ROSYZ-01C 机器人平台

用户手册

Rev: 3.0 (JAN, 2021)

深圳市璞数技术有限公司

§1 简介

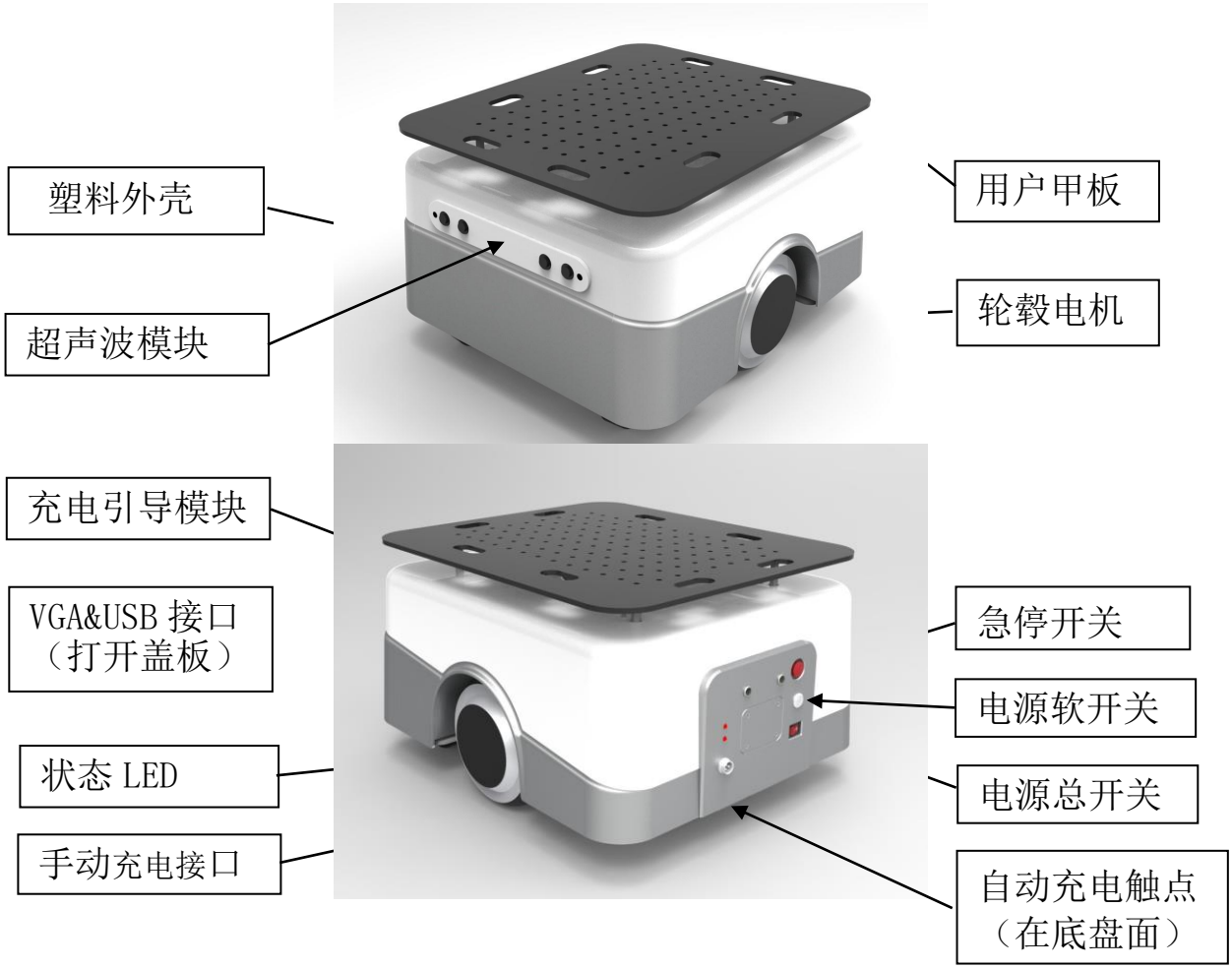
§1. 1.1 ROSYZ-01C 机器人平台特性简述

ROSYZ-01C机器人平台是一款基于ROS架构的双轮差速大载荷机器人运动底盘平台，非常适合于 ROS 机器人爱好者、大专院校学生和中小微企业研发工程师使用。这款运动底盘平台采用的是高效能大载荷的一体化轮毂电机，平台的负荷可以达 50KG，最高行走速度可达 1 米每秒。

ROSYZ-01C 内 INTEL 置酷睿 i5 CPU 高端工控机电脑和 ARM 控制主板。内置的 DCDC 电源转换模块可以提供 5V, 12V，16V 三种不同电压，基本上可以解决机器人需要搭载的绝大多数传感器的供电问题。为了方便开发人员做深入研究，这款 ROS 平台的运动控制主板和 DCDC 电源板的电路原理和详细接口图纸我们也一同提供。

