



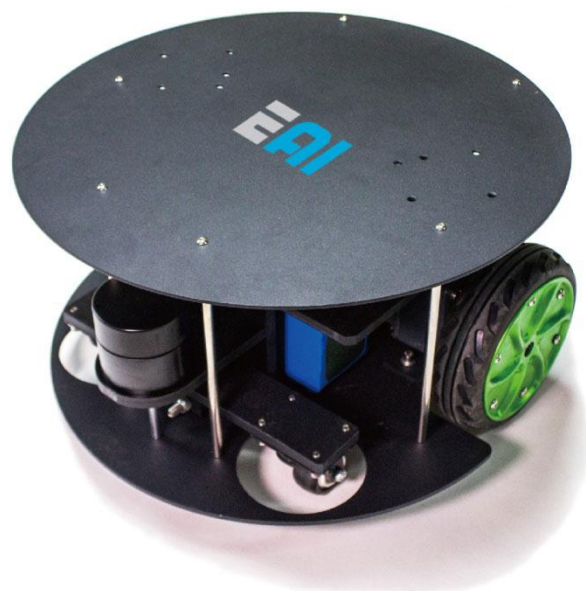
DASHGO

Build Your Robot

Dashgo E1

瞬驰机器人移动平台

使用说明



序言

尊敬的用户：

您好！感谢关注和支持EAI产品，EAI将与您一起技术创新，让人工智能融入生活！

深圳玩智商科技有限公司（玩智商 Enjoy AI，简称 EAI），成立于2015年，专注于机器人移动，客户群体面向全球。通过持续的创新，EAI科技致力于为机器人行业用户提供性能最强、体验最佳的智能移动产品和解决方案。

EAI科技的主要产品有激光雷达、定位导航模块和机器人移动平台。通过技术创新，公司把自主研发且拥有完全知识产权的核心技术：光磁无线技术，运用到激光雷达上，大大延长了激光雷达的使用寿命，确保了激光雷达长寿命、高可靠、高精度的性能。结合配套研发且拥有相关知识产权的定位导航模块，可以让机器人实现定位导航、路径规划、避障避险、物体跟踪等功能。机器人移动平台的通用性、可靠性、耐用性深受机器人企业、科研机构及高校教学、创客的欢迎，自主研发的核心结构保证了高精度、载重大、动力足、续航长和扩展性强的性能。

EAI技术团队不断完善技术方案，及时响应客户需求，再次感谢您的支持！

目录

序言	1
第 1 章 全面认识 EAI 智能移动产品	1
1.1 安全说明.....	1
1.1.1 符号及其含义	1
1.1.2 操作防范	1
1.1.3 电池安全	2
1.1.4 安全储存	2
1.2 接口说明.....	3
第 2 章 E1 之基本使用开发.....	4
2.1 蓝牙控制 E1 移动.....	4
2.2 蓝牙的开发使用	6
2.2.1 E1 的物理参数.....	6
2.2.2 速度控制指令	7
2.3 下位机的开发使用	9
2.3.1 连接方式	9
2.3.2 E1 的物理参数.....	9
2.3.3 支持指令	9
2.4 上位机通过串口给 E1 下发指令	12
第 3 章 E1 之 ROS 开发.....	13
3.1 导航模块的工作空间目录介绍	13
3.2 准备工作.....	14

3.2.1	ROS 系统安装.....	15
3.2.2	搭建 Dashgo 运行环境.....	19
3.3	移动控制.....	20
3.3.1	键盘控制移动.....	21
3.3.2	命令行 topic 控制移动.....	24
3.3.3	手机 APP 控制移动.....	25
3.4	精度校准.....	26
3.4.1	不带陀螺仪.....	27
3.4.2	带有陀螺仪.....	28
3.4.3	其他数据输出.....	29
3.5	E1 与激光雷达 F4 坐标校正.....	30
3.5.1	设置导航模块与 PC 的配置文件.....	31
3.6	VIM 的基本使用.....	37
3.7	不带陀螺仪建图导航.....	37
3.7.1	扫描建图.....	38
3.7.2	保存地图.....	41
3.7.3	自主导航.....	42
3.8	带陀螺仪建图导航.....	45
3.8.1	启动扫描建图.....	45
3.8.2	保存地图.....	48
3.8.3	自主导航.....	49
3.9	多点连续导航.....	53

3.9.1 设置起点 54

3.9.2 设置连续目标点 55

第 1 章 全面认识 EAI 智能移动产品

1.1 安全说明

感谢您购买 EAI 产品 E1、PS1000C、F4。为了您的安全,请在使用 EAI 产品前阅读说明书并特别注意以下安全标识。

1.1.1 符号及其含义

手册使用以下符号中的部分需要特别注意安全。



危险 该标记表示「极可能导致死亡或者重伤」的相关内容。



警告 该标记表示「极可能导致伤害或财产损害」的相关内容。



警惕 忽视此符号指令可能会有人身伤害的风险。



严令禁止 该图形标记表示不可实施的内容。



强制要求 该图形标记表示必须实施的内容。

1.1.2 操作防范



警惕



平台表面为金属材质，请勿与电路板直接接触。



平台边缘锋利，小心接触，防止划伤。



操控平台时避免速度过快，引起碰撞。



搬运时以及设置作业时，请勿落下或倒置。



严令禁止



非专业人员，不要私自对 EAI 产品进行拆卸。



不使用非原厂标配的电池、电源、充电座。




E1 运行时请勿用手触碰。



不要在有水的地方，存在腐蚀性、易燃性气体的环境内和靠近可燃性物质的地方使用。



不要放置在加热器或者大型卷线电阻器等发热体周围。


 切勿将电机直接与商用电源连接。


1.1.3 电池安全

为延长电池的寿命，避免充电过程事故的发生，请注意以下警示。





警告


 充电器在充电工作时，会向外界散发一定的热量，充电器与产品应一起放在通风干燥的环境中使用。

 正常充电时，充电指示灯为红色，当转为绿色时为充满。

 停止充电时，应先拔下 220V 插头，然后取下电池端插头。

 产品长期不用，需三个月至半年补充一次电。

 产品电池不可将电完全用完，否则会严重受损，容易造成不可修复。


 充电时间长于 24 小时无人看护的状态下，应切断电源，禁止长时间挂充。

1.1.4 安全储存


为防止事故发生和减小伤害，请不要在下面列出的条件下存放 EAI 产品。





警惕


 避免处在低于 20 摄氏度或高于 40 摄氏度以上温度。

 避免长期放置于阳光直射位置。

 避免处于泥土和多灰尘的环境。

 远离较强的振动环境。

 远离高湿度环境。

 远离静电环境。

1.2 接口说明





第 2 章 E1 之基本使用开发

2.1 蓝牙控制 E1 移动

E1 底盘内置蓝牙，通过手机 App 可以控制小车底盘的移动。

首先，在 Android 手机上安装 DashApp.apk，资料包附带。

安装 app 之后，打开软件便进入主界面，主界面主要是提供连接方法的选择，如下图所示：



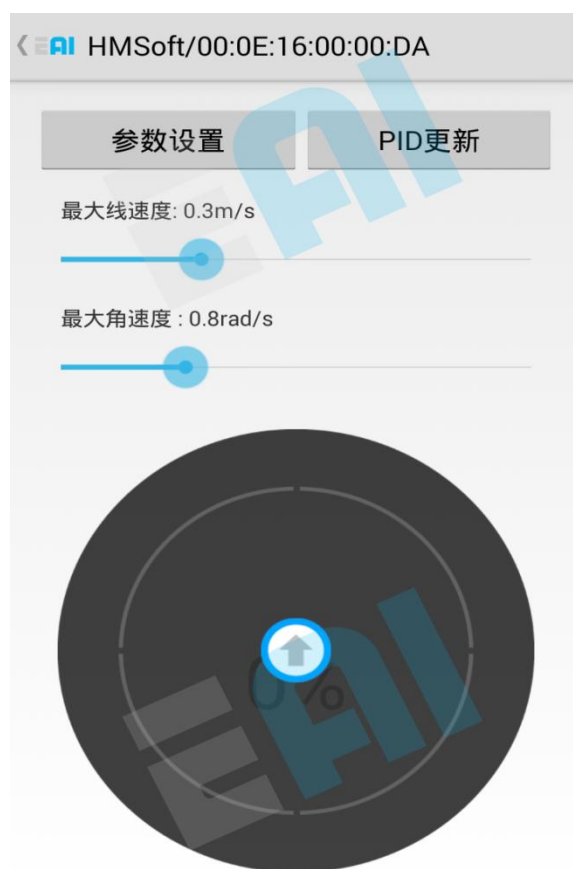
选择“**蓝牙**”便进入到蓝牙搜索界面，蓝牙搜索会自动进行，若不能便点击右上方的“**搜索**”按钮搜索蓝牙设备，如下图所示：



搜索蓝牙设备时，标题栏会有转圈，提示正在搜索设备，若搜索到便以列表的形式显示出来，显示的内容包括蓝牙的名称与蓝牙的地址，蓝牙设备名为“BT05”或“HMSoft”或以 E 加数字，如：E1202。

注意： 若有个别手机蓝牙搜索不到，请换别的手机试试。

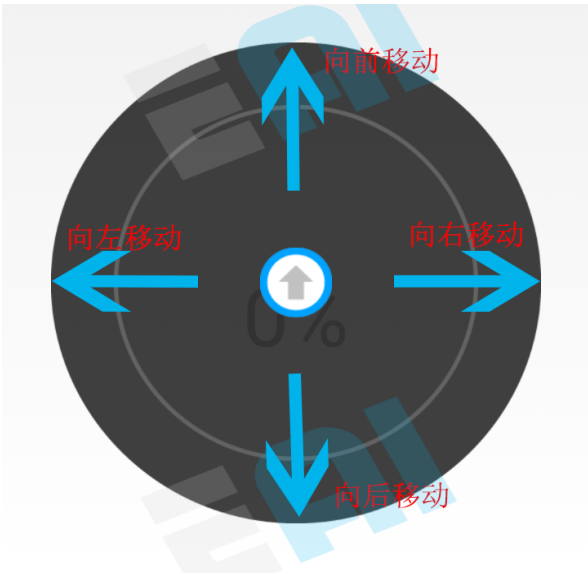
蓝牙搜索的时间为 6s，若没有搜索到想要使用的设备，就可点击标题栏右边的“搜索”按钮重新搜索蓝牙设备。若搜索到要使用的设备，便可点击选择该设备，连接成功后便进入蓝牙控制界面，如下图所示：



底盘移动控制，即下方圆形方向控制操作杆，以圆心为 0 m/s 的速度，圆半径为最大线速度，用手指从圆心向外滑动，速度就从 0%向 100%增加，滑动到圆边缘线上便达到最大的移动速度，默认最大线速度 0.2m/s,最大角度速度 0.5rad/s。

由圆心向前滑动，便控制底盘向前移动；由圆心向后滑动，便控制底盘向后移动；由圆心向左滑动，便控制底盘向左移动；由圆心向右滑动，便控制底盘向右移动。

当手指滑动的方向是水平向左或水平向右，便实现底盘原地向左或向右转。



注意：1. 第一次操作时，从圆心缓慢向外滑动，速度不要过快，以免发生碰撞伤到人或物。

2、如果蓝牙无法控制底盘移动，请注意红色的急停开关是否被按下了，如果按下了，请拧起来，再尝试。

2.2 蓝牙的开发使用

通过蓝牙向 E1 发送移动控制指令。

2.2.1 E1 的物理参数

wheel_diameter: 0.1620	//轮子直径,单位：米
wheel_track: 0.3300	//两个轮子的间距,单位：米
encoder_resolution: 915	
PID_RATE: 30	//PID 调节 PWM 值的频率

2.2.2 速度控制指令

基本的速度指令格式，如下：

```
z 20 -20;
```

说明：指令用于设置在单位时间（1/PID_RATE 秒）内 E1 移动期望编码器的脉冲数。

第一个参数用于设置左轮的速度，第二参数用于设置右轮的速度，正数代表轮子向前移动，负数代表轮子向后移动。

注意：这里所说的速度不是以 m/s 的速度，而是需要一定的计算来和 m/s 进行换算。

1. 只有线速度

假如 E1 以 0.3m/s 的速度前进，指令的参数计算方法如下：

E1 轮子转动一圈，移动的距离为轮子的周长：

```
wheel_diameter * 3.1415926
```

E1 轮子转动一圈，编码器产生的脉冲信号为：

```
encoder_resolution
```

所以每移动 1 米产生脉冲信号为：

```
ticks_per_meter
= encoder_resolution / (wheel_diameter * 3.1415926)
= 1200 / (0.12 * 3.1415926)
= 3183.10
```

E1 以 V=0.3m/s 的速度前进，1 秒钟内产生的脉冲信号为：

```
V * ticks_per_meter
```

又因为 PID 的频率是 1 秒钟 30 次。所以，指令的参数计算方法为：

```
int(V * ticks_per_meter / PID_RATE)
= int(0.3 * 3183.10 / 30)
= 32
```

所以，如果设置 E1 以 0.3m/s 的速度前进，指令输入为：

```
z 32 32;
```

如果以 0.3m/s 的速度后退，指令输入为：

```
z -32 -32;
```

2. 只有角速度

假如 E1 自转，就需要一个轮子正转，一个轮子反转。例如，需要 E1 以 $V=1$ 弧度/秒的速度转动。计算方法如下：

```
左轮  $v_l = V * \text{wheel\_track} / 2.0$   
右轮  $v_r = -1 * V * \text{wheel\_track} / 2.0$ 
```

然后,按公式 $\text{int}(V * \text{ticks_per_meter} / \text{PID_RATE})$ 分别计算左轮和右轮的速度参数。

若以 1 弧度/秒的速度转动，指令输入为

```
z 19 -19;
```

3. 线速度与角速度都有

假如 E1 向左转弯，就需要一个轮子向前转，一个轮子向后转。例如，需要 E1 以 $V_1=1$ 弧度/秒的速度转动， $V_2=0.2\text{m/s}$ 的速度前进。计算方法如下：

```
左轮  $v_l = V_2 + V_1 * \text{wheel\_track} / 2.0$   
右轮  $v_r = V_2 - V_1 * \text{wheel\_track} / 2.0$ 
```

然后,按公式 $\text{int}(V * \text{ticks_per_meter} / \text{PID_RATE})$ 分别计算左轮和右轮的速度参数。

若以 1 弧度/秒的速度转动，指令输入为

```
z 40 3;
```

注意：当需要 E1 持续运动是，需要不断地下发指令，如果下位机 2 秒内没收到指令，E1 将停止运行。

2.3 下位机的开发使用

2.3.1 连接方式

使用 E1 配送的 USB-B 型接口线与上位机（如：PC，树莓派等）相连

- 官网下载 **arduino IDE** 并安装（包含 **arduino 驱动安装**）
- 打开 **arduino IDE** 或其他串口调试工具，如：**SerialPortUtility** 等，波特率设置为 **115200**

2.3.2 E1 的物理参数

wheel_diameter: 0.1620	//轮子直径,单位：米
wheel_track: 0.3300	//两个轮子的间距,单位：米
encoder_resolution: 915	
PID_RATE: 30	//PID 调节 PWM 值的频率

2.3.3 支持指令

注意：指令输入中的 **\r** 代表按下回车键。

1. 获取波特率的值

输入：b\r
输出：115200\r

说明：E1 默认设置下位机和上位机通讯的串口比特率为 115200。

该指令(b)总是会返回固定值 115200。该指令主要用于验证刚开机时下位机和上位机通讯是否正常。

2. 读取编码器当前值

输入：e\r
输出：20 20\r

输出类型：int

说明：指令(e)返回 E1 左右轮编码器当前值。

根据编码器的分辨率（`encoder_resolution`），就可以推算出 E1 移动的距离、朝向、单位时间内的线速度和角速度。

注意： 编码器当前值是累加的，输出类型为 `int`，取值范围在 `-32768` 到 `+32767` 之间，需处理最大和最小值溢出问题。

3. 重置编码器的值

输入： `r\r`
输出： `ok\r`

说明：指令(`r`)用于把 E1 左右轮编码的计数值重置为零。

每次开机时需要重置下左右轮编码的计数值，防止起始值不正确导致推算出错误的 E1 状态。

4. 设置速度的期望值

输入： `z 20 -20;\r`
输出： `ok\r`

说明：指令(`z 20 -20;`)用于设置在单位时间（`1/PID_RATE` 秒）内 E1 移动期望编码器的脉冲数。

第一个参数用于设置左轮的速度，第二参数用于设置右轮的速度，正数代表向前移动，负数代表向后移动。

注意： 这个速度的单位不是 `m/s`， 需要一定的计算来和 `m/s` 进行换算。

举例：

假如期望 E1 以 `0.3m/s` 的速度前进，指令的参数计算方法如下：

E1 轮子转动一圈，移动的距离为轮子的周长：

`wheel_diameter * 3.1415926`

E1 轮子转动一圈，编码器产生的脉冲信号为：

```
encoder_resolution
```

所以每移动 1 米产生脉冲信号为:

```
ticks_per_meter
= encoder_resolution / (wheel_diameter * 3.1415926)
= 1200 / (0.1260 * 3.1415926)
= 3031.52278
```

