



# DASHGO

## Build Your Robot

Dashgo D1

瞬驰机器人移动平台

产品简介与规格书



## 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
<b>第 1 章 全面认识 D1.....</b>	<b>1</b>
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    产品说明 .....	3
1.2.1    发货清单 .....	3
1.2.2    产品概述 .....	3
1.2.3    规格说明 .....	4
1.3    装配说明 .....	5
1.3.1    平台加层（可选支架和螺栓） .....	5
1.3.2    加装雷达 .....	11
1.4    接口说明 .....	15
<b>第 2 章 D1 之基本使用开发 .....</b>	<b>18</b>
2.1    蓝牙控制 D1 移动 .....	18
2.2    蓝牙的开发使用 .....	20
2.2.1    D1 的物理参数 .....	20
2.2.2    速度控制指令 .....	20
2.3    下位机的开发使用 .....	22

2.3.1	连接方式.....	22
2.3.2	D1 的物理参数.....	23
2.3.3	支持指令.....	23
<b>第 3 章 D1 之 ROS 开发 .....</b>		<b>26</b>
3.1	PC 安装 ROS 系统.....	26
3.1.1	安装准备 .....	26
3.1.2	配置 Ubuntu 软件仓库 .....	27
3.1.3	配置 ROS 的 apt 源.....	28
3.1.4	安装 ROS 软件包.....	28
3.1.5	配置环境变量 .....	29
3.1.6	测试 ROS 安装是否成功.....	29
3.2	搭建 DASHGO 运行环境 .....	30
3.3	ROS 控制 D1 移动.....	31
3.3.1	键盘控制 .....	31
3.3.2	命令行控制 .....	32
3.3.3	手机控制 .....	33
3.4	精度校准.....	35
3.5	搭建激光雷达 F4 建图环境 .....	36
3.5.1	安装 ROS 依赖包.....	36
3.5.2	下载 F4 驱动包.....	36
3.6	激光雷达 F4 与 D1 的坐标校正 .....	37
3.6.1	运行地图扫描节点 .....	37

3.6.2	运行 <i>rviz</i> 进行校正.....	37
3.6.3	参数调整.....	42
3.6.4	<i>vim</i> (文本编辑命令) 的安装与基本使用.....	43
3.6.5	前后方向校正，只需修改第 4 个参数.....	44
3.6.6	左右方向校正，只需要修改第 5 个参数.....	46
3.7	D1 通过 ROS 建图.....	47
3.7.1	运行扫地图主程序.....	48
3.7.2	PC 端运行图形界面.....	49
3.7.3	控制 D1 移动扫描地图.....	50
3.7.4	扫描完地图后，保存地图.....	52
3.8	D1 通过 ROS 实现自主导航.....	53
3.8.1	修改 <i>launch</i> 文件，引用已经保存好的地图.....	53
3.8.2	运行地图自主导航节点.....	54
3.8.3	运行 <i>rviz</i> 图形界面.....	54
3.8.4	设置起点.....	54
3.8.5	设置目标点.....	55

# 第 1 章 全面认识 D1

## 1.1 安全说明

感谢您购买 EAI 产品的 D1。为了您的安全，请在使用 D1 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



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。



非专业人员，不要私自对 D1 进行拆卸；



不使用非原厂标配的电池、电源、充电座；



运行时请勿用手触碰 D1；



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

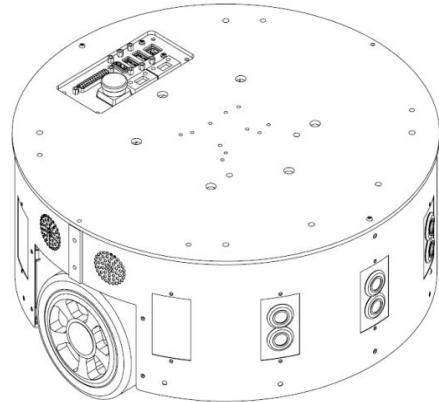
为防止事故发生和减小伤害，请不要在下面列出的条件下存放 D1。

#### 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 产品说明

### 1.2.1 发货清单



**D1**

备注：以上为标准配置清单，平台加层的零件按客户需求选配。

### 1.2.2 产品概述

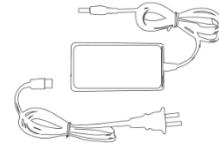
D1 是 EAI 科技专门针对 ROS 开发的移动平台，自主研发的核心结构保证了精度高、载重大、动力足、续航长和扩展性强的性能，深受机器人企业和科研机构的欢迎。

#### 主要特性

- 易于使用 - 由整机及其附件组成，到货后无需再做繁杂的装配；
- 可靠耐用 - 坚固耐用；
- 精度高 - 工业级编码器，定位精准；
- 软件开发工具包 - 提供配套的 ROS 开发包，帮助客户加快机器人项目的开发；
- 可定制化 - 轻松地从数十种支持和测试的配件中选择并使用合适的配件；
- 技术支持 - 齐全的软硬件使用文档，同时也可以获得 EAI 团队专业的技术支持。

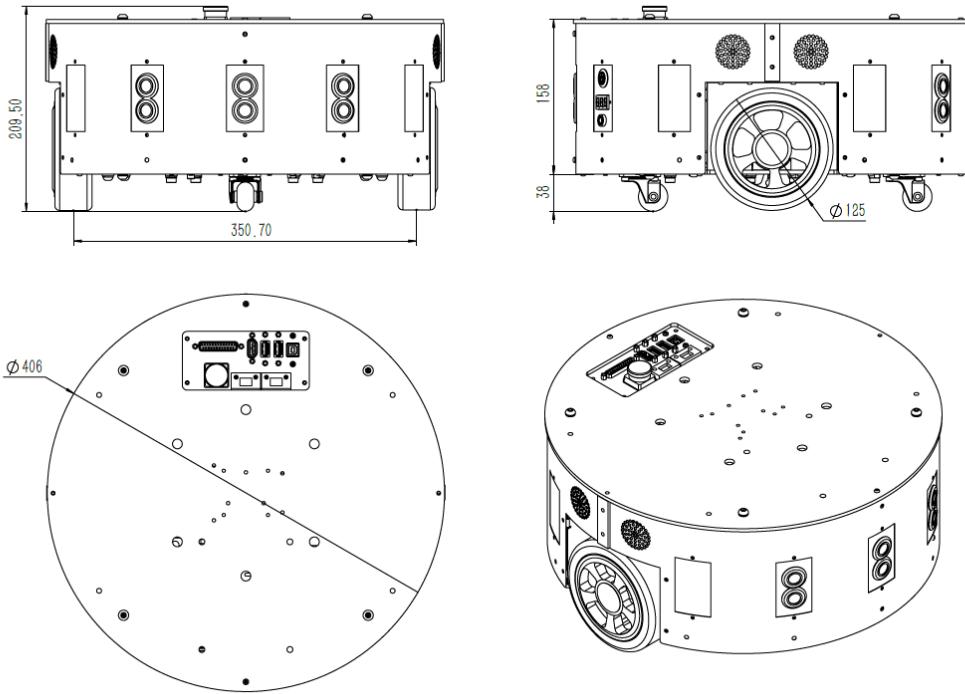


**USB-B 数据线**



**电源充电器**

### 1. 2. 3 规格说明

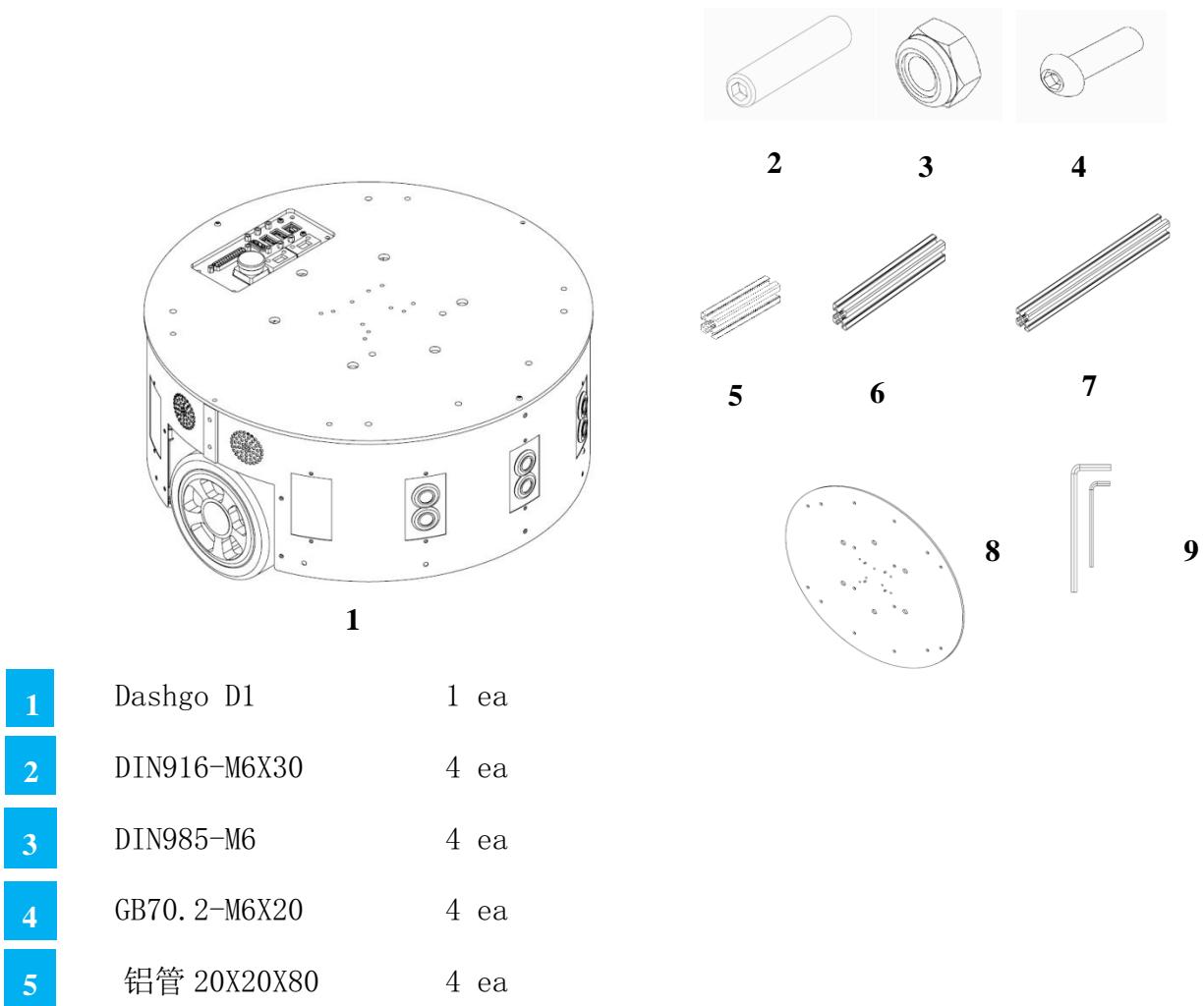


Dashgo 型号	D1
整体尺寸 (mm)	Φ406X210
轮子数量	4
驱动方式	差分驱动
主动轮直径 (mm)	125
悬空高度 (mm)	38
负载 (kg)	50
净重 (kg)	13.7
最大速度 (m/s)	0.8
续航时间 (h)	8
电池容量 (mAh)	12V 14000
超声波数量 (个)	4
通信接口	USB-UART / Bluetooth
电源输入	12V/2A DC
5V 电源输出	有

12V 电源输出	有
电量显示	标配
预留传感器安装位 (个)	5
预留扬声器安装位 (个)	2
预留散热风扇安装位 (个)	2
预留防跌落传感器 (个)	2
预留拓展安装位置 (个)	6

### 1.3 装配说明

#### 1.3.1 平台加层（可选支架和螺栓）



<b>6</b>	铝管 20X20X150	4 ea
<b>7</b>	铝管 20X20X230	4 ea
<b>8</b>	加层板 $\Phi 400 \times 3$	1 ea
<b>9</b>	内六角扳手	2 ea

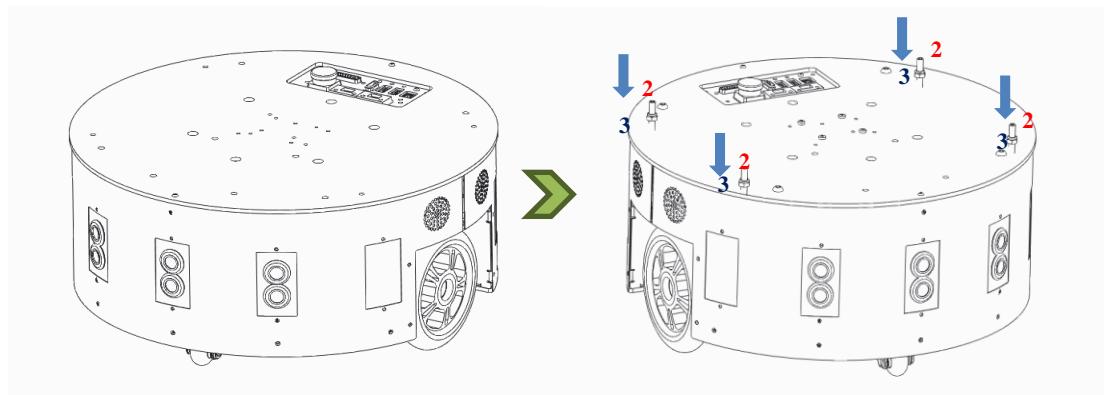
### 加装零件附件列表:

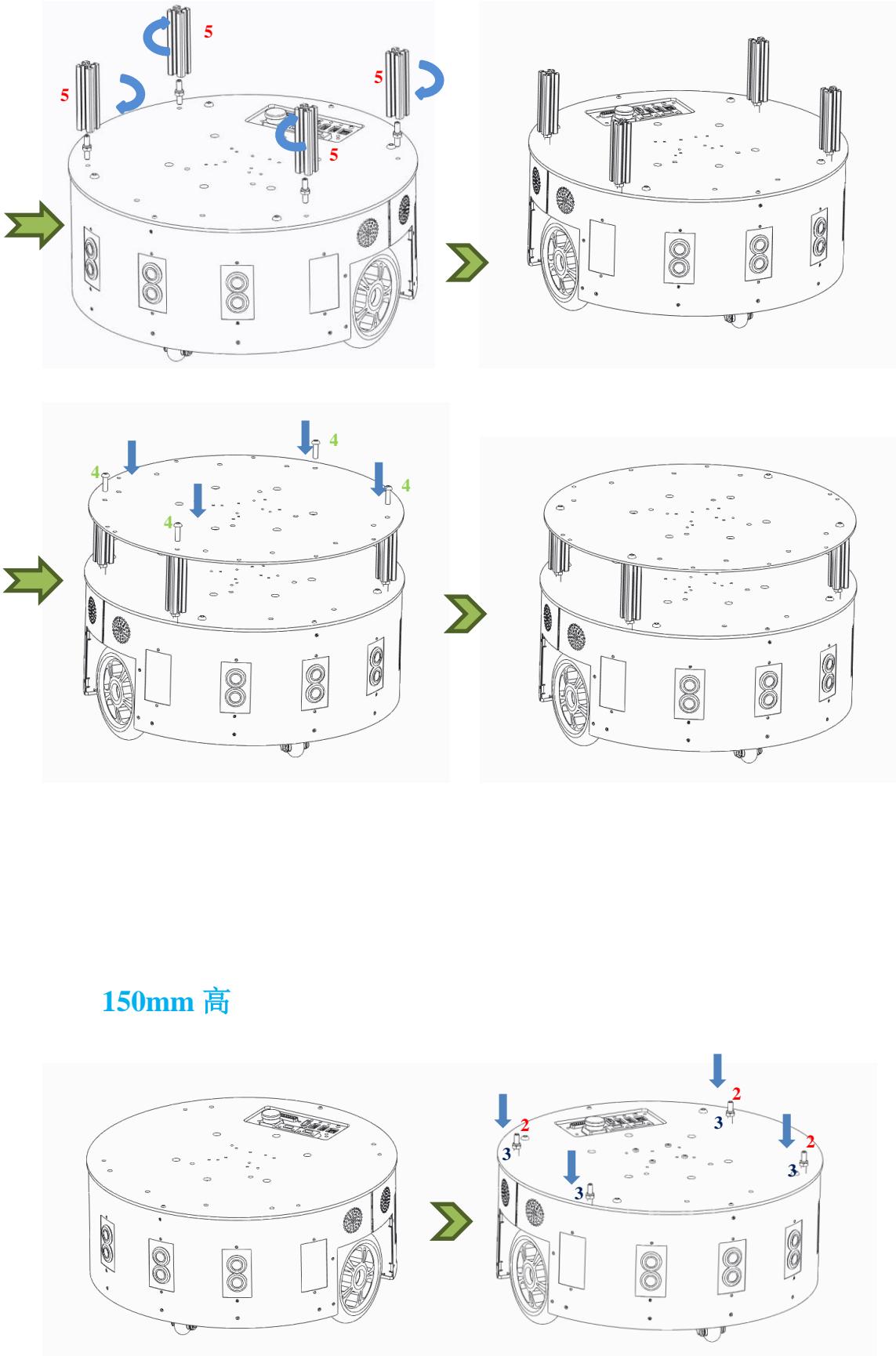
类型	数量	Dashgo D1	DIN916-M6X30	DIN985-M6	GB70.2-M6X20	加层板 $\Phi 400 \times 3$
加装一层平台	1	4	4	4	4	1
加装两层平台	1	4	4	4	12	2
加装三层平台	1	4	4	4	20	3
加装一层雷达	1	4	4	4	4	1

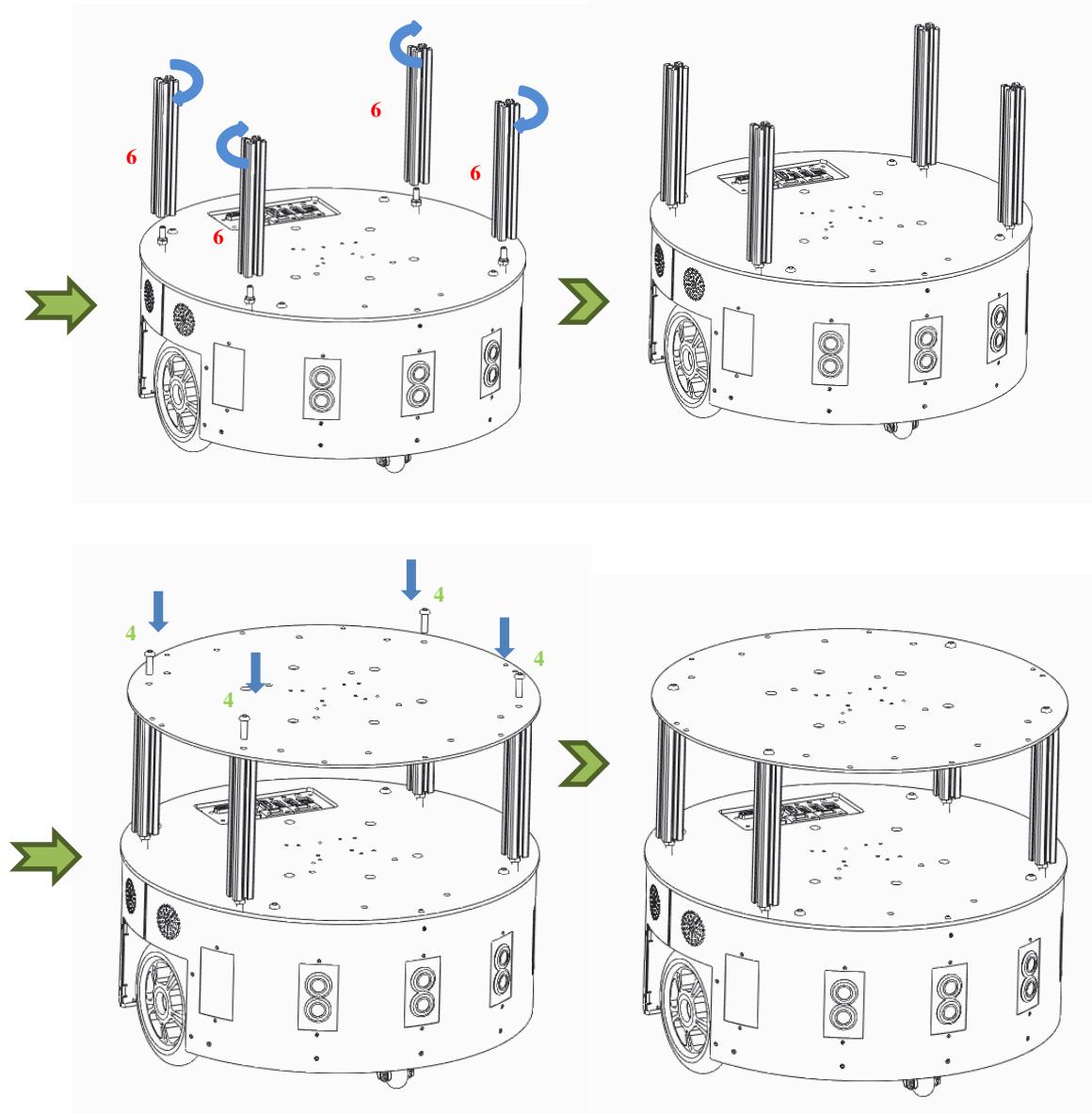
### 1. 加装一层平台

加装一层平台，可选高度有 80mm、150mm、230mm 的支撑杆。具体安装步骤如下列图示教程所示，您可以参照图示自由组合自己的平台。

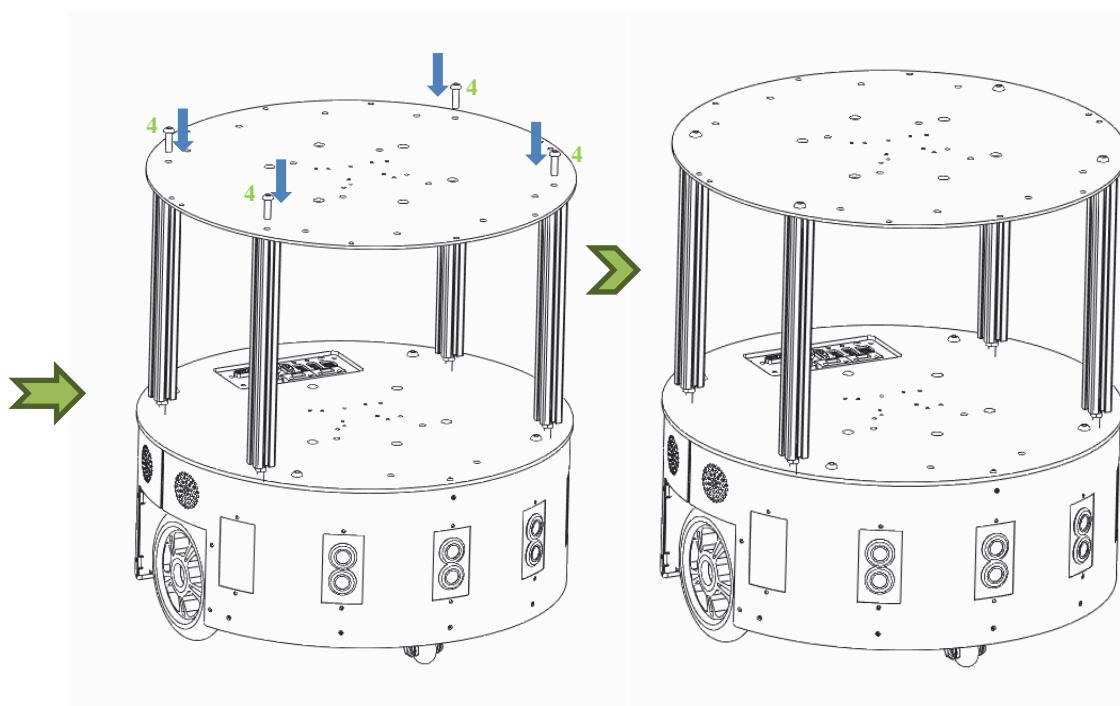
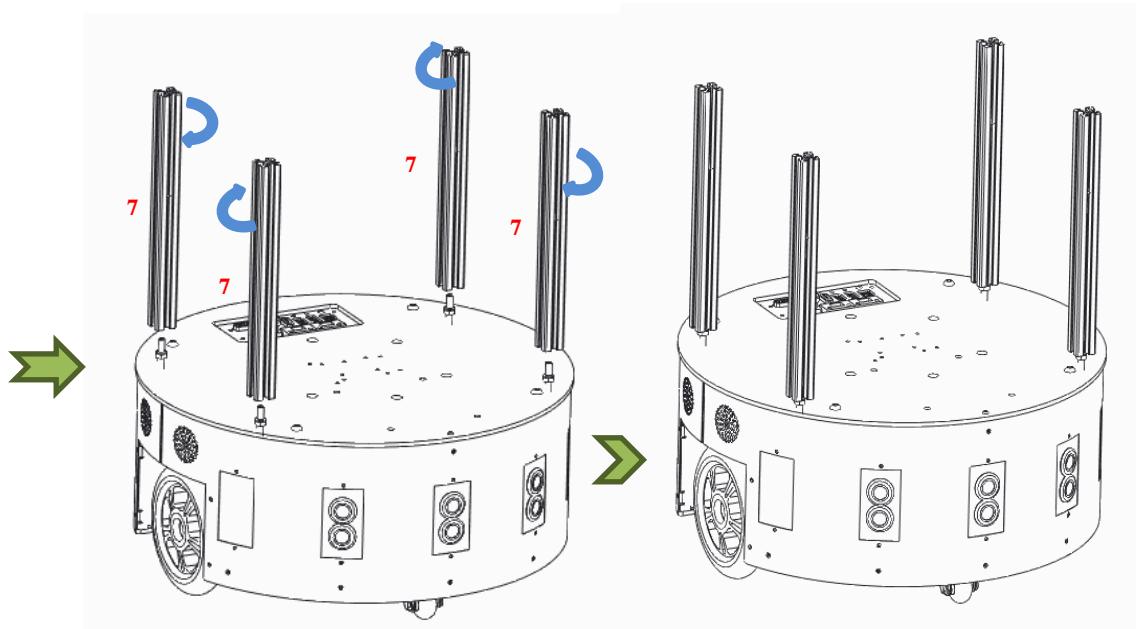
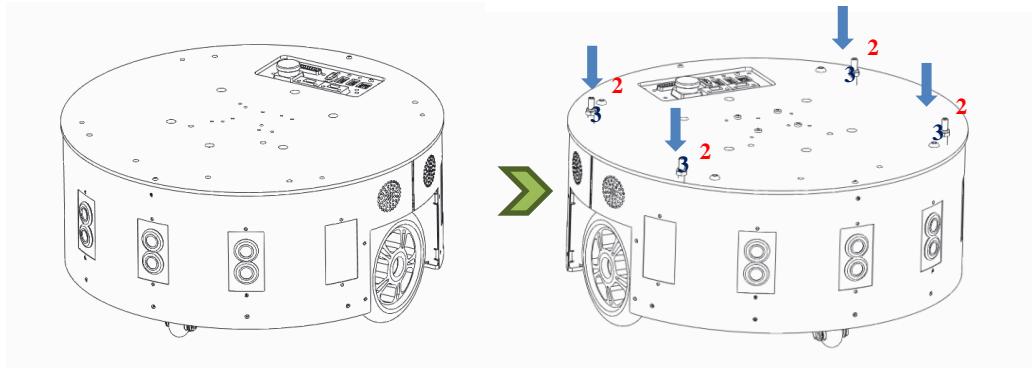
#### 80mm 高