同时，和其他 ROS 平台机器人一样， ROSYZ-01C 也提供开源的基本 ROS 应用示例和基础的运动驱动节点程序，让开发者能非常容易上手使用。

§1.2ROSYZ-01C 的主要构成部件

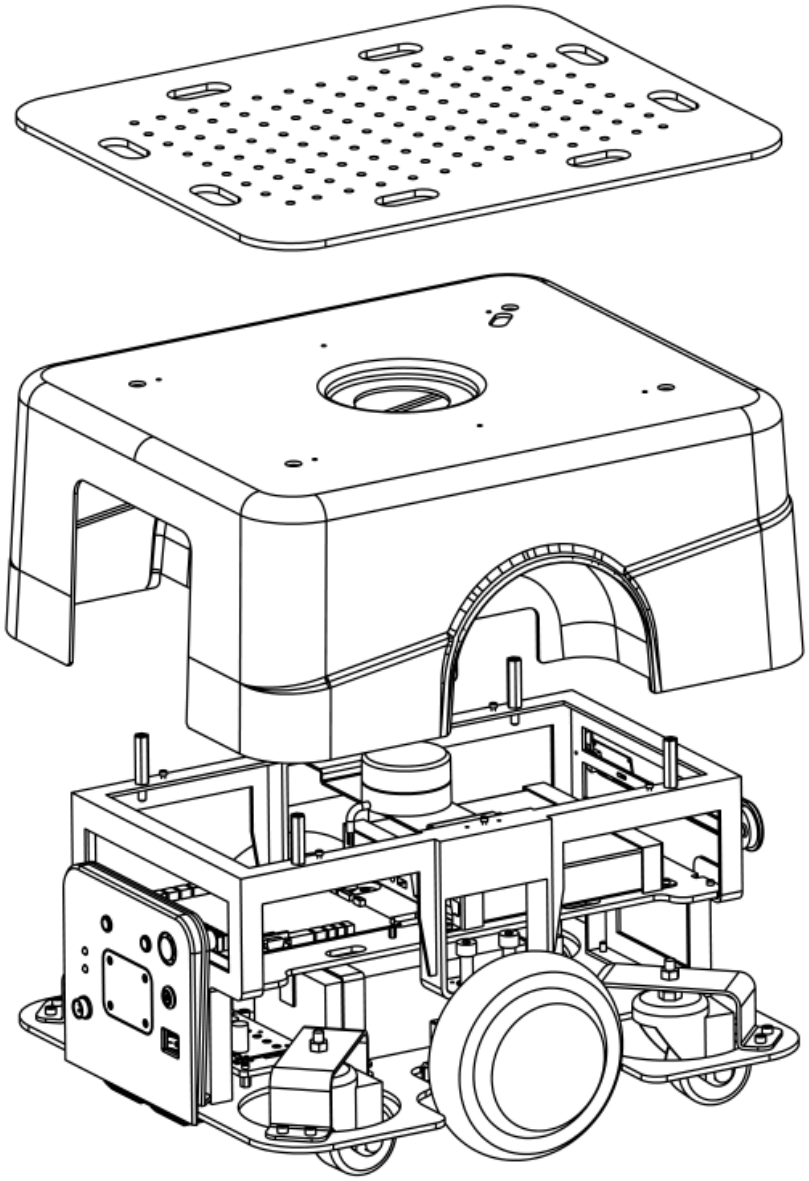


指示灯状态介绍

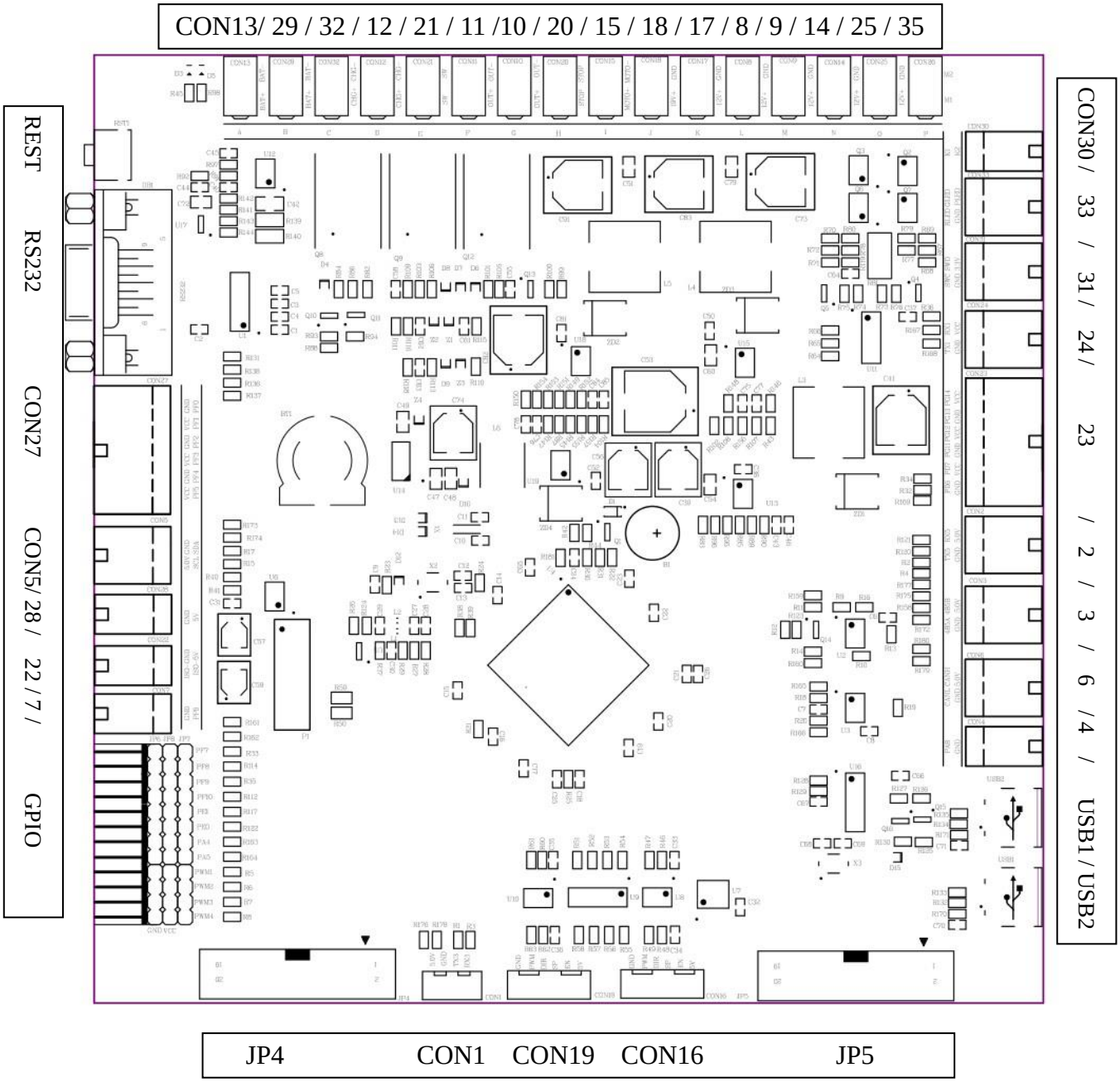
电源指示灯:红色常亮（POWER ON）

状态指示灯: 红色: 充电中；绿闪: 正常工作中；红闪: 电池电量低

ROSYZ-01C 的部件结构爆炸图



§1.3ROSYZ-01C 控制板连接器定义



§1.4 连接器定义：

电源类接口

CON13 电池输入（16V）
CON29 电池输入 (16V)
CON32 充电器输入
CON12 充电器输入
CON21 电源开关（硬开关）
CON10 电池（16V）输出
CON11 电池（16V）输出
CON20 急停开关
CON15 行走电机驱动板供电
CON18 +16V 输出
CON17 +12V 输出
CON8 +12V 输出
CON9 +12V 输出
CON14 +12V 输出
CON25 +12V 输出
CON26 +12V 输出（隔离）
CON28 +5V 输出
CON22 +5V 输出（隔离）

通信和其他类接口

RS232 接口,与工控机通信.
CON27: 前方超声测距模块检测接口
CON5: NA
CON7:NA
CON30: 开机按键（ON/OFF 开关）
CON33: 接指示灯接口板
CON31: SWD 下载口
CON24 : 厂家调试专用
CON23:后方超声测距模块检测接口
CON2: TTL 串口
CON3:默认 TTL 串口输出，可改成 RS485
CON6: NA
CON4: 接后防碰撞条
USB1: 厂家 USB 调试专用
USB2:NA
JP4: AGV 专用
JP5: AGV 专用
CON1: 红外避障通信板接口/RDID 模块接口
CON19: 左电机驱动信号

CON16:右电机驱动信号

JP6, JP7,JP8: GPIO

JP6: GND,电源地

JP8: VCC,电源正

JP7 说明如下:

PF7:开机信号

PF8:NA

PF9,PF10:自动充电引导输入

PE0,PE1:升降限位开关检测

PA4,PA5:NA

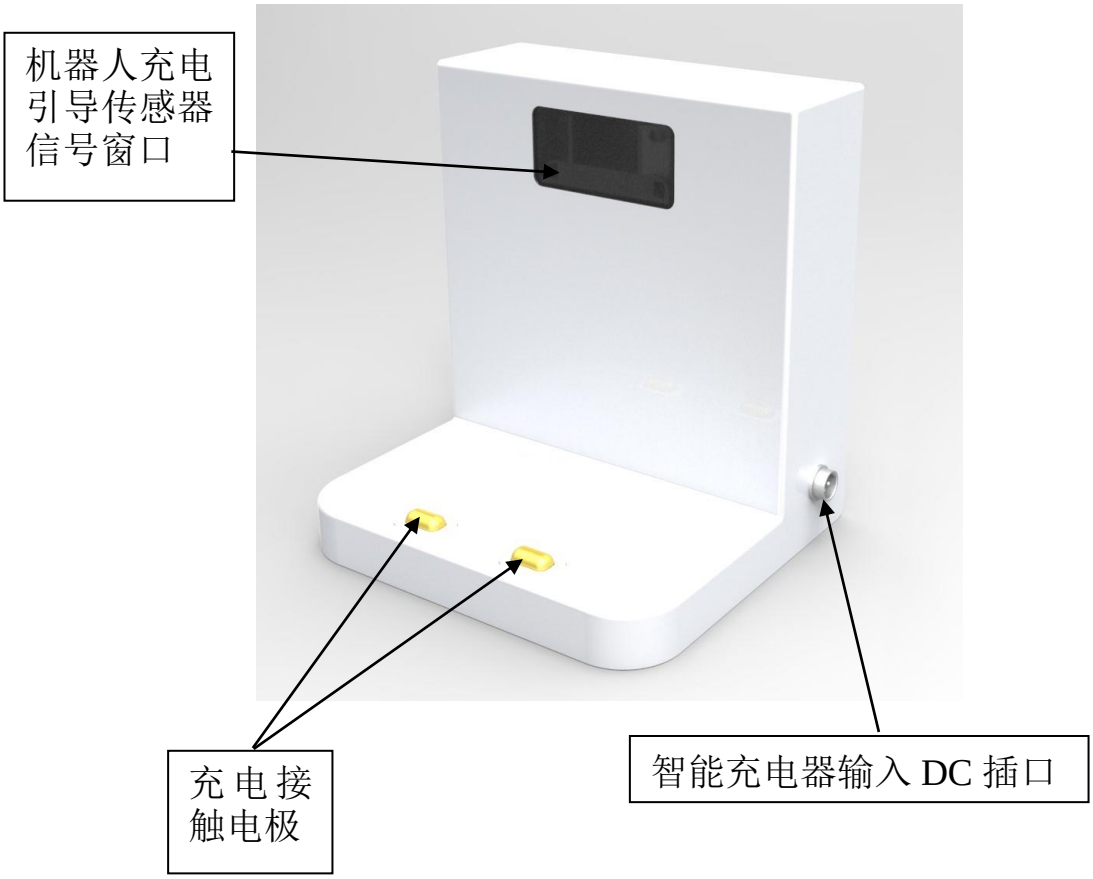
PWM1,PWM2:头部运动控制

PWM3:NA

PWM4: 接前防撞条

§1.5 自动充电桩介绍（选配部件）

ROSYZ-01B 配置有自动充电桩，示意图如下：



安装：充电桩的背侧需要靠墙，建议固定到墙上。充电桩前方 1.8 米 120 度扇形区域内不要有障碍物，否则可能会影响充电引导信号。

§1.6 ROSYZ-01C 机器人标配装箱清单：标配部件

名　　称	数量	单位	备　　注
机器人运动主机	1	台	不含电池
用户甲板	1	个	
支撑立柱	4	条	
18V8A 智能充电器	1	个	型号 G168-180080
充电器电源线	1	条	
M4 螺杆	4	个	
内六角扳手	1	个	配合 M3 M6
16V21Ah 铁锂电池	1	只	
驱动软件和 DEMO	1	套	电子档发送
用户手册	1	份	提供电子版 PDF

选配部件（客户订单中需要选购）

激光雷达	1	pc	EAI G4
自动充电桩	1	Set	Green-Digial
IMU 模组	1	pc	Miiboo
RGBD 深度摄像头	1	pc	Astra pro
无线鼠标键盘	1	set	Rapoo
7 寸便携式显示器		pc	DC12V
无线遥控手柄	1	pc	BT

§1.7 机器人内置工控机预装软件清单

机器人内置工控机内部已经安装的软件清单如下：

1. LINUX 系统：UBUNTU 16.04　用户名：robot，密码：1
2. ROS 系统：KINETIC FULL-DESK
3. ROS 工作目录 WS 下各文件夹如下

3.1 build : 编译中生成的配置文件夹

3.2 devel : 编译中生成的目标文件夹

3.3 install : 璞数技术提供的私有定位和导航包

3.4 src : 机器人底盘和传感器驱动源文件及演示示例源文件

3.4.1 YZBOT_STM32CONNECT ：底盘的核心驱动包

3.4.2 ROBOT_MSGS ：底盘的自定义信息包

3.4.3 YZBOT_DESCRIPTION ：底盘描述包

3.4.4 ROS-NAVIGATION : ROS 官方导航源码压缩包

3.4.5 YZBOT_NAV ：YZ 提供的示例文件夹，主要子文件夹如下

3.4.5.1　launch 文件夹：

rs_amcl_startup.launch ：启动 ROS 开源的 amcl

rs_movebase_startup.launch ：启动 ROS 开源的 movebase

yzbot_joystick_startup.launch ：启动手柄驱动

yzbot_keyboard_teleop.launch ：启动键盘控制驱动

yzbot_loadmodel.launch ：启动加载机器人模型

Laser/lidar_laser.launch ：启动选 EAI 雷达

Laser/lslidar_laser ：启动选配镭神雷达

yzbot_dmcl_startuo.launch ：启动璞数定位

yzbot_gmapping_startuo.launch ：启动单独建图

yzbot_movebase_startup.launch ：启动璞数导航

yzbot_stm32control.launch ：启动底盘

yzbot_createmap.launch ：启动建图，启动底盘，手柄驱动

yzbot_navigation_byAndroid.launch ：启动底盘，璞数定位导航，APP 接口

yzbot_navigation_startup.launch ：启动底盘，璞数定位/导航

rs_navigation_startup.launch ：启动底盘，雷达，开源 amcl/movebase

3.4.5.2　maps 文件夹：保存已经建好的地图

3.4.5.3　nodes　文件夹：保存了示例脚本源程序（PYTHON 编写）

3.4.5.4　config 文件夹：保存了示例用到的相关配置文件

3.5 YZBOT_SENSORS ：包含以下所有传感器文件夹

3.5.1 YDLIDAR ：选配的 EAI 雷达第三方驱动包

3.5.2 MIIBOO_IMU（选配的 IMU 第三方驱动包）

3.5.3 ASTRA_CAMERA （选配的第三方深度摄像头驱动包）

3.5.4 JOY （选配的第三方遥控驱动包）

3.5.5 LS01BV2（选配的镭神雷达第三方驱动包）

3.6 YZBOT_MSATER_CONTROL : 安卓 APP 应用层控制器

§1.7 ROSYZ-01B 机器人主要传感器清单

机器人配置的传感器清单如下：

1. 超声避障传感器(前 2 后 2)
2. 车轮里程编码器（霍尔方式）
3. 激光雷达（需要客户选配）
4. 自动充电引导模块（需要客户选配）
5. IMU 惯量测量模块（需要客户选配）
6. 无线鼠标键盘（需要客户选配）
7. 无线遥控手柄（需要客户选配）
8. RGBD 深度摄像头（需要客户选配）
9. 12V 便携式 7 寸 TFT 显示器（需要客户选配）

§2 使用简介

§2.1 重要说明

ROSYZ-01C 机器人运动平台是针对 ROS 机器人开发者而设计的，操作者必须要有 ROS 机器人的基础知识。使用之前请先通读一遍本手册，特别要仔细阅读最后页的注意事项。

§2.2 ROSYZ-01C 机器人初次使用准备工作

2.2.1 检查配件是否齐全：

打开机器人的包装纸箱，取出机器人运动底盘和所有零部件，注意要对照一下装箱单，检查有无漏装、错装。

2.2.2 安装电池

为了运输安全，ROSYZ-01C 机器人的电池包和机体是分开单独包装的。请按下述方法安装电池：

1. 用内六角扳手松开侧面安装板的螺丝，取下盖板，把电池固定到电池架板上。
2. 把电池的插头插入电池接口插座上，注意插接之前务必注意电池极性不要搞反。把侧板装回去。

2.2.3 上电检查

打开后侧的电源总开关，然后按下顶部的圆形开关按钮，此时电源指示灯应该是红色亮起，机器人内的蜂鸣器鸣响两下，状态指示灯应该是绿色闪动，表示 ROSYZ-01C 已经进入正常工作模式了。

若电源灯不亮，则应该是电池没接好，或者是电池电量严重不足，请先用充电器给电池充电，如何充电见下一节。

2.2.4 给电池充电

ROSYZ-01C 可以使用便携式充电器充电，也可以使用专用智能充桩。使用便携式充电器，请直接把 DC 插头插入机器人后部的充电插孔即可。

若选购了自动充电桩，充电桩的安置请参考 1.3 小结。使用时将机器人移动到充电桩前方 1 米左右的位置，若此时机器人的 ROS 节点发布了自动充电话题（详见附录 A），则机器人会在 20 秒钟之后自动对接到充电桩充电。若机器人完全无电不能正常工作，请把机器人缓慢推行到充电桩上（后面的充电金属片要能触碰到充电桩金属片。此时 ROSYZ-01C 的状态指示灯和电源指示灯一样都是红色。

对于完全空置的电池充满电大约需要 4 小时。充满电之后充电桩会自动进入浮充维护模式，不用担心过充。

2.2.5 把机器人接入本地 WIFI 网络

首先拆掉机器人顶板， 找到工控机接口面板，插入鼠标、键盘和 VGA 显示器。让机器人接入工作区域的 WIFI 网络，然后记录下机器人的 IP 地址。（注：机器人 UBUNTU 系统的用户名为 robot, 密码为 1）

WIFI 配置完成之后，请拔掉鼠标、键盘、显示器的连线）

2.2.6 使用无线摇杆手柄

若你需要用摇杆控制机器人行走（比如构建地图时），请先把你的摇杆 USB 插到机器人工控机的 USB 插口。注意若你的摇杆不是我公司选配的部件，使用前请先自行安装摇杆 ROS 驱动包，并验证可以控制机器人前后行走和左右转弯。