E1 以 $V=0.3\text{m/s}$ 的速度前进, 1 秒钟内产生的脉冲信号为:

```
V * ticks_per_meter
```

又因为 PID 的频率是 1 秒钟 30 次。所以, 指令的参数计算方法为:

```
int(V * ticks_per_meter / PID_RATE)
= int(0.3 * 3031.52278 / 30)
= 30
```

所以, 如果要设置 E1 以 0.3m/s 的速度前进, 指令输入为:

```
z 30 30;\r
```

如果以 0.3m/s 的速度后退, 指令输入为:

```
z -30 -30;\r
```

假如 E1 需要转弯, 就需要一个轮子正转, 一个轮子反转。例如, 需要 E1 以 $V=1$ 弧度/秒的速度转动。计算方法如下:

```
左轮 vl = V * wheel_track / 2.0
右轮 vr = -1 * V * wheel_track / 2.0
```

然后再按公式 $\text{int}(V * \text{ticks_per_meter} / \text{PID_RATE})$ 分别计算左轮和右轮的速度参数。

如果以 1 弧度/秒 的速度转动, 指令输入为

```
z 17 -17;\r
```

注意: 当需要 E1 持续运动是, 需要不断地下发指令, 如果下位机 2 秒内没收到指令, E1 将停止运行。

5. 超声波测距

输入: p\r
输出: 179 340 10 240\r

输出距离单位: 厘米(cm)

前面 3 个超声波, 后面 1 个超声波。输出的值顺序是: 前面左边、前面中间、前面右边、后面中间。

6. 更新电机控制 PID

输入: u 10:12:0:50\r
输出: ok\r

默认 PID 值(Kp:Kd:Ki:Ko)为 20:0:0:50

说明: 该参数用于电机对期望速度的自我调整, 一般情况下不需要改动。

7. 注意事项

1.输入指令的格式: 以小写字母开头, 跟若干个参数, 每个参数之间以空格分隔, 最后以单字符\r (相当于“回车”) 或者英文字符“;”作为结束符。形如:

z 20 -20;\r

2.输出指令的格式: 一个或多个返回值, 以空格分隔最后以单字符\r (相当于“回车”) 作为结束符。形如:

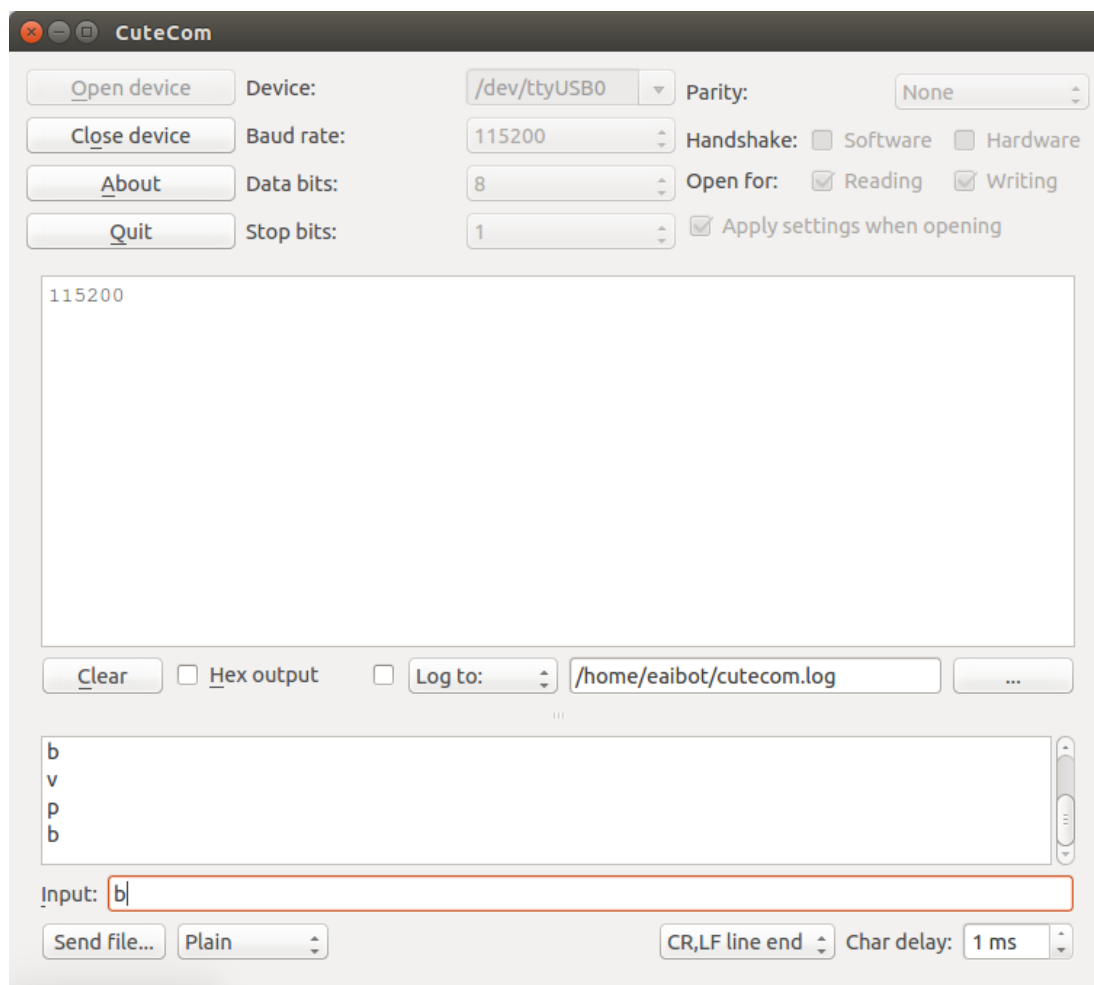
ok\r

2.4 上位机通过串口给 E1 下发指令

1.在 linux 中安装好串口工具, 例如 `sudo apt-get install cutecom` 安装, 并在终端中 `cutecom` 启动。

2.用串口线连接地盘和上位机 (pc), 打开串口工具如下:

根据实际选择底盘串口(/dev/ttyUSB0),点击 open device 打开串口,然后选择 CR,LF line end 表示回车换行, 最后在 input 中输入 b 并按下回车, 正常会返回串口波特率 115200。



第 3 章 E1 之 ROS 开发

3.1 导航模块的工作空间目录介绍

导航模块的默认固定 IP 是 192.168.31.200, 默认用户名为 eaibot , 密码为 eaibot 。

打开终端, 使用 ssh 登录到导航模块

```
$ ssh eaibot@192.168.31.200
```

```
eaibot@DashgoD1:~ $ cd dashgo_ws/
eaibot@DashgoD1:~/dashgo_ws $ ls
build devel src
```

dashgo_ws 便是一个 ROS 工作空间，dashgo_ws 目录下有三个文件夹：build 和 devel 是编译生成的文件夹，src 才是存放源码的文件夹。若有必要，可以自行删除 build 和 devel 文件夹，再使用 catkin_make 命令重新编译源码。

```
eaibot@DashgoD1:~/dashgo_ws $ cd src
eaibot@DashgoD1:~/dashgo_ws/src $ ls
CMakeLists.txt  dashgo
eaibot@DashgoD1:~/dashgo_ws/src $ cd dashgo/
eaibot@DashgoD1:~/dashgo_ws/src/dashgo $ ls
dashgo_description  dashgo_nav  dashgo_tools  flashgo  README.md
dashgo_driver        dashgo_rviz  docs          pathgo_imu
eaibot@DashgoD1:~/dashgo_ws/src/dashgo $
```

重新编译源码操作如下：

```
$ cd ~/dashgo_ws
$ rm -rf devel
$ rm -rf build
$ catkin_make
```

src 下文件夹可以是 ROS 包也可以是项目包，项目包包括多个 ROS 包。dashgo 便是项目包，dashgo 下包括 dashgo_driver、dashgo_nav、flashgo 等多个不同功能的 ROS 包：

dashgo_driver 是 Dashgo 小车的 ROS 驱动包；

dashgo_description 是 Dashgo 小车 3D 模型文件包；

dashgo_nav 是基于 dashgo_driver 的建图导航包；

dashgo_rviz 是 rviz 图形界面包；

dashgo_tools 是 Dashgo 小车调试工具包；

flashgo 是 Flash Lidar F4 的 ROS 驱动包；

pathgo_imu 是陀螺仪程序包；

docs 是用于存放说明文档；

README.md 是简单的工程说明文件。

3.2 准备工作

拷贝资料包内含 dashgo_ws 文件夹到 PC 端，然后使用 catkin_make 编译工程，让其在

PC 端也生效。

3.2.1 ROS 系统安装

1. 安装准备

一台普通的 PC，笔记本和台式机均可，32 位或 64 位都行。建议内存 4G 或以上(内存太小，有些 3D 模拟可能运行不起来)。

ROS 是需要运行在 Ubuntu 操作系统之上，建议使用 Ubuntu 16.04 和 ROS Kinetic。

注意：目前最新导航模块版本是 Ubuntu 16.04+ ROS Kinetic，新用户需要按文档在电脑上安装好 Ubuntu 16.04 和 ROS Kinetic，老用户可以维持原来电脑环境不变（电脑已安装 Ubuntu 14.04 +ROS Indigo+ Indigo 版本的 dashgo_ws 工程目录），使用方式和原来的一样，电脑远程到导航模块启动扫图 gmapping，然后再电脑上启动 dashgo_rviz 显示地图，这样可以快速与新导航模块对接用起来。

在操作过程中，建议使用有线网络，以免出现意外错误。

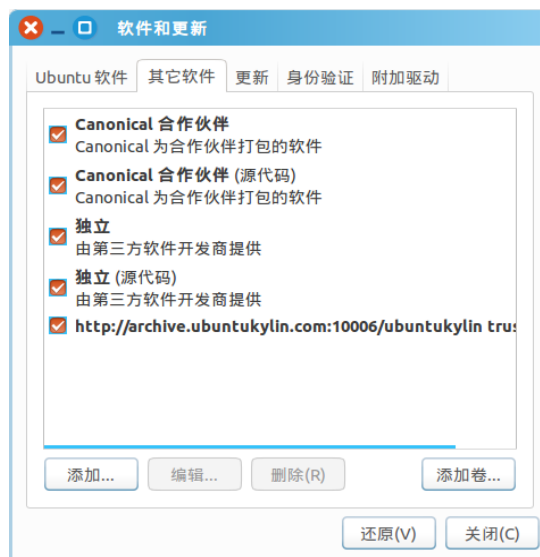
2. 配置 Ubuntu 软件仓库

配置你的 Ubuntu 软件仓库(repositories) 以允许 “restricted”、“universe” 和 “multiverse”这三种安装模式，服务器要选择国内的。

系统设置 》 软件和更新 》 Ubuntu 软件，将设置修改成如下图所示：



系统设置 》 软件和更新 》 其它软件，将设置修改成如下图所示：



点击 关闭(C) 按钮，等待缓存更新完成。

3. 配置 ROS 的 apt 源

ROS 的 apt 源有官方源、国内 USTC 源或新加坡源可供选择，选择其一就可以了，建议使用国内 USTC 源或新加坡源，安装速度会快很多。（安装过程中，建议使用有线网络，不容易出错。）

◆ 方式一：官方源

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

```
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key \
421C365BD9FF1F717815A3895523BAEEB01FA116
$ sudo apt-get update
```

方式二：国内 USTC 源

URL : <http://mirrors.ustc.edu.cn/ros/>

```
$ 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'
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key \
421C365BD9FF1F717815A3895523BAEEB01FA116
$ sudo apt-get update
```

◆ 方式三：新加坡源

URL : <http://mirror-ap.packages.ros.org/>

```
$ sudo sh -c ' ./etc/lsb-release && echo "deb http://mirror-ap.packages.ros.org/ros/ubuntu/ \
$DISTRIB_CODENAME main" > /etc/apt/sources.list.d/ros-latest.list'
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key \
421C365BD9FF1F717815A3895523BAEEB01FA116
$ sudo apt-get update
```

sudo apt-get update 执行更新有时因为网络原因可能出现错误（若不是 ros 安装源错误均可继续 ros 安装操作），可重新执行命令进行更新。

4. 安装 ROS 软件包

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

升级了 71 个软件包，新安装了 799 个软件包，要卸载 0 个软件包，有 314 个软件包未被升级。

需要下载 390 MB 的软件包。

解压缩后会消耗掉 1,620 MB 的额外空间。

`sudo apt-get install ros-kinetic-desktop-full` 安装 ROS Kinetic 时，如果在下载完时，没有进行解压，再/opt/ 下没有 ROS 目录，可能是更新源选错了，导致没下载完，无法解压安装 ROS，需要更换到国内源，然后 `sudo apt-get update` 重新安装

5. 配置环境变量

```
$ sudo rosdep init
$ rosdep update

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

6. 测试 ROS 安装是否成功

在终端输入 `roscore -h`，输出如下所示，表示安装成功。

```
$ roscore -h
Usage: roscore [options]

roscore will start up a ROS Master, a ROS Parameter Server and a rosout
logging node

Options:
-h, --help show this help message and exit
-p PORT, --port=PORT master port. Only valid if master is launched
-v verbose printing

See http://www.ros.org/wiki/roscore
```

在终端输入 `roscore`，输出如下所示，表示环境配置成功，ros 正常运行。

```
eaibot@eaibot:~$ roscore
... logging to
/home/eaibot/.ros/log/45d93ed8-a23a-11e6-99b1-4437e63de0fc/roslaunch-eaibot-3460.log
Checking log directory for disk usage. This may take awhile.
```

```

Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://eaibot:35377/
ros_comm version 1.11.20

SUMMARY
=====

PARAMETERS
* /roscdistro: kinetic
* /rosversion: 1.11.20

NODES

auto-starting new master
process[rosmaster]: started with pid [3472]
ROS_MASTER_URI=http://eaibot:11311/

setting /run_id to 45d93ed8-a23a-11e6-99b1-4437e63de0fc
process[roscout-1]: started with pid [3485]
started core service [/roscout]

```

3.2.2 搭建 Dashgo 运行环境

注意：老用户可以保持原来的环境不变（Ubuntu 14.04 +ROS Indigo+Indigo 版本的 dashgo_ws 工程目录），按照原来的方式与新导航模块模块（Ubuntu 16.04 +ROS Kinetic）对接，不必重装系统 Ubuntu 16.04 和 ROS Kinetic， 这样方便使用。

1. 设置用户的串口读取权限

```
$ sudo usermod -a -G dialout your_user_name
```

your_user_name 替换为实际用户名。

2. 安装依赖包

```
$ sudo apt-get install git python-serial ros-kinetic-serial g++ \
ros-kinetic-turtlebot-rviz-launchers ros-kinetic-teleop-twist-keyboard \
ros-kinetic-move-base-msgs libghc-sdl-image-dev libsdl-image1.2-dev \
ros-kinetic-navigation ros-kinetic-slam-gmapping ros-kinetic-teb-local-planner
```

3. 获取并编译 dashgo_ws 工程包

请确认自己的环境是 Ubuntu 14.04 +ROS Indigo, 还是 Ubuntu 16.04 +ROS Kinetic, 并从资料包内选择适合的 dashgo_ws 包版本, 然后把 dashgo_ws 文件夹放在当前用户主文件夹中, (即 ~/ 目录中)。

