SCOUT MINI Lite 教育套装

SCOUT MINI Lite 科研教育套装开发手册

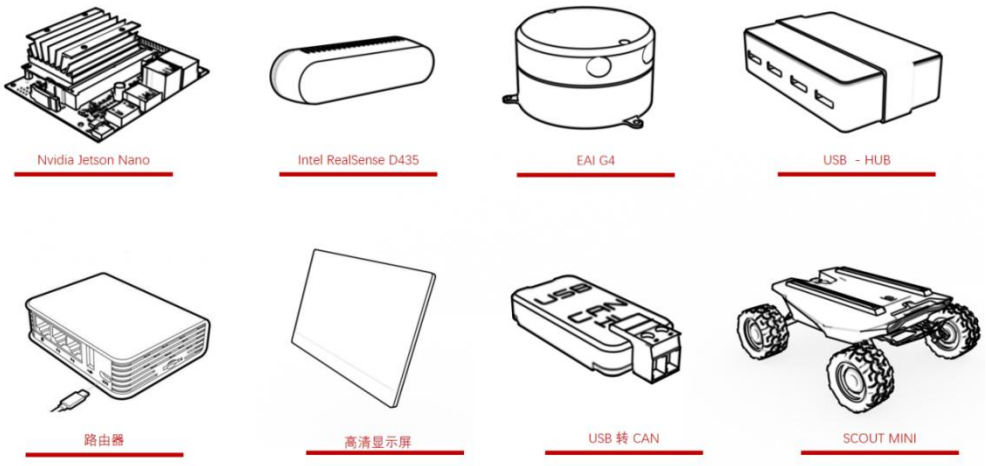
产品概述

SCOUT MINI Lite 是松灵机器人专为 ROS 科研教育应用研发定制的 ROS 开发者入门版本和进阶套装，该套装基于松灵机器人 ROS 生态体系统，集成高性能工控、高精度 LiDAR、多传感器搭配于一身，可实现移动机器人运动控制、通讯、导航、地图构建等应用，提供完善的开发者文档及 DEMO 资源，轻巧便携，极具科技感的工业设计，专属传感器定制支架，为您提供教育科研、产品预研、课题、产品论证等多方向应用快速 ROS 二次开发最佳实验平台。

1、配置清单以及参数说明

1.1、SCOUT MINI Lite

名称	数量	型号
工控机	1	Nvidia Nano 4G
激光传感器	1	EAI-G4
视觉传感器	1	Intel RealSense D435
路由器	1	GL.iNet GL-AR750S
底盘	1	SCOUT MINI
遥控器	1	FS-I6S
充电器	1	Agilex UY360
稳压模块	1	24V 转 12V
稳压模块	1	12V 转 5V
高清显示屏	1	
USB 转 CAN	1	
USB HUB	1	

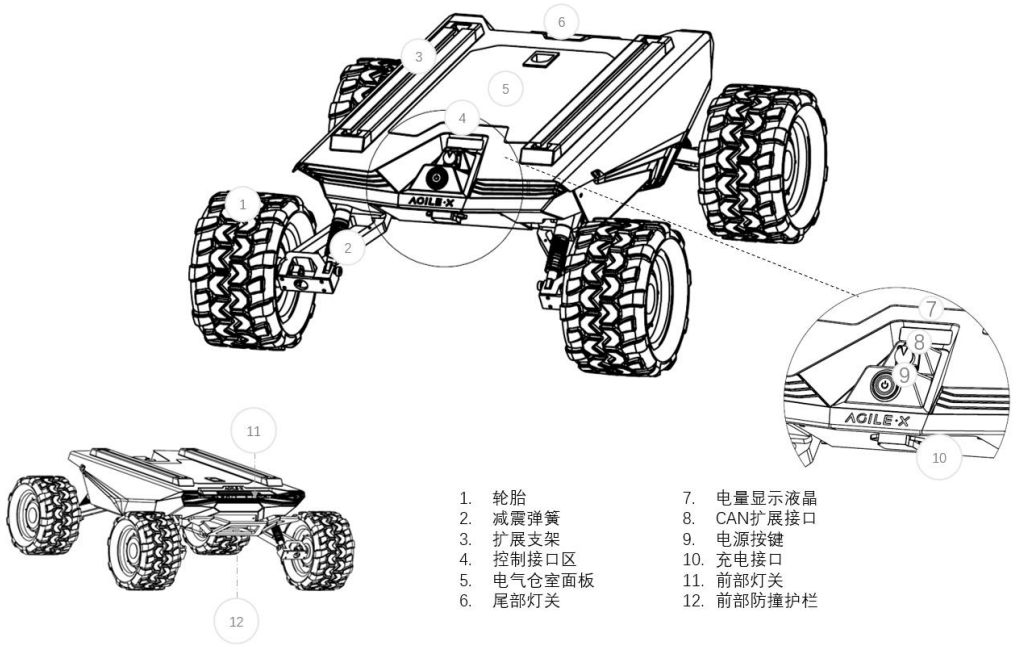


SCOUT MINI Lite 主要配置图

2、SCOUT MINI 产品介绍

2.1、SCOUT MINI 简介

SCOUT MINI 智能移动底盘采用四轮四驱，具备强悍的越野性能，身形小巧，真正实现“灵巧似燕，驰骋如心”。SCOUT MINI 继承了 SCOUT 四轮差速底盘系列四轮驱动、独立悬挂、原地自转等优点，并在轮毂电机的设计上取得了创新，底盘最小转弯半径为 0m，爬坡角度接近 30 度。SCOUT MINI 在体积上比 SCOUT 缩小一半的同时，仍具备卓越的越野性能，同时突破性实现了 10.8km/h 的高速精准稳定可控的动力控制系统。SCOUT MINI 开发平台自带控制核心，支持标准 CAN 总线通讯，可接入标准 CAN 总线通讯，以及各类外部设备，在此基础上支持 ROS/Autoware 二次开发和更高级的及机器人开发系统的接入。配置有标准航模航空器，24V@15Ah 锂电池动力系统，续航里程可达 10KM。



1. 轮胎
2. 减震弹簧
3. 扩展支架
4. 控制接口区
5. 电气仓室面板
6. 尾部灯关
7. 电量显示液晶
8. CAN扩展接口
9. 电源按键
10. 充电接口
11. 前部灯关
12. 前部防撞护栏

2.2、SCOUT MINI 快速上手使用

2.2.1、检查 SCOUT MINI

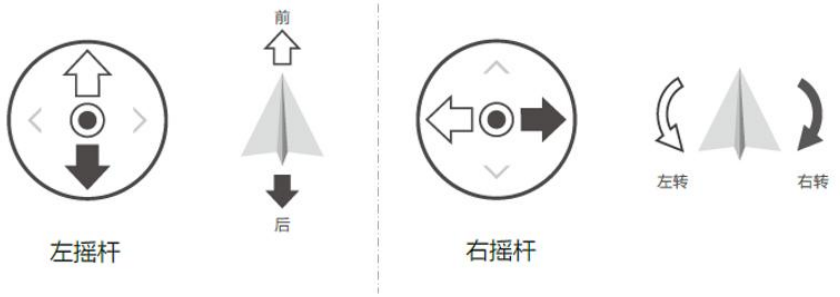
- 按下机身电源按键，等待数秒
- 查看电量显示液晶，看电量是否低于 30%，若低于 30%，请充电
- 若电量低于 30%，请使用标准配置的充电器给车辆充电，充电时关闭小车电源
- 从低电量充电至满电状态需要约 1 个半小时

2.2.2、准备遥控器

- 检查遥控器电池是否已经安装
- 将拨杆 SWA、SWB、SWC、SWD 拨至最上方
- 同时按住电源开关按键 1 和 2 直至开机

2.2.3、使用遥控器

遥控器出厂已经预置了按键的映射，请勿随意更改按键映射，更改可能会导致无法正常控制。拨杆 SWB 切换控制模式，SWC 为手动灯关控制开关，SWD 控制速度模式，左摇杆控制前进后退，右遥控控制车子左旋转和右旋转。值得注意得是，在内部控制上移动底盘是根据百分比映射的，因此当摇杆处于同一个位置时，其速度是恒定的。



- SWB 为控制模式切换按钮，处于最上方为指令模式，处于中间为遥控控制模式，下方为恒速模式。
- SWC 为灯关控制按钮，处于底部时为常闭模式，中间时为常开模式，上方为呼吸灯模式。注意灯光控制选项设置仅在遥控控制模式下有效，其他模式无效。
- SWD 为速度挡位选择模式，处于上方为低速挡模式(最快速度约 10km/h)，下方为高速档（最快速度约为 20km/h）。注意挡位设置按钮仅在遥控控制模式下有效，其他模式无效。

2.2.4、准备 SCOUT MINI

首次操作使用 SCOUT MINI 请在相对开阔的区域进行，以免使用不当，造成不必要的问题。

- 按下 SCOUT MINI 电源按键，等待数秒即可；
- 将 SWB 拨至中间，选址要控制控制挡位；
- 可尝试手动切换灯光模式，确定模式选择时候正确；
- 尝试将左边摇杆轻轻往前推，推一小部分即可，可见小车缓慢速度往前移动；
- 尝试将左边摇杆轻轻往后推，推一小部分即可，可见小车缓慢速度往后移动；
- 释放左边摇杆，小车停下；
- 尝试将右边摇杆轻轻往左推，推一小部分即可，可见小车缓慢往左旋转；
- 尝试将右边摇杆轻轻往右推，推一小部分即可，可见小车缓慢往右旋转；
- 释放右边摇杆，小车停下；
- 可尝试在相对空旷的区域自由控制，熟悉车辆移动速度。

2.2.5、关闭 SCOUT MINI

按下 SCOUT MINI 电源按键，释放即可。

2.2.6、关闭遥控器

同时按下遥控器电源开关按键 1/2 数秒，即可关闭遥控器

2.3、SCOUT MINI 基本参数说明

SCOUT MINI 基本参数说明	
尺寸	627mm × 550mm × 252mm
轴距	452mm
轮距	450mm
整备质量	20kg
最小离地间隙	107mm
额定进行载重	10kg(摩擦系数 0.5 地面测试)
最高速度	20km/h
最小转弯半径	0(可原地自旋)
最大爬坡角度	30°(带负载)
越障能力	70mm
最大行程	10km(空载)
驱动形式	四轮独立驱动
工作温度	-20℃~60℃
充电器	AC 220 独立充电器
充电时间	1.5H
对外供点	24V
电池参数	24V/15Ah
码盘参数	1024 光电增量式码盘
通讯接口	标准 CAN
防护等级	IP22(可定制 IP64)
悬挂形式	摇摆臂独立悬挂

3、Nvidia Jetson Nano 介绍

Nvidia Jetson Nano 是一款功能强大的小型计算机，专为支持入门级边缘 AI 应用程序和设备而设计。依托完善的 NVIDIA JetPack™ SDK 包含用于深度学习、计算机视觉、图形、多媒体等方面的加速库。搭载在 SCOUT MINI Lite 版本，可以用于拓展机器人导航定位、图像处理、语音识别等技术的拓展。

GPU	128-Core Maxwell
CPU	Quad-core ARM57 @1.43Ghz
内存	4GB 64Bit LPDDR4 25.6GB/s
存储	Micro SD 卡（默认）
视频编码	4K@30	4 X 1080p@30	9 X 720p@30(H.264/H.265)
视频解码	4K@60| 2X 4K@30	8X 1080p@30	18 X 720p@30(H.264/H.265)
摄像头	2 X MIPI CSI-2 DPHY lanes
联网	千兆以太网，M.2 Key E 接口外扩
显示	HDMI X 1, DP X 1
USB	4 X USB 3.0, USB 2.0 Micro-B
扩展接口	GPIO，I2C，I2S，SPI，UART

4、Intel RealSense D435 介绍

双目视觉传感器，在机器人视觉测量、视觉导航等机器人行业方向中均有大范围的应用场景和需求，目前我们甄选了了在科研教育行业常见的视觉传感器。英特尔实感深度摄像头 D435 配备全局图像快门和宽视野，能够有效地捕获和串流移动物体的深度数据，从而为移动原型提供高度准确的深度感知。

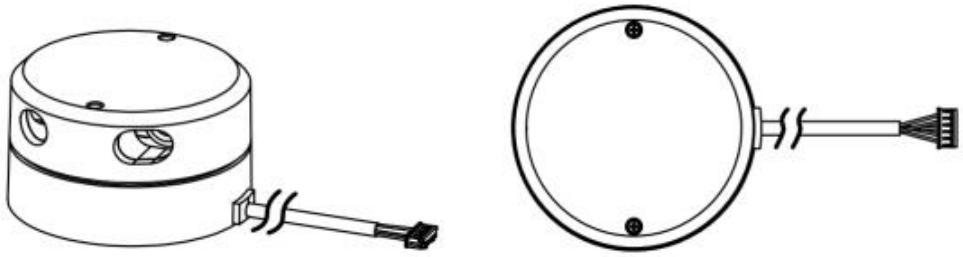
	型号	Intel Realsense D435
基本特征	应用场景	户外/室内
	测量距离	约 10 米
	深度快门类型	全局快门/3um X 3um
	是否支持 IMU	否
深度相机	深度技术	有缘红外
	FOV	86° x 57°（±3°）
	最小深度距离	0.105m
	深度分辨率	1280 x 720
	最大测量距离	约 10 米