§2.3 用开源 ROS 包操控机器人的使用示例—本地模式

*说明：有两种方式可操控 YZ 机器人，一是**本地模式**，就是直接在机器人机载工控机上接入鼠标、键盘、摇杆、VGA 显示器（需要打开机器人后方的 VGA&USB 接口的盖板，见右下图示），输入 ROS 命令行来操控机器人；另一种方式是**局域网模式**，需要先把机器人机载工控机接入您的无线局域网，参看上述 2.2.4，然后通过局域网里的其他电脑（比如您的笔记本电脑）来运行 ROS 指令。*

如果您希望用局域网模式来远程操控机器人，请直接跳到 2.4 节。

2.3.1 建地图的示例

1）开启 2 个终端，分别启动以下节点

终端一 $ roslaunch yzbot_nav yzbot_createmap.launch (建图命令，默认使 YZ 选配的用无线遥控手柄来操控)

终端二 $ roslaunch yzbot_nav keyboard_teleop.launch（此为使用键盘控制机器人移动，若使用无线遥控手柄，可不用运行此命令行）

2）待以上节点运行起来之后，打开第 3 个终端

$ rosrun rviz rviz -d `rospack find yzbot_nav`/gmapping.rviz 查看 RVIZ 此时应该能看到原点附件的地图雏形。

此时可以遥控机器人（用要无线手柄或键盘）慢慢在实验室行走，直到走完全通道然后返回的出发点。无线手柄的使用方法参看附录 B。

3) 开启第 4 个终端

切换目录 roscd yzbot_nav/maps

再运行 rosrun map_server map_saver -f map

你地图的名字就是上面的“map”，可以在 ~/ws/src/yzbot_nav/maps 文件夹里看到。至此，建图结束。若对地图不满意可以再来一遍。

2.3.2 让机器人在建好的地图里行走到鼠标指定的位置。

1) 假设你已经完成了上面 2.3.1 的 SLAM 建图实验，生成的地图位于 ~/ws/src/yzbot_nav/maps 目录下，地图名字是 map。

2）开启 2 个终端，分别运行以下节点：

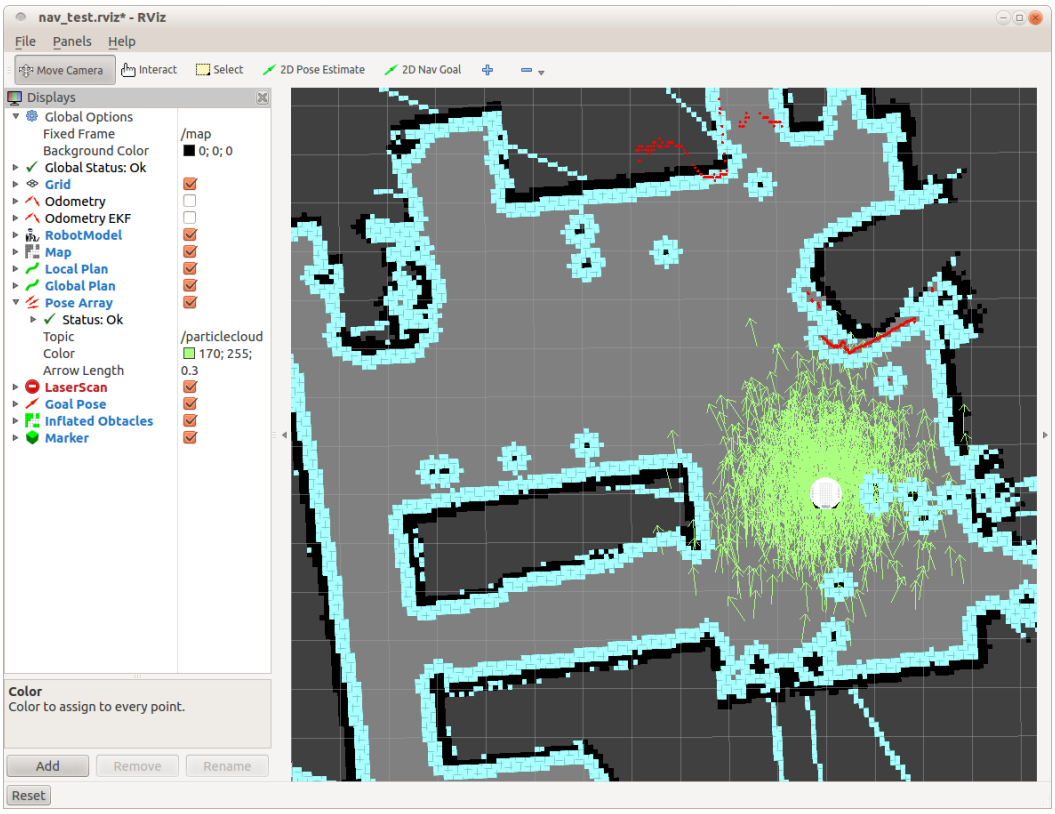
终端一 $ roslaunch yzbot_nav rs_navigation_startup.launch

终端二 $ rosrun rviz rviz -d `rospack find yzbot_nav`/nav_test.rviz

【2D Pose Estimate】按钮用来设定机器人当前位置

【2D Nav Goal】按钮用来给机器人设定一个需要行走到的目标位置。

【Publish Point】按钮用来查看鼠标指定位置的地图坐标值



先用【2D Pose Estimate】按钮把机器人初始位置设定好（激光斑点与地图上的障碍物轮廓基本重合）然后用【2D Nav Goal】按钮，选取一个地图位置，机器人就会自动走过去

2.3.3 机器人在建好的地图里按任务规划行走。

当你已经完成了上述两个任务，接下来让机器人做一个稍微复杂一点的任务：选定 5 个不同的地点，让机器人在这 5 个不同地点之间来回走动，并在屏幕终端上打印出任务执行的成功率、累计行走的距离、累计运行的时间。这个程序是用 Python 脚本编写的，您可以稍作修改，实现自己定义的任务（比如：从 A 点到 B 点再到 C，每个点停留 5 分钟，然后循环往复）。

1）用文本编辑器（比如 gedit）修改~/ws/src/yzbot_nav/nodes 目录下的 nav_test.py, 把 locations[xx]里的坐标值替换为您办公司地图里期望的 5 个不同位置坐标。这个位置坐标可以预先用 rviz 里的【Publish Point】按钮来查看。

2）开启 3 个终端，分别运行以下节点：

终端一 $ roslaunch yzbot_nav rs_navigation_startup.launch
终端二 $ roscd yzbot_nav/nodes (切换目录)
$ ./nav_rs.py （运行 PYTHON 脚本）
终端三 rosrun rviz rviz -d `rospack find yzbot_nav`/nav_test.rviz

3) 如果一切运行正常，终端三的屏幕上会显示
*** Click the 2D Pose Estimate button in RViz to set the robot's initial…
此时您需要在 rviz 窗口上用【2D Pose Estimate】按钮用来设定机器人当前位置，然后机器人就会自动去执行 nav_test.py 里写好的任务了。任务执行的成功率、累计行走的距离、累计运行的时间等信息都会在终端三的屏幕上显示出来。

您可以仔细阅读上述示例的 launch 批处理命令的脚本内容，详细学习和了解机人是如何利用 ROS KINETIC 官方代码进行定位和导航的。

§2.4 用开源 ROS 包操控机器人的使用示例—局域网模式

上面 2.3 节用的是机器人电脑本地模式来操控机器人的。您还可以通过局域网用远程电脑来操控机器人，这个时候机器人上就可以不用加装 VGA 显示器以及键盘鼠标等，这在很多情况下是非常有实用价值的。

2.4.1 用局域网操控机器人之前，请先把机器人的电脑主机接入局域网 WI-FI，这个工作需要临时接入 VGA 显示器和鼠标键盘，让机器人接入工作区局域网 WIFI，用 ifconfig 查看机器人电脑的 IP 地址。

2.4.2 请登录 WI-FI 路由器管理界面，把刚刚看到的机器人的 IP 地址和机器人网卡绑定，这样今后机器人开机后会自动连接这个 WI-FI，而且机器人的 IP 地址会固定不变的（比如绑定为 192.168.1.101）。上述工作完成后，以后就可以不用连接屏幕键盘鼠标了。