```
eaibot@eaibot:~$ cd ~
eaibot@eaibot:~$ cd dashgo_ws
eaibot@eaibot:~$ sudo chmod 777 ./* -R
eaibot@eaibot:~/dashgo_ws$ ls
build  devel  src
eaibot@eaibot:~/dashgo_ws$ rm -rf build/
eaibot@eaibot:~/dashgo_ws$ rm -rf devel/
eaibot@eaibot:~/dashgo_ws$ catkin_make
```

dashgo_ws 文件夹复制完成后, 放在当前用户主文件夹中, 切换到 dashgo_ws 下将 build 与 devel 文件夹使用 rm 命令删掉, 重新使用 catkin_make 编译。

catkin_make 编译完成后, 添加 Dashgo 环境变量 ~/.bashrc 文件中。

```
$ echo "source ~/dashgo_ws/devel/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

source ~/.bashrc 使环境变量的配置生效。

3.3 移动控制

导航模块的默认固定 IP 是 192.168.31.200, 默认用户名为 eaibot, 密码为 eaibot。

3.3.1 键盘控制移动

1、不带陀螺仪的底盘驱动

打开一个终端，运行以下命令，启动底盘驱动(带平滑加减速)

```
$ ssh eaibot@192.168.31.200
$ roslaunch dashgo_driver demo.launch
```

运行成功后，如下图所示

```
NODES
/
  dashgo_driver (dashgo_driver/dashgo_driver.py)
  nodelet_manager (nodelet/nodelet)
  velocity_smoother (nodelet/nodelet)

auto-starting new master
process[master]: started with pid [4829]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 4f480ee6-c106-11e6-95bb-b827eb607381
process[rosout-1]: started with pid [4842]
started core service [/rosout]
process[dashgo_driver-2]: started with pid [4857]
process[nodelet_manager-3]: started with pid [4861]
process[velocity_smoother-4]: started with pid [4862]
[DEBUG] [WallTime: 1481614358.525881] init_node, name[/dashgo_driver], pid[4857]
[DEBUG] [WallTime: 1481614358.527090] binding to 0.0.0.0 0
[DEBUG] [WallTime: 1481614358.527965] bound to 0.0.0.0 35218
[DEBUG] [WallTime: 1481614358.529374] ... service URL is rosrpc://DashgoK1:35218
[DEBUG] [WallTime: 1481614358.530279] [/dashgo_driver/get_loggers]: new Service
instance
[DEBUG] [WallTime: 1481614358.536734] ... service URL is rosrpc://DashgoK1:35218
[DEBUG] [WallTime: 1481614358.537633] [/dashgo_driver/set_logger_level]: new Ser
vice instance
Connecting to Arduino on port /dev/dashgo ...
Connected at 115200
Arduino is ready.
[INFO] [WallTime: 1481614359.591525] Connected to Arduino on port /dev/dashgo at
115200 baud
Updating PID parameters
[INFO] [WallTime: 1481614359.768607] Started base controller for a base of 0.42m
wide with 860 ticks per rev
[INFO] [WallTime: 1481614359.770944] Publishing odometry data at: 10.0 Hz using
base_footprint as base frame
```

打开另一个终端，运行以下命令，实现键盘控制移动

```
$ ssh eaibot@192.168.31.200
$ rosrn dashgo_tools teleop_twist_keyboard.py
```

运行操作结果，如下图所示

```

eaibot@DashgoD1:~ $ rosrun dashgo_tools teleop_twist_keyboard.py
the rosdep view is empty: call 'sudo rosdep init' and 'rosdep update'

Reading from the keyboard and Publishing to Twist!
-----
Moving around:
  u      i      o
  j      k      l
  m      ,      .
For Holonomic mode (strafing), hold down the shift key:
-----
  U      I      O
  J      K      L
  M      <      >
t : up (+z)
b : down (-z)
anything else : stop
q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
CTRL-C to quit

currently:      speed 0.3      turn 0.6
iiiiii,jjjjjjjj

```

2、带有陀螺仪的底盘驱动

打开一个终端，运行以下命令，启动底盘驱动(带陀螺仪)

```

$ ssh eaibot@192.168.31.200
$ roslaunch dashgo_driver demo_imu.launch

```

运行成功后，如下图所示

```

NODES
/
  dashgo_driver (dashgo_driver/dashgo_driver.py)
  nodelet_manager (nodelet/nodelet)
  velocity_smoother (nodelet/nodelet)

auto-starting new master
process[roscout-1]: started with pid [4829]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 4f480ee6-c106-11e6-95bb-b827eb607381
process[roscout-1]: started with pid [4842]
started core service [/roscout]
process[dashgo_driver-2]: started with pid [4857]
process[nodelet_manager-3]: started with pid [4861]
process[velocity_smoother-4]: started with pid [4862]
[DEBUG] [WallTime: 1481614358.525881] init_node, name[/dashgo_driver], pid[4857]
[DEBUG] [WallTime: 1481614358.527090] binding to 0.0.0.0 0
[DEBUG] [WallTime: 1481614358.527965] bound to 0.0.0.0 35218
[DEBUG] [WallTime: 1481614358.529374] ... service URL is rosrpc://DashgoK1:35218
[DEBUG] [WallTime: 1481614358.530279] [/dashgo_driver/get_loggers]: new Service
instance
[DEBUG] [WallTime: 1481614358.536734] ... service URL is rosrpc://DashgoK1:35218
[DEBUG] [WallTime: 1481614358.537633] [/dashgo_driver/set_logger_level]: new Ser
vice instance
Connecting to Arduino on port /dev/dashgo ...
Connected at 115200
Arduino is ready.
[INFO] [WallTime: 1481614359.591525] Connected to Arduino on port /dev/dashgo at
115200 baud
Updating PID parameters
[INFO] [WallTime: 1481614359.768607] Started base controller for a base of 0.42m
wide with 860 ticks per rev
[INFO] [WallTime: 1481614359.770944] Publishing odometry data at: 10.0 Hz using
base_footprint as base frame

```

打开另一个终端，运行以下命令，实现键盘控制移动

```

$ ssh eaibot@192.168.31.200
$ roslaunch dashgo_tools teleop_twist_keyboard.py

```

运行操作结果，如下图所示

```

eaibot@DashgoD1:~ $ rosrn dashgo_tools teleop_twist_keyboard.py
the rosdep view is empty: call 'sudo rosdep init' and 'rosdep update'

Reading from the keyboard and Publishing to Twist!
-----
Moving around:
   u   i   o
   j   k   l
   m   ,   .
For Holonomic mode (strafing), hold down the shift key:
-----
   U   I   O
   J   K   L
   M   <   >
t : up (+z)
b : down (-z)
anything else : stop
q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
CTRL-C to quit

currently:      speed 0.3      turn 0.6
iiiiii,jjjjjjjj

```

3.3.2 命令行 topic 控制移动

1、不带陀螺仪的底盘驱动

打开一个终端，运行以下命令，启动底盘驱动(带平滑加减速)

```

$ ssh eaibot@192.168.31.200
$ roslaunch dashgo_driver demo.launch

```

打开另一个终端，运行以下命令，实现命令行控制移动

```

$ ssh eaibot@192.168.31.200
$ rostopic pub -r 10 /smoother_cmd_vel geometry_msgs/Twist \
  '{linear: {x: 0.2, y: 0, z: 0}, angular: {x: 0, y: 0, z: 0}}'

```

运行操作结果，如下图所示

```

eaibot@DashgoD1:~ $ rostopic pub -r 10 /smoother_cmd_vel geometry_msgs/Twist '{l
linear:{x: 0.2,y: 0,z: 0},angular:{x: 0,y: 0,z: 0}}'

```

注意： 命令行运行后，小车便会一直运行，Ctrl + C 停止命令行。

2、带有陀螺仪的底盘驱动

打开一个终端，运行以下命令，启动底盘驱动(带陀螺仪)

```
$ ssh eaibot@192.168.31.200  
$ roslaunch dashgo_driver demo_imu.launch
```

打开另一个终端，运行以下命令，实现命令行控制移动

```
$ ssh eaibot@192.168.31.200  
$ rostopic pub -r 10 /smoother_cmd_vel geometry_msgs/Twist \  
'{linear: {x: 0.2, y: 0, z: 0}, angular: {x: 0, y: 0, z: 0}}'
```

3.3.3 手机 APP 控制移动

启动一个终端，运行以下命令启动底盘驱动(带平滑加减速)

```
$ ssh eaibot@192.168.31.200  
$ roslaunch dashgo_driver demo.launch
```

或者启动底盘驱动(带陀螺仪)

```
$ ssh eaibot@192.168.31.200  
$ roslaunch dashgo_driver demo_imu.launch
```

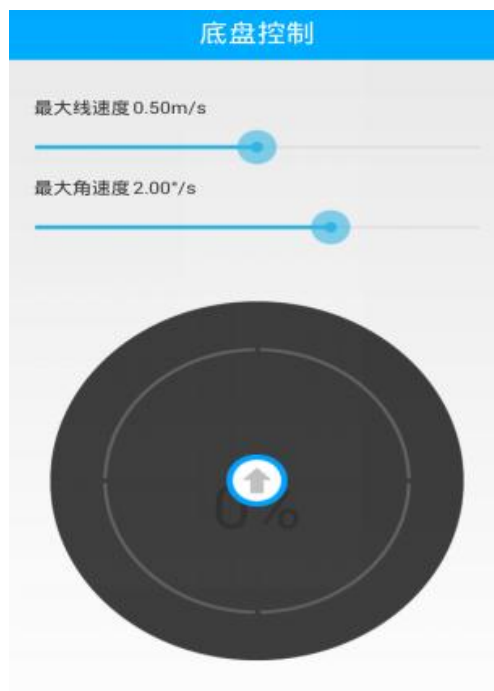
通过 EAI 团队开发的手机 APP 控制，目前仅支持 Android。

在 APP 启动界面，选择“WIFI”便进入到 WiFi 连接界面，如下图所示：

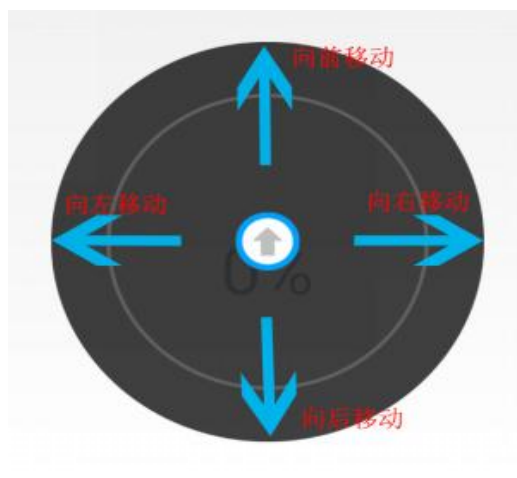


需要输入的 Master IP 是导航模块的 IP 地址，即 192.168.31.200。

然后点击“连接”，连接成功后，界面如下：



方向的操控，如下图所示：



注意：ROS 系统的 IP 必须与手机端的 IP 在同一个网段，即两者要连在同一个路由器上。

3.4 精度校准

导航模块的默认固定 IP 是 192.168.31.200，默认用户名为 eaibot，密码为 eaibot。

底盘运行的精准度是衡量小车的重要标准。主要关注走直线的误差和转动角度的误差。

3.4.1 不带陀螺仪

打开一个终端，启动底盘驱动(带平滑加减速)

```
$ ssh eaibot@192.168.31.200
$ roslaunch dashgo_driver demo.launch
```

然后在另一个终端运行测试脚本。

- 测试一：前进 1 米

```
$ ssh eaibot@192.168.31.200
$ rosrundashgo_tools check_linear.py
```

- 测试二：原地转动 360 度

```
$ ssh eaibot@192.168.31.200
$ rosrundashgo_tools check_angular.py
```

误差应该控制在 1% 以下。

如果误差过大，可以通过调整底盘的轮子直径大小、两个动力轮的轮间距和动力系数三个值。

这三个值在 `~/dashgo_ws/src/dashgo/dashgo_driver/config/my_dashgo_params.yaml` 中。

```
=== Robot drivetrain parameters
wheel_diameter: 0.1260 # 动力轮轮子直径
wheel_track: 0.3500 # 两个动力轮的轮间距
gear_reduction: 1.0 # 动力系数
```

校准策略：

- 优先校准走 1 米直线，这个误差达到要求后再校准转动角度。

原因：走 1 米直线只和轮子直径有关，转动角度既和轮子直径有关，还和轮间距有关。

- 校准走 1 米直线：实际运行超过 1 米时，调大轮子直径；实际运行不足 1 米时，调小轮子直径。

- 校准转动 360 度：实际转动超过 360 度时，调小轮子间距；实际转动不足 360 度时，调大轮子间距。

- 如果轮子直径和轮间距已明显高于轮子实际的直径和间距，就需要通过调整动力系数是运行达到精准。

另外，如果觉得测量 1 米没说服力，可以通过修改
~/dashgo_ws/src/dashgo/dashgo_tools/scripts/check_linear.py 中的

```
self.test_distance = rospy.get_param('~test_distance', 1.0) # meters
```

把测量距离有 1.0 米修改为 3.0 米、5.0 米和 10.0 等等。

同理，也可以修改测量角度，修改
~/dashgo_ws/src/dashgo/dashgo_tools/scripts/check_angular.py 中的

```
self.test_angle = radians(rospy.get_param('~test_angle', 360.0))
```

需要注意的是，修改的角度不能超过 360 度。

当然，还可以修改小车运行的线速度和角速度的大小。

3.4.2 带有陀螺仪

打开一个终端，启动底盘驱动(带陀螺仪)

```
$ ssh eaibot@192.168.31.200
$ roslaunch dashgo_driver demo_imu.launch
```

然后在另一个终端运行测试脚本。

- 测试一：前进 1 米

```
$ ssh eaibot@192.168.31.200
$ rosrundashgo_tools check_linear_imu.py
```

- 测试二：原地转动 360 度

```
$ ssh eaibot@192.168.31.200
$ rosrundashgo_tools check_angular_imu.py
```

校准策略：

- 校准走 1 米直线的具体操作跟不带陀螺仪的一样。
- 校准转动 360 度：

在另一个终端运行陀螺仪输出脚本

```
$ ssh eaibot@192.168.31.200
$ rostopic echo /imu_angle。
```

查看陀螺仪输出的偏航角度

角度范围在-180 ~ 180 ， 小车刚启动朝向位置为 0 度，逆时针旋转，角度增大。

真实情况下旋转 180 度、360 度， 观察 imu_angle 的变化与真实角度的变化是否符合，如果不符合， 调整 yaw 角度校对系数。

陀螺仪偏航角 yaw 角度校对系数，陀螺仪的配置参数保存在
~/dashgo_ws/src/dashgo/pathgo_imu/params/imu_params.yaml 中

```
angle_offset: 1.02
```

注意： 校对系数过大时，陀螺仪水平旋转半圈数值的绝对值就大于 180；过小时，水平旋转半圈数值绝对值就小于 180。在校准 陀螺仪角度时，会有 8 度左右的偏差，这是脚本引起，后续会修正。

3.4.3 其他数据输出

在另一个终端运行编码器的输出脚本，查看编码器的输出值

左编码器值：

```
$ ssh eaibot@192.168.31.200
$ rostopic echo /Lencoder
```

右编码器值：

```
$ ssh eaibot@192.168.31.200
$ rostopic echo /Rencoder
```

在终端运行速度输出脚本，查看小车的速度

```
$ ssh eaibot@192.168.31.200
$ rostopic echo /smoother_cmd_vel
```

3.5 E1 与激光雷达 F4 坐标校正

导航模块的默认固定 IP 是 192.168.31.200，默认用户名为 eaibot，密码为 eaibot。

运行环境：E1 + F4 + PS1000C(导航模块)

只有激光雷达与 E1 的坐标一致才能正常建地图

注意：首先确保已经按照前面的操作把导航模块的工程 scp 拷贝到 ubuntu 本地，并编译好，设置好相应环境变量，因为下面操作需要用到该工程的 rviz 功能包

E1 的前方是有三个超声波模块的，中间的超声波模块就是 E1 的正前方

E1 的后方是单独一个超声波模块的，也是 E1 的正后方

注意： 雷达和导航模块的摆放必须正确，如下图所示，雷达的 0 刻度必需朝底盘正前方（如果是 E1 地盘，有三个超声波一方为正前方，电源一方为后方），并且雷达要固定，在行走过程中不能挪动，否则影响建图效果， 导航模块（带有陀螺仪）必须正面摆放（有两个洞一面为反面，另一面为正面，如下图所示）。

