机器人操作系统（ROS2）入门与实践

全书源代码

刘相权 张万杰 整理

第 1 章 Linux Ubuntu 入门基础

1.2 安装 Ubuntu 22.04

1.2.1 准备工作

3）下载 Ubuntu 22.04 系统安装镜像；下载网址为：https://cn.ubuntu.com/desktop，选择 Ubuntu 22.04.3 LTS 进行下载。

4）win32diskimager 软件；下载网址为：https://sourceforge.net/projects/win32diskimager/。

1.3 VMware 虚拟机安装 Ubuntu22.04

1.3.1 安装 VMware 虚拟机

VMware 虚拟机下载网址：https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html

Ubuntu 网址为：https://cn.ubuntu.com/desktop，选择 Ubuntu 22.04.3 LTS 进行下载。

1.4 Ubuntu 22.04 使用入门

1.4.8 时间同步问题

```
sudo apt update
sudo apt install ntpdate
sudo ntpdate time.windows.com
sudo hwclock --localtime --systohc
```

1.4.9 修改默认引导系统

```
sudo gedit /etc/default/grub
sudo update-grub
```

第 2 章 ROS2 安装与系统架构

2.3 ROS2 安装与配置

2.3.1 安装 ROS2

官方安装步骤网页：
https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html

locale
sudo apt update && 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
sudo apt update
sudo apt install software-properties-common -y
sudo add-apt-repository universe
sudo apt install curl -y
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg
wget http://packages.ros.org/ros.key
sudo cp ros.key /usr/share/keyrings/ros-archive-keyring.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
sudo apt update
sudo apt upgrade
sudo apt install ros-humble-desktop
echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc
sudo apt install ros-dev-tools

ros2

2.4 从 APT 软件源安装应用程序

sudo apt update
sudo apt install ros-humble-rqt-robot-steering
ros2 run rqt_robot_steering rqt_robot_steering
ros2 run turtlesim turtlesim_node

2.5 ROS2 的工作空间

mkdir -p ~/ros2_ws/src
cd ~/ros2_ws
colcon build

2.6 从 Github 下载安装应用程序

sudo apt install git
cd ~/ros2_ws/src/
git clone https://github.com/6-robot/wpr_simulation2.git
git clone https://gitee.com/s-robot/wpr_simulation2.git
cd wpr_simulation2/scripts/
./install_for_humble.sh
cd ~/ros2_ws/
colcon build
source install/setup.bash
ros2 launch wpr_simulation2 wpb_simple.launch.py
ros2 run rqt_robot_steering rqt_robot_steering

2.7 Visual Studio Code 编辑器

2.7.1 安装 Visual Studio Code

在浏览器中打开 VSCode 官方下载地址：https://code.visualstudio.com。

先输入“sudo dpkg -i code_”然后按下键盘的[Tab]键，让命令行自动补齐后面的文件名。

2.8 Terminator 终端工具

sudo apt update

第 3 章 ROS2 编程基础

3.1 节点 Node 和软件包 Package

3.1.1 创建软件包 Package

```
cd ~/ros2_ws/src
```

```
ros2 pkg create my_pkg
```

3.1.2 编写节点 Node

```
#include "rclcpp/rclcpp.hpp"

int main(int argc, char * argv[])
{
   rclcpp::init(argc, argv);

   auto node = std::make_shared<rclcpp::Node>("my_node");

   RCLCPP_INFO(node->get_logger(), "Hello world!");
   while (rclcpp::ok())
   {
      ;
   }

   rclcpp::shutdown();

   return 0;
}
```

```
add_executable(my_node src/my_node.cpp)
ament_target_dependencies(my_node "rclcpp")
```

```
install(TARGETS
   my_node
   DESTINATION lib/${PROJECT_NAME})
```

```
<depend>rclcpp</depend>
```

```
cd ~/ros2_ws
```

```
colcon build
```

3.1.3 运行节点 Node

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

```
ros2 run my_pkg my_node
```