2.4.3 用激光雷达构建地图的示例

1）假设你的笔记本电脑已经安装了 ROS KINETIC，请先用局域网文件拷贝命令把机器人电脑中的~/ws/src/yzbot_nav 文件拷贝到你的笔记本电脑的 ROS 工作文件夹的 src 目录下，并且编译通过。接下来将用这台笔记本电脑通过 SSH 来控制操作你的机器人，完成任务。
$ ssh robot@192.168.1.101 (机器人 IP)，登录密码是 1

2）在笔记本电脑上开启两个 SSH 终端，分别启动以下节点
注意：在 SSH 终端打开后，请先运行
$ export ROS_HOSTNAME= 机器人 IP
$ export ROS_IP= 机器人 IP
终端一 $ roslaunch yzbot_nav yzbot_createmap.launch
终端二 $ roslaunch yzbot_nav keyboard_teleop.launch（此为使用键盘控制机器人移动，若使用无线遥控操手柄控请忽略本条）

3）在笔记本开启一个新终端，运行 RVIZ 查看画面(终端 3)

注意这不要 SSH，但是需要先在一个新终端里运行 ROS MASTER 指向。

$ export ROS_HOSTNAME= 笔记本电脑 IP

$ export ROS_MASTER_URI=http://机器人 IP:11311

$ export ROS_IP= 笔记本电脑 IP

$ rosrun rviz rviz -d `rospack find yzbot_nav`/gmapping.rviz

此时应该能看到原点附件的地图雏形

4) 开启一个新的 SSH 终端：

先运行$ roscd yzbot_nav/maps

再运行$ rosrun map_server map_saver -f map

你地图的名字就是上面的“map”， 可以在机器人电脑的 yzbot_nav/maps 文件夹里看到。至此，建图结束。若对地图不满意，可以再来一遍。

2.4.4 控制机器人在建好的地图里走到鼠标指定的位置。

1) 假设你已经完成了上面 2.4.3 的 SLAM 建图实验，生成的地图位于机器人电脑的 yzbot_nav/maps 目录下，地图名字是 map。

2）用你的笔记本电脑通过 SSH 登录到机器人电脑，开启 1 个 SSH 终端，分别运行以下节点：

注意：在 SSH 终端打开后，请先运行

$ export ROS_HOSTNAME= 机器人 IP

$ export ROS_IP= 机器人 IP

终端一 $ roslaunch yzbot_nav rs_navigation_startup.launch

3）在笔记本电脑的终端上（终端 2）运行 RVIZ。

此时实验室的地图应该出现在 RVIZ 的视窗内，如下图示。

注意这不要 SSH，但是需要先在一个新终端里运行 ROS MASTER 指向。

$ export ROS_HOSTNAME= 笔记本电脑 IP

$ export ROS_MASTER_URI=http://机器人 IP:11311

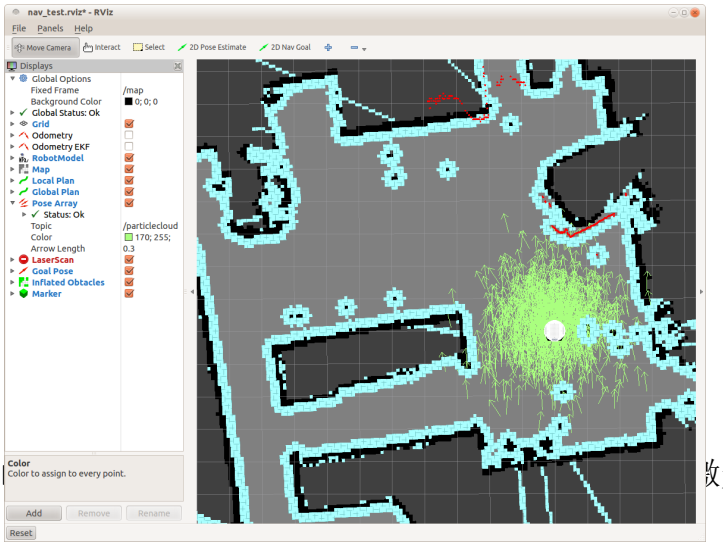
$ export ROS_IP= 笔记本电脑 IP

$ rosrun rviz rviz -d `rospack find yzbot_nav`/rs_nav.rviz 运行 RVIZ

【2D Pose Estimate】按钮用来初始化机器人当前位置

【2D Nav Goal】按钮用来给机器人设定一个需要行走到的目标位置。

【Publish Point】按钮用来查看鼠标指定位置的地图坐标值



先用【2 ... 数光斑点与地图上

的障碍物轮廓基本重合）然后用【2D Nav Goal】按钮，选取一个地图位置，机器人就会自动走过去

2.4.5 机器人在建好的地图里按任务规划行走。

当你已经完成了上述两个任务，接下来让机器人做一个稍微复杂一点的任务：选定 5 个不同的地点，让机器人在这 5 个不同地点之间来回走动，并在屏幕终端上打印出任务执行的成功率、累计行走的距离、累计运行的时间。这个程序是用 Python 脚本编写的，您可以稍作修改，实现自己定义的任务（比如：从 A 点到 B 点再到 C，每个点停留 5 分钟，然后循环往复）。

1）用 SSH 进入到机器人电脑的 yzbot_nav/nodes 目录，用 vim 或 nano 打开该目录下的 nav_test.py 文件，把 locations[xx]里的坐标值替换为您办公司地图里期望的 5 个不同位置坐标。这个位置坐标可以预先用 rviz 里的【Publish Point】按钮来查看（参看 2.4.4 节）。

2）用 SSH 登录机器人电脑，开启 2 个 SSH 终端：

注意：在 SSH 终端打开后，请先运行

$ export ROS_HOSTNAME= 机器人 IP

$ export ROS_IP= 机器人 IP

终端一 $ roslaunch yzbot_nav rs_navigation_startup.launch

终端二 $ roscd yzbot_nav/nodes

$ ./nav_rs.py

3）在笔记本自己的终端上（终端三）运行 RVIZ。

此时实验室的地图应该出现在 RVIZ 的视窗内。

注意这不要 SSH，但是需要先在一个新终端里运行 ROS MASTER 指向。

$ export ROS_HOSTNAME= 笔记本电脑 IP

$ export ROS_MASTER_URI=http://机器人 IP:11311

$ export ROS_IP= 笔记本电脑 IP

$ rosrun rviz rviz -d `rospack find yzbot_nav`/nav_test.rviz 运行 RVIZ

4) 如果一切运行正常，终端三的屏幕上会显示

*** Click the 2D Pose Estimate button in RViz to set the robot's initial…

此时您需要在 rviz 窗口上用【2D Pose Estimate】按钮用来设定机器人当前位置，然后机器人就会自动去执行 nav_test.py 里写好的任务了。任务执行的成功率、累计行走的距离、累计运行的时间等信息都会在终端三的屏幕上显示出来。

您可以仔细阅读上述示例的 launch 批处理命令的脚本内容，详细学习和了解机人是如何利用 ROS 官方代码进行定位和导航的。

§2.5 用 YZ 专用定位导航包来操控机器人定位、导航

ROS 官网的开源定位导航算法可以给初学者提供很好的入门学习指导，但是这些开源的定位导航算法有一定的局限性，在实际应用中还有很多局限性，定位容易偏差甚至错乱，运动导航效果也不是很好。对于真实的应用场景，深圳璞数技术有限公司开发了专门的有自主知识产权的机器人定位、导航算法，我们也把这些算法的可执行包集成到了 ROSYZ01/02 机器人系统中了，您可以通过替换相关 launch 命令行来直接使用这些有实用价值的定位导航算法。