如下图所示，在底盘正前面放好一个大纸箱（或其他障碍物），如果雷达是 360 扫描的（如 E1），也需要在正后面放一个大纸箱来进行观察，校准，一般校准前面，后面也会准了



3.5.1 设置导航模块与 PC 的配置文件

打开一个终端，远程进入导航模块，模块修改/etc/hosts 文件，让模块知道 PC 端的 IP 与主机名。

```
$ ssh eaibot@192.168.31.200
$ sudo vim /etc/hosts
```

[illegible]

其中 192.168.31.245 是电脑 Ubuntu 的 IP 地址（用 ifconfig 查看），eaibot 是电脑的主机名（用 hostname 查看）。

打开另一个终端，在电脑的/etc/hosts 文件中添加导航模块的 ip 和用户名：

192.168.31.200 DashgoE1。

```
$ sudo vim /etc/hosts
```

[illegible]

注意：若没有设置好电脑和导航模块的/etc/hosts 文件，两端通讯不正常，将会导致在建图时在 rviz 上无法显示地图，或者建完图后，在导航时无法设置起点和终点。

3.5.2 修改雷达与底盘的 tf 坐标转换

修改完配置文件后，打开一个终端，远程进入导航模块运行扫地图主程序

```
$ ssh eaibot@192.168.31.200
$ roslaunch dashgo_nav gmapping_demo.launch
```

运行成功，如下面显示：

```
$ roslaunch dashgo_nav gmapping_demo.launch
... logging to
/home/pi/.ros/log/a05f59f0-a569-11e6-9a1a-b827eb8dfc88/roslaunch-dashgo-E1-web-10624.1
og
Checking log directory for disk usage. This may take awhile.
```

```

Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://dashgo-E1-web:38957/

SUMMARY
=====

CLEAR PARAMETERS
* /move_base/

PARAMETERS
* /arduino/Kd: 20
.....
[ INFO] [1478578384.240421241]: Using plugin "static_layer"
[ INFO] [1478578384.442934859]: Requesting the map...
[INFO] [WallTime: 1478578384.575114] Attempting to connect to mongodb @
localhost:27017
[ INFO] [1478578384.664973875]: Resizing costmap to 544 X 544 at 0.050000 m/pix
[ INFO] [1478578384.764004366]: Received a 544 X 544 map at 0.050000 m/pix
[ INFO] [1478578384.790346896]: Using plugin "obstacle_layer"
[ INFO] [1478578384.804262619]: Subscribed to Topics: scan
[ INFO] [1478578384.949661094]: Using plugin "inflation_layer"
[ INFO] [1478578385.456495715]: Loading from pre-hydro parameter style
[ INFO] [1478578385.578284076]: Using plugin "obstacle_layer"
[ INFO] [1478578385.737670722]: Subscribed to Topics: scan
[ INFO] [1478578385.875052630]: Using plugin "inflation_layer"
[INFO] [WallTime: 1478578386.580523] Attempting to connect to mongodb @
localhost:27017
[ INFO] [1478578386.677315640]: Created local_planner
teb_local_planner/TebLocalPlannerROS
[ WARN] [1478578387.057302959]: TebLocalPlannerROS() Param Warning:
max_vel_x_backwards <= penalty_epsilon. The resulting bound is negative. Undefined
behavior... Change at least one of them!
[ WARN] [1478578387.087761403]: TebLocalPlannerROS() Param Warning:
'alternative_time_cost' is deprecated. It has been replaced by 'selection_alternative_time_cost'.
[ INFO] [1478578387.114672479]: No robot footprint model specified for trajectory
optimization. Using point-shaped model.
[ INFO] [1478578387.118526568]: Parallel planning in distinctive topologies disabled.
[ INFO] [1478578387.119119749]: No costmap conversion plugin specified. All occupied
costmap cells are treaten as point obstacles.
[ WARN] [1478578387.597289377]: TebLocalPlannerROS() Param Warning:
max_vel_x_backwards <= penalty_epsilon. The resulting bound is negative. Undefined
behavior... Change at least one of them!

```

```
[INFO] [WallTime: 1478578388.584415] Attempting to connect to mongodb @
localhost:27017
```

```
[ INFO] [1478578389.330330106]: Recovery behavior will clear layer obstacles
```

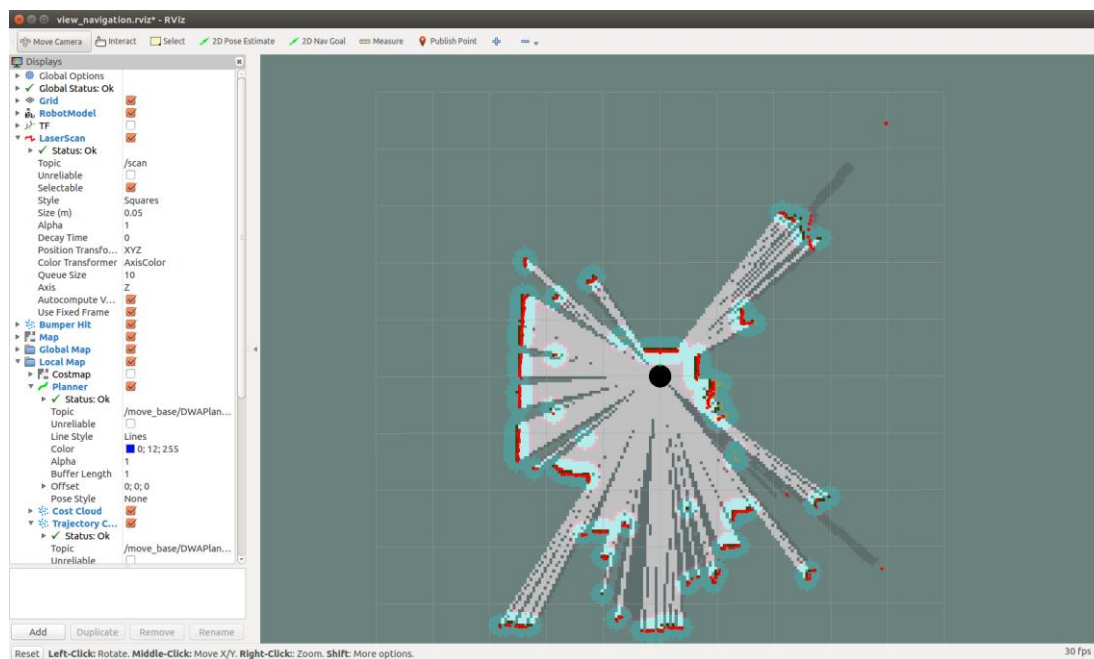
```
[ INFO] [1478578389.543029629]: Recovery behavior will clear layer obstacles
```

```
[ INFO] [1478578389.781773815]: odom received!
```

在电脑上打开终端，运行轻量级的 `rviz`，观察雷达是否校正好了

```
$ export ROS_MASTER_URI=http://192.168.31.200:11311
```

```
$ roslaunch dashgo_rviz view_navigation.launch
```



F4 正常情况下，在 `rviz` 上观察到的情况如上图所示，红色的激光线与底盘基本平行，与障碍物基本重合，如果观察到激光线与底盘不平行，红色激光线相对底盘是倾斜的，那就需要调整雷达参数。

如果雷达驱动为 2.3.0 及以上版本，则仅需要修改 `flashgo/launch` 目录下的 `demo.launch` 和 `flash_lidar.launch` 中 `tf` 坐标转换的参数，具体修改如下：

```
roscd flashgo/launch
vim flash_lidar.launch
```

```

launch>
<node name="flashgo_node" pkg="flashgo" type="flashgo_node" output="screen">
  <param name="port" type="string" value="/dev/flashlidar"/>
  <param name="baudrate" type="int" value="115200"/>
  <param name="frame_id" type="string" value="laser_frame"/>
  <param name="angle_fixed" type="bool" value="true"/>
  <param name="angle_min" type="double" value="-180" />
  <param name="angle_max" type="double" value="180" />
</node>

<node pkg="tf" type="static_transform_publisher" name="laser_frame_to_base_footprint" args="0.0 0.0
0.2 0.06 0.0 0.0 /base_footprint /laser_frame 40" />
</launch>
~
~

```

由于雷达 2.3.0 驱动遵循右手定则，角度变化逆时针增大，雷达底座 0 刻度为 0 度，由 0~180 度，-180~0 其中 180 和 -180 度重合。如上图所示雷达与底座的 tf 转换关系如下：

args="0.0 0.0 0.2 0.06 0.0 0.0 /base_footprint /laser_frame 40

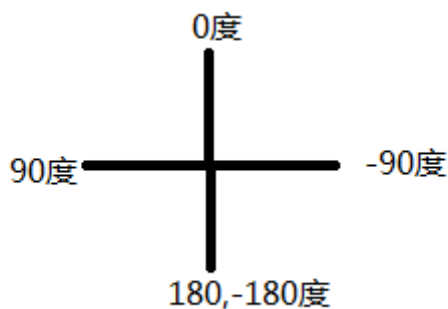
由于雷达和底盘都遵循右手定则，并且在摆放时，雷达 0 度与底盘 0 度（正前方）重合，因此它们的坐标转换关系，仅需要 x, y, z 平移，不需要翻转。

前面 3 个参数表示底盘中心离雷达中心的 x, y, z 距离（底盘正前方为 x 轴方向，左右为 y 轴，上下为 z 轴），这三个参数可直接测量出来。

由于不需要翻转，后面两个参数为 0。第四个参数 0.06 是校准 gmapping 地图上看到激光线的倾斜度，基本为 0。

3.5.3 修改雷达扫描角度

如图所示为（2.3.0 版本及以上）雷达角度图：



在 flash_lidar.launch 中的雷达角度扫描参数


```
<param name="angle_min"      type="double" value="-180" />
<param name="angle_max"      type="double" value="180" />
```

这表示雷达扫描的角度为-180~180 度共 360 度。如果扫描角度为雷达前方的 180 度，则需要改成

```
<param name="angle_min"      type="double" value="-90" />
<param name="angle_max"      type="double" value="90" />
```

注意：1. 雷达扫描角度需要大于 180 度，否则会影响建图效果和效率，影响导航避障效果等。

如果雷达驱动版本为 2.2.1 及之前版本的驱动，建议用最新的驱动替换，在 Linux 下可以 `git clone https://github.com/EAIBOT/flashgo.git` 直接从 github 下载最新雷达驱动。然后把旧的驱动删除，把新的驱动拷贝到原来位置，最后重新编译生效。

旧的雷达驱动（2.2.1 版本及之前），需要修改导航模块的 `dashgo_nav/launch/gmapping_demo.launch`, `navigation.launch` 等 launch 文件中修改雷达的 `tf` 转换参数，用到那个 launch 文件，就需要改那个 launch。例如修改 `gmapping_demo.launch`

```
args="0.18 0.0 0.2 -3.06 3.14 0.0 /base_footprint /laser_frame 40"
```

前面 3 个参数表示底盘中心离雷达中心的 x, y, z 距离，第四个参数表示前后翻转，一般都只需修改它来使激光线平行，第五个参数表示上下翻转，固定 3.14，最后一个参数为 0。

修改雷达的扫描角度在 `flashgo/launch/flash_lidar.launch` 和 `demo.launch`（用到那个改那个），例如这个表示雷达扫描角度为 360 度，

```
<param name="ignore_array"    type="string" value="" />
```

如果需要改成扫描雷达前方 180 度，则为

```
<param name="ignore_array"    type="string" value="90,270" />
```

这表示雷达扫描角度为 0~90 度，270~360 度共 180 度扫描范围。

3.6 vim 的基本使用

终端执行以下命令，根据提示安装 vim

```
$ sudo apt-get install vim
```

以上面修改 gmapping_demo.launch 文件，以将第 4 个参数改成 3.1415926 为例，介绍 vim 的基本使用

- 打开 gmapping_demo.launch 文件（必须要先切换到所要编辑的目录下）

```
$ vim gmapping_demo.launch
```

- 进入文件内容显示界面，此时的文件状态只是显示，还不是编辑状态
- 光标的移动是通过键盘上的上、下、左、右方向键来控制的。
- 通过方向键将光标移动到要修改的 0.0 处
- 按键盘上的 I 键，让文本进入编辑状态，编辑状态下，命令窗口左下角显示

Insert 或 插入 字样

- 通过键盘上的 Delete 或 Backspace 来删除 0.0 ，填写 3.1415926
- 修改好后，按键盘上的 Esc 退出编辑状态
- 按组合键 Shift + ; ，窗口左下角显示 : ，再按 W + Q ，窗口左下角显示 :wq ，