3.2 话题 Topic 和消息 Message

3.2.1 编写话题发布者 Publisher

```
cd ~/ros2_ws/src
ros2 pkg create topic_pkg
```

打开[VSCode]，创建一个名为 publisher_node.cpp 的源代码文件。

```
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"

int main(int argc, char * argv[])
{
  rclcpp::init(argc, argv);

  auto node = std::make_shared<rclcpp::Node>("publisher_node");

  auto publisher = node->create_publisher<std_msgs::msg::String>("/my_topic", 10);

  std_msgs::msg::String message;
  message.data = "Hello World!";

  rclcpp::Rate loop_rate(1);

  while (rclcpp::ok())
  {
    publisher->publish(message);
    loop_rate.sleep();
  }

  rclcpp::shutdown();

  return 0;
}
```

```
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)
```

```
add_executable(publisher_node src/publisher_node.cpp)
ament_target_dependencies(publisher_node "rclcpp"   "std_msgs")
```

```
install(TARGETS
  publisher_node
```

```
  DESTINATION lib/${PROJECT_NAME})
```

```
<depend>rclcpp</depend>
<depend>std_msgs</depend>
```

```
cd ~/ros2_ws
colcon build
source install/setup.bash
ros2 run topic_pkg publisher_node
ros2 topic list
ros2 topic echo /my_topic
```

3.2.2 编写话题订阅者 Subscriber

创建一个名为 subscriber_node.cpp 的源代码文件。

```
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"

std::shared_ptr<rclcpp::Node> node;

void Callback(const std_msgs::msg::String::SharedPtr msg)
{
      RCLCPP_INFO(node->get_logger(),"Receive : %s", msg->data.c_str());
}

int main(int argc, char * argv[])
{
  rclcpp::init(argc, argv);

 node = std::make_shared<rclcpp::Node>("subscriber_node");

  auto subscriber= node->create_subscription<std_msgs::msg::String>("/my_topic", 10,
&Callback);

  rclcpp::spin(node);

  rclcpp::shutdown();

  return 0;
}
```

```
add_executable(subscriber_node src/subscriber_node.cpp)
ament_target_dependencies(subscriber_node "rclcpp"   "std_msgs")
install(TARGETS
```

```
  publisher_node
 subscriber_node
  DESTINATION lib/${PROJECT_NAME})
```

```
cd ~/ros2_ws
colcon build
source install/setup.bash
ros2 run topic_pkg publisher_node
ros2 run topic_pkg subscriber_node
```

3.3 面向对象的节点 Node 实现

3.3.1 话题发布者 Publisher 的类封装

```
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"

class PublisherNode : public rclcpp::Node
{
public:
  PublisherNode()
    : Node("publisher_node")
  {
    publisher_ = create_publisher<std_msgs::msg::String>("/my_topic", 10);
    timer_ = create_wall_timer(
      std::chrono::milliseconds(1000),
      std::bind(&PublisherNode::publishMessage, this)
      );
  }

private:
  void publishMessage()
  {
    message_.data = "Hello World!";
    publisher_->publish(message_);
  }

  rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_;
  std_msgs::msg::String message_;
  rclcpp::TimerBase::SharedPtr timer_;
};

int main(int argc, char * argv[])
{
```

```
    rclcpp::init(argc, argv);
    auto node = std::make_shared<PublisherNode>();
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```

3.3.2 话题订阅者 Subscriber 的类封装

```
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"

class SubscriberNode : public rclcpp::Node
{
public:
    SubscriberNode()
        : Node("subscriber_node")
    {
        subscriber_ = create_subscription<std_msgs::msg::String>(
            "/my_topic",
            10,
            std::bind(&SubscriberNode::Callback, this, std::placeholders::_1));
    }

private:
    void Callback(const std_msgs::msg::String::SharedPtr msg)
    {
        RCLCPP_INFO(get_logger(), "Receive : %s", msg->data.c_str());
    }

    rclcpp::Subscription<std_msgs::msg::String>::SharedPtr subscriber_;
};

int main(int argc, char * argv[])
{
    rclcpp::init(argc, argv);
    auto node = std::make_shared<SubscriberNode>();
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```

