进行 2D 建图，指令如下：

```
ros2 launch wheeltec_slam_toolbox slam_toolbox.launch.py
```

建图页面如图 3-2-4 所示。

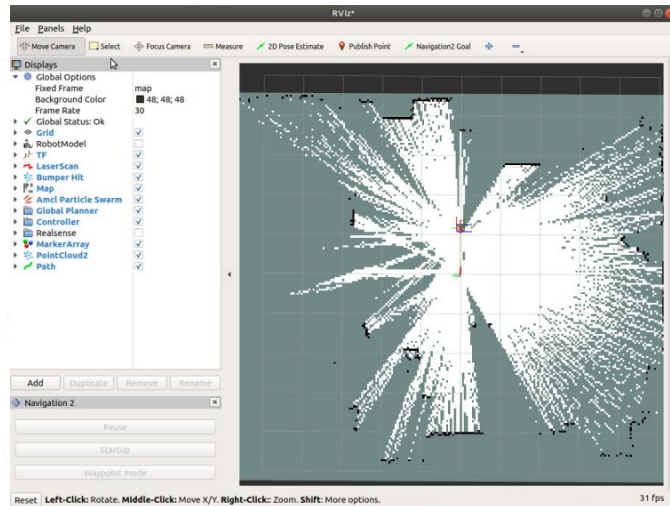


图 3-2-4 使用 `slam_toolbox` 建图页面

使用 `cartographer` 算法进行 2D 建图，指令如下。建图页面如图 3-2-5 所示。

```
ros2 launch wheeltec_cartographer cartographer.launch.py
```

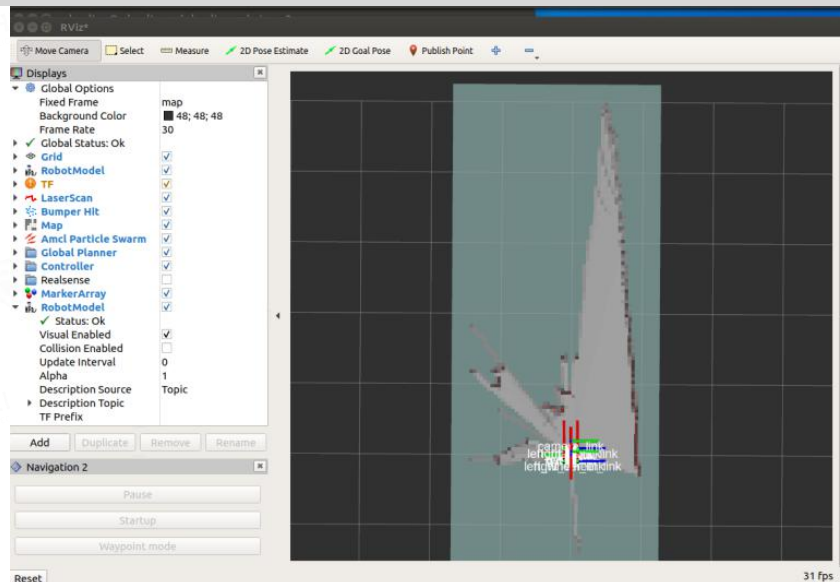


图 3-2-5 使用 `slam_toolbox` 建图页面

1) 打开键盘控制节点控制机器人运动：

```
ros2 run wheeltec_robot_keyboard wheeltec_keyboard
```

2) 打开 `rviz2` 可视化工具：

```
rviz2
```

3) 建图完成，保存地图，指令格式为：

```
ros2 run nav2_map_server map_saver -f ~/wheeltec_robot_ros/src/wheeltec_robot_nav/map WHEELTEC
```

⑤ wheeltec_robot_nav

1) 2D 单点导航

wheeltec_robot_nav 功能包的主要作用是在机器人端实现 2D 导航与其它拓展性功能。

在 wheeltec_navigation.launch.py 文件中，调用了初始化底层控制节点、需导航的地图(地图文件存放在 wheeltec_robot_nav/map 目录下)、以及导航的相关参数(参数文件存放在 wheeltec_robot_nav/param 目录下)以及 localization 功能包。