注意：这些算法仅供 ROSYZ 平台研究学习使用，未经深圳璞数技术有限公司授权许可，请勿将这些软件用作商业用途，也请务拷贝给第三方。

2.5.1 地图相关知识和预处理工作

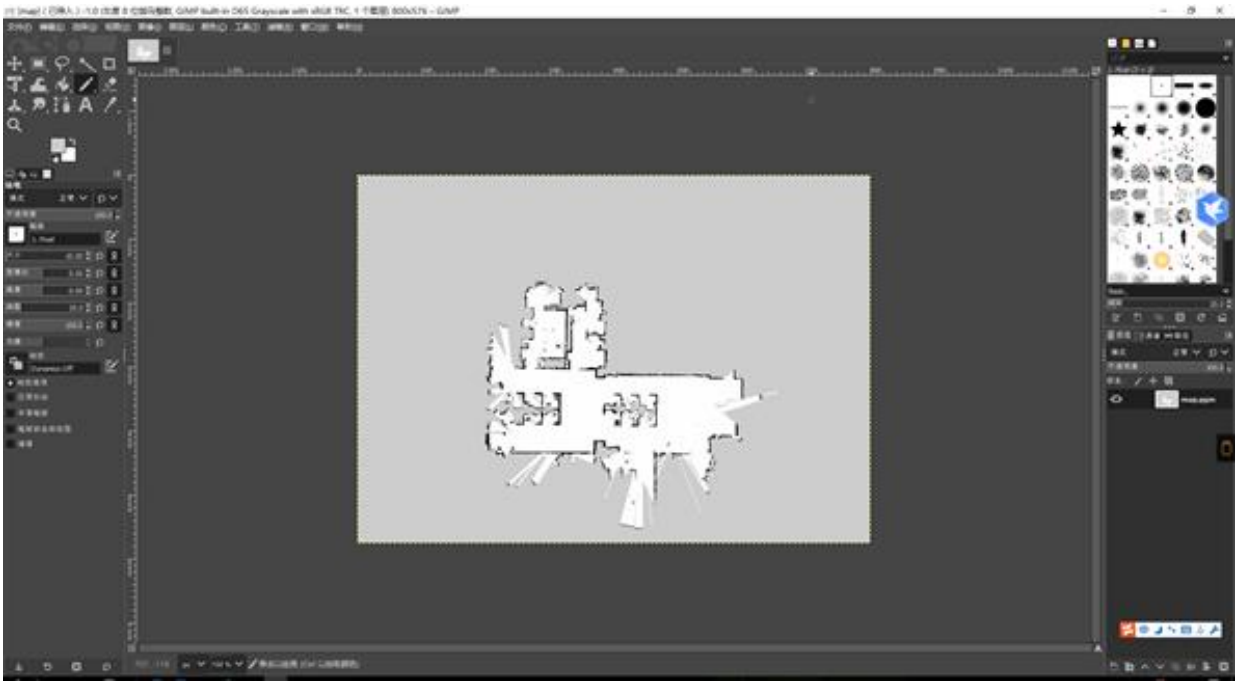
请参考 2.3.1 节或 2.4.3 节，为机器人工作区建好地图，为了能更好的满足客户的实际需要，比如，设定一些禁止机器人走入的区域，我们首先要对这个地图进行适当的预处理工作。

若使用的是机器人本地电脑，请用机器人电脑里的 GMIP 软件打开 ~/ws/src/yzbot_nav/maps 目录下的 map.pgm 图片。若机器人上没有安装屏幕和鼠标键盘，请先通过局域网把这个 map.pgm 图片拷贝到桌面电脑，然后用桌面电脑的 GMIP 或 PhotoShop 软件打开这个图片，示例如下：

设置地图尺寸：点击图像----》画布大小----》输入合适的宽度高度，裁剪出地图。

注意：歪斜的地图并不妨碍机器人定位导航，但是用户可能会看着不舒服，您可以先用地图旋转命令把图片旋转到您满意的角度，然后再裁剪地图

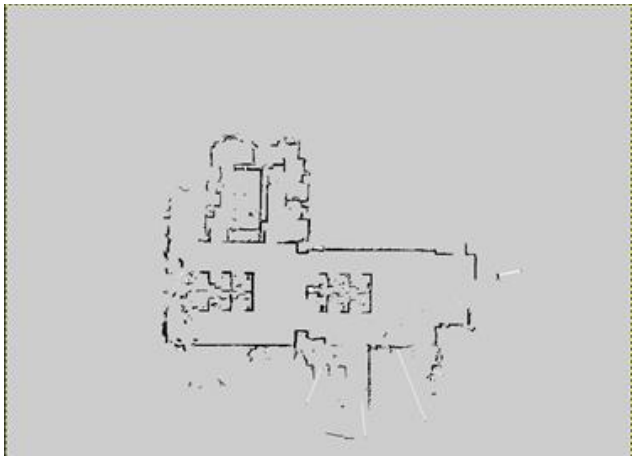
原则：将灰色无效区域裁剪掉，可以适当留一点。如图



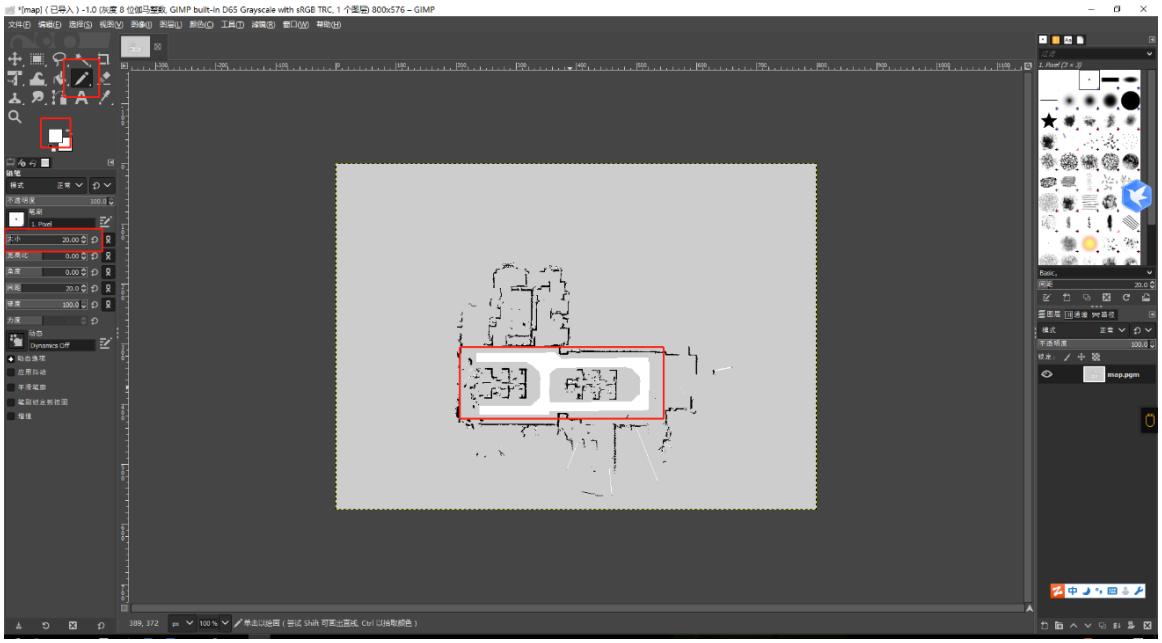
点击改变大小确认。

填充白色区域：

点击“油桶填充工具”，选色（按住 ctrl 后点击灰色区域），选取灰色后，将油桶点击白色区域后 效果如图



图画白色区域：此操作将直接影响机器人可行走空间区域，白色为可行走区域
选择铅笔工具，选色为白色，画笔大小酌情选择
涂画出白色线路，机器人将会只在白色区域内进行规划行走。涂画线路建议使用 shift 快捷键拉线条的方式，这样出来的效果比较平整。如图