RGB	深度帧率	90 fps
	分辨率	1280 x 800
	FOV	69.4° × 42.5°（±3°）
	帧率	30fps
其他信息	尺寸	90mm x 25mm x 25mm
	接口类型	USB-C 3.1

5、EAI-G4 介绍

YDLIDAR G4 激光雷达是一款 360 度二维测距产品。G4 基于三角测距原理，并配以相关光学、电学、算法设计，实现高频高精度的距离测量，在测距的同时，机械结构 360 度旋转，不断获取角度信息，从而实现了 360 度扫描测距，输出扫描环境的点云数据。

EAI G4 激光雷达参数	
项目	参数
测距频率	4000~9000Hz
扫描频率	5~12Hz
测距半径	0.1~16m
扫描角度	360°
角度分辨率	0.26~0.30°
绝对误差	2cm
供电电压	4.8v
激光器波长	775nm

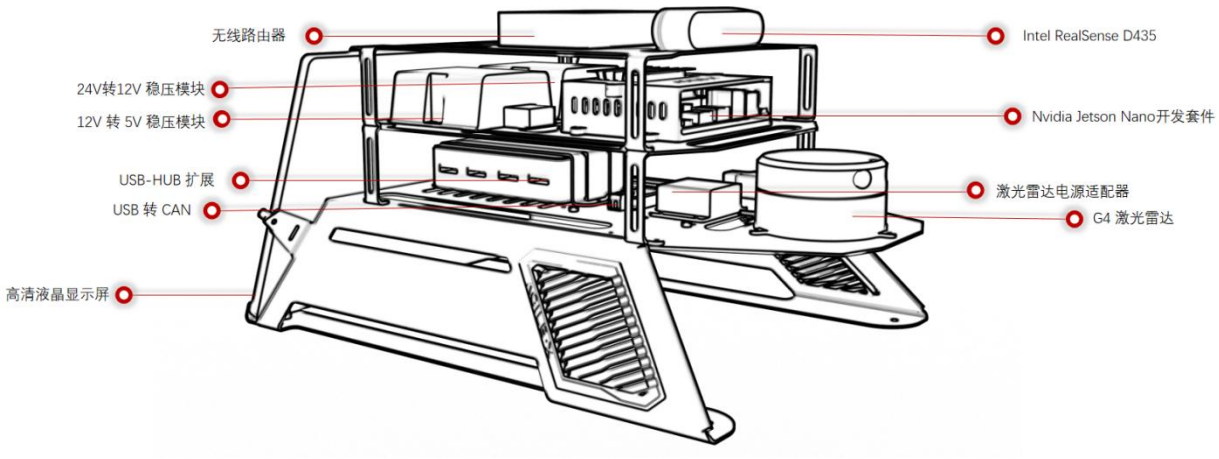


6、硬件安装说明和电气说明

6.1、SCOUT MINI Lite 科研教育套装支架安装说明

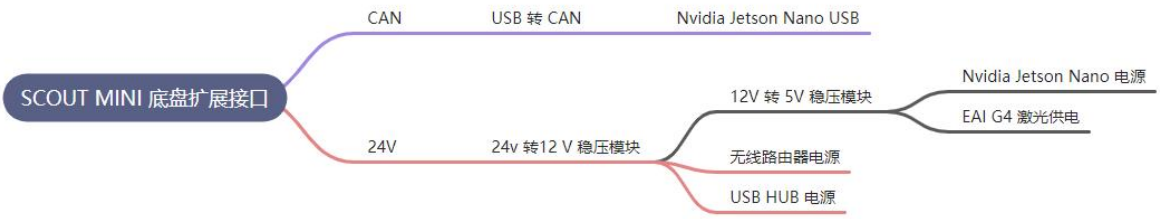
SCOUT MINI Lite 科研教育套件支架整体设计上采用了堆叠式设计，框架采用钣金作为

支撑件，便于客户进行拓展连接，尾部安装了高清显示屏，便于客户调试。底部采用镂空的钣金支架，与 SCOUT MINI 顶部的标准铝型材支架进行装配。第二层主要有 USB-HUB 拓展接口（独立供电）、USB 转 CAN 模块以及 G4 激光雷达。第三层主要为计算单元和稳压模块，顶层目前主要安装有 Intel RealSense D435。具体安装位置与示意图可以参考下图。

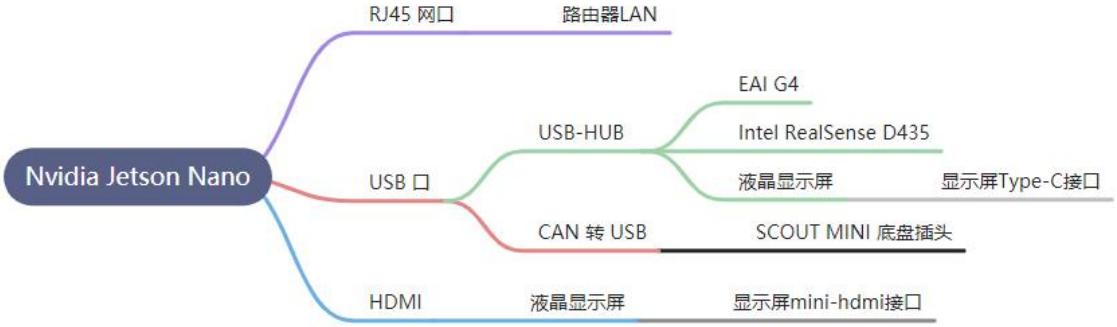


6.2、电气通讯拓扑连接说明

SCOUT MINI 通过底盘的航空拓展接口为上部设备提供电源和通讯接口。SCOUT MINI 底盘电源拓展接口（最大电源输出支持 24V@5A）的供电来自于底盘的电池，未经稳压和调压模块，为了给其他拓展设备供电的时，需要增加稳压调压设备，为了便于客户在后期使用的时候拓展，我们选择了 12V@20A 的稳压模块，稳压主要为 5V 稳压模块、无线路由器、USB-HUB 提供电源输入，5@10A 稳压模块为 EAI-G4 激光雷达与 Nvidia Jetson Nano 提供电源输入。



Nvidia Jetson Nano 为核心计算单元的外部拓扑连接如下示意图。Nano 的网口与路由器网口连接，便于远程桌面连接，访问调试，同时便于拓展其他的网络设备；在 USB 拓展连接上，我们采用了外部增加了 USB-HUB（带外部电源独立供电）的设计，由于 Nano 本身的 USB 带载能力相对较差，故采用此设计，能有效提高 USB 带载能力，USB-HUB 主要拓展连接了 EAI-G4 激光雷达、D435 双目摄像头、液晶显示屏等。



6.3、传感器拓展

外部拓展主要涉及机械安装拓展、电源供电拓展、以及通讯拓展。电源供电拓展，我们在前期电源稳压模块选型时便将此问题纳入考虑，故电源稳压模块均预留有一定的余量；通讯拓展上，设备上增加了 USB-HUB、无线网关，均可拓展接入更多的设备。

7、开发指南

7.1、ROS 开发的基本介绍

7.1.1、ROS 历史介绍

硬件技术的飞速发展在促进机器人领域快速发展和复杂化的同时，也对机器人系统的软件开发提出了巨大挑战。机器人平台与硬件设备越来越丰富，致使软件代码的复用性和模块化需求越发强烈，而已有的机器人系统又不能很好地适应需求。相比硬件开发，软件开发明显力不从心。为迎接机器人软件开发面临的巨大挑战，全球各地的开发者与研究机构纷纷投入机器人通用软件框架的研发工作当中。在近几年里，产生了多种优秀的机器人软件框架，为软件开发工作提供了极大的便利，其中最为优秀的软件框架之一就是机器人操作系统（Robot Operating System，ROS）。ROS 是一个用于编写机器人软件的灵活框架，它集成了大量的工具、库、协议，提供了类似操作系统所提供的功能，包括硬件抽象描述、底层驱动程序管理、共用功能的执行、程序间的消息传递、程序发行包管理，可以极大简化繁杂多样的机器人平台下的复杂任务创建与稳定行为控制。

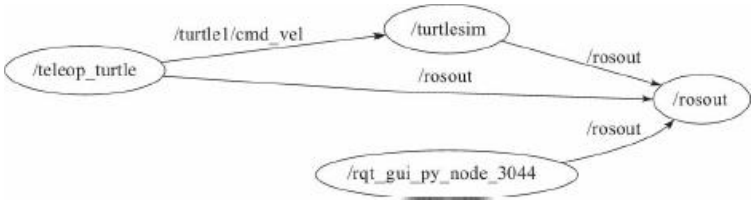
7.1.2、ROS 基本概念

ROS 的概念分为三个层次：文件系统层、计算图层、社区层。ROS 的设计目标是提高机器人研发中的软件复用率，所以它被设计成为一种分布式结构，使得框架中的每个功能模块都可以被单独设计、 编译，并且在运行时以松散耦合的方式结合在一起。ROS 主要为机器人 开发提供硬件抽象、底层驱动、消息传递、程序管理、应用原型等功能和机制，同时

整合了许多第三方工具和库文件，帮助用户快速完成机器人应用的建立、编写和多机整合。而且 ROS 中的功能模块都封装于独立的功能包（Package）或元功能包（Meta Package）中，便于在社区中共享和分发。

7.1.3、ROS 节点

ROS 是一个用于编写机器人软件的灵活框架，它集成了大量的工具、库、协议，提供了类似操作系统所提供的功能，包括硬件抽象描述、底层驱动程序管理、共用功能的执行、程序间的消息传递、程序发行包管理，可以极大简化繁杂多样的机器人平台下的复杂任务创建与稳定行为控制。的通信绘制成如图所示的节点关系图，在这个图中进程就是图中的节点，而端对端的连接关系就是节点之间的连线。

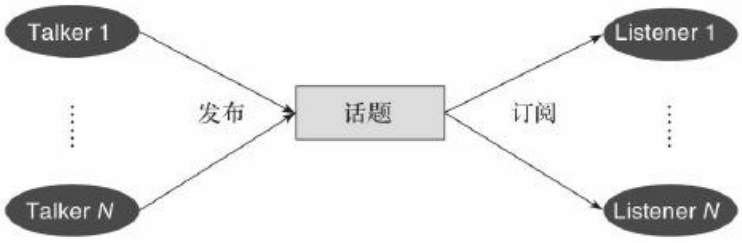


7.1.4、ROS 消息

节点之间最重要的通信机制就是基于发布/订阅模型的消息（Message）通信。每一个消息都是一种严格的数据结构，支持标准数据类型（整型、浮点型、布尔型等），也支持嵌套结构和数组（类似于 C 语言的结构体 struct），还可以根据需求由开发者自主定义。

7.1.5、ROS 话题

消息以一种发布/订阅（Publish/Subscribe）的方式传递（见图 2- 4）。一个节点可以针对一个给定的话题（Topic）发布消息（称为发布者/Talker），也可以关注某个话题并订阅特定类型的数据（称为订阅者/Listener）。发布者和订阅者并不了解彼此的存在，系统中可能同时有多个节点发布或者订阅同一个话题的消息。



7.1.6、ROS 服务

虽然基于话题的发布/订阅模型是一种很灵活的通信模式，但是对于双向的同步传输模式并不适合。在 ROS 中，我们称这种同步传输模式为服务（Service），其基于客户端/服务器（Client/Server）模型，包含两个部分的通信数据类型：一个用于请求，另一个用于应答，

类似于 Web 服务器。与话题不同的是，ROS 中只允许有一个节点提供指定命名的服务。

7.1.7、ROS 文件系统

类似于操作系统，ROS 将所有文件按照一定的规则进行组织，不同功能的文件被放置在不同的文件夹下，如图 2-5 所示。

功能包（Package）：功能包是 ROS 软件中的基本单元，包含 ROS 节点、库、配置文件等。

功能包清单（Package Manifest）：每个功能包都包含一个名为 package.xml 的功能包清单，用于记录功能包的基本信息，包含作者信息、许可信息、依赖选项、编译标志等。

元功能包（Meta Package）：在新版本的 ROS 中，将原有功能包集（Stack）的概念升级为“元功能包”，主要作用都是组织多个用于同一目的的功能包。例如一个 ROS 导航的元功能包中会包含建模、定位、导航等多个功能包。