3.4 Launch 文件

3.4.1 XML 格式的 Launch 文件

新建文件夹，命名为“launch”。新建文件，命名为“pub_sub.launch.xml”。

```
    <launch>

  <node pkg = "topic_pkg" exec = "publisher_node" name = "publisher_node" />

  <node pkg = "topic_pkg" exec = "subscriber_node" name = "subscriber_node" />

</launch>
```

```
install(
DIRECTORY
   launch
DESTINATION
   share/${PROJECT_NAME}
)
```

```
cd ~/ros2_ws/
colcon build
source install/setup.bash
ros2 launch topic_pkg pub_sub.launch.xml
```

3.4.2 YAML 格式的 Launch 文件

新建文件，命名为“pub_sub.launch.yaml”。

```
launch:
- node:
      pkg: "topic_pkg"
      exec: "publisher_node"
      name: "publisher_node"
- node:
      pkg: "topic_pkg"
      exec: "subscriber_node"
      name: "subscriber_node"
```

```
install(
DIRECTORY
   launch
DESTINATION
   share/${PROJECT_NAME}
)
```

```
cd ~/ros2_ws/
colcon build
```

```
source install/setup.bash
ros2 launch topic_pkg pub_sub.launch.yaml
```

3.4.3 Python 格式的 Launch 文件

新建文件，命名为“pub_sub.launch.py”。

```
from launch_ros.actions import Node
from launch import LaunchDescription

def generate_launch_description():

    publisher_cmd = Node(
        package='topic_pkg',
        executable='publisher_node',
        name='publisher_node'
        )

    subscriber_cmd = Node(
        package='topic_pkg',
        executable='subscriber_node',
        name='subscriber_node'
        )

  ld = LaunchDescription()

    ld.add_action(publisher_cmd)
  ld.add_action(subscriber_cmd)

    return ld
```

```
install(
DIRECTORY
  launch
DESTINATION
  share/${PROJECT_NAME}
)
```

```
cd ~/ros2_ws/
colcon build
source install/setup.bash
ros2 launch topic_pkg pub_sub.launch.py
```

第 4 章 ROS2 机器人运动控制

4.2 机器人运动控制的实现

4.2.1 编写速度控制程序

```
cd ~/ros2_ws/src
ros2 pkg create vel_pkg
```

命名新建文件为“vel_node.cpp”。

```
#include <rclcpp/rclcpp.hpp>
#include <geometry_msgs/msg/twist.hpp>

int main(int argc, char** argv)
{
  rclcpp::init(argc, argv);

  auto node = std::make_shared<rclcpp::Node>("velocity_command_node");

  auto vel_pub = node->create_publisher<geometry_msgs::msg::Twist>("/cmd_vel", 10);

  geometry_msgs::msg::Twist vel_msg;
  vel_msg.linear.x = 0.1;
  vel_msg.linear.y = 0.0;
  vel_msg.linear.z = 0.0;
  vel_msg.angular.x = 0.0;
  vel_msg.angular.y = 0.0;
  vel_msg.angular.z = 0.0;

  rclcpp::Rate loop_rate(30);
  while (rclcpp::ok())
  {
    vel_pub->publish(vel_msg);
    loop_rate.sleep();
  }

  rclcpp::shutdown();

  return 0;
}
```

```
find_package(rclcpp REQUIRED)
find_package(geometry_msgs REQUIRED)
```

```
add_executable(vel_node src/vel_node.cpp)
ament_target_dependencies(vel_node "rclcpp" "geometry_msgs")
install(TARGETS    vel_node
DESTINATION    lib/${PROJECT_NAME})
```

```
<depend>rclcpp</depend>
<depend>geometry_msgs</depend>
```

```
cd ~/ros2_ws
colcon build
source install/setup.bash
ros2 run vel_pkg vel_node
ros2 topic list
ros2 topic echo /cmd_vel
```

4.2.2 仿真运行速度控制程序

```
source install/setup.bash
ros2 launch wpr_simulation2 wpb_simple.launch.py
```

```
source install/setup.bash
ros2 run vel_pkg vel_node
```