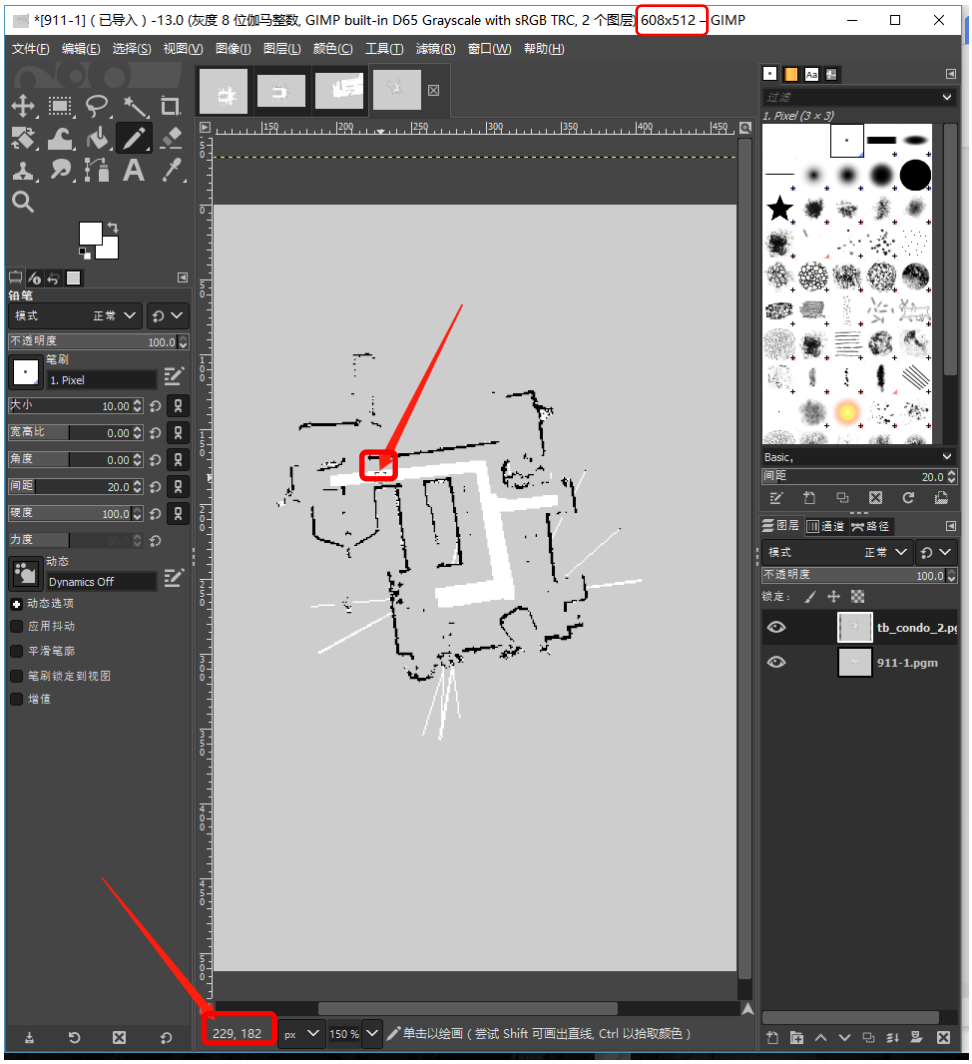


保存地图：
点击 文件---》导出（export as）---->点击导出（注意导出路径）
编辑完成后请将新地图放到/maps 目录下，命名为“map.pgm”.

编辑地图原则:涂灰留白
涂灰色：设置禁行区域，机器人将不会在这些灰色区域内行走。
画白色：设置行走区域，将允许机器人行走的区域涂成白色。

地图的像素与真实世界坐标关系的几个常识

1. 按照地图的使用习惯，我们把地图图片的左下角为设置为真实世界的原点（0，0），
2. 默认情况下，地图像素与真实世界的比例为 0.05 米/像素.这个参数是在生成该地图的 gmapping 批处理参数里设置的。0.05 米/像素适合一般情况下的室内建图，户外建图可以选 0.1 米/像素甚至更大。室内小面积建图也可以采用 0.025 米/像素。
   假设地图的大小为 600 X 500 像素，比例是 0.05 米/像素，那么这个地图对应的真实世界的区域就是 30 米长，25 米宽
3. 常规的电脑地图编辑软件（如 GMIP、PhotoShop 等）显示的图片的左上角像素的编号是（0，0），左下角是（0，H），H 是图像高度。因此图片像素编号和真实世界坐标的对应关系是： 世界坐标 x = 像素 X * 比例尺
   世界坐标 y = （图片高度 - 像素 Y） * 比例尺
   举例如下，此地图的宽度 W 是 608，高度 H=512，



上图中方框位置的像素编号为 229，182 ，这个像素对应的世界 x， y 坐标就是：

x = 229 * 0.05 = 11.45

y = (512 - 182) * 0.05 = 16.5 (512 是地图的总高度像素，不同地图此值会不同)

a 角度 = 朝向右方为 0.0，朝向左方为 3.14，朝向上方为 1.57，朝向下方为 -1.57

结论：上述像素点的坐标参数为 11.45， 16.50， 0.0（机器人面向右方）

2.5.2 用 YZ 专用定位导航包来操控机器人定位、导航

如果您已经完成了上述 2.5.1 的地图构建和地图编辑工作，请按照以下步骤实现用 YZ 专用导航包来操控机器人定位、导航任务：

1. 让机器人在建好的实验室地图里行走到指导的目标位置

基本操作指令与上面的 2.3.2 节或 2.4.4 节几乎相同，将旧地图更换为新地图

以 2.3.2 节为例，只需要把终端一的命令行改为：

终端 1 roslaunch yzbot_nav yzbot_navigation_startup.launch

2. 让机器人在建好的地图里按任务规划行走。

同样的，操作指令与之前的 2.3.3 节或 2.4.5 节几乎相同：

终端 1 $ roslaunch yzbot_nav yzbot_navigation_startup.launch

启动另一个执行任务程序

终端 2 $ roscd yzbot_nav/nodes

$ ./nav_yzbot.py

rviz 查看终端改成：

$ rosrun rviz rviz -d `rospack find yzbot_nav`/nav_test.rviz 运行 RVIZ

3. 机器人定位算法的 amcl 和 dmcl 简单介绍

ROS 官网推荐的定位算法包是 amcl, http://wiki.ros.org/amcl 是 ROS 官网对 amcl 包的详细介绍。要使 amcl 包能准确地实现机器人实时定位功能，先决条件是要将机器人运动前的初始位置 x, y, a(机器人朝向角度)给到 amcl. 有两种方法可以得到机器人的初始位置 x, y, a：

第一种方法就是前例子里提到的 Rviz 图示方式，先用【2D Pose Estimate】按钮把机器人初始位置设定好，让激光斑点与地图上的障碍物轮廓基本重合，此时查看 Rviz 终端窗口的打印内容，机器人的初始坐标值会打印在屏幕上，请记录下这些值。

第二种方法是用图片编辑软件（比如 GIMP 或 PhotoShop）打开地图图片，把鼠标放到机器人当前所在的位置，查看鼠标位置的像素编号（X,Y），按照上一节的方法计算出 x, y 坐标值，a 的取值也参考该节内容。

把得到的 x, y, a 这 3 个值写入到 install/share/yzbot_nav/launch/目录下的 rs_amcl_startup.launch 文档中对应 initial_pose_x ，_y 和_a 三个参数。只要每次机器人开机启动的时候处在这个位置附近，那么在接下来的机器人行走工组中，amcl 包就会实现机器人的实时定位。

ROS 官网提供的开源 amcl 算法有很好的参考学习作用，但是这个开源算法还有很多不足之处，若直接应用到商业化的真实场景中，您就会发现经常出现机器人定位跳变甚至完全错乱的情况。璞数技术有限公司开发的 dmcl 是在开源 amcl 算法基础上增加了动态障碍物剔除、特殊边界约束函数以及粒子生成算法重构等三个重大革新而得到的，经过验证，这个 dmcl 算法在众多复杂的真实场景下都能表现出非常优秀的精确定位作用。这个算法已经集成在了 ROS YZ01/02 机器人的 ROS 系统里面了，您之前运行的 yzbot_navigation_startup.launch 里面调用的就是这个 dmcl.