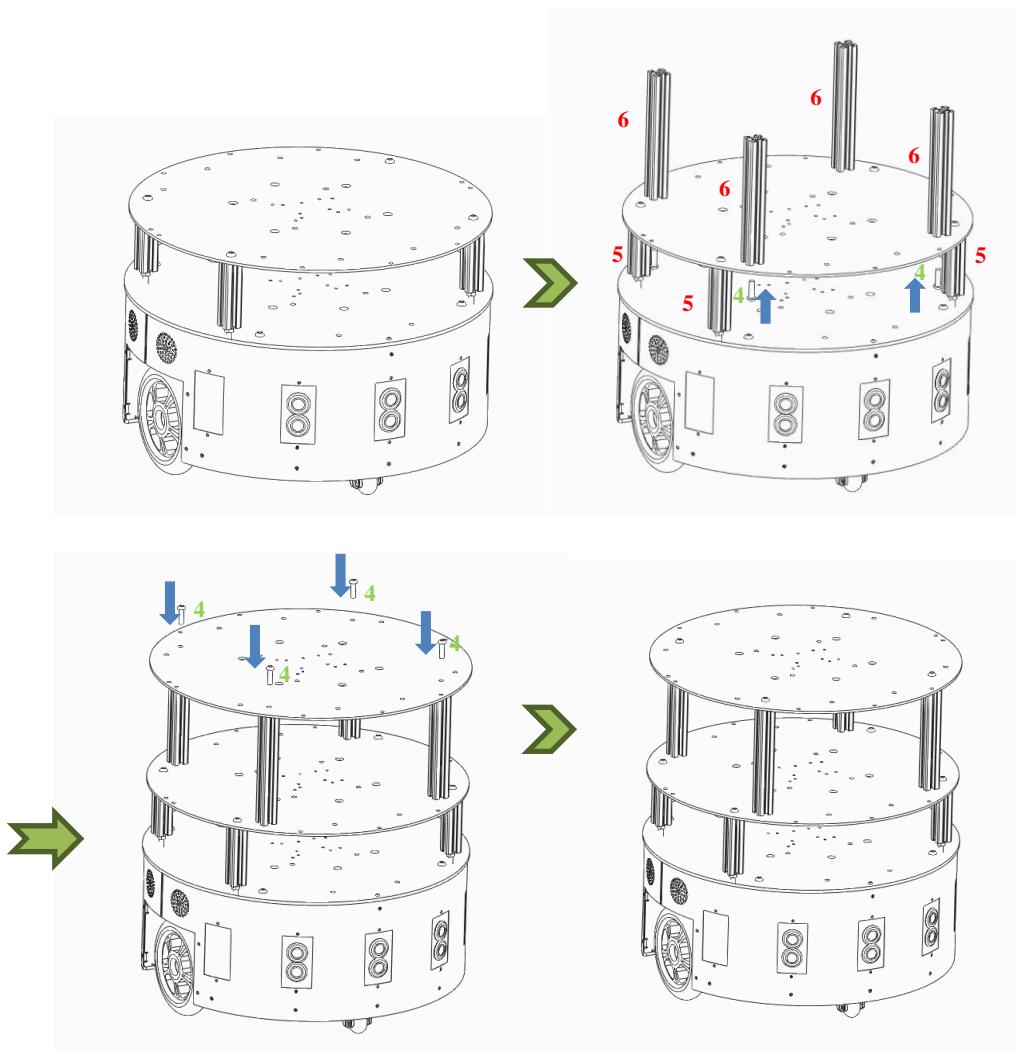


230mm 高

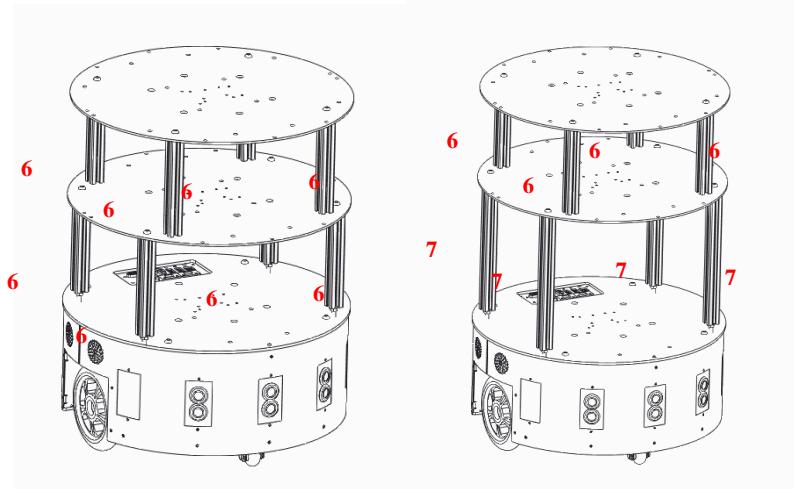


## 2. 加装两层平台

在上面 1 的基础上可以继续再增加一层平台支架。具体安装步骤如下列图示教程所示，您可以参照图示自由组合自己的平台。

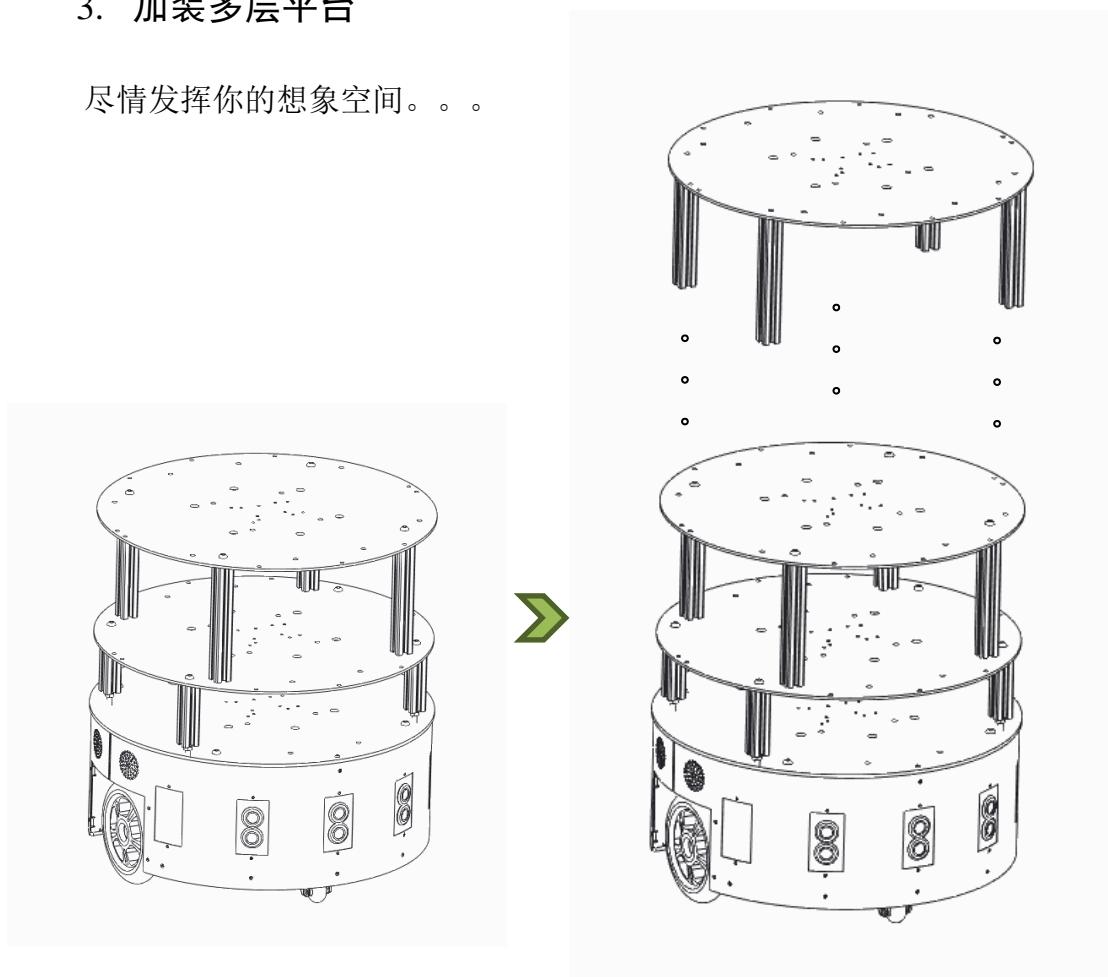


甚至可以任意长度的组合



### 3. 加装多层平台

尽情发挥你的想象空间。。。

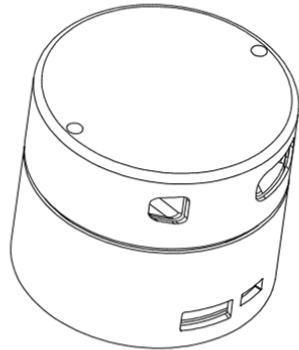


#### 1. 3. 2 加装雷达

##### 1. 加装 Flash Lidar F4

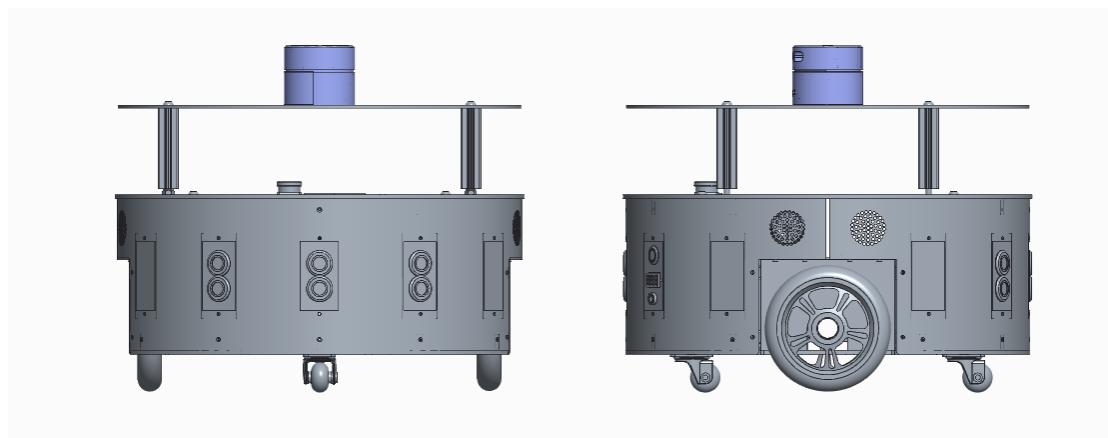
Flash Lidar 系列，F4 激光雷达是 EAI 科技推出的一款高性能的激光雷达。该款雷达的

详细介绍, 请参见 F4 产品说明书。



Flash Lidar 系列 F4 与 D1 安装方式推荐以下两种:

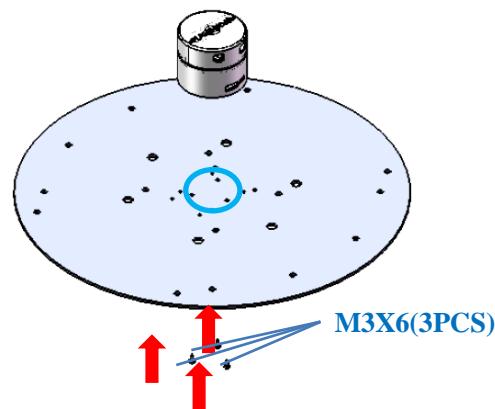
- A. 安装到平台或加层上方。此类安装方式, 雷达四周无遮挡, 软件建图过程不需要进行数据的剔除。



- B.倒装到加层下方。此类安装方式, 雷达放置于加层下部, 平台上可继续载物或摆放其它设备。加装 80mm 高的铝平台支架, 相对应的在上位机中剔除 4 根杆遮挡的位置。