在 wheeltec_navigation.launch.py 文件中，如果设置 autostart: = False，则需要单击 RViz 中的开始按钮以初始化节点。确保 use_sim time 设置为 False，因为我们要使用系统时间而不是 Gazebo 中的时间模拟时间。

执行 2D 导航的指令为：

```
ros2 launch wheeltec_robot_nav wheeltec_navigation.launch.py
```

执行完以上指令后，打开一个终端，输入指令：

```
rviz2
```

打开可视化页面，点击 rviz2 上方的”Navigation2 Goal”设置目标点进行导航。

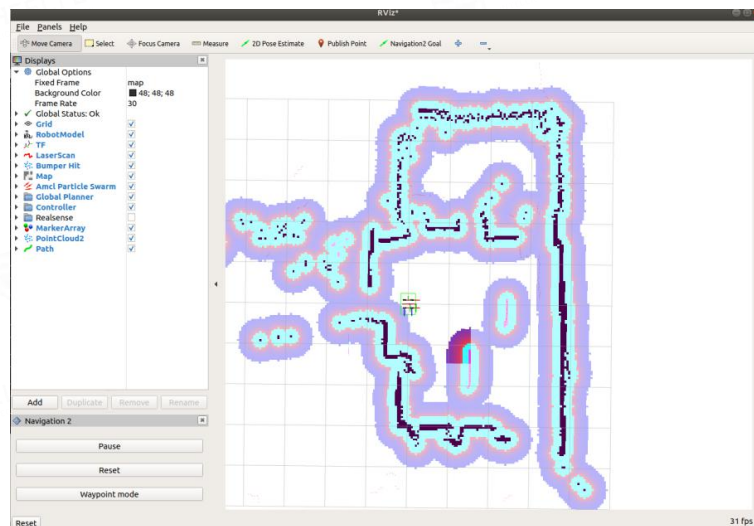


图 3-2-6 rviz2 可视化导航页面

2) 路标跟随演示

路标跟踪是导航系统的基本功能。它告诉我们的系统如何使用导航到达多个目的地。在 2D 单点导航的步骤下，打开 rviz2 后，点击右下角的“waypoint mode”切换成路标跟随模式，如图 3-2-7 所示。