再回车便保存好修改后的文件

3.7 不带陀螺仪建图导航

导航模块的默认固定 IP 是 192.168.31.200，默认用户名为 eaibot ， 密码为 eaibot 。

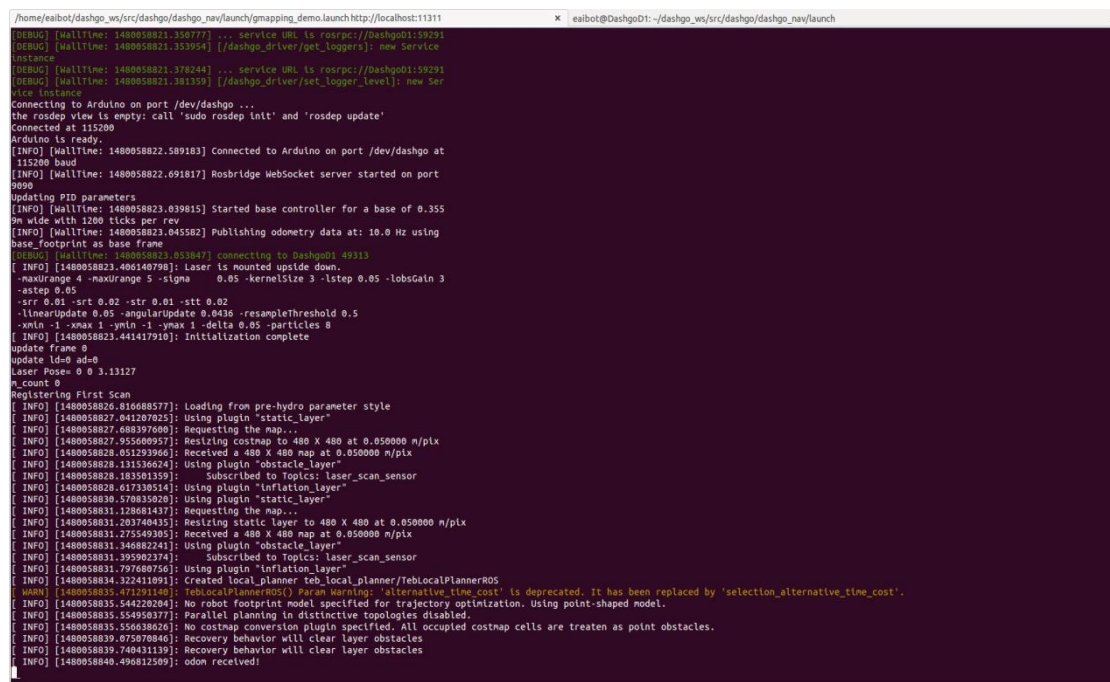
E1 与 F4 的坐标系已经校准正确的情况下

注意：在建图，导航前，必须校准好走 1m 直线，360 度旋转 和 雷达坐标，否则建出来的图不正确，混乱。

3.7.1 扫描建图

打开一个终端，ssh 登录导航模块并启动建图 launch

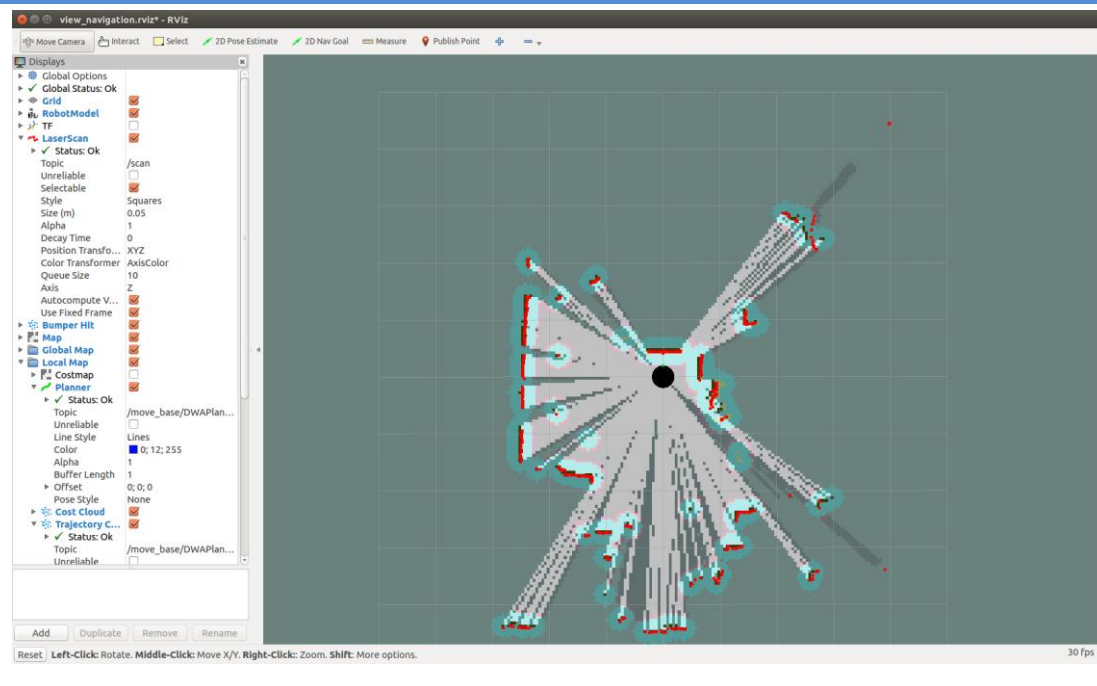
```
$ ssh eaibot@192.168.31.200
$ roslaunch dashgo_nav gmapping_demo.launch
```



```
/home/eaibot/dashgo_ws/src/dashgo/dashgo_nav/launch/gmapping_demo.launch http://localhost:11311
[DEBUG] [WallTime: 1480058821.350777] ... service URL is rospic://dashgo01:59291
[DEBUG] [WallTime: 1480058821.353954] [/dashgo_driver/get_logger]: new Service
Instance
[DEBUG] [WallTime: 1480058821.378244] ... service URL is rospic://dashgo01:59291
[DEBUG] [WallTime: 1480058821.381350] [/dashgo_driver/set_logger_level]: new Ser
vice Instance
Connecting to Arduino on port /dev/dashgo ...
the roscop view is empty: call 'sudo roscop init' and 'roscop update'
Connected at 115200
Arduino is ready.
[INFO] [WallTime: 1480058822.589183] Connected to Arduino on port /dev/dashgo at
115200 baud
[INFO] [WallTime: 1480058822.691817] Rosbridge WebSocket server started on port
9890
Updating PID parameters
[INFO] [WallTime: 1480058823.039815] Started base controller for a base of 0.355
m wide with 1200 ticks per rev
[INFO] [WallTime: 1480058823.045582] Publishing odometry data at: 10.0 Hz using
base_footprint as base frame
[DEBUG] [WallTime: 1480058823.053847] connecting to Dashgo01 49313
[ INFO] [1480058823.406140798]: Laser is mounted upside down.
-maxiRange 4 -maxiRange 5 -signa 0.05 -kernelSize 3 -lstep 0.05 -lobsgain 3
-lstep 0.05
-srr 0.01 -srt 0.02 -str 0.01 -stt 0.02
-linearUpdate 0.05 -angularUpdate 0.0436 -resampleThreshold 0.5
-xmin -1 -xmax 1 -ymin -1 -ymax 1 -delta 0.05 -particles 8
[ INFO] [1480058823.441417910]: Initialization complete
update frame 0
update ld=0 ad=0
Laser Poses: 0 0 3.13127
n_count 0
Registering First Scan
[ INFO] [1480058826.816688577]: Loading from pre-hydro parameter style
[ INFO] [1480058827.041207023]: Using plugin "static_layer"
[ INFO] [1480058827.688397090]: Requesting the map...
[ INFO] [1480058827.956009577]: Resizing costmap to 480 X 480 at 0.050000 m/pix
[ INFO] [1480058828.051293960]: Received a 480 X 480 map at 0.050000 m/pix
[ INFO] [1480058828.131536624]: Using plugin "obstacle_layer"
[ INFO] [1480058828.183501359]: Subscribed to Topics: laser_scan_sensor
[ INFO] [1480058828.617330514]: Using plugin "inflation_layer"
[ INFO] [1480058830.570835020]: Using plugin "static_layer"
[ INFO] [1480058831.128681437]: Requesting the map...
[ INFO] [1480058831.203740435]: Resizing static layer to 480 X 480 at 0.050000 m/pix
[ INFO] [1480058831.275493051]: Received a 480 X 480 map at 0.050000 m/pix
[ INFO] [1480058831.346882241]: Using plugin "obstacle_layer"
[ INFO] [1480058831.395902374]: Subscribed to Topics: laser_scan_sensor
[ INFO] [1480058831.797680750]: Using plugin "inflation_layer"
[ INFO] [1480058834.322411091]: Created local planner teb_local_planner/TeLocalPlannerROS
[ WARN] [1480058835.471391140]: tebLocalPlannerROS() Param Warning: 'alternative_time_cost' is deprecated. It has been replaced by 'selection_alternative_time_cost'.
[ INFO] [1480058835.544220204]: No robot footprint model specified for trajectory optimization. Using point-shaped model.
[ INFO] [1480058835.554950377]: Parallel planning in distinctive topologies disabled.
[ INFO] [1480058835.556838020]: No costmap conversion plugin specified. All occupied costmap cells are treated as point obstacles.
[ INFO] [1480058839.075078046]: Recovery behavior will clear layer obstacles
[ INFO] [1480058839.740431139]: Recovery behavior will clear layer obstacles
[ INFO] [1480058840.490812509]: odom received!
```

在电脑上打开另一个终端，设置节点管理器，并启动 rviz 图形界面观看建图效果

```
$ export ROS_MASTER_URI=http://192.168.31.200:11311
$ roslaunch dashgo_rviz view_navigation.launch
```



注意：1. 若在 **rviz** 上无法显示地图（显示一片黑的），是由于电脑终端没有运行

`export ROS_MASTER_URI=http://192.168.31.200:11311` 命令，或者导航模块的 `/etc/hosts` 文件没修改正确。

2. 若运行 **rviz** 报红色错误，首先请确认是在电脑的终端上运行 **rviz**（导航模块中没有 **rviz** 工具，若是在导航模块中运行 **rviz**，必然报错，这是操作不当引起）。也有可能是电脑内存不足导致 **rviz** 无法运行起来，请把一些不相关的软件关闭，重新运行。

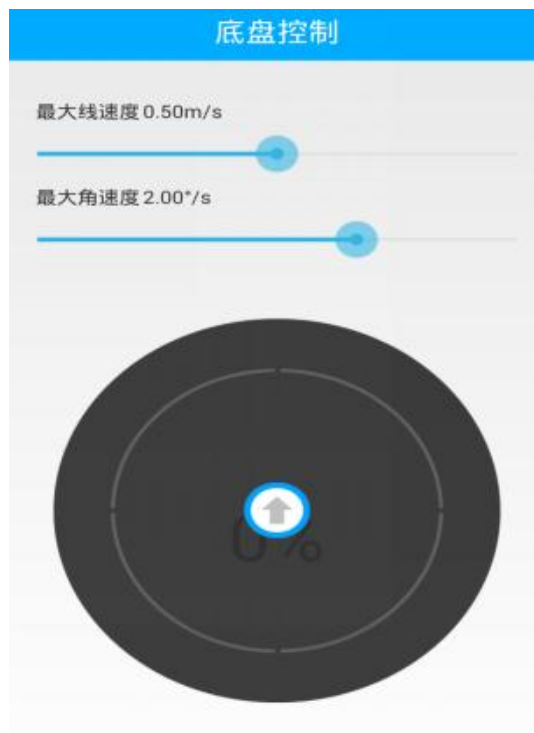
通过 EAI 团队开发的手机 APP 控制小车移动建图，目前仅支持 Android。

在 APP 启动界面，选择“WIFI”便进入到 WiFi 连接界面，如下图所示：

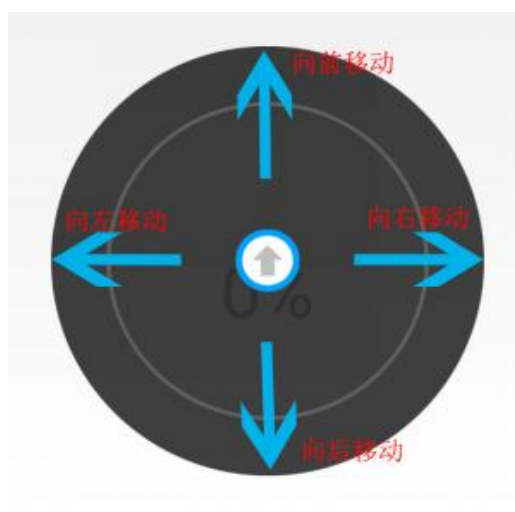


需要输入的 Master IP 是导航模块的 IP 地址，即 192.168.31.200。

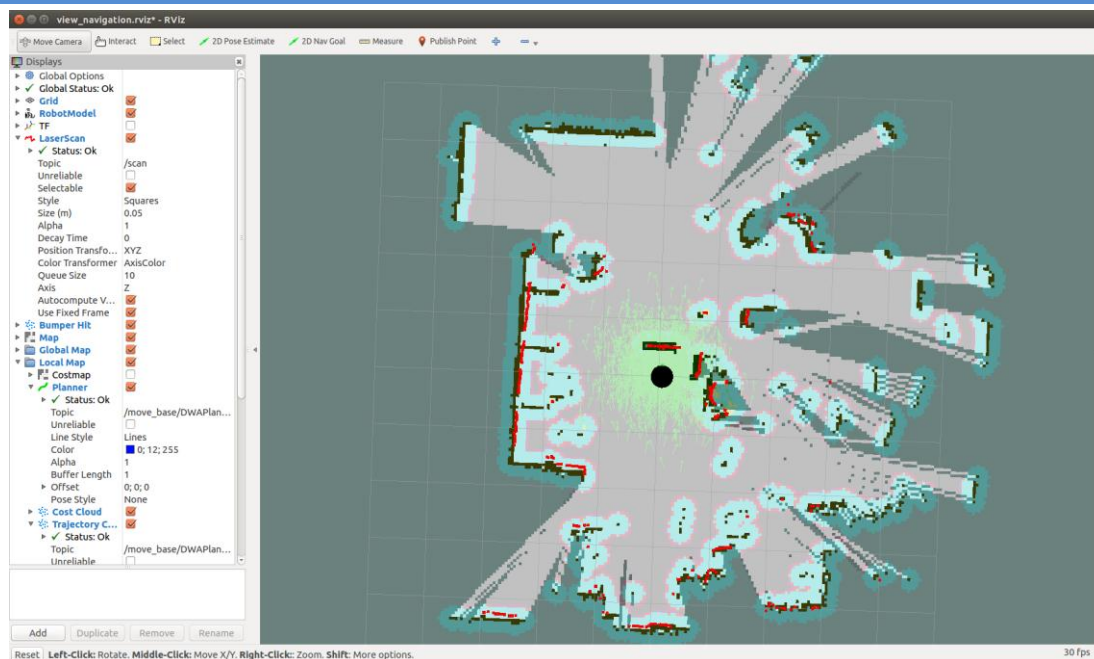
然后点击“连接”，连接成功后，界面如下：



方向的操控，如下图所示：



最后建好的地图如下所示：



3.7.2 保存地图

打开一个终端，ssh 登录导航模块，切换到地图保存地址并使用 `map_server` 对已经建好的地图进行保存

```
$ ssh eaibot@192.168.31.200
$ roscd dashgo_nav/maps
$ rosrn map_server map_saver -f retest_map
```

```
eaibot@Dashgo01:~$ roscd dashgo_nav/maps/
eaibot@Dashgo01:~/dashgo_ws/src/dashgo/dashgo_nav/maps$ rosrn map_server map_saver -f retest_map
[ INFO] [1480065580.082998835]: Waiting for the map
[ INFO] [1480065580.296717531]: Received a 544 X 576 map @ 0.050 n/plx
[ INFO] [1480065580.297476035]: Writing map occupancy data to retest_map.pgm
[ INFO] [1480065580.462529662]: Writing map occupancy data to retest_map.yaml
[ INFO] [1480065580.465395019]: Done
eaibot@Dashgo01:~/dashgo_ws/src/dashgo/dashgo_nav/maps$ ls
c_zoulang01.pgm  c_zoulang01.yaml  my_map.pgm  my_map.yaml  retest_map.pgm  retest_map.yaml  zoulang01.pgm  zoulang01.yaml  zoulang.pgm  zoulang.yaml
eaibot@Dashgo01:~/dashgo_ws/src/dashgo/dashgo_nav/maps$
```

注意：`retest_map` 为自定义地图名，若与已保存地图名称同名，便覆盖原来的地图信息。

导航时，默认使用名为 `retest_map`，如果你想在导航时引用其他地图文件，需要切换到 `launch` 文件目录，使用 `vim` 修改 `navigation_demo.launch` 文件，将 `retest_map.yaml` 改成你要引入的地图名称，操作如下：

```
$roscd dashgo_nav/launch
$vim navigation_demo.launch
```