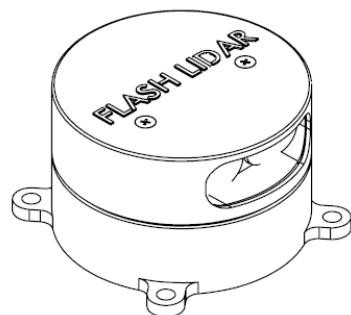


F4 先与加层板装配（如下图），然后再按照平台加层教程安装到 D1 上。安装孔位详细参考 3-4 章节接口说明。



## 2. 加装 Flash Lidar T1

Flash Lidar 系列，T1 激光雷达是 EAI 科技推出的一款小型轻量化的激光雷达。该款雷达的详细介绍，请参见 T1 产品说明书。



Flash Lidar 系列 T1 与 Dashgo 安装方式推荐以下三种：

- A. 安装到平台或加层上方。此类安装方式，雷达四周无遮挡，软件建图过程不需要进行数据的剔除。



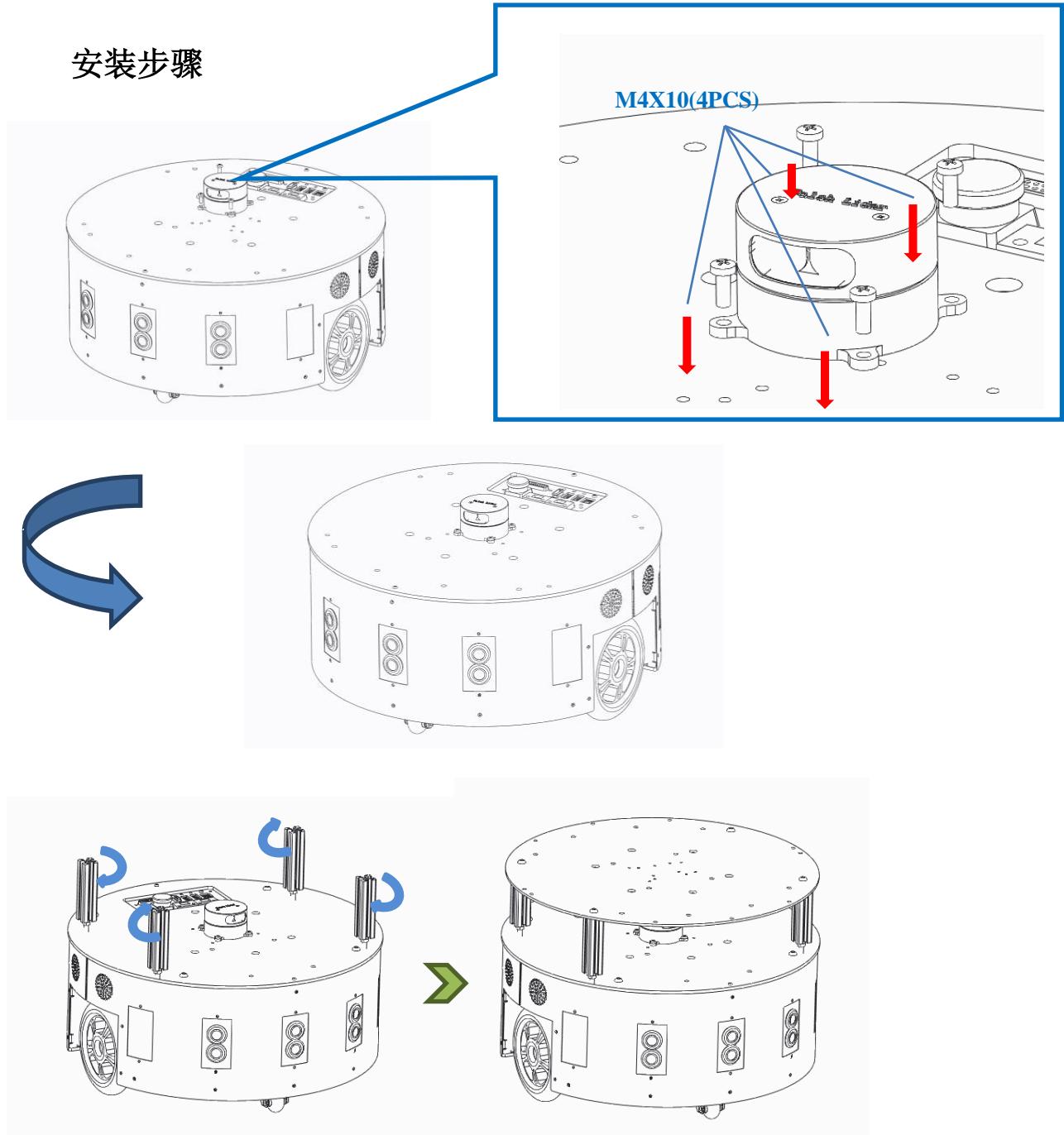
- B. 安装到平台上，加层下方。此类安装方式，雷达放置于加层下部，平台上可继续载物或摆放其它设备。加装 80mm 高的铝平台支架，相对应的在上位机中剔除 4 根杆遮挡的位置。



- C. 倒装到加层下方。此类安装方式，雷达放置于加层下部，平台上可继续载物或摆放其它设备。加装 80mm 高的铝平台支架，相对应的在上位机中剔除 4 根杆遮挡的位置。

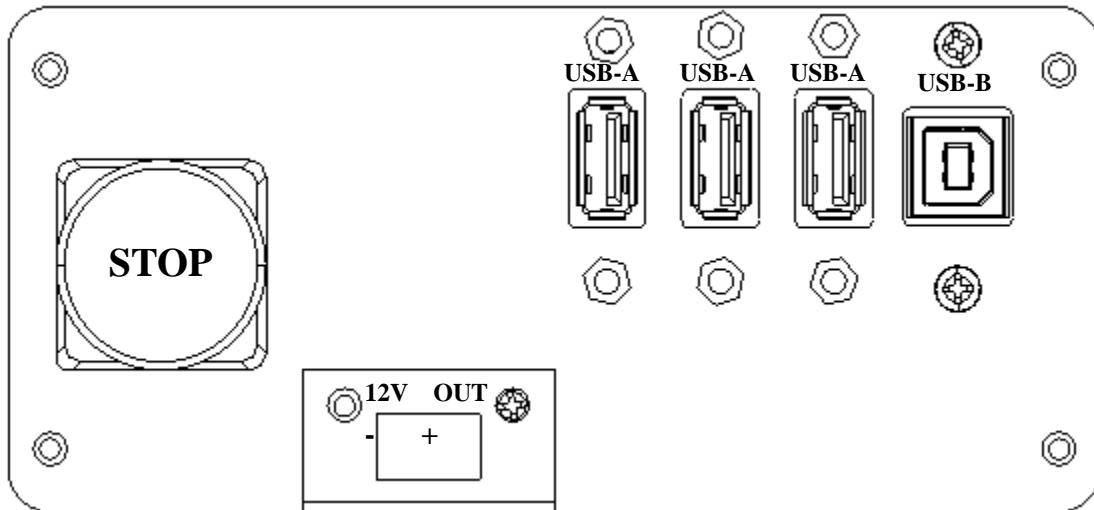


### 安装步骤



### 1.4 接口说明

输入输出接口板端口图示及说明如下图：



通过 USB-B 连接 D1 内置 Arduino 的 USB 通讯;

通过 USB-A 连接 D1 的 USB 5V 电源输出;

12V OUT 是 D1 内置电源输出;

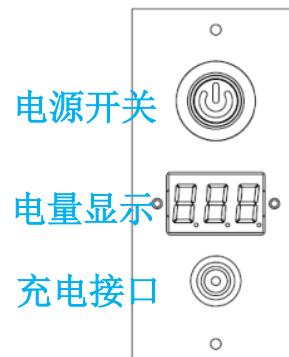
红色的 STOP 按键是急停开关，紧急情况下请按下急停按钮。

电源管理接口板的输入输出接口如右图所示：

通过电源开关控制 D1 开机和关机；

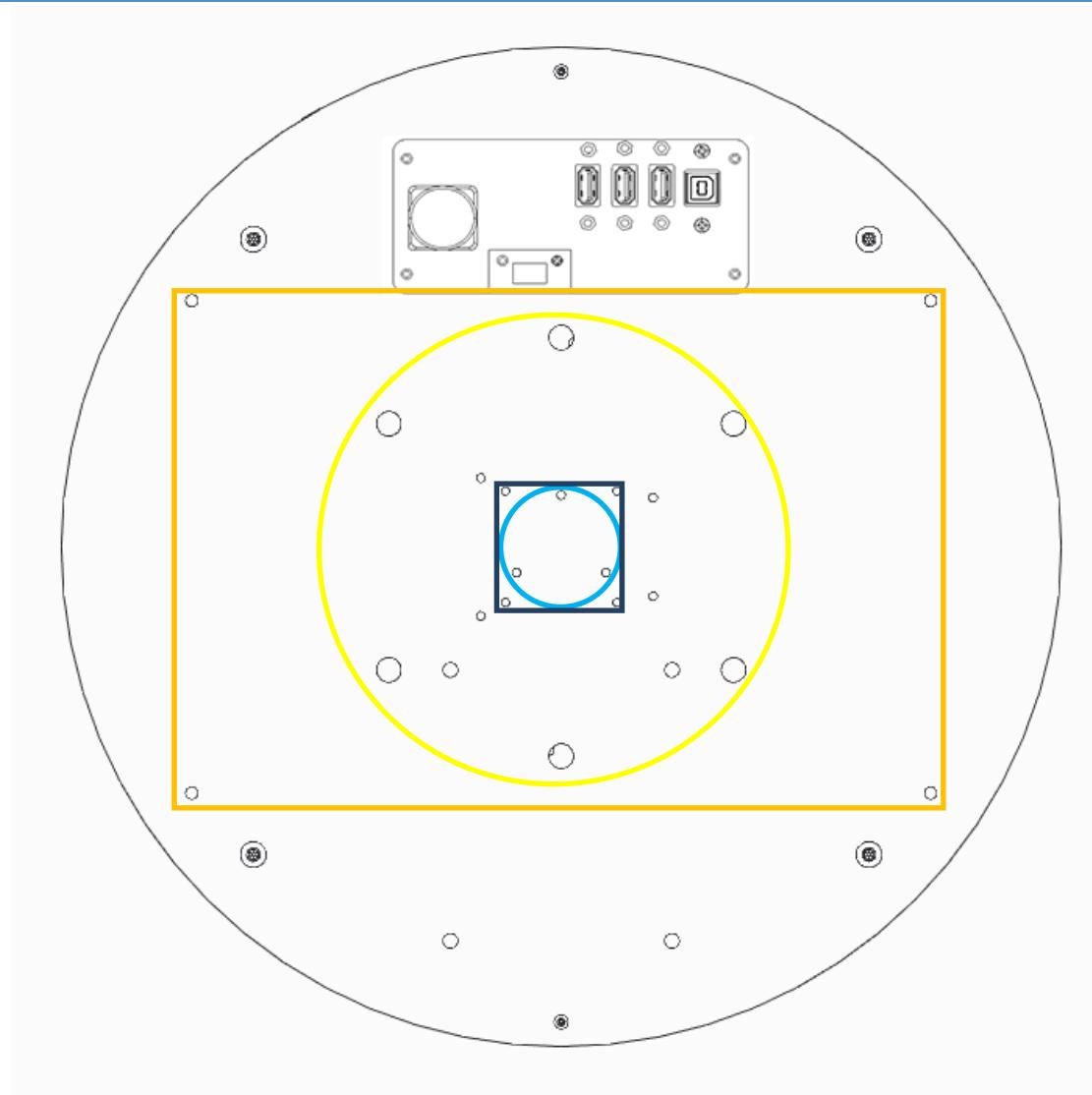
通过数码管显示 D1 内部电池电压的电量；

通过充电接口对 D1 进行充电。



**注意：电量显示11~13V范围均为安全值，数值显示低于12V请充电。充电前要先关闭D1电源开关。**

平台上预留丰富的安装位置，您可以选择多种设备接入。下图所示的是安装孔位及对应预留设备的说明。



预留拓展安装区域说明：

○ Flash Lidar F4

□ Flash Lidar T1

□ 平台加层

○ 预留安装孔

## 第 2 章 D1 之基本使用开发

### 2.1 蓝牙控制 D1 移动

D1 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。**

---

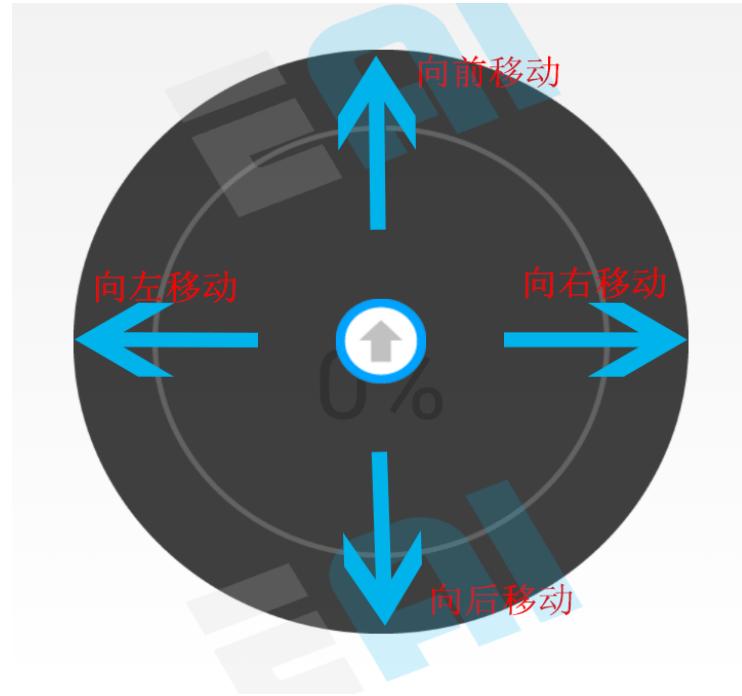
蓝牙搜索的时间为 6s，若没有搜索到想要使用的设备，就可点击标题栏右边的“**搜索**”按钮重新搜索蓝牙设备。若搜索到要使用的设备，便可点击选择该设备，连接成功后便进入蓝牙控制界面，如下图所示：