元功能包清单：在图 2-5 中并未显示，类似于功能包清单，不同之处在于元功能包清单中可能会包含运行时需要依赖的功能包或者声明一些引用的标签。

8、开机启动与关机

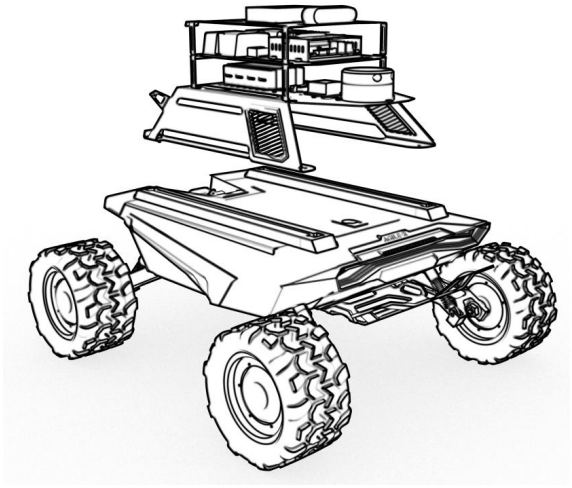
8.1、安装

8.1.1、工具准备

内六角工具一套

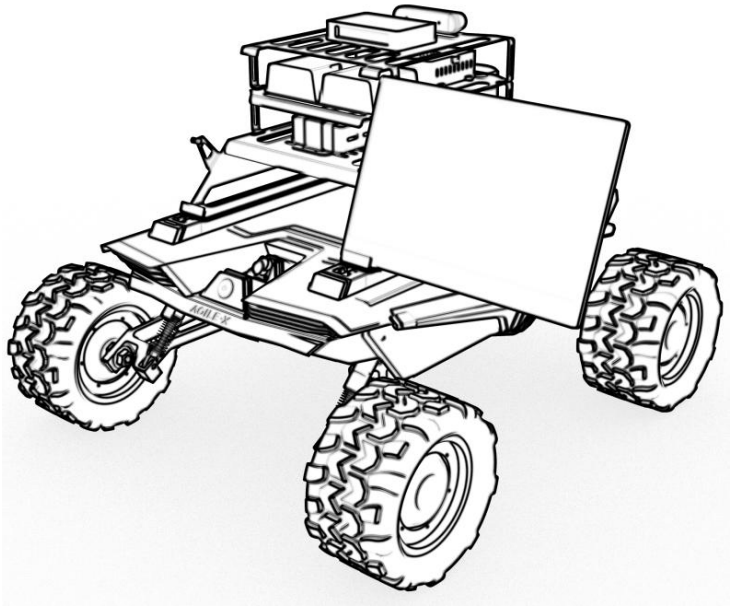
8.1.2、套件与底盘的固定

出库时传感器支架与车身分开，用户需使用工具将传感器支架与车身行台固定，首先将四块滑块螺母平分插入两边行台，随后用内六角工具将依次对应的四个螺丝与行台上滑块螺母拧紧固定，如下图所示。



8.1.3、屏幕的安装

将高清显示屏横向插入显示器支架，如下图所示。随后用螺丝和尼龙柱从显示器支架两边突出的固定插孔固定显示屏。



8.2、开机前检查

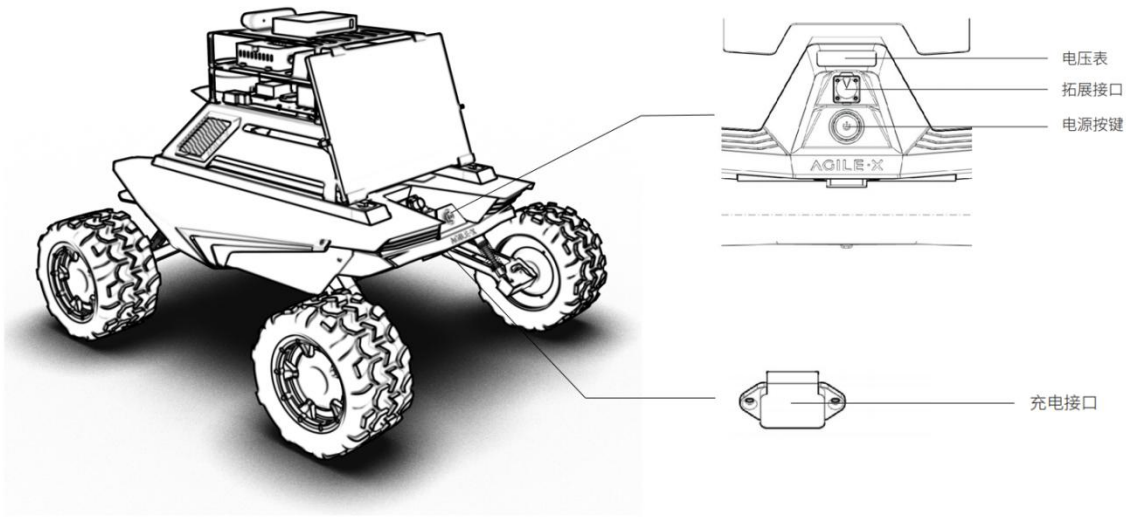
- 检查所有车辆上的线束连接是否有明显的脱落;
- 开机前请确保在相对空旷和无大面积积水的环境下操作，环境相对稳定，周边无易燃易爆等危险品;
- 机器整体完整，线束完好无断裂，传感器无损坏。

8.3、插入鼠标键盘

本科研教育套装不配备鼠标和键盘，用户可根据需求自行配置，可与计算单元的 USB 接口或者 USB HUB 拓展设备进行连接。

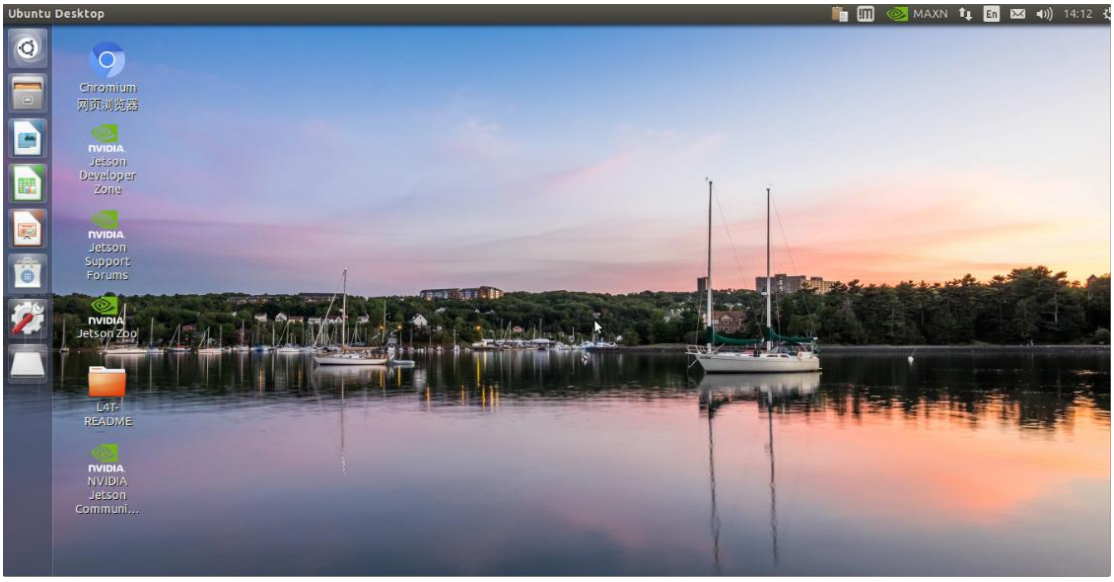
8.4、开启电源与上电

按下车身上的电源按键上电，如下图所示。



8.5、车载电脑登入

按下车身电源按键上电后，工控机将自动登录并显示如下界面，根权限密码和系统登入密码为 agx



8.6、关机

如需关闭系统，因为工控机运行过程中不能直接断电，所以请勿直接按下车身的电源按钮。在终端输入命令 $poweroff 或者点击右上角设置下拉菜单 shut down 关闭工控机，如下图所示，待显示器显示无信号输入时方可按下电源按键关机。

9、SCOUT MINI Lite 的基本开发环境介绍

（Ubuntu 系统版本，ROS 版本，Gazebo 版本，RVIZ 版本）

Ubuntu 系统版本	16.04
ROS 版本	kinetic
Gazebo 版本	7.0
RVIZ 版本	-

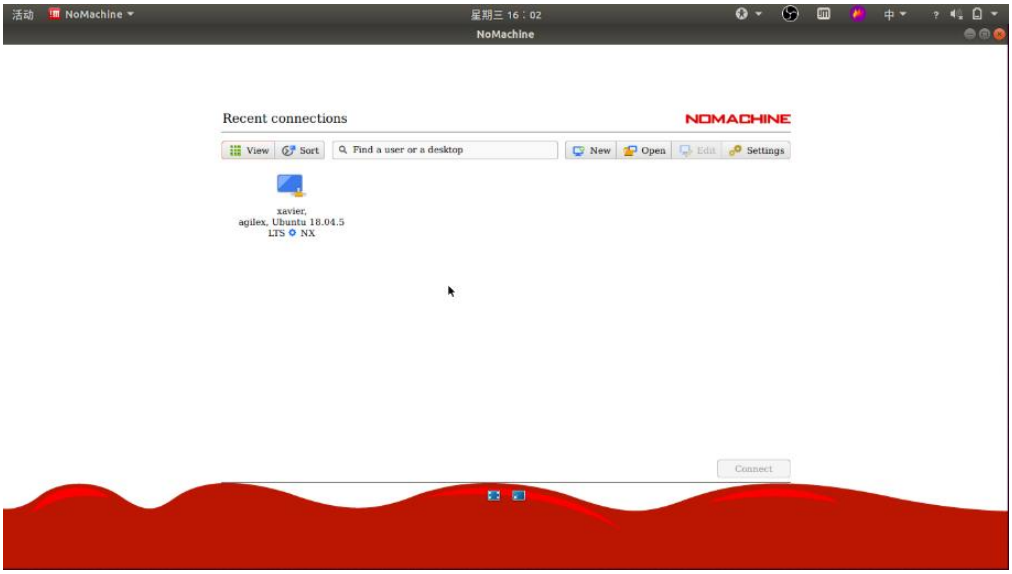
Ubuntu 系统版本	18.04
ROS 版本	melodic
Gazebo 版本	9.0
RVIZ 版本	-

10、远程桌面的使用和连接

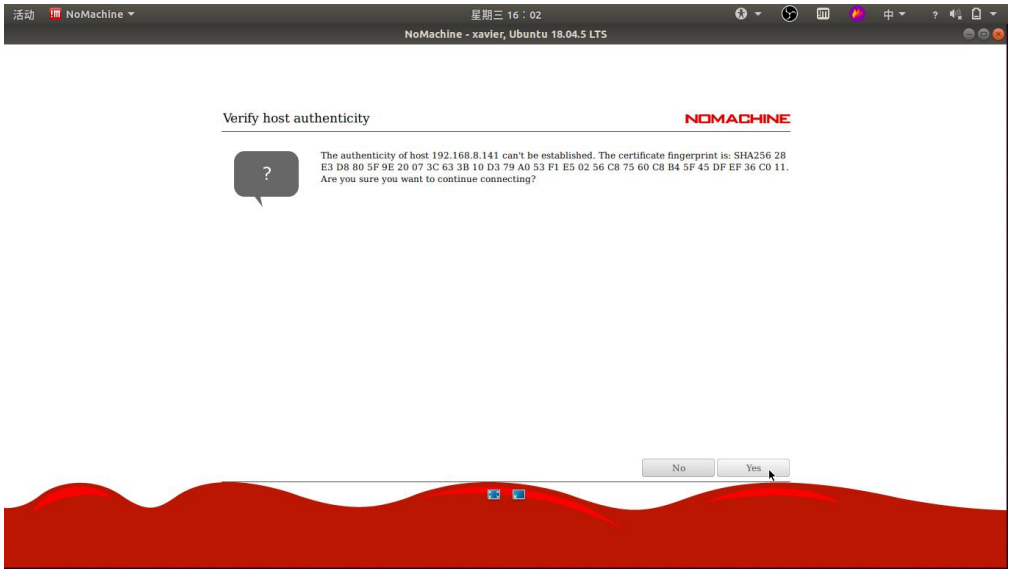
打开主机电脑无线网络设置，显示有来自路由器 GL-AR750 两个信号，请选择连接 5G 频段，密码为 12345678。

10.1、远程连接 Nano(请确保车载计算单元已经开机)

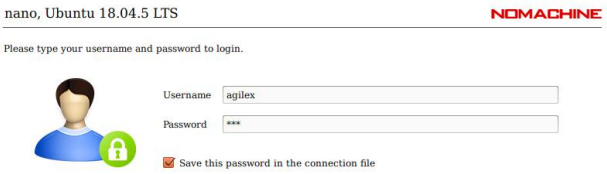
- 下载安装 NoMachine
- 首先在个人电脑下载相应的软件。
- 下载链接：https://www.nomachine.com/download
- 让自己电脑连接到小车的 wifi（让小车与个人电脑在同一个局域网即可）
- 打开已经下载好的 NoMachine



- 点击 Yes



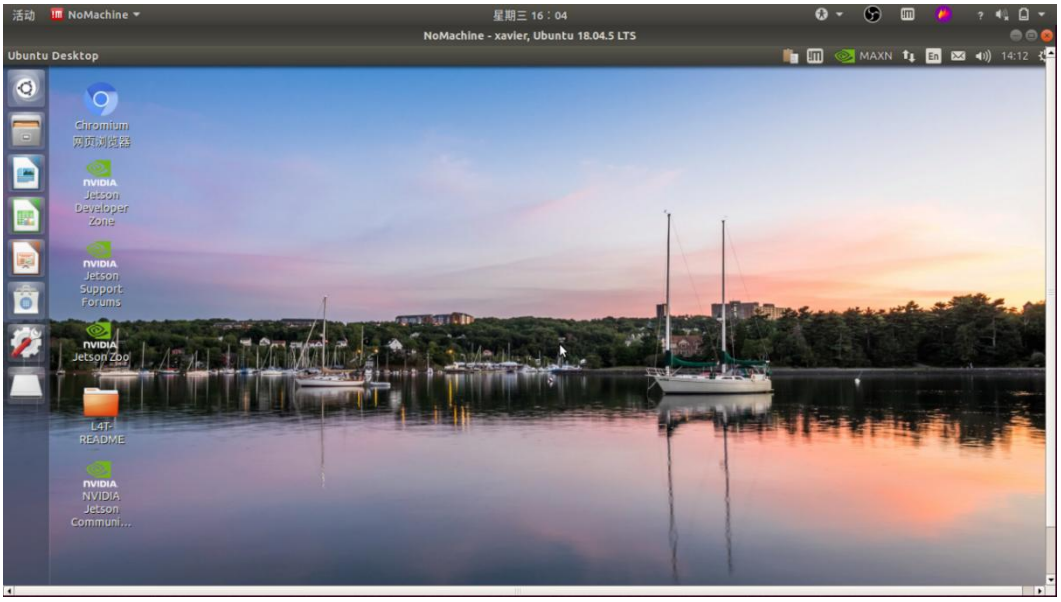
- Username：agilex
- Password：agx
- 勾选保存密码