3.7.3 自主导航

在已启动 gmapping_demo.launch 的终端，Ctrl + C 关闭 gmapping_demo.launch 程序

然后，启动导航 launch

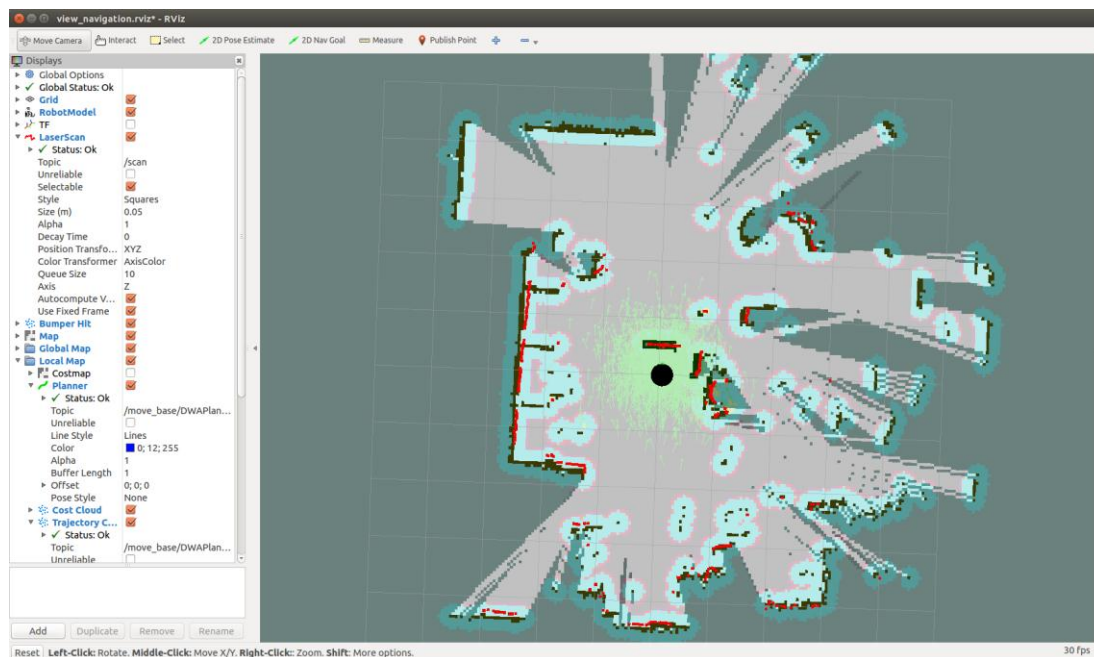
```
$ roslaunch dashgo_nav navigation_demo.launch
```

```
arning: The publisher should be created with an explicit keyword argument 'queue_size'. Please see http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscribers for more information.
  self.rEncoderPub = rospy.Publisher('Rencoder', Int16)
[DEBUG] [WallTime: 1478752655.609766] connecting to dashgo-d1 34481
/home/pi/dashgo_ws/src/dashgo/dashgo_bringup/nodes/dashgo_driver.py:433: SyntaxWarning: The publisher should be created with an explicit keyword argument 'queue_size'. Please see http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscribers for more information.
  self.lPidoutPub = rospy.Publisher('LPidout', Int16)
/home/pi/dashgo_ws/src/dashgo/dashgo_bringup/nodes/dashgo_driver.py:434: SyntaxWarning: The publisher should be created with an explicit keyword argument 'queue_size'. Please see http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscribers for more information.
  self.rPidoutPub = rospy.Publisher('RPidout', Int16)
/home/pi/dashgo_ws/src/dashgo/dashgo_bringup/nodes/dashgo_driver.py:435: SyntaxWarning: The publisher should be created with an explicit keyword argument 'queue_size'. Please see http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscribers for more information.
  self.lVelPub = rospy.Publisher('Lvel', Int16)
/home/pi/dashgo_ws/src/dashgo/dashgo_bringup/nodes/dashgo_driver.py:436: SyntaxWarning: The publisher should be created with an explicit keyword argument 'queue_size'. Please see http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscribers for more information.
  self.rVelPub = rospy.Publisher('Rvel', Int16)
[ INFO] [1478752655.988503647]: Laser is mounted upside down.
-maxUrange 4 -maxUrange 5 -sigma 0.05 -kernelSize 3 -lstep 0.05 -lobsGain 3
-astep 0.05
-srr 0.01 -srt 0.02 -str 0.01 -stt 0.02
-linearUpdate 0.05 -angularUpdate 0.0436 -resampleThreshold 0.5
-xmin -1 -xmax 1 -ymin -1 -ymax 1 -delta 0.05 -particles 8
[ INFO] [1478752655.998159908]: Initialization complete
update frame 0
update ld=0 ad=0
Laser Pose= 0 0 -0.00872675
m_count 0
Registering First Scan
[DEBUG] [WallTime: 1478752657.459001] connecting to dashgo-d1 33088
[ INFO] [1478752657.500477123]: Loading from pre-hydro parameter style
[ INFO] [1478752657.640970320]: Using plugin "static_layer"
[ INFO] [1478752657.889303558]: Requesting the map...
[ INFO] [1478752658.110660276]: Resizing costmap to 608 X 576 at 0.050000 m/pix
[ INFO] [1478752658.209549412]: Received a 608 X 576 map at 0.050000 m/pix
[ INFO] [1478752658.236047346]: Using plugin "obstacle_layer"
[ INFO] [1478752658.250791760]: Subscribed to Topics: scan
[ INFO] [1478752658.410824609]: Using plugin "inflation_layer"
[ INFO] [1478752658.946061131]: Loading from pre-hydro parameter style
[ INFO] [1478752659.062779790]: Using plugin "obstacle_layer"
[ INFO] [1478752659.212247903]: Subscribed to Topics: scan
[ INFO] [1478752659.353407860]: Using plugin "inflation_layer"
[ INFO] [1478752659.868934626]: Created local_planner base_local_planner/TrajectoryPlannerROS
[ INFO] [1478752659.957224547]: Sim period is set to 0.33
[ INFO] [1478752661.142093575]: Recovery behavior will clear layer obstacles
[ INFO] [1478752661.380818833]: Recovery behavior will clear layer obstacles
[ INFO] [1478752661.628205159]: odom received!
```

切换到打开 rviz 的终端，Ctrl + C 关闭 rviz ， 再重新启动 rviz

```
$ roslaunch dashgo_rviz view_navigation.launch
```

rviz 打开后显示的地图如下：



注意：由于 rviz 软件默认小车的位置 在栅格图的中间，但这个位置一般不是小车的实际位置，所以必须设置好小车的起点位置（实际停放的位置），正常情况是红色的激光线必须和地图的障碍物，轮廓相重合，并且每次打开 rviz 都必须设置起点。

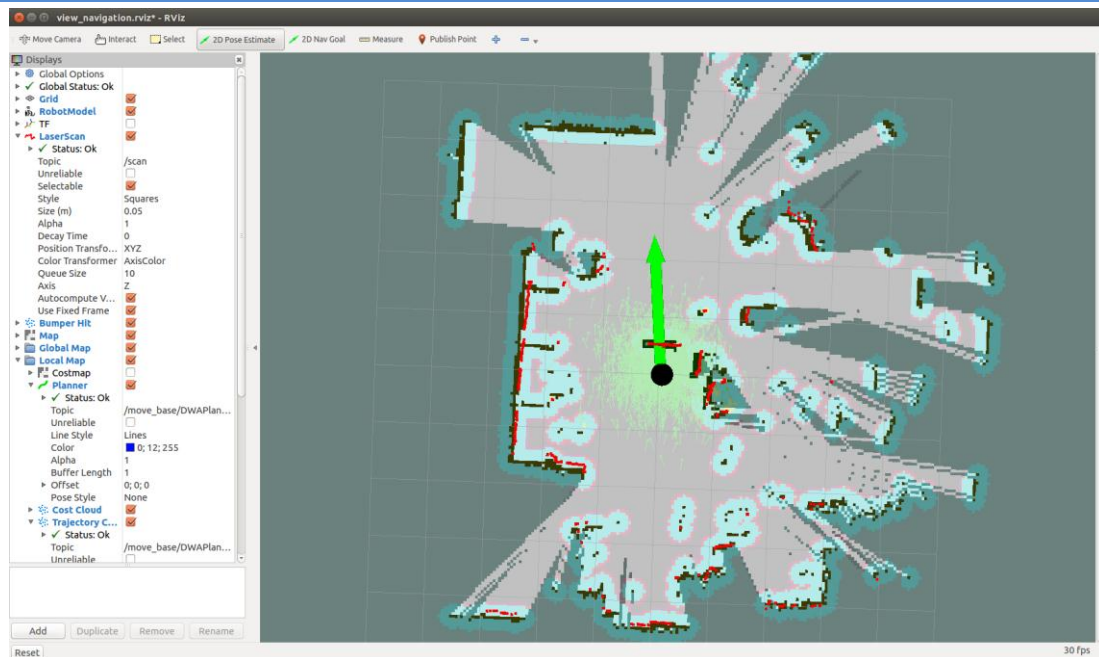
1. 设置起点

rviz 打开后显示 E1 默认所在的位置是栅格的中心点，不一定是 E1 实际所在的位置

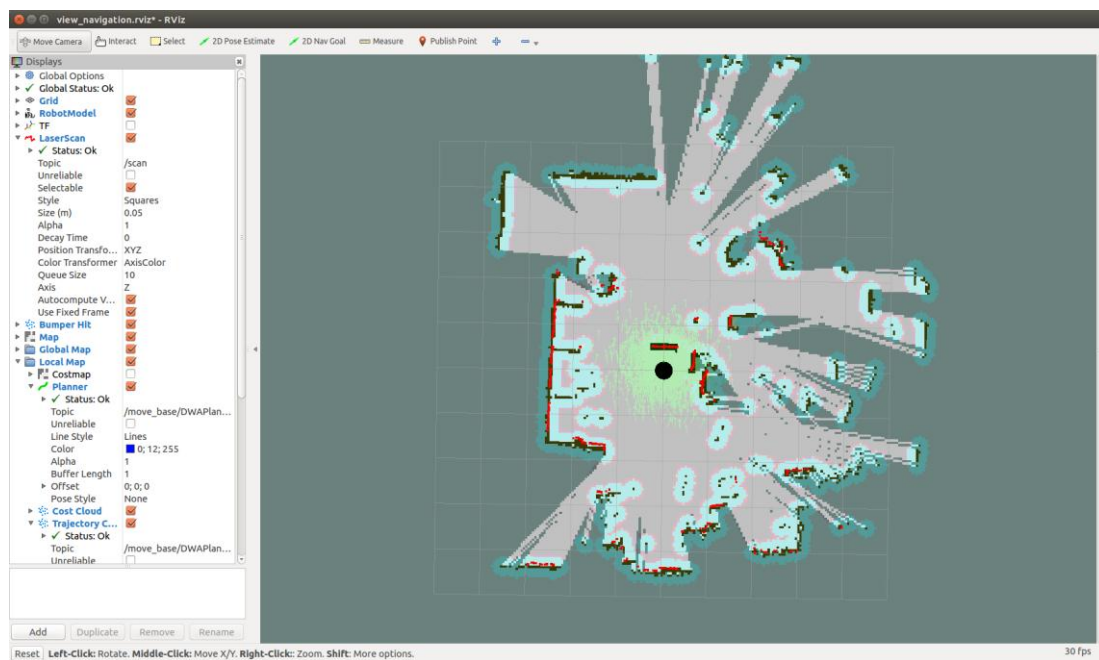
注意：每次打开 rviz 都要检查并设置起点。若无法设置起点（点击地图设置时无反应），可能是导航模块的 `/etc/hosts` 文件没修改正确。

点击 2D Pose Estimate

根据当前 E1 实际位置，在地图上选择正确的位置，并调整好 E1 的正前方方向(正前方的调整：在选择位置时，按住鼠标不放，拖动方向便可)，如下图所示



设置好起点后，如下图所示：激光线和障碍物，轮廓基本重合。

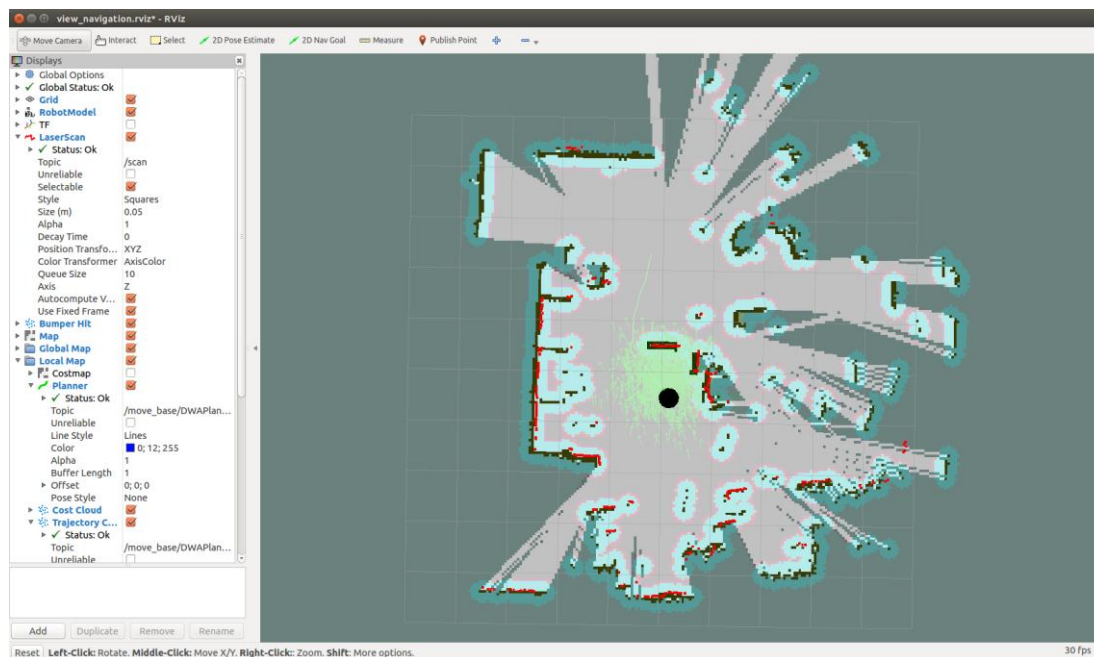


2. 设置目标点

点击 2D Nav Goal

在地图上选择要到达的目标点位置，并调整好 E1 停止时的正前方方向(正前方的调整：在选择位置时，按住鼠标不放，拖动方向便可)

设置好目标点后，导航模块便自主计算路径并控制 E1 向目标点移动



注意：每次导航时，都要先点击 2D Nav Goal 然后选择目标点，小车便自主导航移动。

3.8 带陀螺仪建图导航

导航模块的默认固定 IP 是 192.168.31.200，默认用户名为 eaibot，密码为 eaibot。

注意：在建图，导航前，必须校准好走 1m 直线，360 度旋转 和雷达坐标，否则建出来的图不正确，混乱。

3.8.1 启动扫描建图

打开一个终端，ssh 登录导航模块并启动建图 launch

```
$ ssh eaibot@192.168.31.200
$ roslaunch dashgo_nav gmapping_demo_imu.launch
```

```

/home/eaibot/dashgo_ws/src/dashgo/dashgo_nav/launch/gmapping_demo.launch http://localhost:11311
[DEBUG] [WallTime: 1480058821.350777] ... service URL is rosrpc://dashgo01:59291
[DEBUG] [WallTime: 1480058821.353954] [/dashgo_driver/get_loggers]: new Service
instance
[DEBUG] [WallTime: 1480058821.370244] ... service URL is rosrpc://dashgo01:59291
[DEBUG] [WallTime: 1480058821.381359] [/dashgo_driver/set_logger_level]: new Ser
vice instance
Connecting to Arduino on port /dev/dashgo ...
the rosdep view is empty: call 'sudo rosdep init' and 'rosdep update'
Connected at 115200
Arduino is ready.
[INFO] [WallTime: 1480058822.589183] Connected to Arduino on port /dev/dashgo at
115200 baud
[INFO] [WallTime: 1480058822.691817] Rosbridge WebSocket server started on port
9090
Updating PID parameters
[INFO] [WallTime: 1480058823.039815] Started base controller for a base of 0.355
m wide with 2200 ticks per rev
[INFO] [WallTime: 1480058823.045582] Publishing odometry data at: 10.0 Hz using
base_footprint as base frame
[DEBUG] [WallTime: 1480058823.053847] connecting to Dashgo01 49313
[INFO] [1480058823.4061407980] Laser is mounted upside down.
-maxRange 4 -maxUrange 5 -sigma 0.05 -kernelSize 3 -lstep 0.05 -lbgain 3
-astep 0.05
-srr 0.01 -srt 0.02 -str 0.01 -stt 0.02
-linearUpdate 0.05 -angularUpdate 0.0436 -resampleThreshold 0.5
-xmin -1 -xmax 1 -ymin -1 -ymax 1 -delta 0.05 -particles 8
[INFO] [1480058823.441417910] Initialization complete
update frame 0
update ld=0 ad=0
Laser Pose= 0 0 3.13127
x_count 0
Registering First Scan
[INFO] [1480058826.810688577] Loading from pre-hydro parameter style
[INFO] [1480058827.041207025] Using plugin "static_layer"
[INFO] [1480058827.680397600] Requesting the map...
[INFO] [1480058827.955600957] Resizing costmap to 480 X 480 at 0.050000 m/pix
[INFO] [1480058828.051293960] Received a 480 X 480 map at 0.050000 m/pix
[INFO] [1480058828.131360241] Using plugin "obstacle_layer"
[INFO] [1480058828.183501350] Subscribed to Topics: laser_scan_sensor
[INFO] [1480058828.617330514] Using plugin "inflation_layer"
[INFO] [1480058830.570835020] Using plugin "static_layer"
[INFO] [1480058831.120814373] Requesting the map...
[INFO] [1480058831.203740435] Resizing static layer to 480 X 480 at 0.050000 m/pix
[INFO] [1480058831.275493051] Received a 480 X 480 map at 0.050000 m/pix
[INFO] [1480058831.346882241] Using plugin "obstacle_layer"
[INFO] [1480058831.395902374] Subscribed to Topics: laser_scan_sensor
[INFO] [1480058831.797680750] Using plugin "inflation_layer"
[INFO] [1480058834.322411091] Created local_planner teb_local_planner/TebLocalPlannerROS
WARN [1480058834.471093140] TebLocalPlannerROS() Param Warning: 'alternative_tlcost' is deprecated. It has been replaced by 'selection_alternative_tlcost'.
[INFO] [1480058835.544220204] No robot footprint model specified for trajectory optimization. Using point-shaped model.
[INFO] [1480058835.554950377] Parallel planning in distinctive topologies disabled.
[INFO] [1480058835.556386204] No costmap conversion plugin specified. All occupied costmap cells are treated as point obstacles.
[INFO] [1480058839.073070804] Recovery behavior will clear layer obstacles
[INFO] [1480058839.740411139] Recovery behavior will clear layer obstacles
[INFO] [1480058840.496812509] odom received!

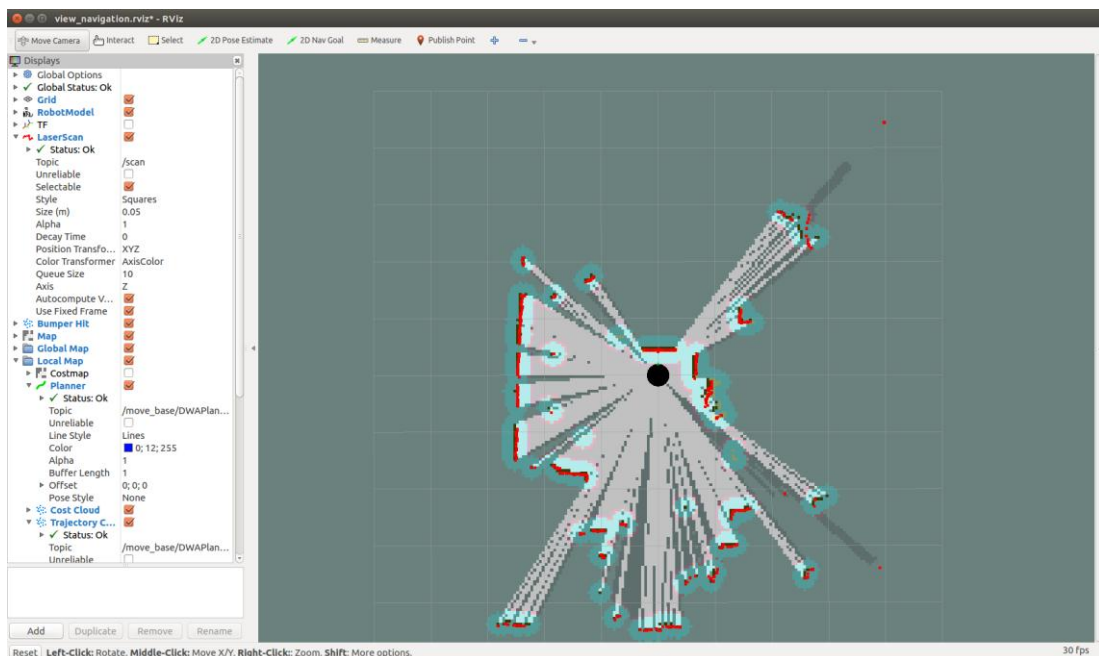
```