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：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。**

## 2.2 蓝牙的开发使用

通过蓝牙向 D1 发送移动控制指令。

### 2.2.1 D1 的物理参数

```
wheel_diameter: 0.1260      //轮子直径,单位: 米  
wheel_track: 0.3500        //两个轮子的间距,单位: 米  
encoder_resolution: 1200    //编码器分辨率,轮子转一圈, 编码器产生的脉冲数  
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（ $1/\text{PID\_RATE}$  秒）内 D1 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 D1 以 0.3m/s 的速度前进，指令的参数计算方法如下：

D1 轮子转动一圈，移动的距离为轮子的周长：

```
wheel_diameter * 3.1415926
```

D1 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
```

D1 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 D1 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 D1 自转，就需要一个轮子正转，一个轮子反转。例如，需要 D1 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，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 D1 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 D1 以 V1=1 弧度/秒的速度转动，V2=0.2m/s 的速度前进。计算方法如下：

```
左轮 vl = V2 + V1 * wheel_track / 2.0  
右轮 vr = V2 - V1 * 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 D1 持续运动时，需要不断地下发指令，如果下位机 2 秒内没收到指令，D1 将停止运行。**

## 2.3 下位机的开发使用

### 2.3.1 连接方式

使用 D1 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 D1 的物理参数

```
wheel_diameter: 0.1260 //轮子直径,单位: 米  
wheel_track: 0.3500 //两个轮子的间距,单位: 米  
encoder_resolution: 1200 //编码器分辨率, 轮子转一圈, 编码器产生的脉冲数  
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: D1 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 D1 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 D1 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 D1 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 D1 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 D1 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 D1 以 0.3m/s 的速度前进, 指令的参数计算方法如下:

D1 轮子转动一圈, 移动的距离为轮子的周长:

wheel\_diameter \* 3.1415926

D1 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
```

D1 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 * 3031.52278 / 30)  
= 30
```

所以，如果要设置 D1 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 D1 需要转弯，就需要一个轮子正转，一个轮子反转。例如，需要 D1 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 D1 持续运动时，需要不断地下发指令，如果下位机 2 秒内没收到指令，D1 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

# 第 3 章 D1 之 ROS 开发

## 3.1 PC 安装 ROS 系统

### 3.1.1 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 14.04 和 ROS indigo。理由是：ROS indigo 和 Ubuntu 14.04 都是 LTS(长期维护版本，支持 5 年)，并且 ROS indigo 是专门为 Ubuntu14.04 量身定做。Ubuntu 的 15.10 和 ROS 的 jade 都不是 LTS, kinetic 版本刚出炉，配套的软件包还不齐全。

在操作过程中，建议使用有线网络，以免出现意外错误。

### 3.1.2 配置 Ubuntu 软件仓库

配置 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。

### 3.1.3 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。

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://pool.sks-keyservers.net --recv-key 0xB01FA116
$ 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://pool.sks-keyservers.net --recv-key 0xB01FA116
$ 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://pool.sks-keyservers.net --recv-key 0xB01FA116
$ 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。**

---

### 3.1.4 安装 ROS 软件包

```
$ sudo apt-get install ros-indigo-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。

### 3.1.5 配置环境变量

```
$ sudo rosdep init  
$ rosdep update  
  
$ echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc  
$ source ~/.bashrc
```

### 3.1.6 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

\* /rosdistro: indigo  
\* /rosversion: 1.11.20

## NODES

auto-starting new master  
process[master]: started with pid [3472]  
ROS\_MASTER\_URI=http://eaibot:11311/  
  
setting /run\_id to 45d93ed8-a23a-11e6-99b1-4437e63de0fc  
process[rosout-1]: started with pid [3485]  
started core service [/rosout]

## 3.2 搭建 Dashgo 运行环境

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。

重启 PC，使配置生效。

安装依赖包

```
$ sudo apt-get install git python-serial ros-indigo-serial g++
```

下载并编译 dashgo 包

```
$ mkdir -p ~/dashgo_ws/src  
$ cd ~/dashgo_ws/src  
$ git clone https://github.com/EAIBOT/dashgo.git  
$ cd ~/dashgo_ws/src/dashgo  
$ git checkout slam_02  
$ cd ~/dashgo_ws
```

```
$ catkin_make
```

**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。

为 D1 的串口增加一个设备别名 `/dev/dashgo`

```
$ cd ~/dashgo_ws/src/dashgo/dashgo_bringup/startup  
$ sudo chmod +x /*  
$ sudo sh create_dashgo_udev.sh
```

重新插拔连接 PC 的 D1 USB 线后，通过如下命令查看是否生效。

```
$ ls -l /dev/dashgo
```

---

**注意：如果没有找到 `/dev/dashgo` 串口，先确定 `/dev` 下是否存在名为 `ttyACM` 开头的串口，若没有，则是 D1 的串口没有连上。**

---

### 3.3 ROS 控制 D1 移动

本教程提供三种方式控制小车的移动。

首先，在终端运行 dashgo 的 D1 驱动节点

```
$ roslaunch dashgo_bringup minimal.launch
```

---

**注意：运行上面命令行是 ROS 控制 D1 移动的基础。**

---

#### 3.3.1 键盘控制

在新的终端先安装键盘控制包 `ros-indigo-teleop-twist-keyboard`

```
$ sudo apt-get install ros-indigo-teleop-twist-keyboard
```

然后，运行

```
$ rosrun dashgo_bringup teleop_twist_keyboard.py
```

得到如下内容

```
Reading from the keyboard and Publishing to Twist!
```

```
-----  
Moving around:
```

```
u i o
```

```
j k l
```

```
m , .
```

```
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%
```

```
anything else : stop
```

```
CTRL-C to quit
```

控制 D1 移动的按键分布如下：

- 前进： i
- 后退： ,
- 左转： j
- 右转： l
- 停止： k

### 3.3.2 命令行控制

在新的终端执行如下命令，给底盘一个恒定的速度一直运动下去。

```
$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist \  
'{linear: {x: 0.2, y: 0, z: 0}, angular: {x: 0, y: 0, z: 0}}'
```

CTRL+C 取消该命令。

说明：

- linear: {x: 0.2, y: 0, z: 0}
  - x:0.2 表示以 0.2m/s 的速度前进，也可以修改为 0.1~0.7 任意值，0.5 以上的速度比较快，要把 D1 放到空旷的地方
  - 该值也支持负数，-0.2m/s 表示以 0.2m/s 的速度后退。

- angular: {x: 0, y: 0, z: 0}
- z:0 表示以 rad/s (弧度/秒) 的速度向左转动，一般取值 0.5~3.0。
- 该值也支持负数， 表示向右转动。

### 3.3.3 手机控制

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 PC 的 IP 地址。

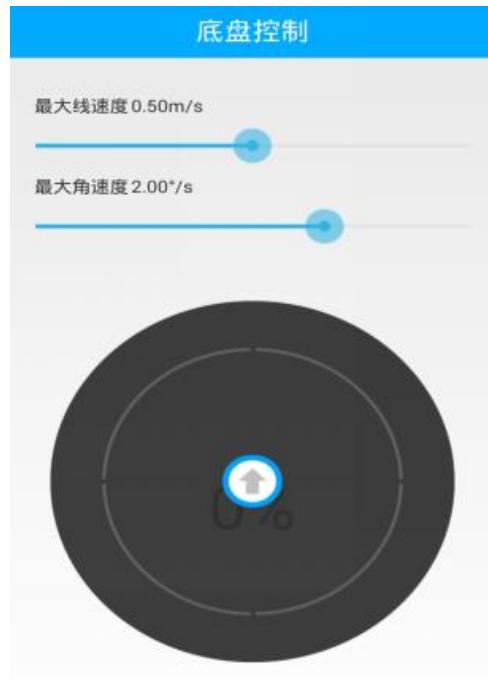
Ubuntu 查看 IP 地址，Ctrl + Alt + T 打开终端，输入 ifconfig，然后回车

```
eaibot@eaibot:~$ ifconfig
eth0 Link encap:以太网 硬件地址 44:37:e6:3d:e0:fc
inet 地址:172.20.105.24 广播:172.20.105.255 掩码:255.255.255.0 //有线网络的 IP 地址
inet6 地址: fe80::4637:e6ff:fe3d:e0fc/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST MTU:1500 跳点数:1
      接收数据包:350 错误:0 丢弃:0 过载:0 帧数:0
      发送数据包:134 错误:0 丢弃:0 过载:0 载波:0
      碰撞:0 发送队列长度:1000
      接收字节:27790 (27.7 KB) 发送字节:18843 (18.8 KB)
      中断:17

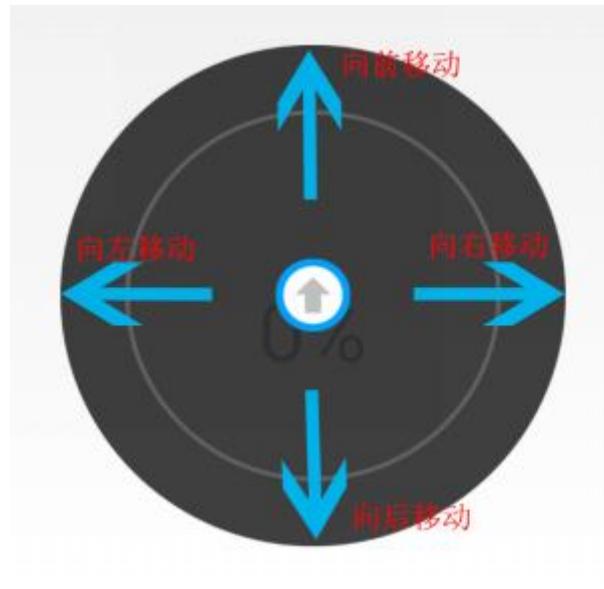
lo Link encap:本地环回
inet 地址:127.0.0.1 掩码:255.0.0.0 //本地的 IP 地址
inet6 地址: ::1/128 Scope:Host
      UP LOOPBACK RUNNING MTU:65536 跳点数:1
      接收数据包:167 错误:0 丢弃:0 过载:0 帧数:0
      发送数据包:167 错误:0 丢弃:0 过载:0 载波:0
      碰撞:0 发送队列长度:1
```

接收字节:12903 (12.9 KB) 发送字节:12903 (12.9 KB)

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

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。

打开一个终端，启动底盘驱动

```
$ roscl dashgo_bringup/scripts/  
$ sudo chmod +x /*  
$ roslaunch dashgo_bringup minimal.launch
```

然后在另一个终端运行测试脚本。

- 测试一：前进 1 米

```
$ rosrun dashgo_bringup check_linear.py
```

- 测试二：原地转动 360 度

```
$ rosrun dashgo_bringup 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_bringup/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 Bringup/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 Bringup/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。

## 3.5 搭建激光雷达 F4 建图环境

此节需要用到激光雷达 F4。

### 3.5.1 安装 ROS 依赖包

Ctrl + Alt + T 打开终端，运行命令

```
$ sudo apt-get install ros-indigo-serial ros-indigo-turtlebot-rviz-launchers \
ros-indigo-move-base-msgs libghc-sdl-image-dev libsdl-image1.2-dev \
ros-indigo-navigation ros-indigo-slam-gmapping ros-indigo-teb-local-planner
```

### 3.5.2 下载 F4 驱动包

Ctrl + Alt + T 打开终端，运行命令

```
$ cd ~/dashgo_ws/src
$ git clone https://github.com/EAIBOT/flashgo.git
```

```
$ cd flashgo  
$ git checkout 2.0.0
```

然后编译

```
$ cd ~/dashgo_ws  
$ catkin_make
```

创建 flash lidar 的串口别名的访问

```
$ rosdep flashgo/startup  
$ sudo chmod +x /*  
$ sudo sh initenv.sh
```