- 一路选择默认 OK

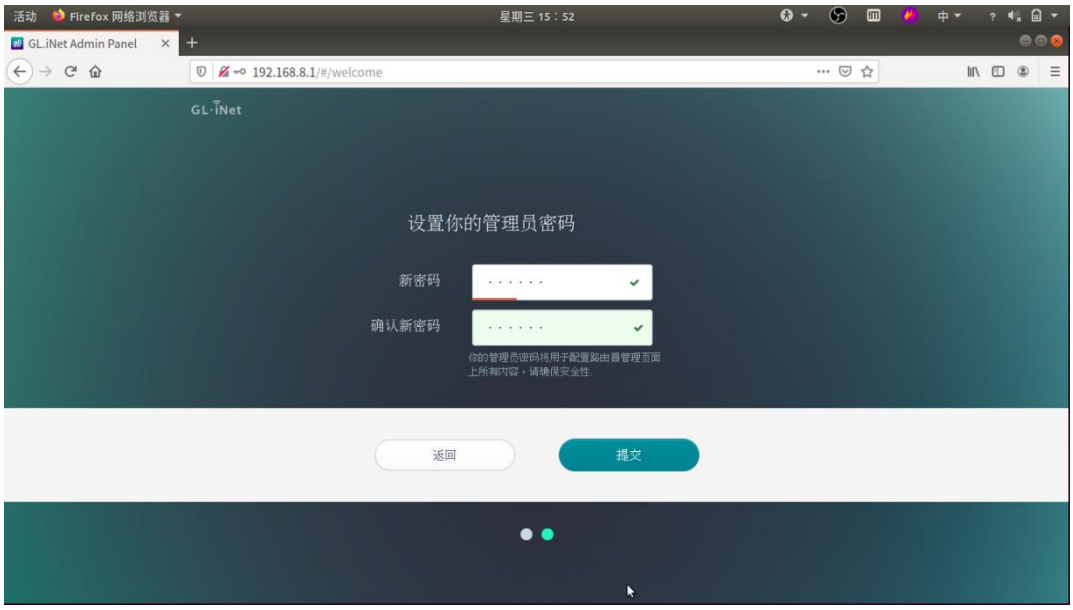
- 输入默认密码 agx

- 连接成功

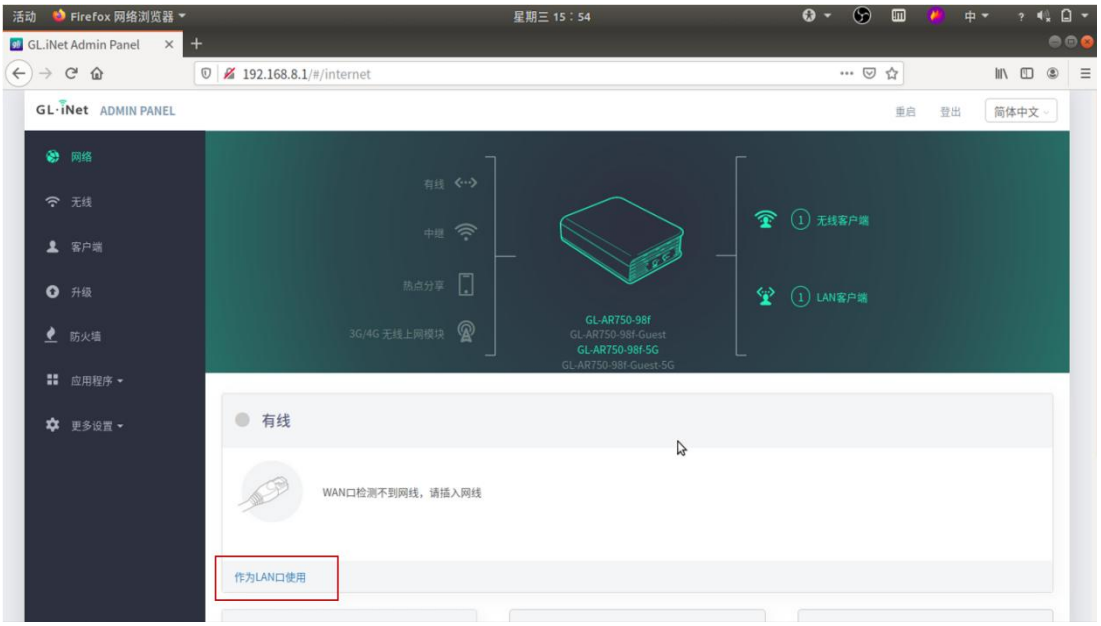


10.2、网口连接远程桌面访问

进入路由器，默认地址为 192.168.8.1，选择语言并设置密码。如果 192.168.8.1 登录不上的话，就用 192.168.1.1 登录。



成功进入路由器设置，点击作为 LAN 口使用



成功设置为 LAN 口模式，可用网线将套件与主机网口相连进行调试访问

10.3、ROS 的安装

安装源

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

或来自中国的源

```
sudo sh -c '. /etc/lsb-release && echo "deb http://mirrors.ustc.edu.cn/ros/ubuntu/ $DISTRIB_CODENAME main" > /etc/apt/sources.list.d/ros-latest.list'
```

设置 key

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6B ADE8868B172B4F42ED6FBAB17C654
```

更新

```
sudo apt-get update
```

ROS Desktop-Full 安装

Ubuntu16.04

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

Ubuntu18.04

```
sudo apt-get install ros-melodic-desktop-full
```

解决依赖

```
sudo rosdep in
rosdep update
```

环境设置

Ubuntu16.04

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Ubuntu18.04

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

安装 rosinstall，便利的工具

```
sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

11、传感器的基础节点使用

11.1、USB 转 CAN 驱动的安装和测试

使能 gs_usb 内核模块

```
sudo modprobe gs_usb
```

打开 can 设备且设置波特率

```
sudo ip link set can0 up type can bitrate 500000
```

如果在前面的步骤中没有发生错误，那么您现在应该可以通过使用 command 查看 CAN

```
ifconfig -a
```

安装和使用 can-utils 工具来测试硬件

```
sudo apt install can-utils
```

测试指令

```
# receiving data from can0
$ candump can0
# send data to can0
$ cansend can0 001#1122334455667788
```

11.1、SCOUT MINI 底盘的节点 ROS Package

scout_bringup: 启动和配置底盘的 ROS 节点。
scout_base: 基于 ugv_sdk 控制和监测底盘的功能包。
scout_description: scout_mini 的 URDF 模型，可供搭载传感器的定制机器人仿真使用。
scout_msgs:scout_mini 相关消息格式的定义。

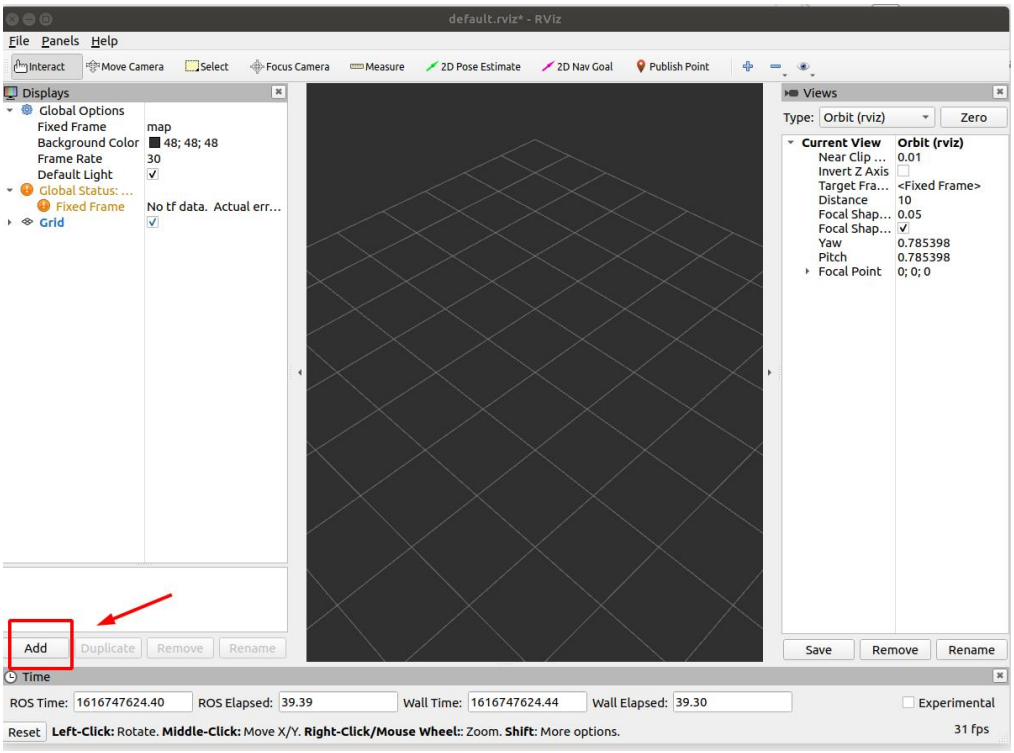
11.2、RealSense D435 节点 ROS Package 以及 RVIZ 可视化

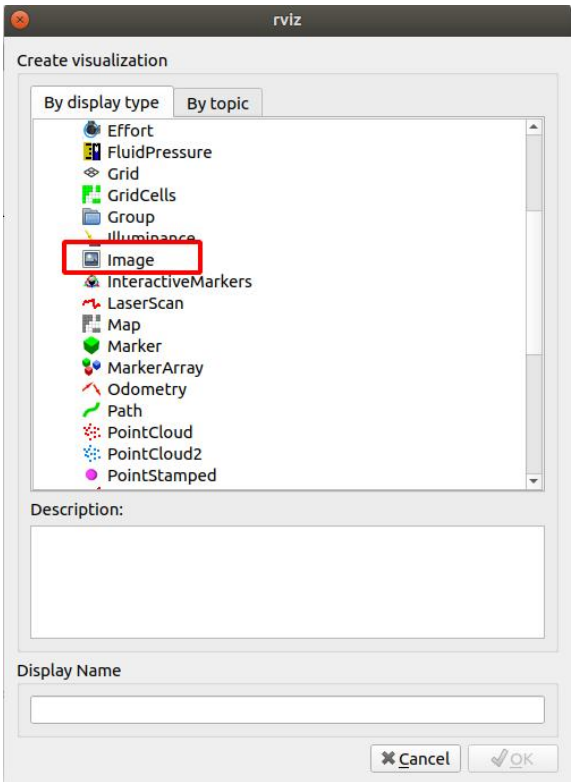
realsense2_camera: 设置与启动深度相机功能包，并且可以使用 rviz 可视化工具查看彩色图、深度图、稠密点云图等等。