在电脑上打开另一个终端，设置节点管理器，并启动 rviz 图形界面观看建图

```

$ export ROS_MASTER_URI=http://192.168.31.200:11311
$ roslaunch dashgo_rviz view_navigation.launch

```



注意：1. 若没有设置好电脑和导航模块的/etc/hosts 文件，两端通讯不正常，将会导致在建图时在 rviz 上无法显示地图，或者在建完图后，在导航时无法设置起点和终点

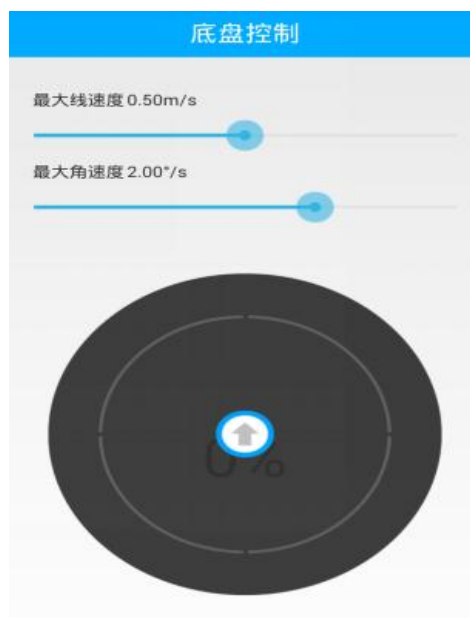
通过 EAI 团队开发的手机 APP 控制小车移动建图，目前仅支持 Android。

在 APP 启动界面，选择“WIFI”便进入到 WiFi 连接界面，如下图所示：

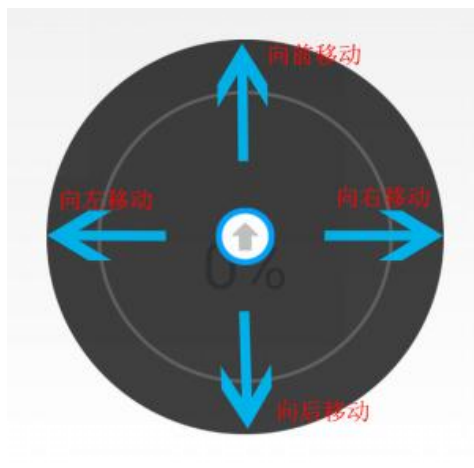


需要输入的 Master IP 是导航模块的 IP 地址，即 192.168.31.200。

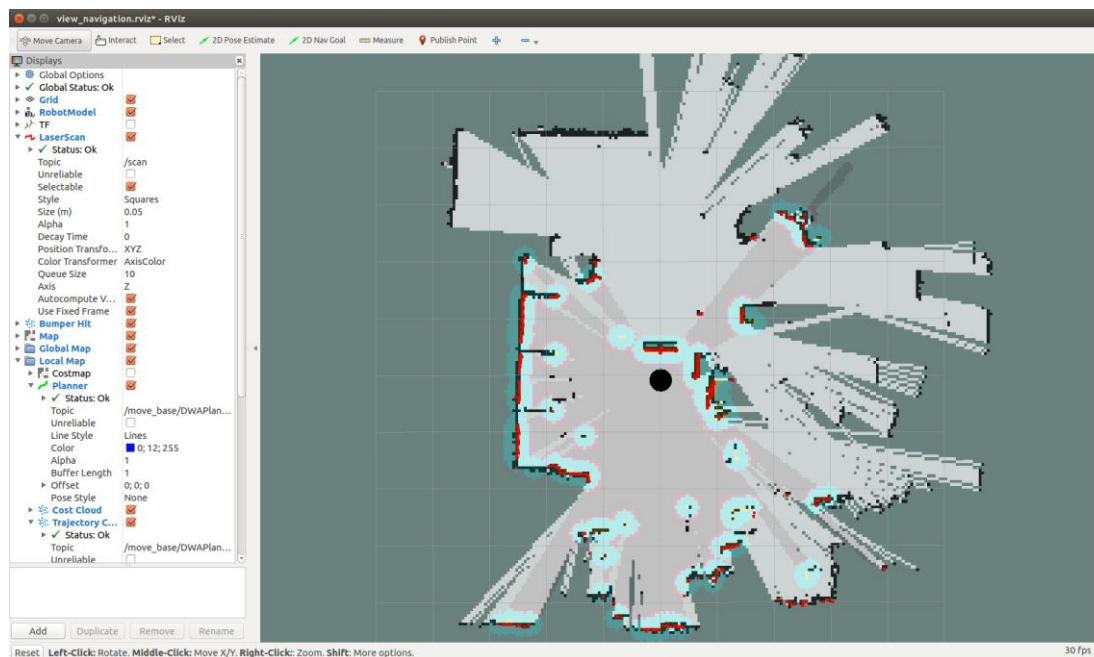
然后点击“连接”，连接成功后，界面如下：



方向的操控，如下图所示：



最后建好的地图如下所示：



3.8.2 保存地图

打开一个终端，ssh 登录导航模块，切换到地图保存地址并使用 `map_server` 对已经建好的地图进行保存

```
$ ssh eaibot@192.168.31.200
$ roscd dashgo_nav/maps
$ rosrn map_server map_saver -f retest_map
```

```
eaibot@dashgo01:~$ roscd dashgo_nav/maps/
eaibot@dashgo01:~/dashgo_ws/src/dashgo/dashgo_nav/maps$ rosrn map_server map_saver -f retest_map
[ INFO ] [1480065580.082998835]: Waiting for the map
[ INFO ] [1480065580.296717531]: Received a 544 X 576 map @ 0.050 m/pix
[ INFO ] [1480065580.297476035]: Writing map occupancy data to retest_map.pgm
[ INFO ] [1480065580.462529662]: Writing map occupancy data to retest_map.yaml
[ INFO ] [1480065580.463395019]: Done
eaibot@dashgo01:~/dashgo_ws/src/dashgo/dashgo_nav/maps$ ls
c_zoulang01.pgm c_zoulang01.yaml my_map.pgm my_map.yaml retest_map.pgm retest_map.yaml zoulang01.pgm zoulang01.yaml zoulang.pgm zoulang.yaml
eaibot@dashgo01:~/dashgo_ws/src/dashgo/dashgo_nav/maps$
```

注意：`retest_map` 为自定义地图名，若与已保存地图名称同名，便覆盖原来的地图信息。

导航时，默认使用名为 `retest_map2`，如果你想在导航时引用其他地图文件，需要切换到 `launch` 文件目录，使用 `vim` 修改 `navigation_demo_imu.launch` 文件，将 `retest_map.yaml` 改成你要引入的地图名称，操作如下：

```
$roscd dashgo_nav/launch
```

```
$vim navigation_demo_imu.launch
```

3.8.3 自主导航

在已启动 `gmapping_demo_imu.launch` 的终端，**Ctrl + C** 关闭
`gmapping_demo_imu.launch` 程序

切换到 `launch` 文件目录，使用 `vim` 修改 `navigation_demo_imu.launch` 文件，引用已经保存好的地图

然后，启动导航 launch

```
$ roslaunch dashgo_nav navigation_demo_imu.launch
```



```

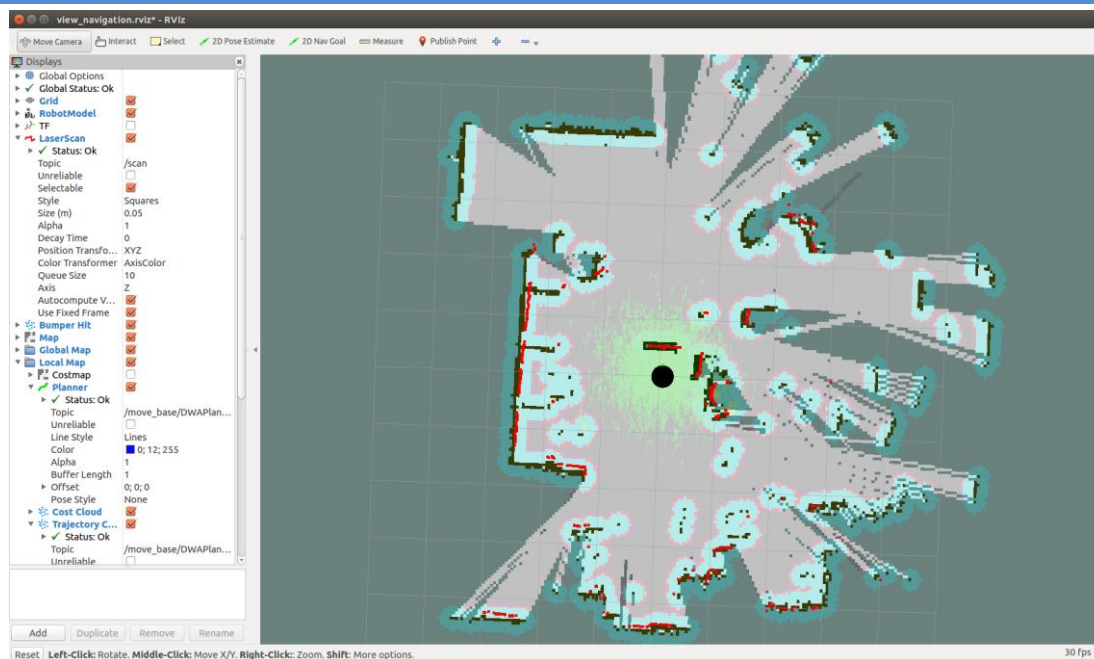
arning: The publisher should be created with an explicit keyword argument 'queue
_size'. Please see http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscri
bers for more information.
    self.rEncoderPub = rospy.Publisher('Rencoder', Int16)
[DEBUG] [WallTime: 1478752655.609766] connecting to dashgo-d1 34481
/home/pi/dashgo_ws/src/dashgo/dashgo_bringup/nodes/dashgo_driver.py:433: SyntaxW
arning: The publisher should be created with an explicit keyword argument 'queue
_size'. Please see http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscri
bers for more information.
    self.lPidoutPub = rospy.Publisher('Lpidout', Int16)
/home/pi/dashgo_ws/src/dashgo/dashgo_bringup/nodes/dashgo_driver.py:434: SyntaxW
arning: The publisher should be created with an explicit keyword argument 'queue
_size'. Please see http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscri
bers for more information.
    self.rPidoutPub = rospy.Publisher('Rpidout', Int16)
/home/pi/dashgo_ws/src/dashgo/dashgo_bringup/nodes/dashgo_driver.py:435: SyntaxW
arning: The publisher should be created with an explicit keyword argument 'queue
_size'. Please see http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscri
bers for more information.
    self.lVelPub = rospy.Publisher('Lvel', Int16)
/home/pi/dashgo_ws/src/dashgo/dashgo_bringup/nodes/dashgo_driver.py:436: SyntaxW
arning: The publisher should be created with an explicit keyword argument 'queue
_size'. Please see http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscri
bers for more information.
    self.rVelPub = rospy.Publisher('Rvel', Int16)
[ INFO] [1478752655.988503647]: Laser is mounted upside down.
-maxUrange 4 -maxUrange 5 -sigma 0.05 -kernelSize 3 -lstep 0.05 -lobsGain 3
-astep 0.05
-srr 0.01 -srt 0.02 -str 0.01 -stt 0.02
-linearUpdate 0.05 -angularUpdate 0.0436 -resampleThreshold 0.5
-xmin -1 -xmax 1 -ymin -1 -ymax 1 -delta 0.05 -particles 8
[ INFO] [1478752655.998159908]: Initialization complete
update frame 0
update ld=0 ad=0
Laser Pose= 0 0 -0.00872675
m_count 0
Registering First Scan
[DEBUG] [WallTime: 1478752657.459001] connecting to dashgo-d1 33088
[ INFO] [1478752657.500477123]: Loading from pre-hydro parameter style
[ INFO] [1478752657.640970320]: Using plugin "static_layer"
[ INFO] [1478752657.889303558]: Requesting the map...
[ INFO] [1478752658.110660276]: Resizing costmap to 608 X 576 at 0.050000 m/pix
[ INFO] [1478752658.209549412]: Received a 608 X 576 map at 0.050000 m/pix
[ INFO] [1478752658.236047346]: Using plugin "obstacle_layer"
[ INFO] [1478752658.250791760]: Subscribed to Topics: scan
[ INFO] [1478752658.410824609]: Using plugin "inflation_layer"
[ INFO] [1478752658.946061131]: Loading from pre-hydro parameter style
[ INFO] [1478752659.062779790]: Using plugin "obstacle_layer"
[ INFO] [1478752659.212247903]: Subscribed to Topics: scan
[ INFO] [1478752659.353407860]: Using plugin "inflation_layer"
[ INFO] [1478752659.868934626]: Created local_planner base_local_planner/Traject
oryPlannerROS
[ INFO] [1478752659.957224547]: Sim period is set to 0.33
[ INFO] [1478752661.142093575]: Recovery behavior will clear layer obstacles
[ INFO] [1478752661.380818833]: Recovery behavior will clear layer obstacles
[ INFO] [1478752661.628205159]: odom received!