第 5 章 激光雷达在 ROS2 中的使用

5.2 在 RViz2 中查看激光雷达数据

5.2.1 RViz2 的使用

```
source ~/ros2_ws/install/setup.bash
ros2 launch wpr_simulation2 wpb_simple.launch.py
```

```
source ~/ros2_ws/install/setup.bash
rviz2
```

5.2.3 界面配置的加载

```
rviz2 -d ~/lidar.rviz
```

5.3 激光雷达数据获取

5.3.1 编写雷达数据获取程序

```
cd ~/ros2_ws/src
ros2 pkg create lidar_pkg
```

命名新建文件为“lidar_data.cpp”。

```
#include <rclcpp/rclcpp.hpp>
#include <sensor_msgs/msg/laser_scan.hpp>

std::shared_ptr<rclcpp::Node> node;

void LidarCallback(const sensor_msgs::msg::LaserScan::SharedPtr msg)
{
    int nNum = msg->ranges.size();

    int nMid = nNum / 2;
    float fMidDist = msg->ranges[nMid];
    RCLCPP_INFO(node->get_logger(), "ranges[%d] = %f m", nMid, fMidDist);
}

int main(int argc, char** argv)
{
    rclcpp::init(argc, argv);

    node = std::make_shared<rclcpp::Node>("lidar_data_node");
```

```
    auto lidar_sub = node->create_subscription<sensor_msgs::msg::LaserScan>("/scan", 10, LidarCallback);

    rclcpp::spin(node);

    rclcpp::shutdown();

    return 0;
}
```

```
find_package(rclcpp REQUIRED)
find_package(sensor_msgs REQUIRED)
add_executable(lidar_data src/lidar_data.cpp)
ament_target_dependencies(lidar_data "rclcpp" "sensor_msgs")
install(TARGETS    lidar_data
DESTINATION    lib/${PROJECT_NAME})
```

```
<depend>rclcpp</depend>
<depend>sensor_msgs</depend>
```

```
cd ~/ros2_ws
colcon build
```

5.3.2 仿真运行雷达数据获取程序

```
source install/setup.bash
ros2 launch wpr_simulation2 wpb_simple.launch.py
```

```
source install/setup.bash
ros2 run lidar_pkg lidar_data
```

5.4 基于激光雷达的避障实现

5.4.1 编写雷达避障程序

命名新建文件为“lidar_behavior.cpp”。

```
#include <rclcpp/rclcpp.hpp>
#include <sensor_msgs/msg/laser_scan.hpp>
#include <geometry_msgs/msg/twist.hpp>

std::shared_ptr<rclcpp::Node> node;
rclcpp::Publisher<geometry_msgs::msg::Twist>::SharedPtr vel_pub;
int nCount = 0;

void LidarCallback(const sensor_msgs::msg::LaserScan::SharedPtr msg)
```

```
{
    int nNum = msg->ranges.size();

    int nMid = nNum / 2;
    float fMidDist = msg->ranges[nMid];
    RCLCPP_INFO(node->get_logger(), "ranges[%d] = %f m", nMid, fMidDist);

    if (nCount > 0)
    {
        nCount--;
        return;
    }

    geometry_msgs::msg::Twist vel_msg;
    if (fMidDist < 1.5f)
    {
        vel_msg.angular.z = 0.3;
        nCount = 100;
    }
    else
    {
        vel_msg.linear.x = 0.1;
    }
    vel_pub->publish(vel_msg);
}

int main(int argc, char** argv)
{
    rclcpp::init(argc, argv);

    node = std::make_shared<rclcpp::Node>("lidar_behavior_node");

    vel_pub = node->create_publisher<geometry_msgs::msg::Twist>("/cmd_vel", 10);
    auto lidar_sub = node->create_subscription<sensor_msgs::msg::LaserScan>("/scan", 10,
LidarCallback);

    rclcpp::spin(node);

    rclcpp::shutdown();

    return 0;
}
```

```
find_package(rclcpp REQUIRED)
```

```
find_package(sensor_msgs REQUIRED)
find_package(geometry_msgs REQUIRED)
```

```
add_executable(lidar_behavior src/lidar_behavior.cpp)
ament_target_dependencies(lidar_behavior "rclcpp" "sensor_msgs" "geometry_msgs")
```