获取 rgb 图

```
roslaunch realsense2_camera rs_camera.launch
```

终端输入 rviz 打开 rviz，点击 Add 添加 image 组件





Fixed frame 选择 camera_link

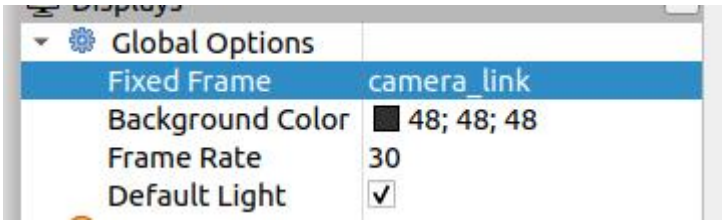
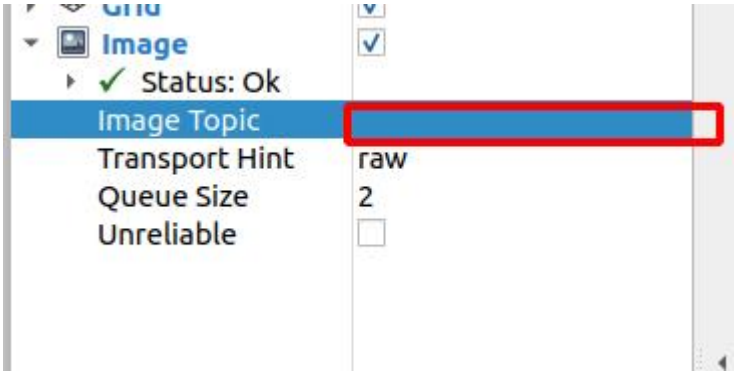
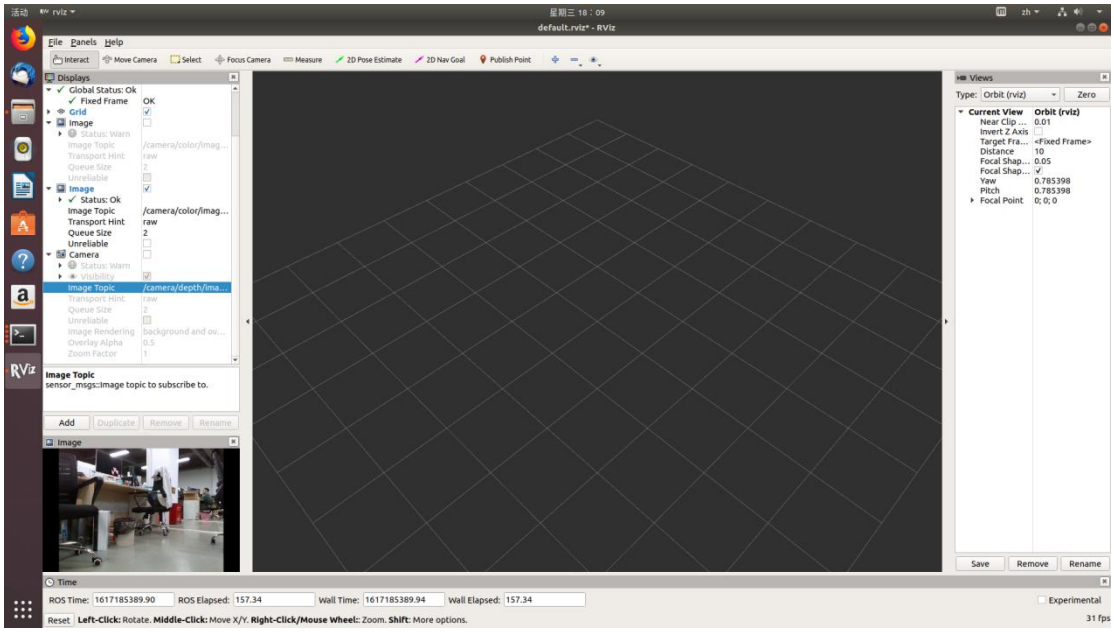


image 组件填入对应的话题获取 rgb 图片

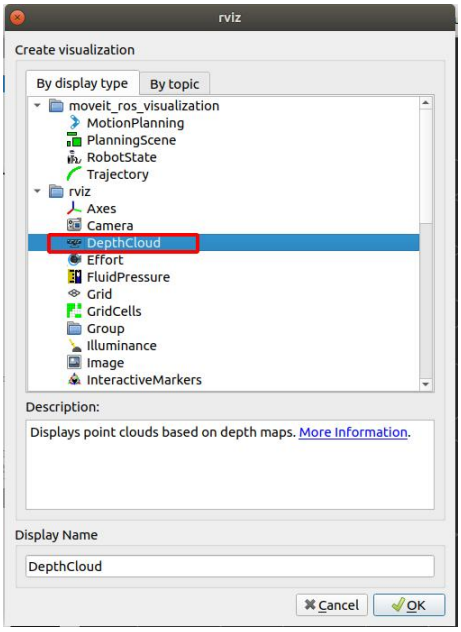




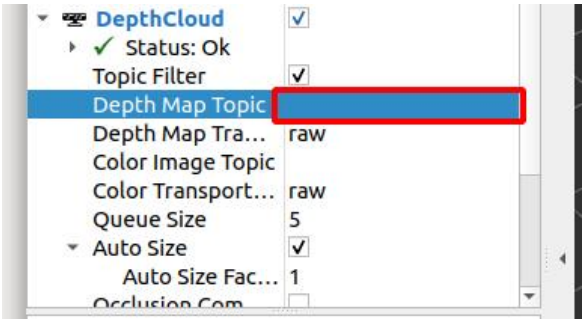
获取 rgb+深度图

```
roslaunch realsense2_camera rs_camera.launch
```

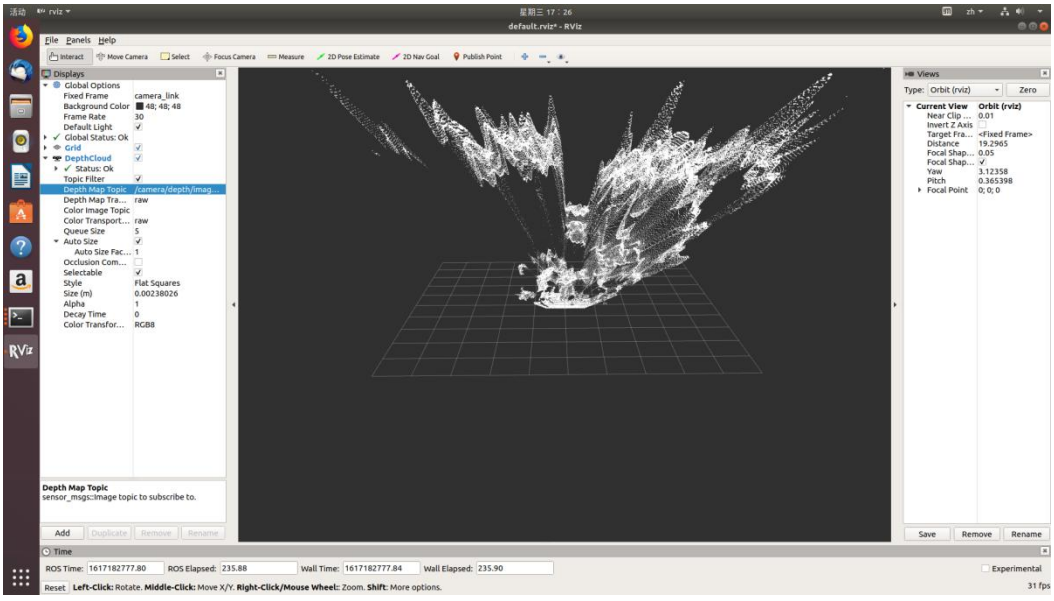
打开 rviz，点击 add 添加 DepthCloud 组件



fixed frame 选择 camera_link, DepthCloud 组件选择对应的话题



显示深度图



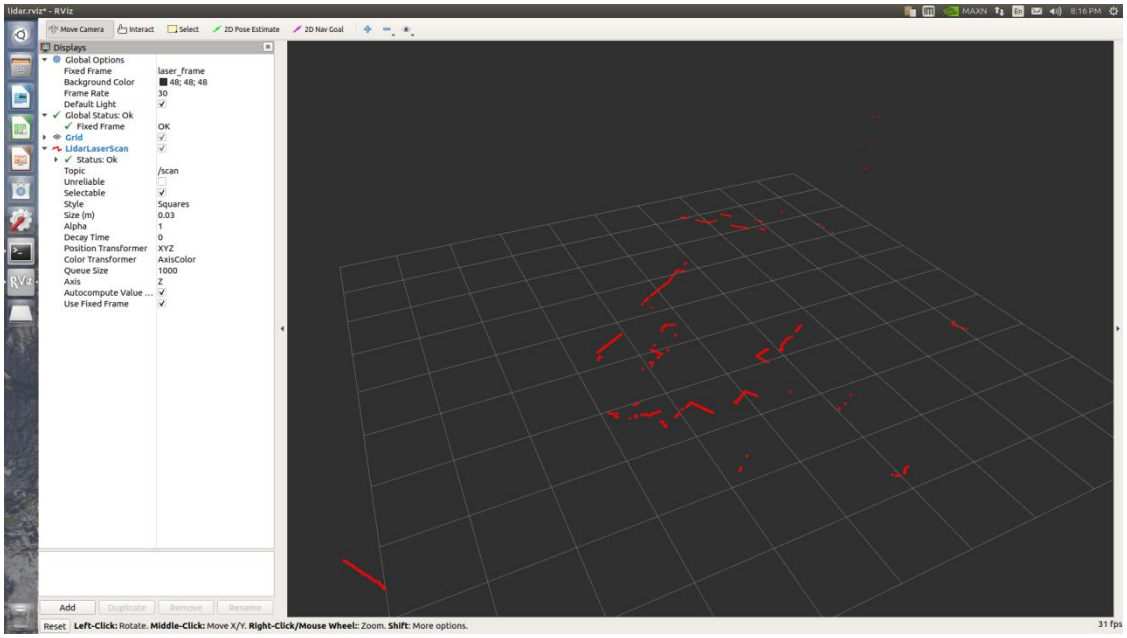
11.3、单线激光雷达 EAI-G4 ROS Package

ydlidar_ros:启动 EAI G4 单线激光雷达功能包，用户可以根据具体的应用场景去设置雷达的一些参数。

RVIZ 显示雷达扫描结果

```
cd ydlidar/launch
```

```
roslaunch ydlidar lidar_view.launch
```



启动激光雷达，发布 base_link-><laser_link>的坐标变换

```
roslaunch scout_bringup open_rslidar.launch
```

12、基于 Gmapping 开源架构的导航与定位

12.1、2D 激光雷达导航算法基本介绍

move_base 是机器人进行导航路径规划的核心算法，它通过订阅激光雷达的/scan，map 地图，/odom 里程，amcl 的定位数据，然后利用融合进 move_base 中的路径规划插件全局路径的规划插件，包括：

- navfn:ROS 中比较旧的代码实现了 dijkstra 和 A*全局规划算法。
- global_planner：重新实现了 Dijkstra 和 A*全局规划算法,可以看作 navfn 的改进版。

- parrot_planner: 一种简单的算法实现全局路径规划算法。

局部路径的规划插件包括:

- base_local_planner: 实现了 Trajectory Rollout 和 DWA 两种局部规划算法。

- dwa_local_planner: 实现了 DWA 局部规划算法，可以看作是 base_local_planner 的改进版本。

12.2、2D 激光雷达地图构建

ROS 中主流的 slam 算法有以下几种:

1. gmapping：需要激光雷达/scan 数据和里程计/odom 数据，采用的是 PF（粒子滤波）。
2. hector ： 基于优化的算法（解最小二乘问题），优缺点：不需要里程计，但对于雷达帧率要求很高 40Hz，估计 6 自由度位姿，可以适应空中或者地面不平坦的情况。初值的选择对结果影响很大，所以要求雷达帧率较高。
3. Cartographer：累计误差较前两种算法低，能天然的输出协方差矩阵，后端优化的输入项。成本较低的雷达也能跑出不错的效果。

综合这几个 slam 算法，该套件采用 gmapping 算法来进行 slam 建图。slam_gmapping 节点接收 sensor_msgs/LaserScan 消息并构建一个 map (nav_msgs/OccupancyGrid)。地图可以通过 ROS 的 topic 或 service 查看。

gmapping 的话题和服务

	名称	类型	描述
话题订阅	**tf**	tfMessage	需要进行激光，用于坐标系、基准和测距的相关框架转换
	scan	sensor_msgs/LaserScan	激光雷达扫描数据
话题发布	**map_metadata**	nav_msgs / MapMetaData	发布地图 Meta 数据
	map	nav_msgs / OccupancyGrid	发布地图栅格数据
	entropy	std_msgs / Float64	发布机器人姿态分布熵的估计
服务	**dynamic_map**	nav_msgs / GetMap	调用此服务以获取地图数据

gmapping 坐标变换

	TF 变换	描述
必须的 TF 变换	base_link-><laser_link>	底盘坐标系和激光雷达坐标系直接的变换
	odom->base_link	里程坐标系和底盘坐标系之间的变换，一般由里程计发

		布
发布的 TF 变换	map->odom	地图坐标系和里程计坐标系之间的变换，估计机器人在地图中的位姿

12.3、2D 激光雷达的自主导航实践

slam 建图实践：

（1）启动激光雷达，发布 base_link-><laser_link>的坐标变换

```
roslaunch scout_bringup open_rslidar.launch
```

（2）启动 gmapping 建图节点

```
roslaunch scout_bringup gmapping.launch
```

（3）用手柄遥控小车去在场景中走动，构建完地图之后，把地图保存到指定目录（一般是保存到 description）

```
rosrun map_server map_saver -f ~/catkin_ws/src/scout_base/scout_description/maps/map
```

注意：在建好图后，需要将终端所有在执行的指令关闭，才可以执行自主导航实践。可以通过 Ctrl+c 进行关闭。

自主导航实践：

（1）启动激光雷达，发布 base_link-><laser_link>的坐标变换

```
roslaunch scout_bringup open_rslidar.launch
```