同样的，dmcl 也需要把机器人的初始位置的数值填写进去才能发挥精确的定位作用。请修改 install/share/yzbot_nav/launch/目录下的 yzbot_dmcl_startup.launch 参数。

截图如下：请将红色框里面的 3 个 0 修改为上面得到的 3 个新值

```
<?xml version="1.0"?>
<launch>
  <arg name="use_map_topic" default="false"/>
  <arg name="scan_topic" default="scan"/>
  <node name="map_server" pkg="map_server" type="map_server"
args="/home/robot/robot_dsr/map.yaml"/>
  <!--node pkg="amcl" type="amcl" name="amcl" clear_params="true"
output="screen"-->
  <node pkg="amcl_wyj2" type="amcl_wyj2" name="amcl_wyj2"
clear_params="true" output="screen">
    <param name="use_map_topic" value="$(arg use_map_topic)"/>
    <!-- Publish scans from best pose at a max of 10 Hz -->

    <param name="initial_pose_x" value="0"/>
    <param name="initial_pose_y" value="0"/>
    <param name="initial_pose_a" value="0"/>

    <param name="odom_model_type" value="diff"/>
    <param name="odom_alpha5" value="0.1"/>
    <param name="gui_publish_rate" value="10.0"/>
    <param name="laser_max_beams" value="60"/>
    <param name="laser_max_range" value="8.0"/>
    <param name="min_particles" value="500"/>
    <param name="max_particles" value="1000"/>
    <param name="kld_err" value="0.01"/>
    <param name="kld_z" value="0.99"/>
    <param name="odom_alpha1" value="0.2"/>
    <param name="odom_alpha2" value="0.2"/>
    <!-- translation std dev, m -->
    <param name="odom_alpha3" value="0.2"/>
    <param name="odom_alpha4" value="0.2"/>
    <param name="laser_z_hit" value="0.95"/>
    <param name="laser_z_short" value="0.1"/>
```

4. 机器人导航算法 move_base 和 yzbot_move_base 简单介绍

ROS 开源资料包里面常用的导航算法是 move_base，我们前面的 2.3.2、2.3.3、2.4.4、2.4.5 等例子中用到的就是 ROS 官网提供的 move_base 导航包，您可以访问这里来了解更详细的介绍 http://wiki.ros.org/move_base。

同样的，这个开源包对于用来学习基本的导航算法是很有帮助的，但是若将这个包直接用到商业产品上，那将会看到比较糟糕的运动效果。

本节 1、2 小节两个例子里调用的是 yzbot_move_base 这个专用算法，该算法

是璞数技术有限公司针对室内行走的服务机器人专门开发的算法，有很不错的运动表现，您可以在自己的产品中试试看。

§3 注意事项

3.1 预充电：机器人初次使用之前请先放置到充电桩上充电至少 3 小时以上，因为运输的原因，出厂的电池组基本上只有很少电量。

3.2 充电环境温度：特别注意 充电时请必须在 0~35 摄氏度的室温环境下充电。过高或过低的环境温度会损坏电池！

3.4 电源接口：请严格按照电源接口图示的脚位和极性接入外接电池和输出 DCDC 电源。错误的接线会损坏接口板或其他设备。使用机器人提供的 DCDC 电源给传感器或电脑主板供电时，请务必先确认设备使用的最大电流不会超过 DCDC 板的限定电流值。

3.4 故障排除：机器人在使用当中若出现功能异常，请先按下急停按钮，若故障不能排除，请关闭电源，然后重新启动，一般情况下机器人都会恢复正常，若重新启动后机器人仍然不能正常使用，请尽快通知芸众科技的技术服务人员进行远程指导。

3.5 紧急处理：请谨记以下重要安全事项

机器人行走异常时，请立即按下机器侧面的红色急停开关！

机器人或充电器有烟雾或异味出现时，请立即关闭机器电源！

机器人发生严重碰撞事故或摔倒事故时，请立即关闭电源！

3.6 ROSYZ 机器人必须由请由熟悉 ROS 系统的技术人员进行操作。

ROSYZ-01C 主要技术参数

运动底盘尺寸	52X48X29 CM，金属底盘框架+塑胶壳罩
运动底盘重量	25KG（含电池的裸机重量）
驱动方式	双路差速驱动
电池配置	16V21AH LIFEPO4
电机规格	12V-24V 高效能轮毂电机
配置传感器	EAI 激光雷达选配，超声波传感器
内置电脑	酷睿 i5 处理器工控机
电脑通信接口	USB X4 / VGA / 以太网口
电源输出接口	5V2A、12V2A、16V4A
运动速度范围	0.1-1.0 米每秒
最大承载重量	50KG
标配充电器	18.0V8A 智能充电器
自动充电桩	选配. 充电引导为激光+红外方式
底盘控制板 **MCU**	STM32F10X ARM 主芯片
紧急停车方式	拍按红色急停开关
上层平台高度	5CM
软件硬件支持	可提供软件示例代码、电路原理图

附录 A 底盘驱动节点发布/订阅的话题

一. 启动节点说明

1.底盘驱动节点的启动： roslaunch stm32_connect start_basecontrol.launch

二. 话题说明

1.自动充电话题: 话题名称：“Auto_Charging”

话题参数 auto_charging_flag 0:无效 1:开始充电

【例子】：输入终端命令：

rostopic pub /Auto_Charging robot_msgs/Charging_Control “auto_charging_flag:1”

即可开始充电

2.电量话题： 话题名称：“PMS_get_status”

话题参数：pms_charging_flag 0：非充电状态 1：充电状态

Pms_battary_level: n :电量百分比

【例子】：输入终端命令：rostopic echo /PMS_get_status

即可开始查询电量和充电状态

3.超声波避障话题：话题名称 ：“Ultrasound_result”

话题参数 ：cs_obs：0：无触发 1：前方触发 16：后方触发

17：前后触发

【例子】：输入终端命令：rostopic echo /Ultrasound_result

即可开始查询超声波触发状态

4.急停开关话题： 话题名称：”Wheel_Switch”

话题参数：Switch 0：松开 1：按下

【例子】：输入终端命令：rostopic echo /Wheel_Switch

即可开始查询急停开关状态

5. 速度话题：话题名称：“cmd_vel” 控制和显示底盘的线速度角速度

【例子】：输入终端命令：

rostopic pub /cmd_vel geometry_msgs/Twist “linear：

x:0.3 y:0.0 z:0.0 angular: x:0.0 y:0.0 z:0.1”

设备即可开始以 0.3 米每秒的线速度以及 0.1 米每秒的角速度运动。

6. Odom 话题：话题名称：“odom” ， odom 数据话题

【例子】：输入终端命令：rostopic echo /odom

即可开始查询 odom 话题信息

7. 超声波话题：话题名称“sonar1”显示超声波探测信息 sonar1~sonar4，分别是左前，右前，左后，右后超声探测传感器。

话题参数：field_of_view：超声探测锥形弧度

min_range：最小探测距离（米）

max_range：最大探测距离（米）

range：当前探测距离（米）

【例子】：输入终端命令：rostopic echo /sonar1

即可显示左前方超声探测传感器的信息

附录 B: 遥控手柄介绍及如和构建地图

一． 认识遥控控制手柄

初次使用时构建场地地图以及手动遥控机器人时，都必须依赖遥控手柄，下图是遥控手柄的正视图，各按键分布如下：

注： 手柄上的 X, Y, BACK, START 等四个按键在此没有使用，无定义。

二 使用介绍

机器人首先应该处于开启状态，此时机器人的液晶显示屏应该出现时钟，显示当前的时间和充电状态。

使用手柄控制机器人之前，请先按下手柄上方中间位置的圆形“模式选择键”，轻按一次就会正常开启手柄的遥控功能，此时手柄状态指示灯（共 4 个）的第 1 和第 3 会出现绿色常量。*（注意：若四个指示灯同时闪烁，表示机器人故障或者手柄距离机器人太远从而导致手柄收不到机器人信号；若出现 2 个指示灯灯闪，表示手柄内置的电池电量太低，请用标准 USB 线先给手柄充电 1 小时再用；若手柄指示灯的第 2 和第 4 亮起，表示模式和机器人不匹配，请再按下“模式选择键”）*

2.1 遥控机器人移动行走

机器人左右转向：按下“LB 键”的同时推动“机器人转向推杆”向左或右。

机器人前进后退：按下“LB 键”的同时推动“机器人行走推杆”向前或后。

2.2 遥控机器人去充电桩充电

当机器人在充电桩正前方 1 米左右时，此时若同时按下“LB 键”+“A 键”，机器人就会自动退回充电桩充电。

当机器人不在充电桩正前方 1 米左右时，请先用遥控手柄将机器人移动到充电桩正前方 1 米左右，然后再同时按下“LB 键”+“A 键”即可。

对于已经建好地图而且机器人正在按按地图进行定位导航运动时，若此时同时按下“LB 键”+“RB 键”+“A”键这 3 个键，机器人会被终止当前所有任务，自行回去充电。

附录 C: 产品外型尺寸图