```

切换到打开 `rviz` 的终端，`Ctrl + C` 关闭 `rviz`，再重新启动 `rviz`

```
$ roslaunch dashgo_rviz view_navigation.launch
```

`rviz` 打开后显示的地图如下：



注意：由于 **rviz** 软件默认小车的位置 在栅格图的中间，但这个位置一般不是小车的实际位置，所以必须设置好小车的起点位置（实际停放的位置），正常情况是红色的激光线必须和地图的障碍物，轮廓相重合，并且每次打开 **rviz** 都必须设置起点。

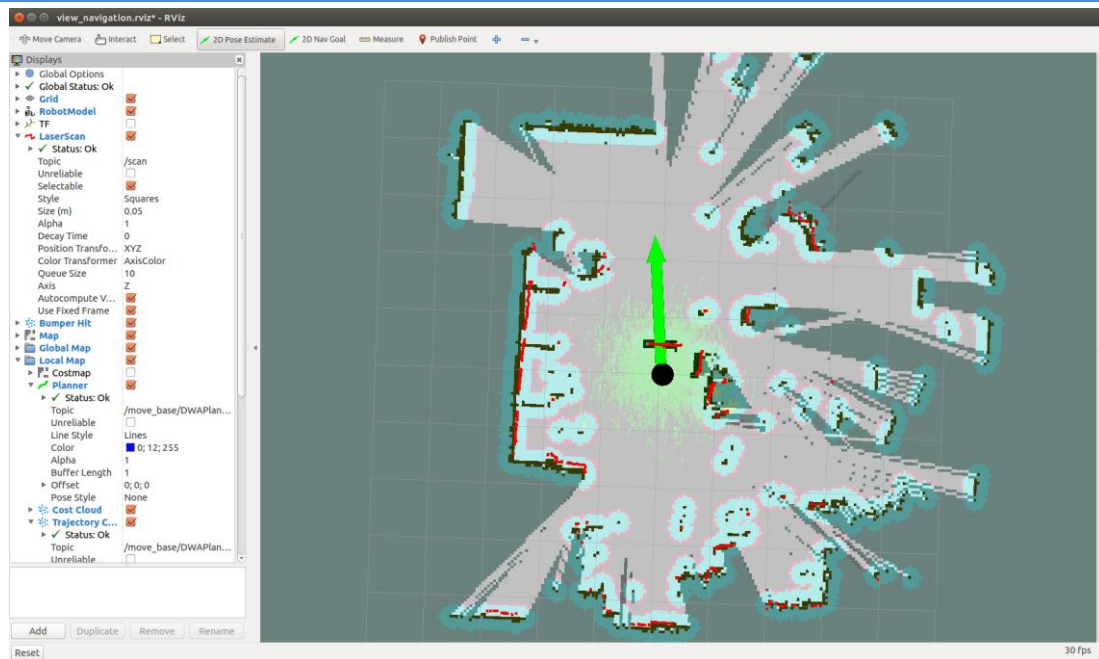
1. 设置起点

rviz 打开后显示 **E1** 默认所在的位置是栅格的中心点，不一定是 **E1** 实际所在的位置

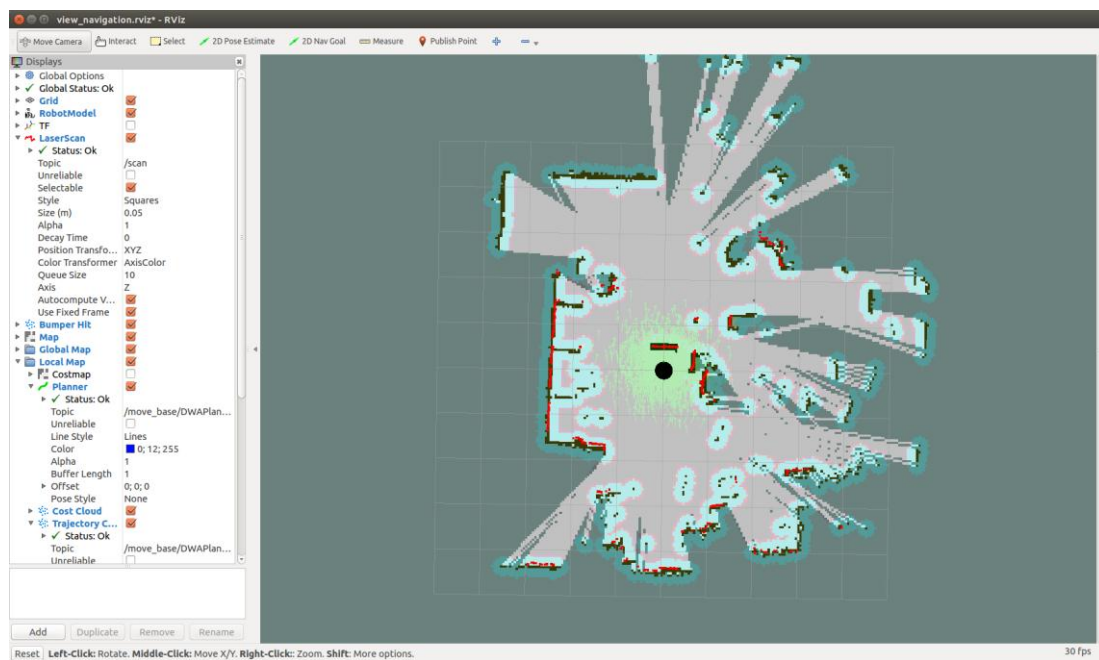
注意：每次打开 **rviz** 都要检查并设置起点。

点击 2D Pose Estimate

根据当前 **E1** 实际位置，在地图上选择正确的位置，并调整好 **E1** 的正前方方向(正前方的调整：在选择位置时，按住鼠标不放，拖动方向便可)，如下图所示



设置好起点后，如下图所示：激光线和障碍物，轮廓基本重合

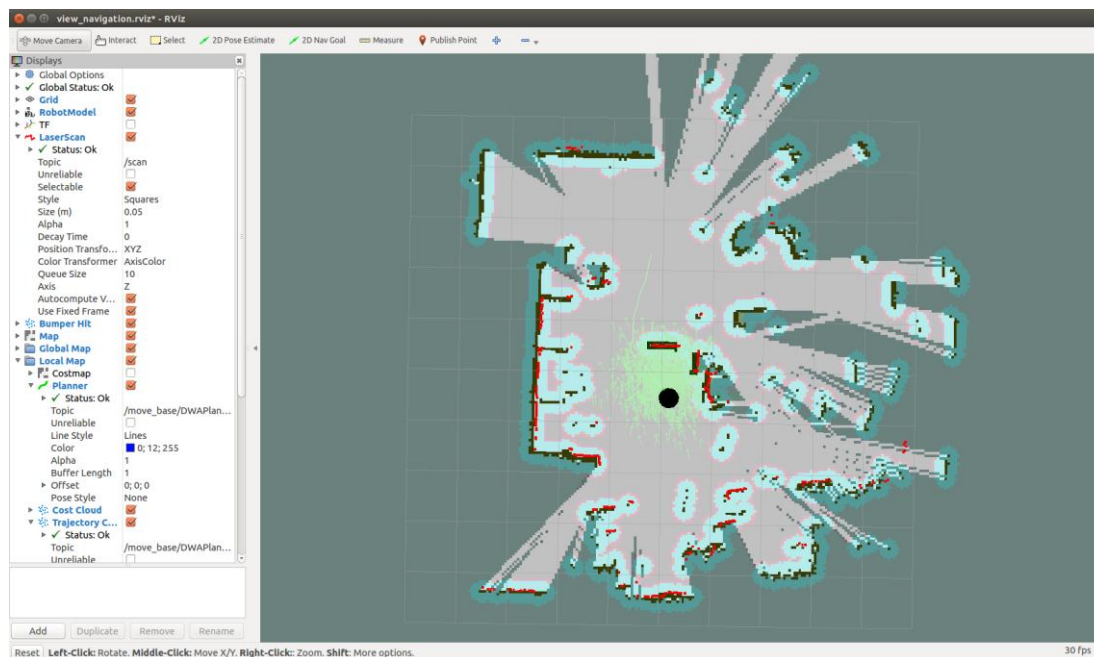


2. 设置目标点

点击 2D Nav Goal

在地图上选择要到达的目标点位置，并调整好 E1 停止时的正前方方向(正前方的调整：在选择位置时，按住鼠标不放，拖动方向便可)

设置好目标点后，导航模块便自主计算路径并控制 E1 向目标点移动



注意：每次导航时，都要先点击 2D Nav Goal 然后选择目标点，小车便自主导航移动。

3.9 多点连续导航

导航模块的默认固定 IP 是 192.168.31.200，默认用户名为 eaibot，密码为 eaibot。

确保已经按照上面“不带陀螺仪或带陀螺仪建图导航”的方法建好地图，并保存好地图。

注意：“不带陀螺仪”保存的地图名为 retest_map，“带陀螺仪”保存的地图名为 retest_map2。

导航时默认会导入该名字的地图，若要修改导入的地图名，可以修改相对应的导航 launch 文件，如 navigation_demo_multi.launch 文件中修改 retest_map.yaml 为你要导入的地图名 `<arg name="map_file" default="$(find dashgo_nav)/maps/retest_map.yaml"/>`

打开一个终端，ssh 登录导航模块，运行导航 launch

不带陀螺仪

```
$ ssh eaibot@192.168.31.200
```

```
$ roslaunch dashgo_nav navigation_demo_multi.launch
```

带陀螺仪

```
$ ssh eaibot@192.168.31.200
$ roslaunch dashgo_nav navigation_demo_imu_multi.launch
```

再打开一个终端，运行以下命令打开 rviz

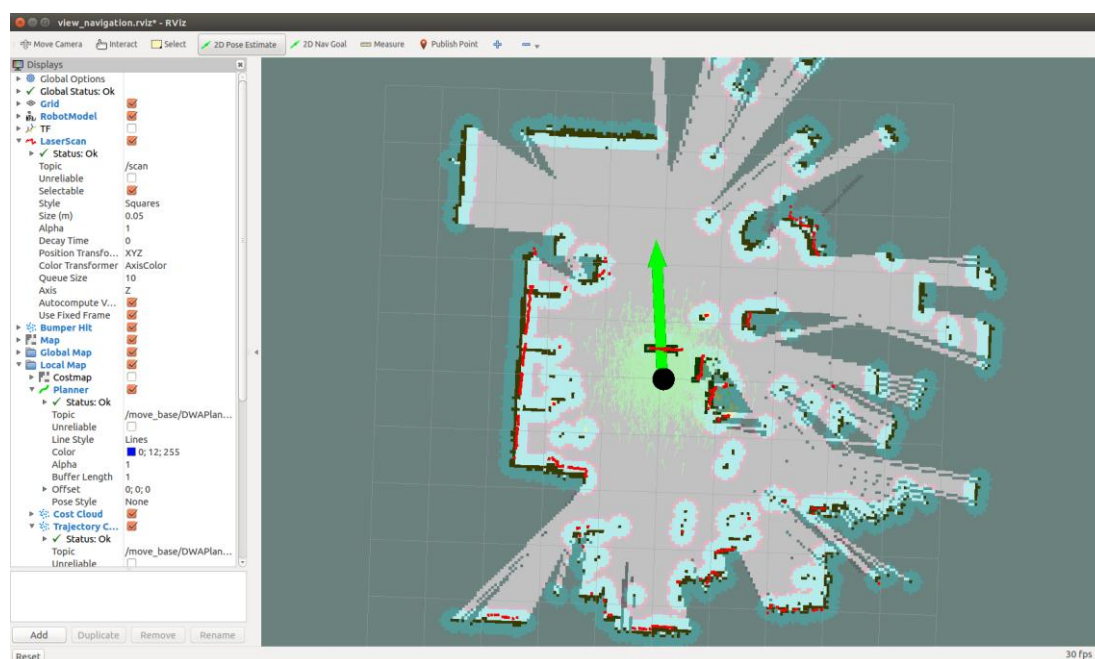
```
$ export ROS_MASTER_URI=http://192.168.31.200:11311
$ roslaunch dashgo_rviz multi_goal.launch
```

3.9.1 设置起点

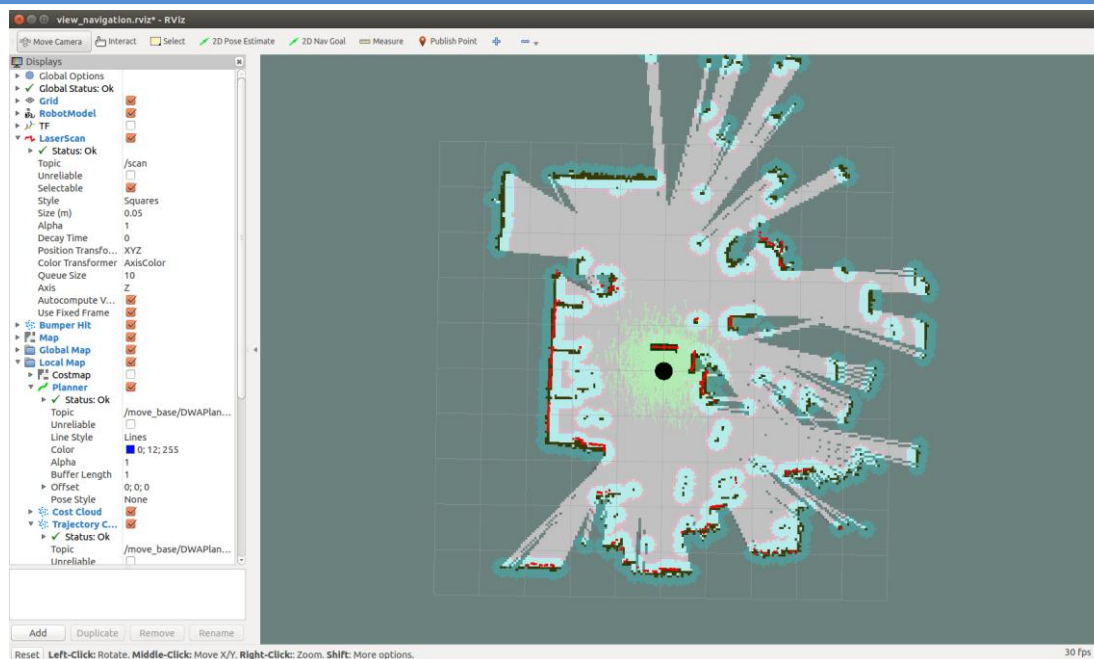
rviz 打开后显示 E1 默认所在的位置是栅格的中心点，不一定是 E1 实际所在的位置

注意：每次打开 rviz 都要检查并设置起点。

点击 2D Pose Estimate

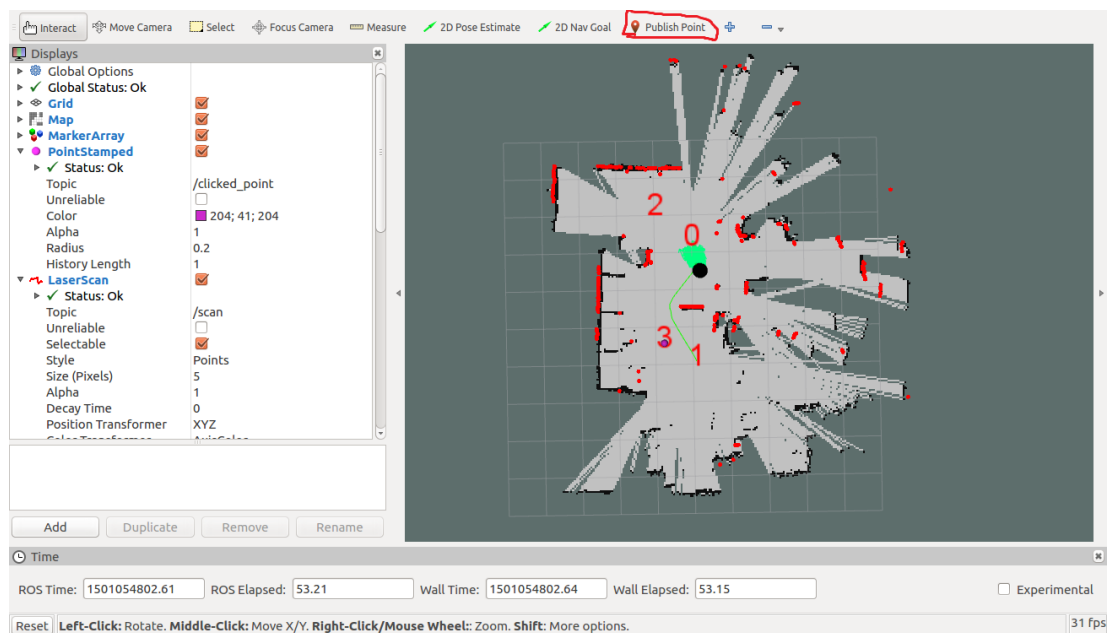


根据当前 E1 实际位置，在地图上选择正确的位置，并调整好 E1 的正前方方向(正前方的调整：在选择位置时，按住鼠标不放，拖动方向便可)



3.9.2 设置连续目标点

Publish Point 便是设置连续目标点, 每选择一个目标点之前都要先点击 Publish Point, 目标点的序号在 rviz 中以 0, 1, 2.....的数字显示出来



一个目标点到达后便向下一个目标点移动