重新插拔 F4 连接电脑的 USB 线

查看激光雷达 USB 口是否存在

```
$ ls -l /dev/flashlidar
```

## 3.6 激光雷达 F4 与 D1 的坐标校正

此节需要用到激光雷达 F4。

### 3.6.1 运行地图扫描节点

Ctrl + Alt + T 打开终端，运行命令

```
$ roslaunch dashgo_nav gmapping_demo.launch
```

### 3.6.2 运行 rviz 进行校正

Ctrl + Alt + T 打开终端，安装 rviz，如下：

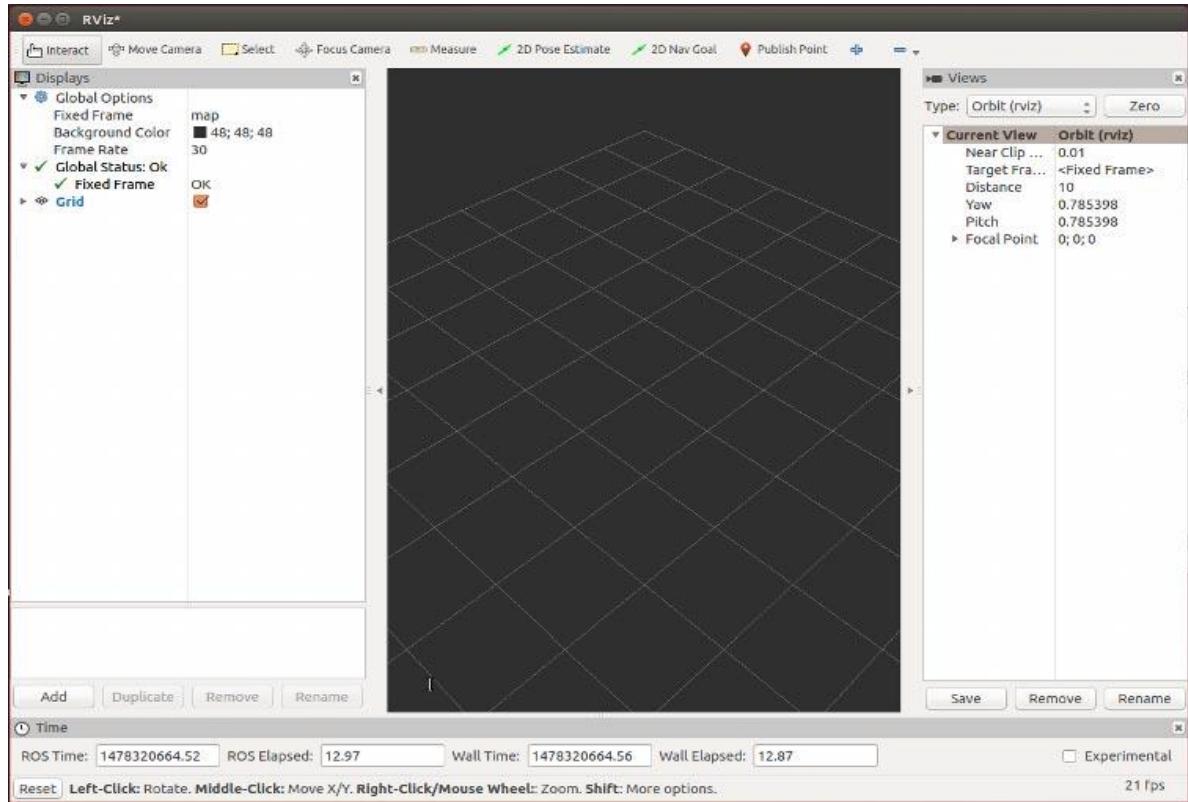
```
$ sudo apt-get install ros-indigo-serial ros-indigo-turtlebot-rviz-launchers
```