最后在安装规则里添加新的节点：

```
install(TARGETS    lidar_data    lidar_behavior
DESTINATION    lib/${PROJECT_NAME})
```

```
<depend>rclcpp</depend>
<depend>sensor_msgs</depend>
<depend>geometry_msgs</depend>
```

```
cd ~/ros2_ws
```

```
colcon build
```

5.4.2 仿真运行雷达避障程序

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

```
ros2 launch wpr_simulation2 wpb_simple.launch.py
```

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

```
ros2 run lidar_pkg lidar_behavior
```

第 6 章 IMU 在 ROS2 中的使用

6.2 IMU 数据获取

6.2.1 编写 IMU 数据获取程序

```
cd ~/ros2_ws/src
ros2 pkg create imu_pkg
```

命名新建文件为“imu_data.cpp”。

```
#include <rclcpp/rclcpp.hpp>
#include <sensor_msgs/msg/imu.hpp>
#include <tf2/LinearMath/Quaternion.h>
#include <tf2/LinearMath/Matrix3x3.h>

std::shared_ptr<rclcpp::Node> node;

void IMUCallback(const sensor_msgs::msg::Imu::SharedPtr msg)
{
    tf2::Quaternion tf2_quaternion;
    tf2_quaternion.setX(msg->orientation.x);
    tf2_quaternion.setY(msg->orientation.y);
    tf2_quaternion.setZ(msg->orientation.z);
    tf2_quaternion.setW(msg->orientation.w);

    tf2::Matrix3x3 matrix(tf2_quaternion);

    double roll, pitch, yaw;
    matrix.getRPY(roll, pitch, yaw);
    roll = roll * 180 / M_PI;
    pitch = pitch * 180 / M_PI;
    yaw = yaw * 180 / M_PI;
    RCLCPP_INFO(node->get_logger(), "roll= %.0f pitch= %.0f yaw= %.0f", roll, pitch,
yaw);
}

int main(int argc, char** argv)
{
    rclcpp::init(argc, argv);

    node = std::make_shared<rclcpp::Node>("imu_data_node");

    auto sub = node->create_subscription<sensor_msgs::msg::Imu>("/imu", 10, IMUCallback);
```

```
    rclcpp::spin(node);

    rclcpp::shutdown();

    return 0;
}
```

```
find_package(rclcpp REQUIRED)
find_package(sensor_msgs REQUIRED)
find_package(tf2 REQUIRED)
```

```
add_executable(imu_data src/imu_data.cpp)
ament_target_dependencies(imu_data "rclcpp" "sensor_msgs" "tf2")
```

```
install(TARGETS    imu_data
DESTINATION    lib/${PROJECT_NAME})
```

```
<depend>rclcpp</depend>
<depend>sensor_msgs</depend>
<depend>tf2</depend>
```

```
cd ~/ros2_ws
```

```
colcon build
```

6.2.2 仿真运行 IMU 数据获取程序

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

```
ros2 launch wpr_simulation2 wpb_simple.launch.py
```

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

```
ros2 run imu_pkg imu_data
```