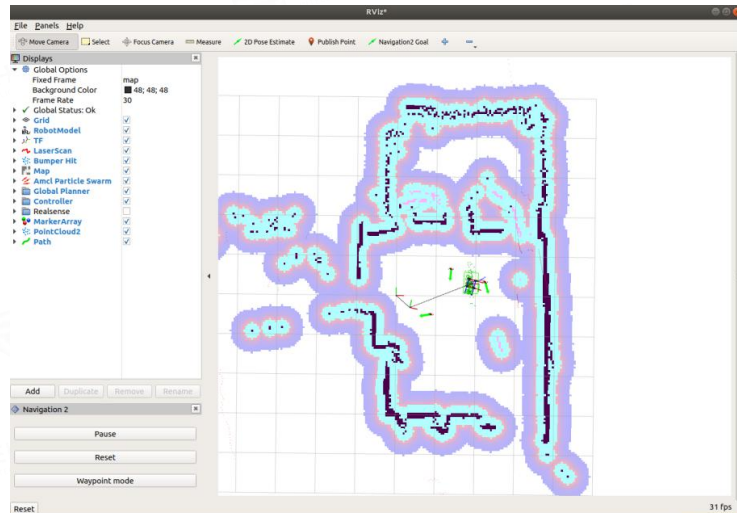


图 3-2-7 waypoint mode 设置页面

到这一步，再点击 rviz2 上方的“Navigation2 Goal”开始设定路标位置，标完位置后，点击右下角的“stat”开始实现路标跟随。

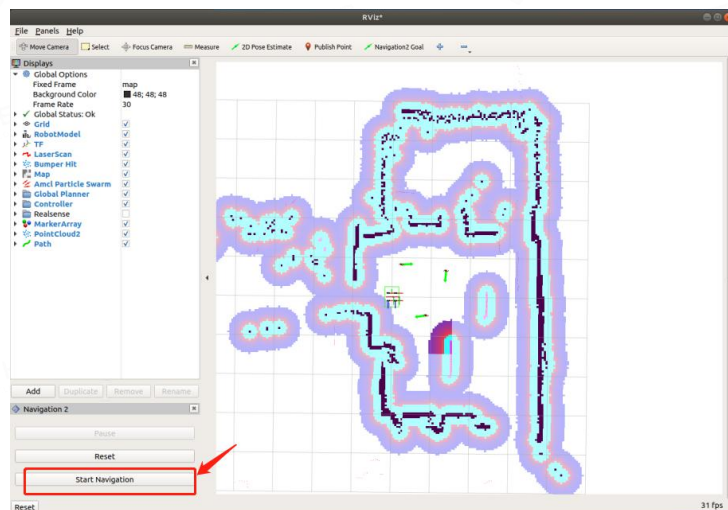


图 3-2-7 waypoint mode 开始导航页面

3.3 ROS1 与 ROS2 的通信

ROS2 的目标并不是取代 ROS1，所以会在很长的一段时间内，二者会一直并存。为了使用户能够使用 ROS1 和 ROS2 的功能包进行项目开发，可以通过 `ros1_bridge` 实现两个版本的连接与公用，该程序包提供了一个网桥，使 ROS 1 和 ROS 2 之间可以交换消息。

`ros1_bridge` 实际上是 ROS2 的一个功能包，用于自动或手动建立信息、话题和服务的映射，并在 ROS1 和 ROS2 之间进行通信。使用步骤示例：

- 1) 在 ROS1 中运行 `roscore`

```
roscore
```

- 2) 在 ROS1 的终端打开初始化底盘控制节点

```
roslaunch turn_on_wheeltec_robot turn_on_wheeltec_robot.launch
```

- 3) 在 ROS2 的终端打开 `ros1_bridge`

```
initros2
ros2 run ros1_bridge dynamic_bridge
```

- 4) 在 ROS2 的终端打开键盘控制节点

```
ros2 run wheeltec_robot_keyboard wheeltec_keyboard
```

以上示例实现了速度话题/`cmd_vel`从 ROS2 到 ROS1 的通信，如果想实现从 ROS1 到 ROS2 的通信的话，反过来也能完成。运行后的效果如图 3-3-1 所示。

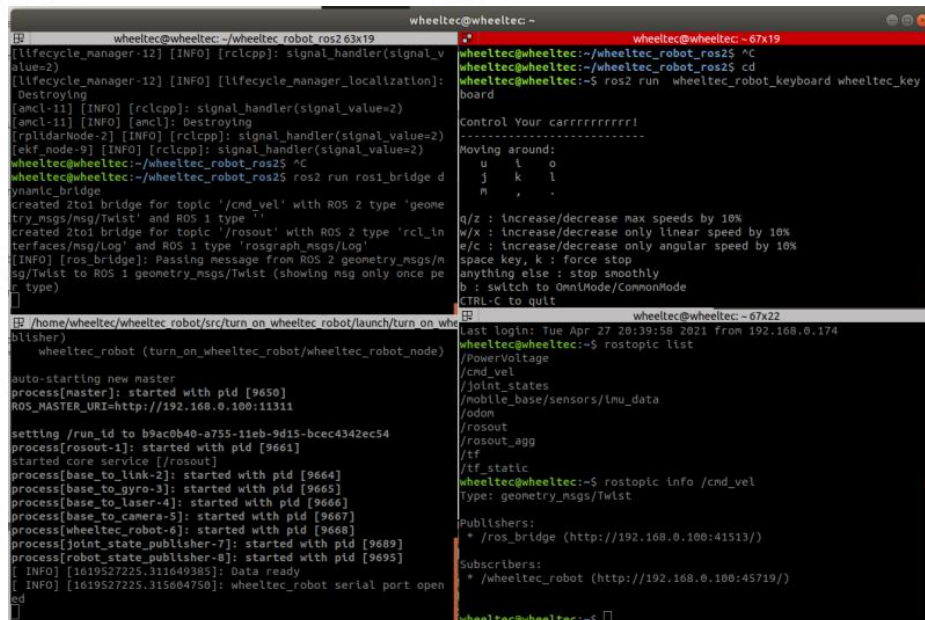


图 3-3-1 ROS1 与 ROS2 之间的服务通信