运行轻量级的 rviz

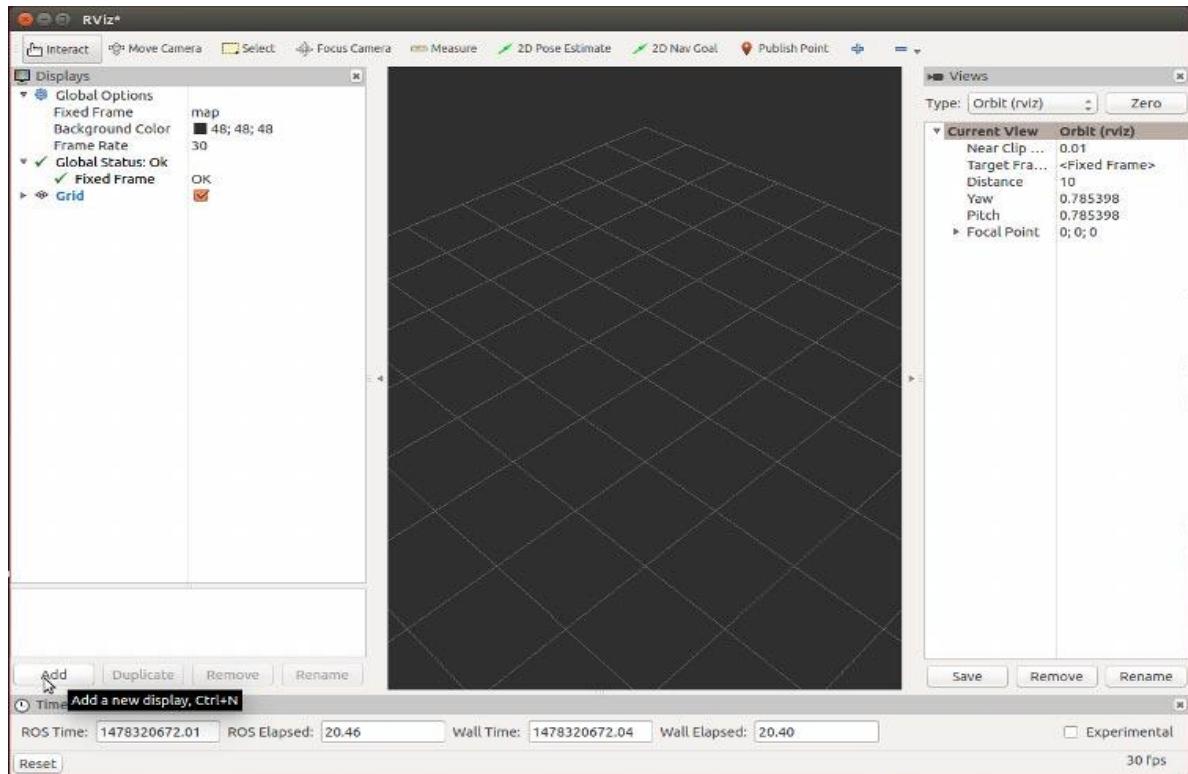
```
$ rviz
```

rviz 简易配置

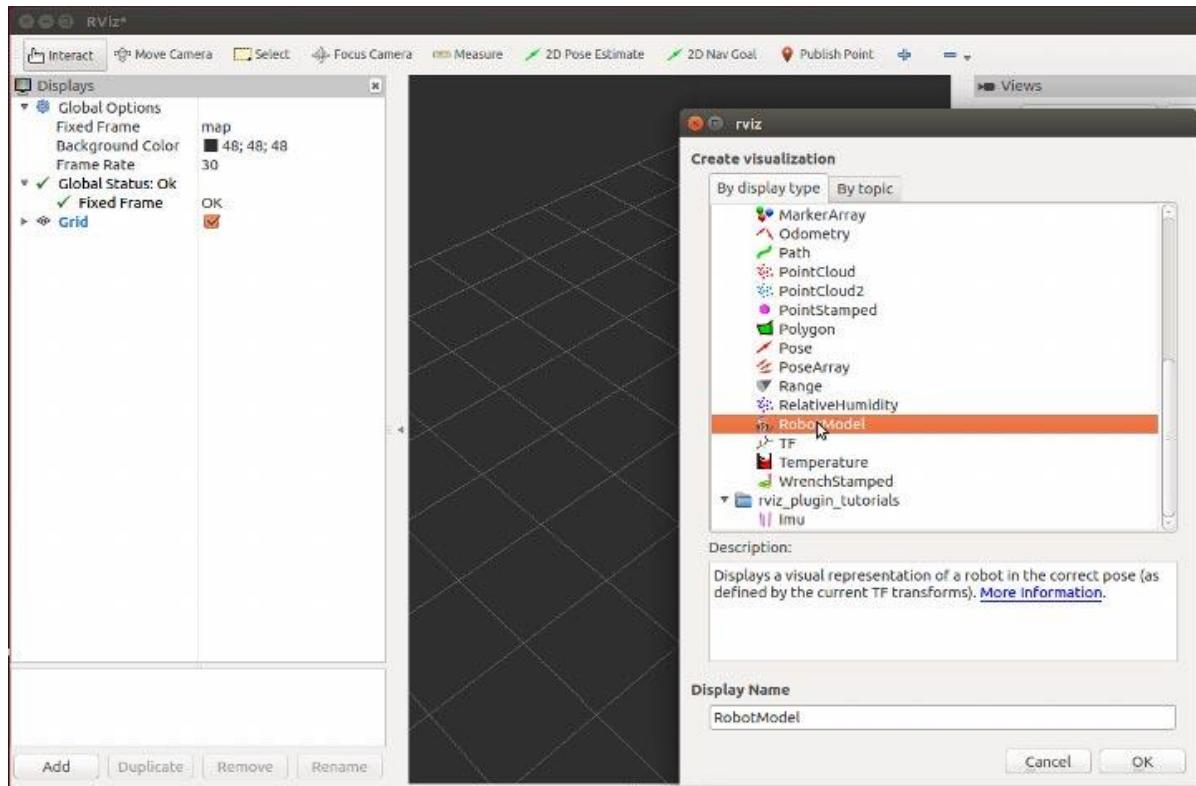
运行 rviz 显示的界面如下：



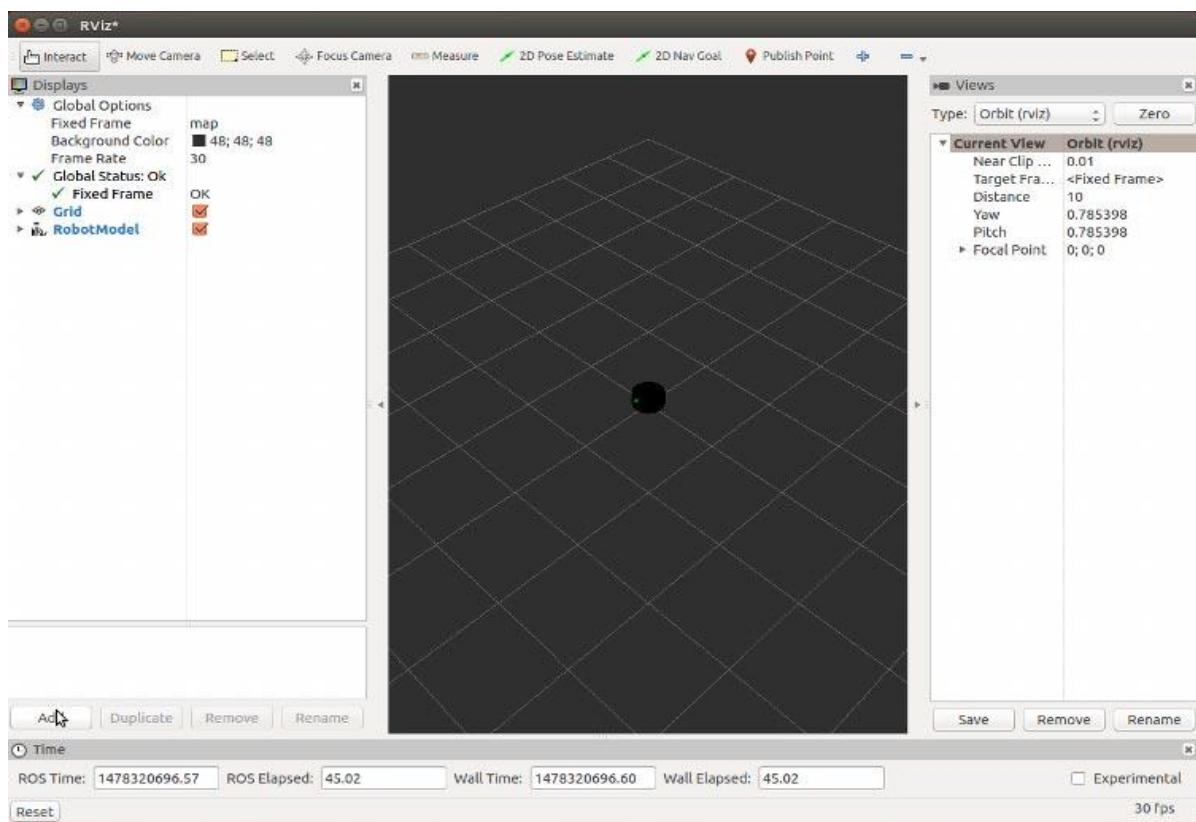
对 rviz 界面显示进行简单的配置，点击 Add 添加控件



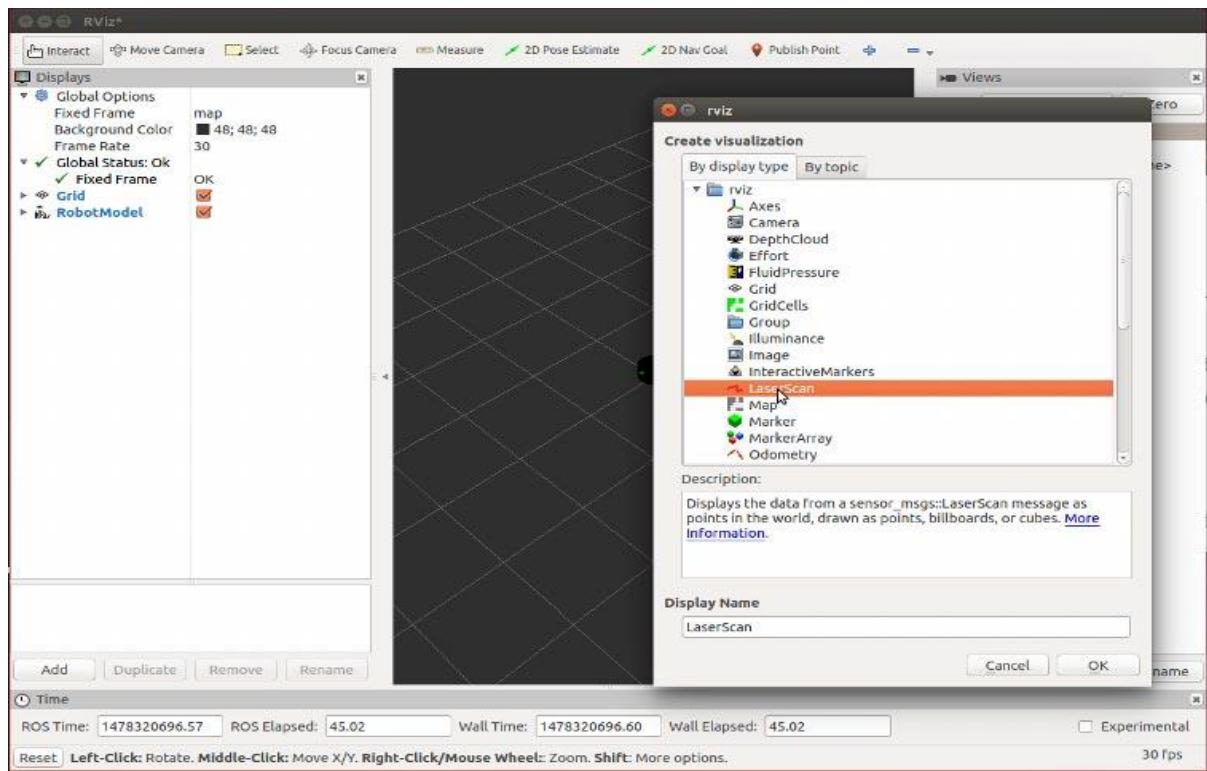
在弹出的窗口，选择 RobotModel，点击 OK，添加 D1 在 rviz 中的显示控件



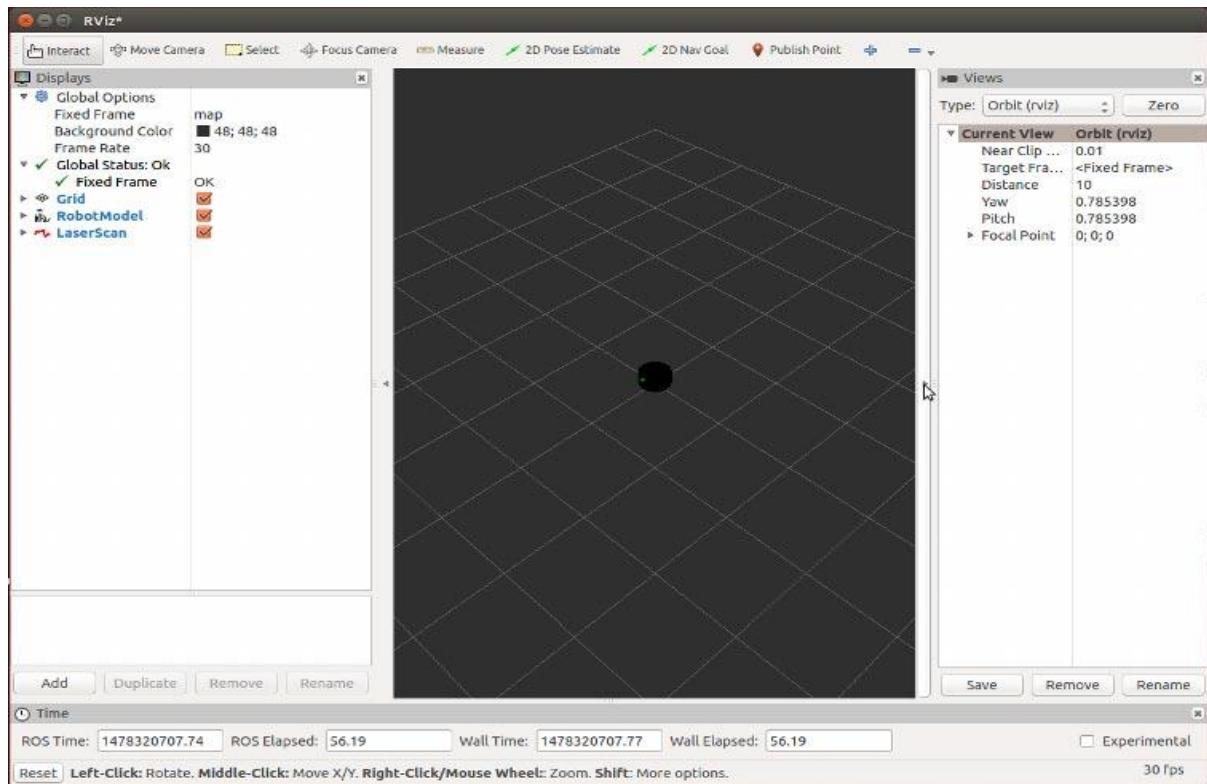
D1 显示控件添加成功后，如下图所示（图中那淡绿色的点便 D1 的正前方）



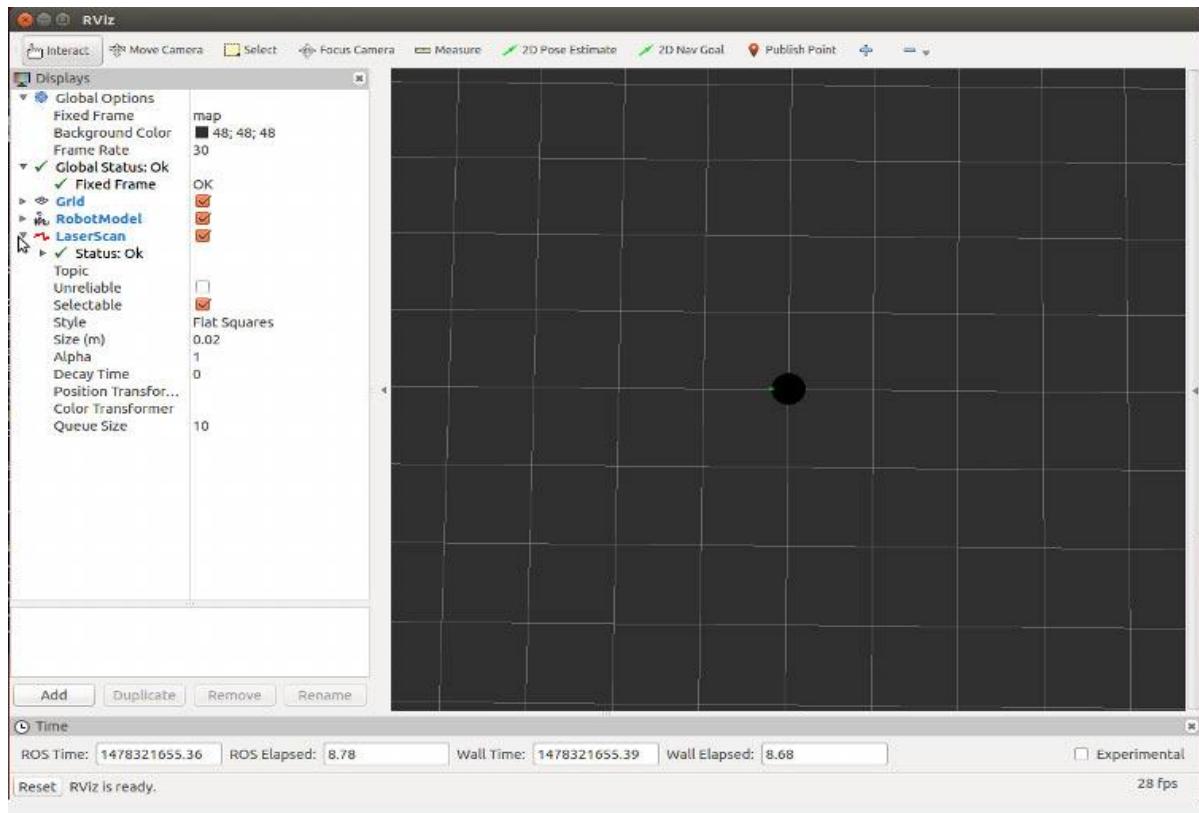
然后，再点击 Add，选择 LaserScan，点击 OK，添加激光雷达 F4 在 rviz 中的显示图层



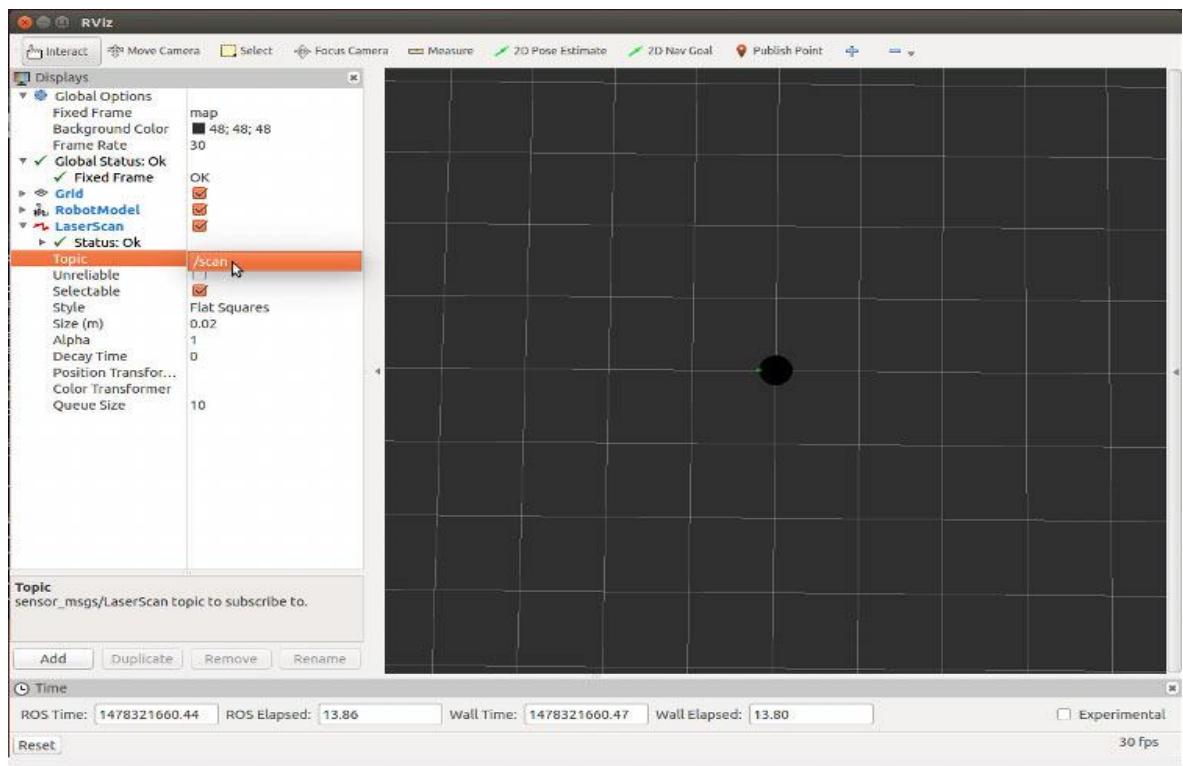
添加 激光雷达 F4 显示图层后，显示并没有变化，原因在于还没有选择要订阅的 topic