6.3 基于 IMU 的航向锁定实现

6.3.1 编写航向锁定程序

新建文件“imu_behavior.cpp”。

```
#include <rclcpp/rclcpp.hpp>
#include <sensor_msgs/msg/imu.hpp>
#include <geometry_msgs/msg/twist.hpp>
#include <tf2/LinearMath/Quaternion.h>
#include <tf2/LinearMath/Matrix3x3.h>

std::shared_ptr<rclcpp::Node> node;
rclcpp::Publisher<geometry_msgs::msg::Twist>::SharedPtr vel_pub;
```

```
void IMUCallback(const sensor_msgs::msg::Imu::SharedPtr msg)
{
    tf2::Quaternion tf2_quaternion;
    tf2_quaternion.setX(msg->orientation.x);
    tf2_quaternion.setY(msg->orientation.y);
    tf2_quaternion.setZ(msg->orientation.z);
    tf2_quaternion.setW(msg->orientation.w);

    tf2::Matrix3x3 matrix(tf2_quaternion);

    double roll, pitch, yaw;
    matrix.getRPY(roll, pitch, yaw);
    roll = roll * 180 / M_PI;
    pitch = pitch * 180 / M_PI;
    yaw = yaw * 180 / M_PI;
    RCLCPP_INFO(node->get_logger(), "roll= %.0f pitch= %.0f yaw= %.0f", roll, pitch,
yaw);

    double target_yaw = 90;

    geometry_msgs::msg::Twist vel_msg;
    double diff_angle = target_yaw - yaw;
    vel_msg.angular.z = diff_angle * 0.01;
    vel_msg.linear.x = 0.1;
  vel_pub->publish(vel_msg);
}

int main(int argc, char** argv)
{
    rclcpp::init(argc, argv);

    node = rclcpp::Node::make_shared("imu_behavior_node");

    auto sub = node->create_subscription<sensor_msgs::msg::Imu>("imu", 10, IMUCallback);

    vel_pub = node->create_publisher<geometry_msgs::msg::Twist>("/cmd_vel", 10);

    rclcpp::spin(node);

    rclcpp::shutdown();

    return 0;
}
```

```
find_package(rclcpp REQUIRED)
find_package(sensor_msgs REQUIRED)
find_package(tf2 REQUIRED)
find_package(geometry_msgs REQUIRED)
```

```
add_executable(imu_behavior src/imu_behavior.cpp)
ament_target_dependencies(imu_behavior "rclcpp" "sensor_msgs" "tf2" "geometry_msgs")
```

```
install(TARGETS    imu_data    imu_behavior
DESTINATION    lib/${PROJECT_NAME})
```

```
<depend>rclcpp</depend>
<depend>sensor_msgs</depend>
<depend>tf2</depend>
<depend>geometry_msgs</depend>
```

```
cd ~/ros2_ws
```

```
colcon build
```

6.3.2 仿真运行航向锁定程序

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

```
ros2 launch wpr_simulation2 wpb_simple.launch.py
```

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

```
ros2 run imu_pkg imu_behavior
```

第 7 章 ROS2 中的 SLAM 环境建图

7.2 SLAM Toolbox 简介

```
sudo apt install ros-<ROS2 版本名称>-slam-toolbox
sudo apt install ros-humble-slam-toolbox
```

7.3 仿真环境实现 SLAM 建图

7.3.1 SLAM Toolbox 的启动

```
cd ~/ros2_ws/src
ros2 pkg create slam_pkg
```

新建文件为“slam.launch.py”。

```
import os
from launch import LaunchDescription
from launch_ros.actions import Node
from ament_index_python.packages import get_package_share_directory

def generate_launch_description():

    slam_params = {
        "use_sim_time": True,
        "base_frame": "base_footprint",
        "odom_frame": "odom",
        "map_frame": "map"
    }
    slam_cmd = Node(
        package="slam_toolbox",
        executable="sync_slam_toolbox_node",
        parameters=[slam_params]
    )

    rviz_file = os.path.join(get_package_share_directory('wpr_simulation2'), 'rviz', 'slam.rviz')
    rviz_cmd = Node(
        package='rviz2',
        executable='rviz2',
        name='rviz2',
        arguments=['-d', rviz_file]
    )

    ld = LaunchDescription()
```

```
    ld.add_action(slam_cmd)
    ld.add_action(rviz_cmd)

    return ld
```

```
install(
DIRECTORY
  launch
DESTINATION
  share/${PROJECT_NAME})
```

```
cd ~/ros2_ws
colcon build
```

7.3.2 仿真环境建图

```
source install/setup.bash
ros2 launch wpr_simulation2 robocup_home.launch.py
```

```
source install/setup.bash
ros2 launch slam_pkg slam.launch.py
```

```
source install/setup.bash
ros2 run wpr_simulation2 keyboard_vel_cmd
```