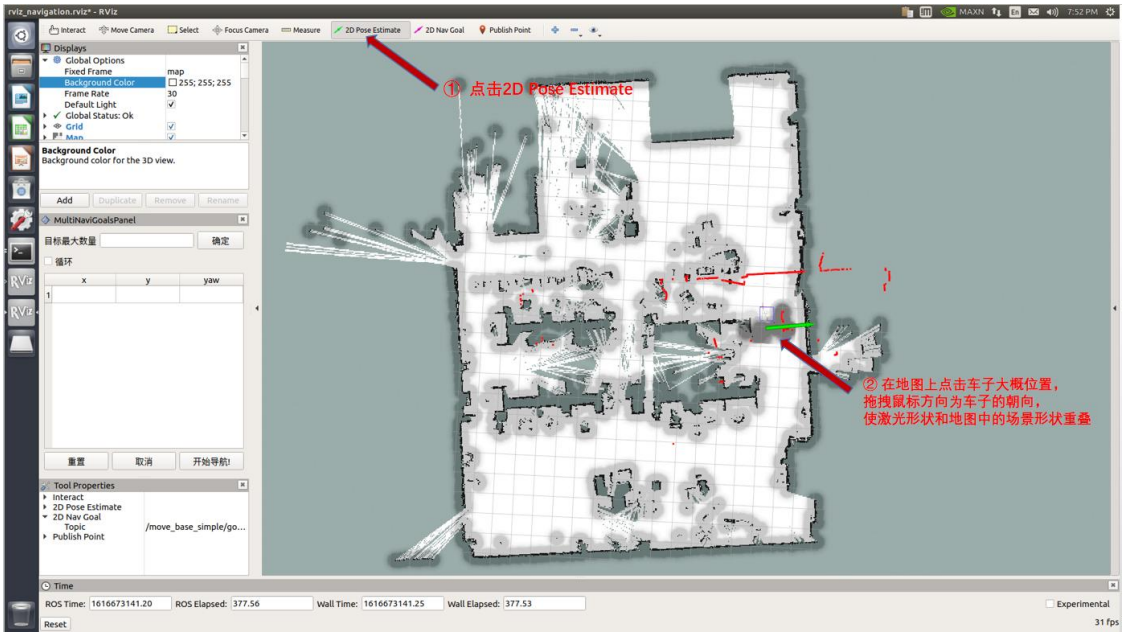
（2）启动 move_base 导航

```
roslaunch scout_bringup navigation_4wd.launch
```

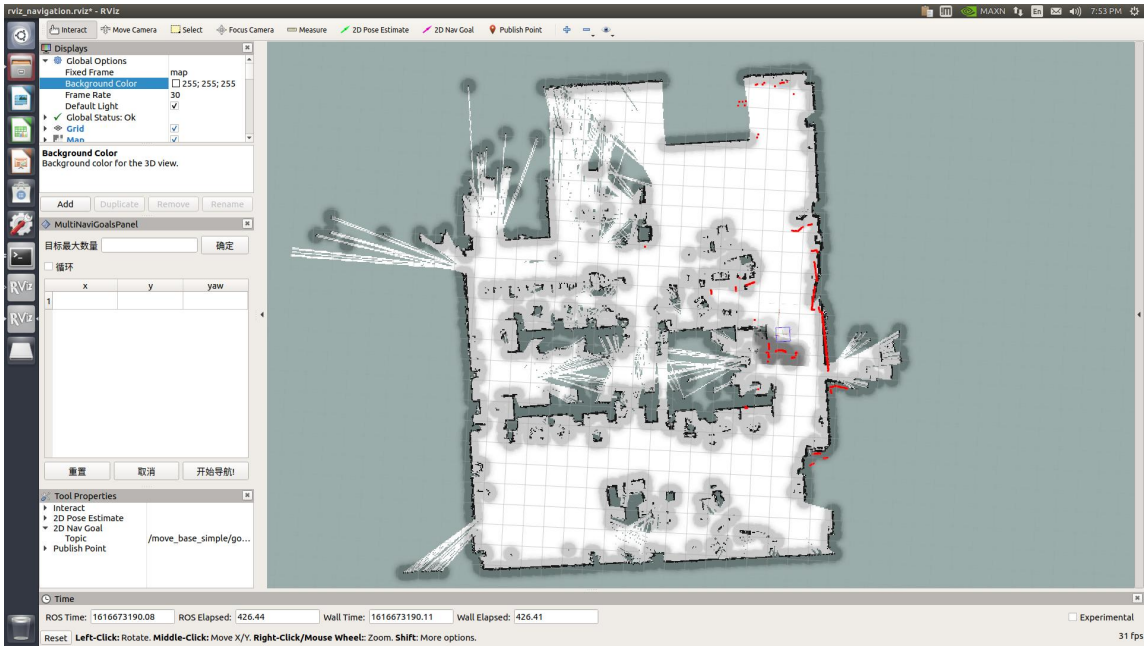
注意：如需自定义打开的地图，请打开 navigation.launch 文件修改参数，如下图所示，请在标记横线处修改为需要打开的地图名称。

```
<?xml version="1"?>
<!--
  Simulate a carlike robot with the teb_local_planner in stage:
  - stage
  - map_server
  - move_base
  - static map
  - amcl
  - rviz view
 -->
<launch>

        <!--  ************** Navigation ***************  -->
  <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
    <rosparam file="$(find scout_description)/param/4wd/costmap_common_params.yaml" command="load" ns="global_costmap" />

    <rosparam file="$(find scout_description)/param/4wd/costmap_common_params.yaml" command="load" ns="local_costmap" />

    <rosparam file="$(find scout_description)/param/4wd/local_costmap_params.yaml" command="load" />

    <rosparam file="$(find scout_description)/param/4wd/global_costmap_params.yaml" command="load" />

    <rosparam file="$(find scout_description)/param/4wd/base_local_planner_params.yaml" command="load" />

    <rosparam file="$(find scout_description)/param/4wd/move_base_params.yaml" command="load" />
  </node>


        <!--  ****** Maps *****  -->
<node name="map_server" pkg="map_server" type="map_server" args="$(find scout_description)/maps/map2.yaml" output="screen">
    <!--  <node name="map_server" pkg="map_server" type="map_server" args="/home/nvidia/map/2.yaml" output="screen">-->
     <param name="frame_id" value="map"/>
      </node>

      <node pkg="amcl" type="amcl" name="amcl" output="screen">
            <rosparam file="$(find scout_description)/param/amcl_params.yaml" command="load" />
            <param name="initial_pose_x"            value="0"/>
            <param name="initial_pose_y"            value="0"/>
            <param name="initial_pose_a"            value="0"/>
      </node>

  <!--  **************** Visualisation ****************  -->
  <node name="car_rviz" pkg="rviz" type="rviz" args="-d $(find scout_description)/rviz/rviz_navigation.rviz">
  </node>
  <include file="$(find scout_bringup)/launch/scout_volecity_smoother.launch"/>
  <include file="$(find scout_base)/launch/scout_mini_base.launch">
      <arg name="port_name" default="can0" />
      <arg name="simulated_robot" default="false" />
  </include>
</launch>
```

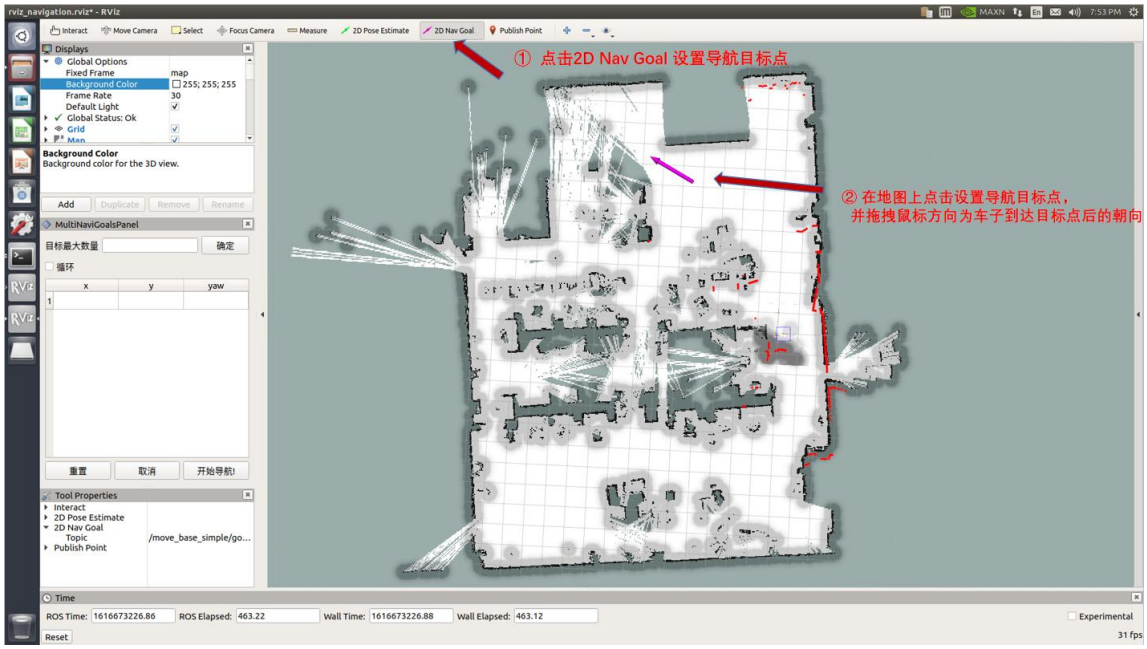
（3）在 rviz 中显示的地图上矫正底盘在场景中实际的位置，通过发布一个大概的位置加上手柄是底盘旋转校正。当激光形状和地图中的场景形状重叠的时候，校正完成



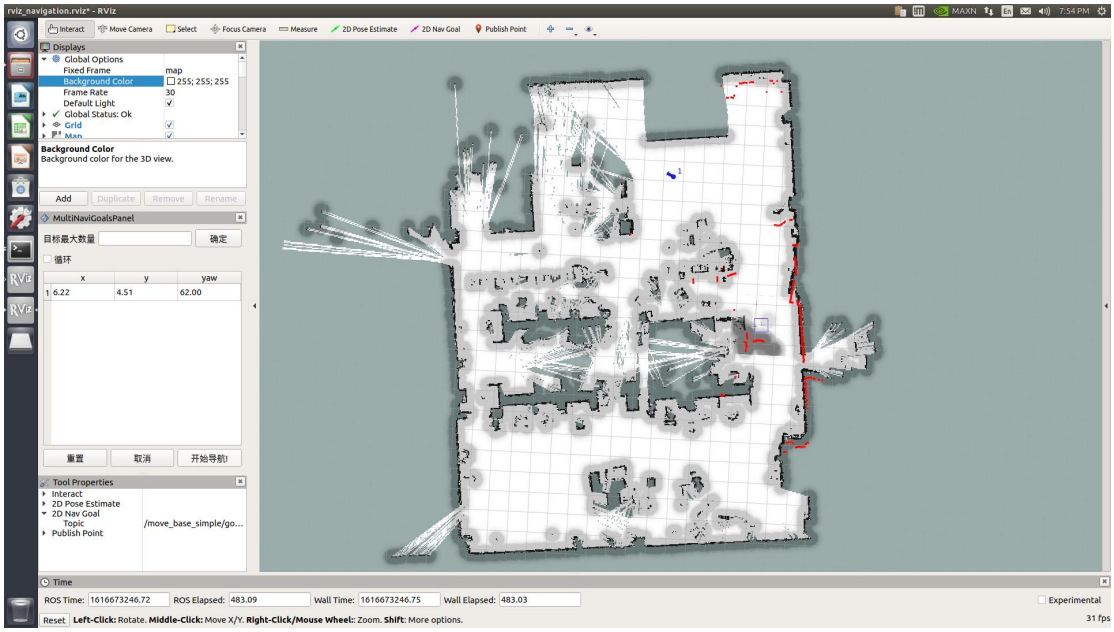
设置定位后，激光形状和地图中的场景形状已基本重合，校正完成。如下图所示



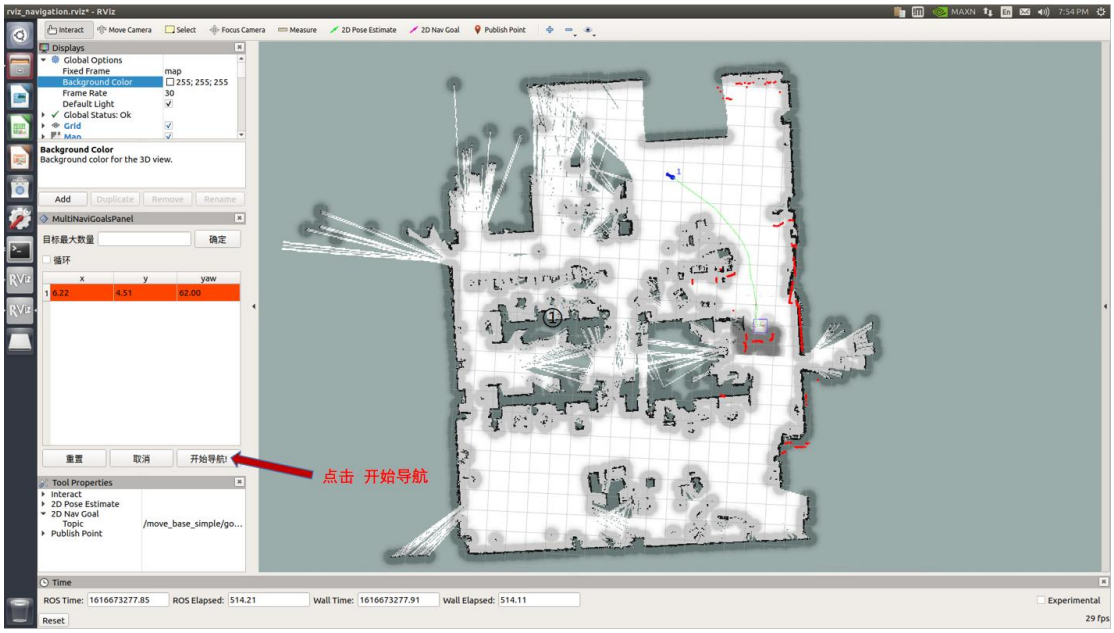
（4）在左侧栏多点导航的插件上设置目标点，点击开始导航，切换手柄为指令控制模式。