7.3.3 地图的保存

```
ros2 run nav2_map_server map_saver_cli -f map
```

7.3.4 SLAM Toolbox 的参数设置

```
https://github.com/SteveMacenski/slam_toolbox
https://gitee.com/s-robot/slam_toolbox
```

```
cd ~/ros2_ws
```

```
colcon build
```

```
ros2 launch wpr_simulation2 robocup_home.launch.py
ros2 launch slam_pkg slam.launch.py
ros2 run wpr_simulation2 keyboard_vel_cmd
```

第 8 章 ROS2 中的 NAV2 自主导航

8.2 使用 NAV2 进行自主导航

8.2.1 NAV2 的安装

```
sudo apt install ros-<ros2-distro>-navigation2
sudo apt install ros-humble-navigation2
```

```
sudo apt install ros-<ros2-distro>-nav2-bringup
sudo apt install ros-humble-nav2-bringup
```

8.2.2 使用 NAV2 实现自主导航

```
source ~/ros2_ws/install/setup.bash
ros2 launch wpr_simulation2 slam.launch.py
```

```
source ~/ros2_ws/install/setup.bash
ros2 run wpr_simulation2 keyboard_vel_cmd
```

```
ros2 run nav2_map_server map_saver_cli -f map
```

```
cd ~/ros2_ws/src
ros2 pkg create nav_pkg
```

新的 Launch 文件命名为 nav.launch.py。

```
import os
from launch import LaunchDescription
from launch_ros.actions import Node
from ament_index_python.packages import get_package_share_directory
from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource

def generate_launch_description():

    map_file = os.path.join(
        get_package_share_directory('wpr_simulation2'),
        'maps',
        'map.yaml'
    )

    nav_param_file = os.path.join(
        get_package_share_directory('wpr_simulation2'),
```

```
        'config',
        'nav2_params.yaml'
    )

    nav2_launch_dir = os.path.join(
        get_package_share_directory('nav2_bringup'),
        'launch'
    )

    navigation_cmd = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([nav2_launch_dir, '/bringup_launch.py']),
        launch_arguments={
            'map': map_file,
            'use_sim_time': 'True',
            'params_file': nav_param_file}.items(),
    )

    rviz_file = os.path.join(get_package_share_directory('wpr_simulation2'), 'rviz', 'navi.rviz')
    rviz_cmd = Node(
            package='rviz2',
            executable='rviz2',
            name='rviz2',
            arguments=['-d', rviz_file]
        )

    ld = LaunchDescription()
    ld.add_action(navigation_cmd)
    ld.add_action(rviz_cmd)

    return ld
```

```
install(
DIRECTORY
  launch
DESTINATION
  share/${PROJECT_NAME})
```

```
cd ~/ros2_ws
colcon build
```

8.2.3 仿真环境运行自主导航

```
ros2 launch wpr_simulation2 robocup_home.launch.py
source install/setup.bash
```

```
ros2 launch nav_pkg nav.launch.py
```

8.3 开源导航插件的使用

8.3.1 安装导航插件

```
cd ~/ros2_ws/src
git clone https://github.com/6-robot/wp_map_tools.git
git clone https://gitee.com/s-robot/wp_map_tools.git
```

```
cd ~/ros2_ws/src/wp_map_tools/scripts/
./install_for_humble.sh
```

```
cd ~/ros2_ws/
colcon build
```

8.3.2 添加航点

```
cd ~/ros2_ws
source install/setup.bash
ros2 launch wp_map_tools add_waypoint_sim.launch.py
```

```
source install/setup.bash
ros2 run wp_map_tools wp_saver
```

8.3.3 启动导航服务

创建一个 Launch 文件“waypoint_nav.launch.py”。

```
import os
from launch import LaunchDescription
from launch_ros.actions import Node
from ament_index_python.packages import get_package_share_directory
from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource

def generate_launch_description():

    map_file = os.path.join(
        get_package_share_directory('wpr_simulation2'),
        'maps',
        'map.yaml'
    )

    nav_param_file = os.path.join(
        get_package_share_directory('wpr_simulation2'),
```