已生成目标点



点击开始导航，地图已生成路径（绿色），将自动导航到目标点



导航与视觉抓取复合项目操作指导

1. USB 转 CAN 驱动的启动

使能 gs_usb 内核模块

```
sudo modprobe gs_usb
```

打开 can 设备且设置波特率

```
sudo ip link set can0 up type can bitrate 500000
```

检查CAN通信正常:

```
candump can0
```

(回车后有数据刷新即数据通信已打通，终端即可关闭)

2. 启动雷达和导航:

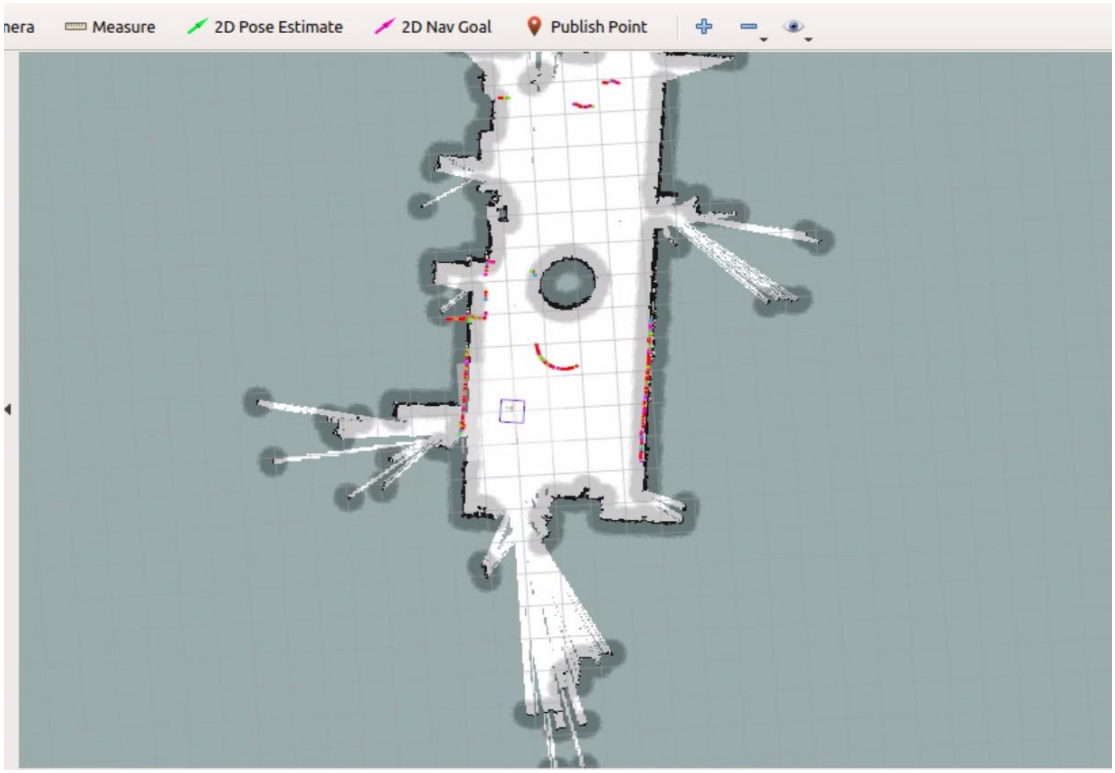
启动激光雷达，发布 base_link-><laser_link>的坐标变换

```
roslaunch scout_bringup open_rslidar.launch
```

启动 move_base 导航

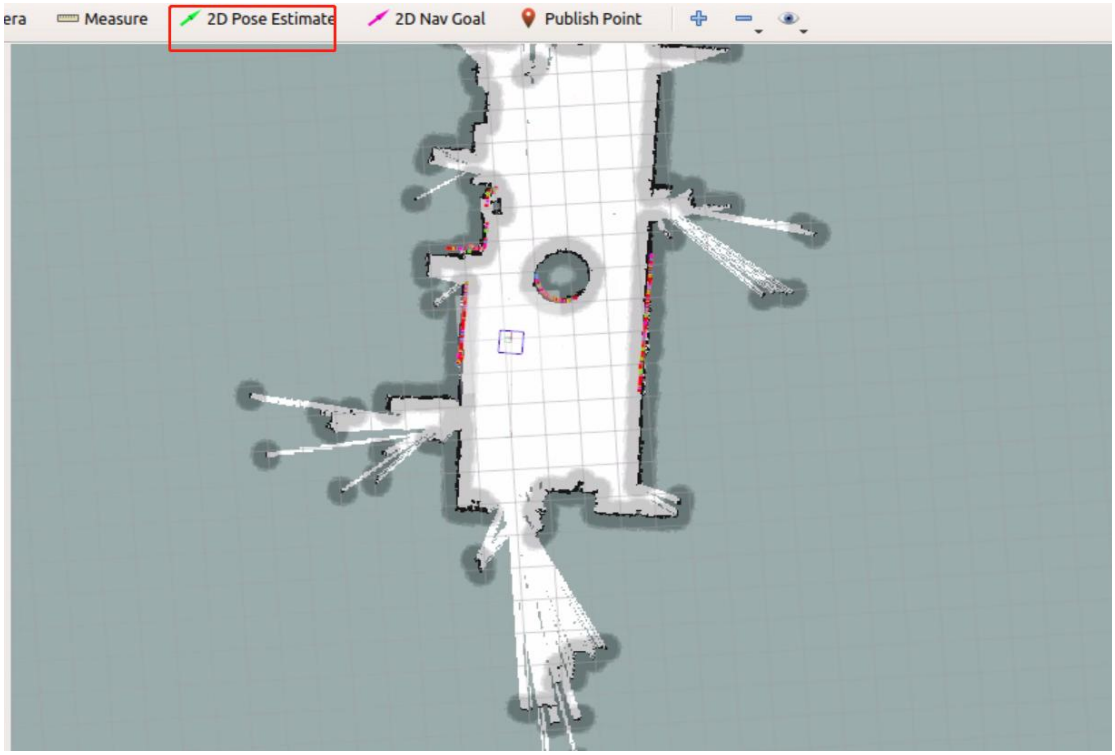
```
roslaunch scout_bringup navigation_4wd.launch
```

3.抓取点（机械臂操作点）获取

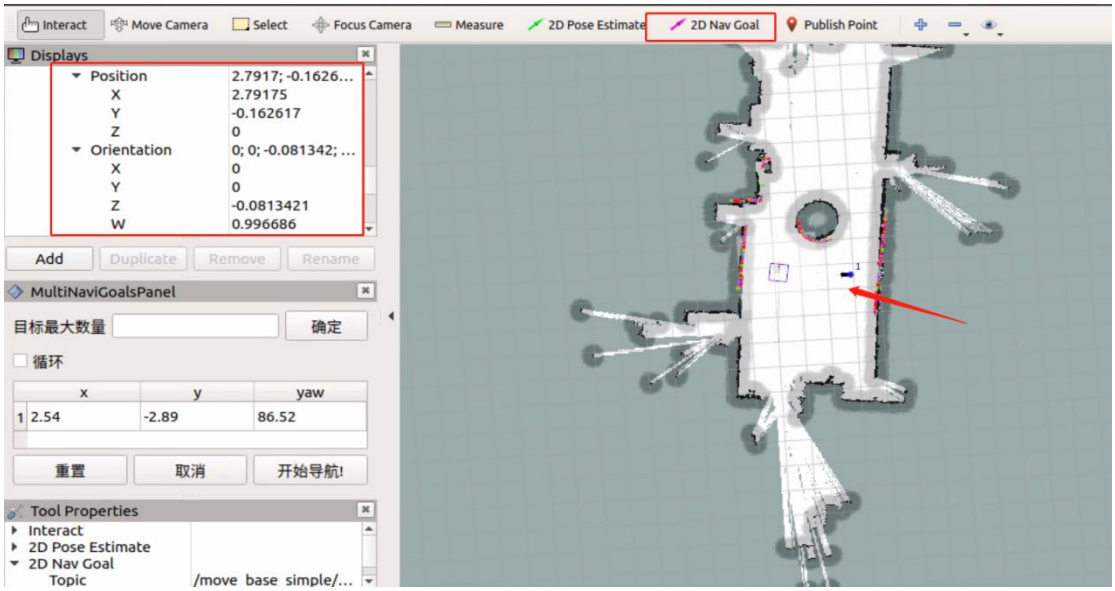


如图所示,如果小车不在原点启动,红线和障碍物不会重合,需要通过上方绿色的2D Pose Estimate来校准,点击2D Pose Estimate后，在地图上面找到下车实际所在的位置，鼠标左击后拖动调整箭头方向为小车车头方向，做完基础校准后，如仍旧不准，可以使用遥控器原地旋转360度，直到

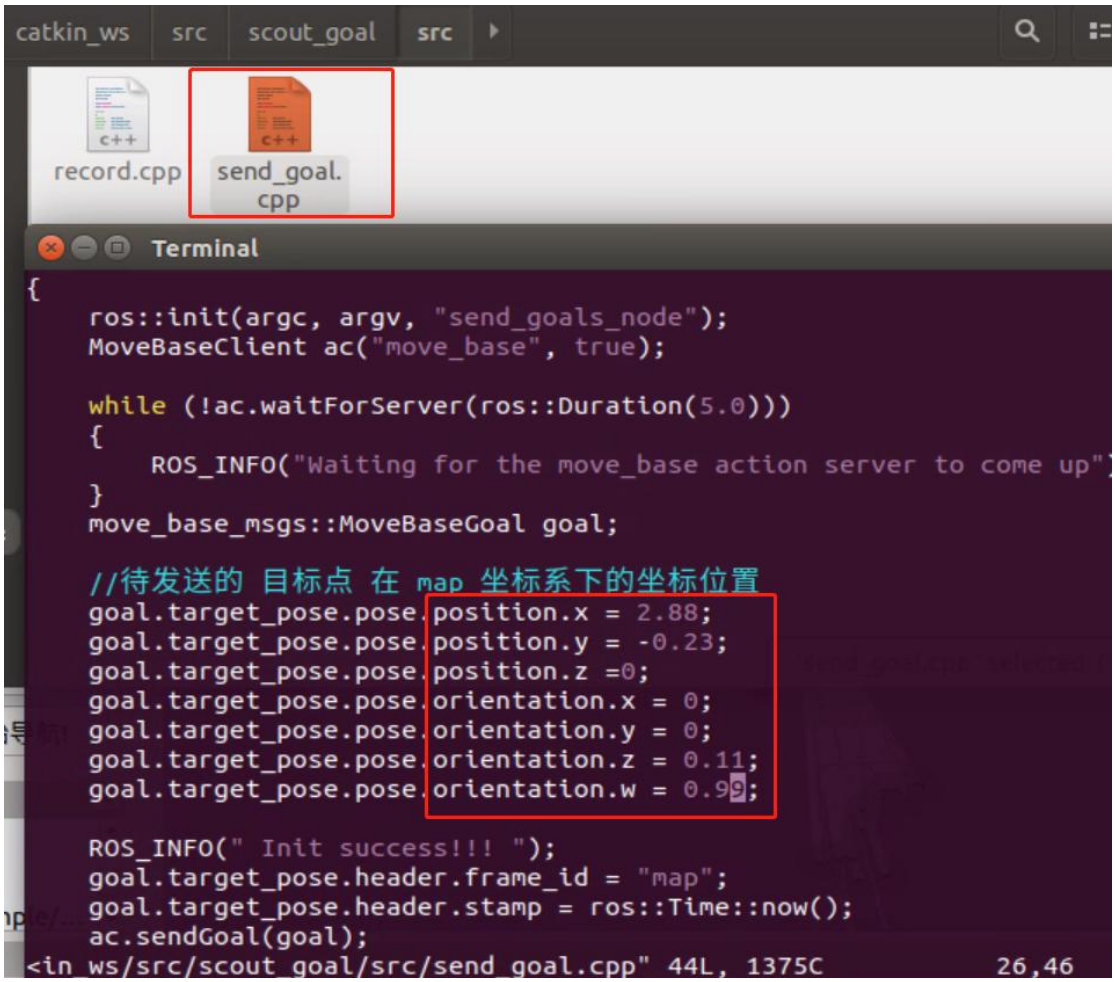
雷达显示红色特征点与静态障碍物一致即可，矫正后如下图所示:



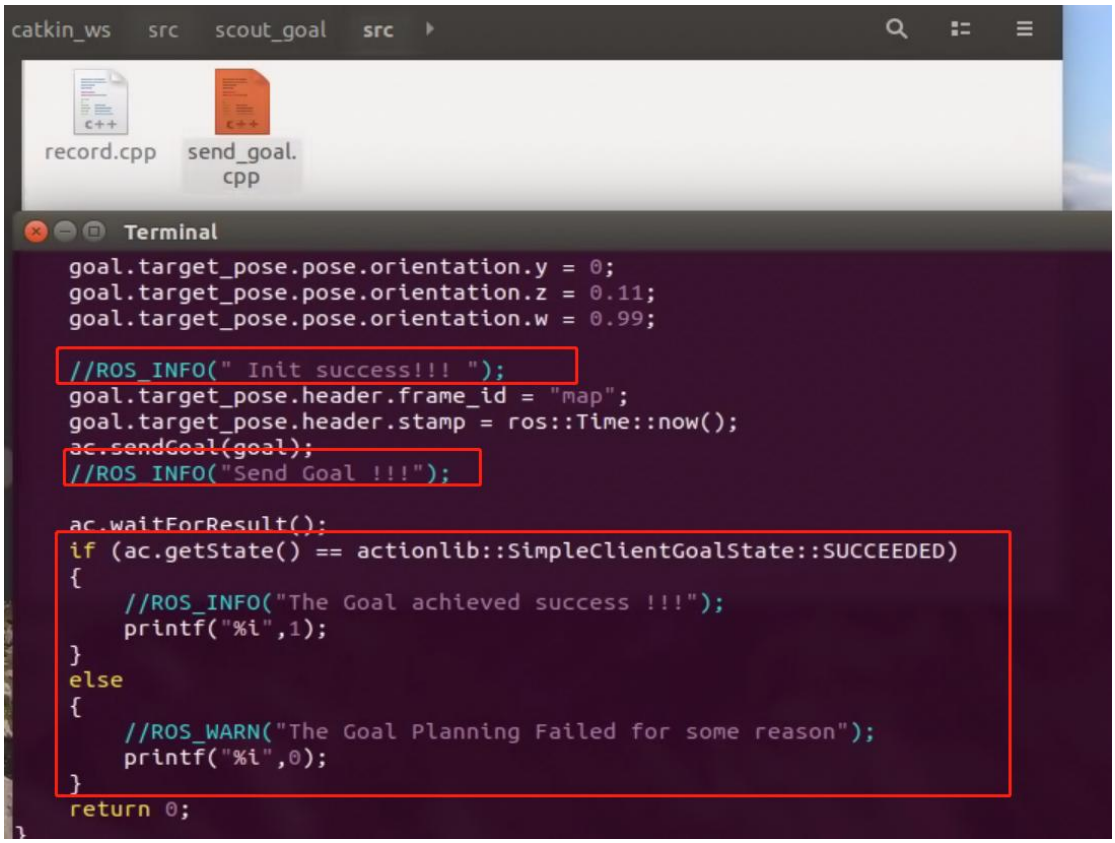
此时在图中选择一个需要抓取的点位,点击2D Nav Goal,然后在地图中选择位置和方向,如下图所示:



需要记录 TF >Frames>Base_link 下的(Position 和 Orientation)几个参数,然后一一对应填到 send_goal.cpp 文件中,



同时还需要在该.cpp 文件中修改代码，修改为导航与机械臂通信模式,具体更改为如下:



更改后需要重新编译.cpp 文件,此时该点即为抓取物料的点,然后将小车移动到其它地图内任一点就绪

在 catkin_ws 下面(src 上面一级目录)做编译然后刷新

catkin_make

source devel/setup.bash

4. 结合机械臂识别抓取

先查询 USB 串口，通过插拔机械臂与工控机接的 USB 线，确认串口号（默认为 USB0）

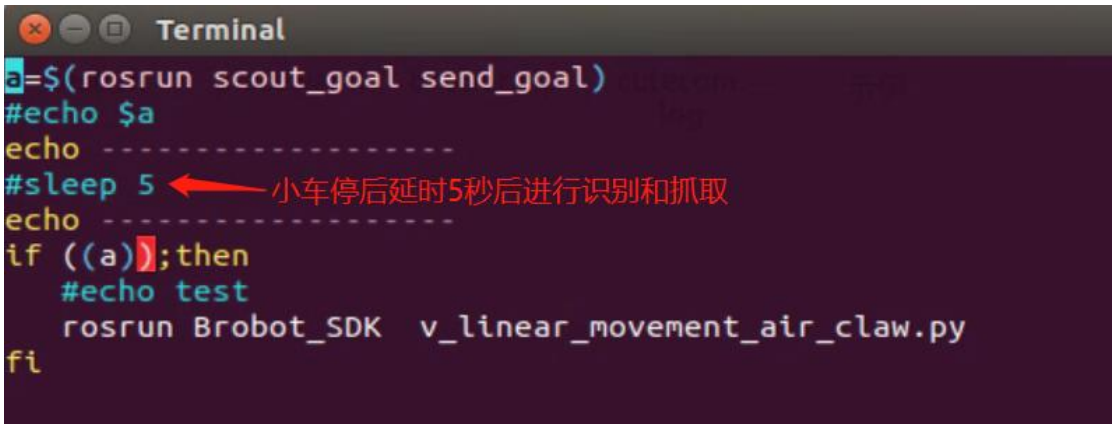
ls -l /dev/ttyUSB*

给串口权限

sudo chmod 777 /dev/ttyUSB*

(1) 在 home 下创建一个 test.sh 脚本,给权限,内容如下:

```
sudo chmod 777 test.sh
```



(2) 启动 RGB 相机节点(确保机械臂的串口正确且有权限)

```
roslaunch usb_cam usb_cam-test.launch
```

(3) 启动 yolov3 目标检测节点

```
roslaunch darknet_ros darknet_ros.launch
```

(4) 运行 home 目录下的 test.sh 脚本即可,

```
./test.sh
```

说明: test.sh 脚本中运行的是三爪的程序,可以在程序中更改机械臂的各项参数,打开如下图所示:

Home | catkin_ws | src | darknet_ros | Brobot_SDK | scripts

v_linear_movement_air_claw.py (~/catkin_ws/src/darknet_ros/Brobot_SDK/scripts) - gedit

Open | v_linear_movement_air_claw.py ~/catkin_ws/src/darknet_ros/Brobot_SDK/scripts

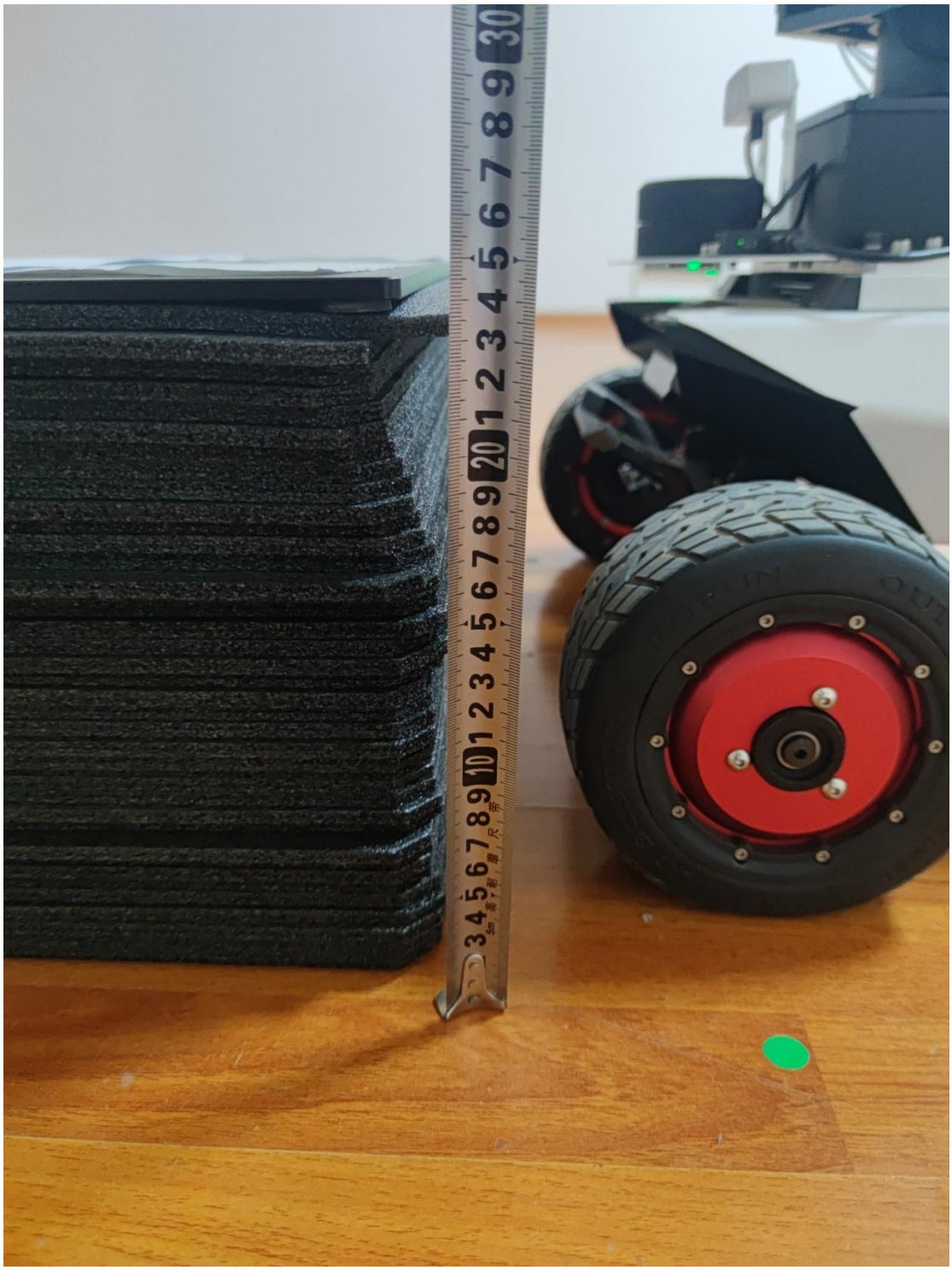
```
#! /usr/bin/env python3
# -*- coding: utf-8 -*-

# 视觉识别+控制机械臂气动夹爪抓取
#1.导包

import rospy
from std_msgs.msg import Float32MultiArray
from darknet_ros_msgs.msg import BoundingBoxes
from PythonAPI import *

# x,y,z为识别时机械臂的位置
x = 0
y = -355
z = 350
# m为单位像素对应的实际尺寸
m = 0.495


if __name__ == "__main__":
    # 开启端口
    open_port('/dev/ttyUSB0',115200)
    """
    /dev/ttyUSB0 : 端口号
    115200 : 波特率
```

```
    a = (msg.bounding_boxes[0].xmin + msg.bounding_boxes[0].xmax) / 2
    b = (msg.bounding_boxes[0].ymin + msg.bounding_boxes[0].ymax) / 2

    # 将坐标从像素坐标系转换到机械臂坐标
    x = x + 2.75 - (a - 320) * m - 5
    y = y + 55.5 + (b - 240) * m + 2

    # 高度需根据物体的实际高度来调整
    linear_movement(x, y, 80)

    time.sleep(10)

    air_pump_action(1)
    """
    0 停
    1 吸
    2 吹
    """
    time.sleep(1)
    #
    linear_movement(200, -300, 350)
    time.sleep(10)
    air_pump_action(2)
    time.sleep(1)
    air_pump_action(0)
    # 设置结束控制信号
    set_control_signal(5)
```

夹取小球后放置的点位

5.物料抓取台的要求

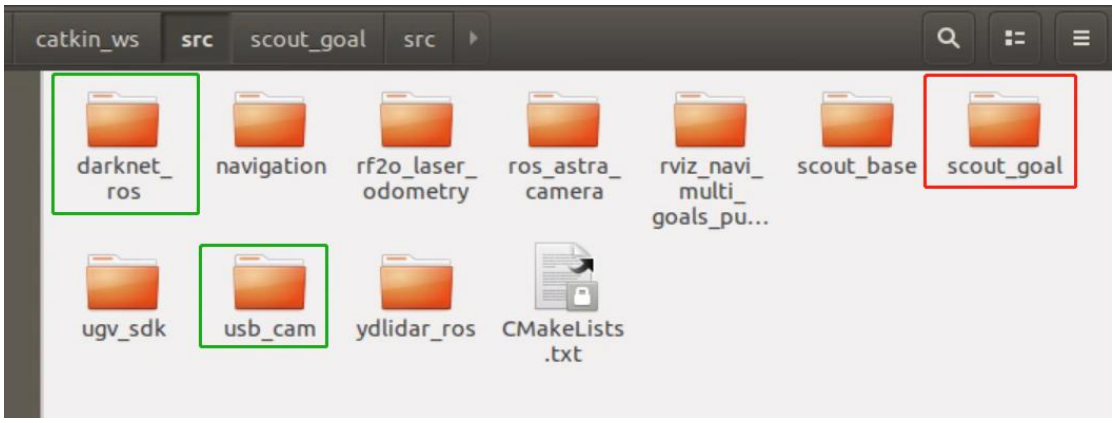
物料抓取台的高度不能低于机械臂底座的高度,同时不能高于雷达的高度,建议高度如图所示:高度在 24.5cm 左右,



物料抓取台的放置位置要求:由于物料抓取台的高度没有超过雷达高速,雷达暂时检测不到,所以小车到达台面附近的时候需要以车头对着台面的方向进去,便于机械臂识别和抓取,当识别和抓取之后,小车应以车尾对着台面的姿态离开(即离开后到达的导航点位不能越过台面)

6.开发包说明

如下图所示,绿色框中的为摄像头和机械臂的开发包,红色框中的为小车底盘移动的开发包



注意：机械臂在如图下图的姿态进行开机（大概的姿态），机械臂才初始化才会成功。