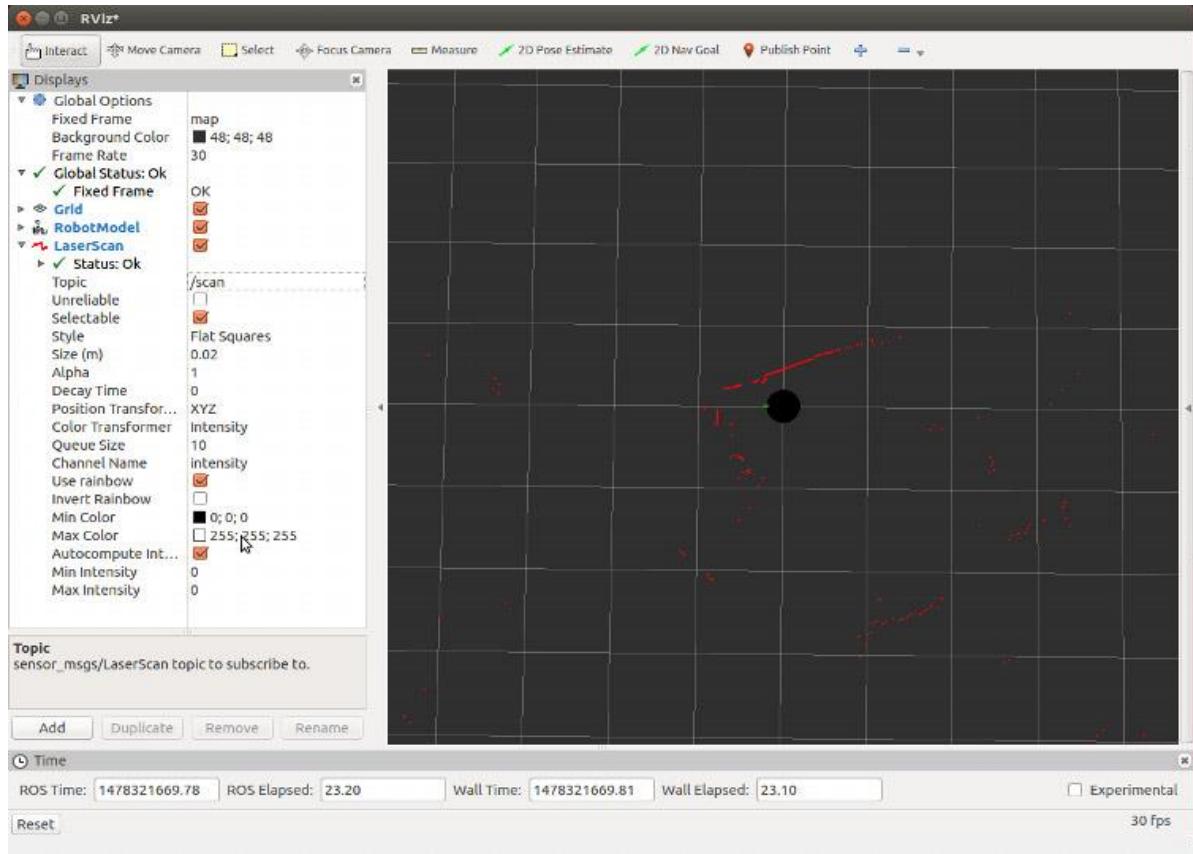
## 展开 LaserScan 的菜单



选择 Topic 项，选择要订阅的 topic 为 /scan，再回车



此时，F4 的扫描数据便可在 rviz 上显示，如下图所示



### 3. 6. 3 参数调整

F4 与 D1 的坐标校对，是通过对 launch 文件中的参数进行修改来使两者坐标一致的。

要运行哪个 launch 文件就要对哪个 launch 文件中的参数进行修改，使两者坐标一致的。

在本教程中，便是要修改 gmapping\_demo.launch 文件

运行命令，找到 gmapping\_demo.launch 文件所在的位置

```
$ roscd dashgo_nav/launch
```

使用 vim 编辑命令编辑 gmapping\_demo.launch

```
$ vim gmapping_demo.launch
```

```
<launch>
<node name="arduino" pkg="dashgo_bringup" type="dashgo_driver.py" output="screen">
```

```
<rosparam file="$(find dashgo_bringup)/config/my_dashgo_params.yaml" command="load">
</node>

<node name="flashgo" pkg="flashgo" type="flashgo_node" output="screen">
<param name="serial_port" type="string" value="/dev/flashlidar"/>
<param name="serial_baudrate" type="int" value="230400"/>
<param name="frame_id" type="string" value="laser_frame"/>
<param name="inverted" type="bool" value="false"/>
<param name="angle_compensate" type="bool" value="true"/>
</node>

<include file="$(find dashgo_description)/launch/dashgo_description.launch"/>
<node pkg="tf" type="static_transform_publisher" name="base_link_to_laser4"
args="0.0 0.0 0.2 0.0 3.1415926 0.0 /base_link /laser_frame 40" //修改 args 的参数值

<include file="$(find dashgo_nav)/launch/gmapping.launch"/>
<include file="$(find dashgo_nav)/launch/move_base.launch"/>
</launch>
```

args="0.0 0.0 0.2 0.0 3.1415926 0.0 /base\_link /laser\_frame 40"

在 /base\_link 前面的 6 个参数，只需要调整前面 5 个参数，最后一个默认为 0.0 即可，其他参数分别表示

- 这 5 个参数都是相对于 D1 的坐标系来调整的。
- 前面 3 个，分别表示激光雷达 F4 在 X、Y、Z 轴（右手定则）上距离 (0,0,0) 点的坐标位置，(0,0,0) 点是 D1 的坐标系原点，该点是 D1 的重心点。
- 后面 2 个，分别表示沿着 D1 中心线（正前方与正后方连成的直线）左右方向、上下方向移动偏离的角度，大小范围为 -3.1415926 ~ 3.1415926，-3.1415926 为-180 度，3.1415926 为 180 度。

## 3. 6. 4 vim(文本编辑命令) 的安装与基本使用

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. 6. 5 前后方向校正，只需修改第 4 个参数

D1 的前方是有三个超声波模块的，中间的超声波模块就是 D1 的正前方。

D1 的后方是单独一个超声波模块的，也是 D1 的正后方。

D1 的前后方向了解之后，便调整 F4 的坐标方向与 D1 的一致即可。

一般来说，F4 已经固定在 D1 已设定的位置上，而且 F4 在不停旋转，便要借用较大块、较平整的挡板，如：大纸箱，来确定 F4 当前的方位。

先将挡板横摆在 D1 正前方，挡板中心与 D1 正前方那个超声波模块成一条直线，挡板的面要与直线成 90 度角。

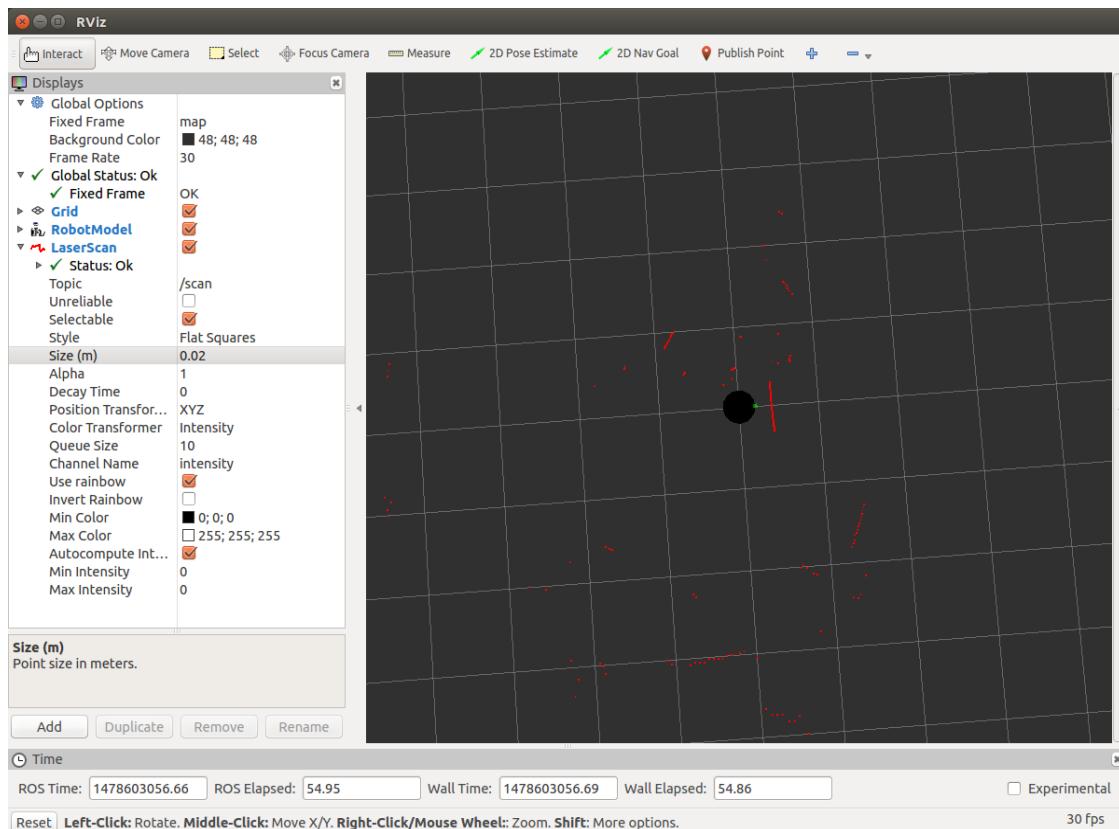
然后，对照 rviz 中 F4 的扫描显示，D1 正前方是否横画着一条红线障碍物。

若有，则说明前方坐标正确，再将挡板横摆在 D1 正后方，若 rviz 中同样在 D1 正后方横画着一条红线障碍物，则 D1 前后方向已经正确。

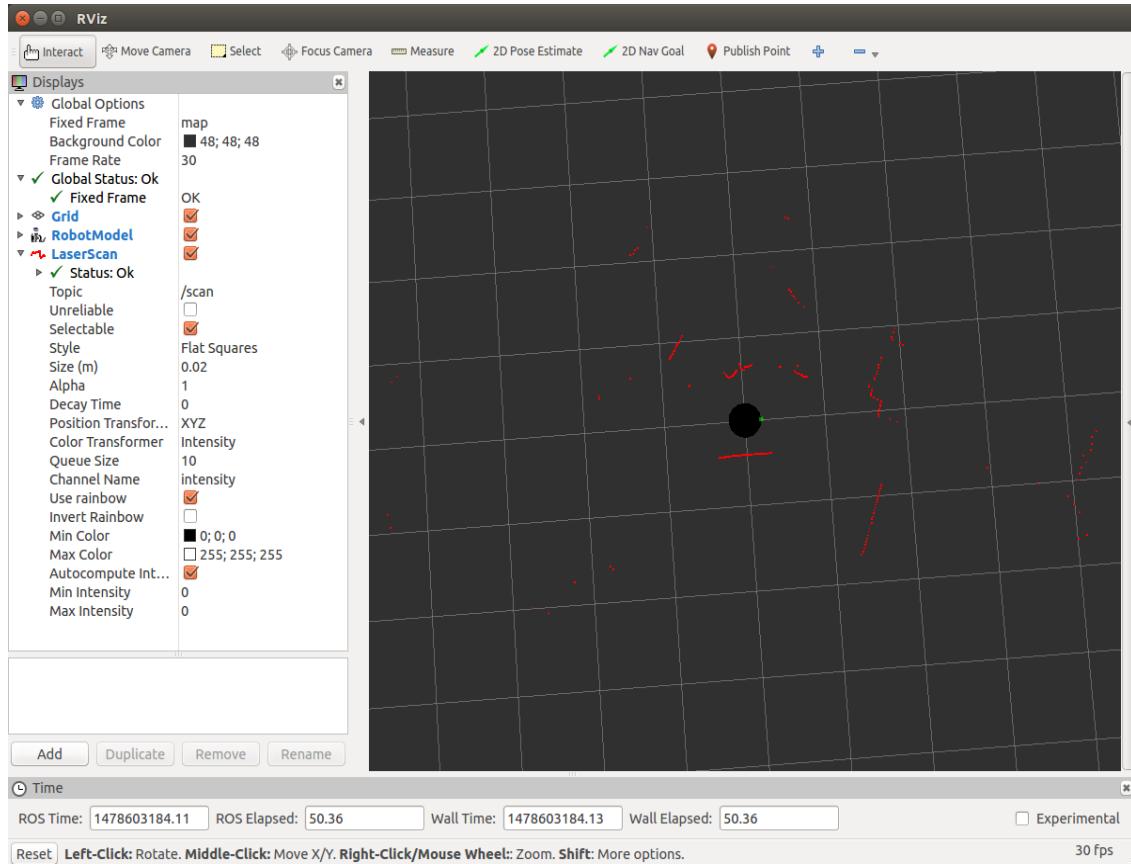
若实际中挡板的位置与 rviz 显示的位置不一致，则根据对前面 参数调整 的了解，相对应地对参数进行调整。

具体操作，如下：

挡板横摆在 D1 正前方，rviz 显示应该如下图：



但 rviz 实际显示，如下图所示：



Ctrl + C , 关闭 rviz , 也关闭正在运行的 launch 文件，打开该 launch 文件

先将第 4 个参数改大（注意数值范围），保存 launch 文件，接着启动 launch 文件并  
打开 rviz，查看效果

若显示的红线向正确的红线位置靠近，则再将参数慢慢加大以修正；若偏离正确位置，  
则将参数改小来修正。

正前方的位置校正后，将挡板横摆在 D1 正后方，rviz 显示与实际情况是相符的。

### 3. 6. 6 左右方向校正，只需要修改第 5 个参数

先将挡板横摆在 D1 正左方，挡板中心与 D1 中心成一条直线，挡板的面要平行于  
D1 正前方与正后方所成的直线

然后，对照 rviz 中 F4 的扫描显示，D1 正左方是否横画着一条红线障碍物

若有，则说明左方坐标正确，再将挡板横摆在 D1 正右方，若 rviz 中同样在 D1 正  
右方横画着一条红线障碍物，则 D1 左右方向已经正确，不用修改参数。

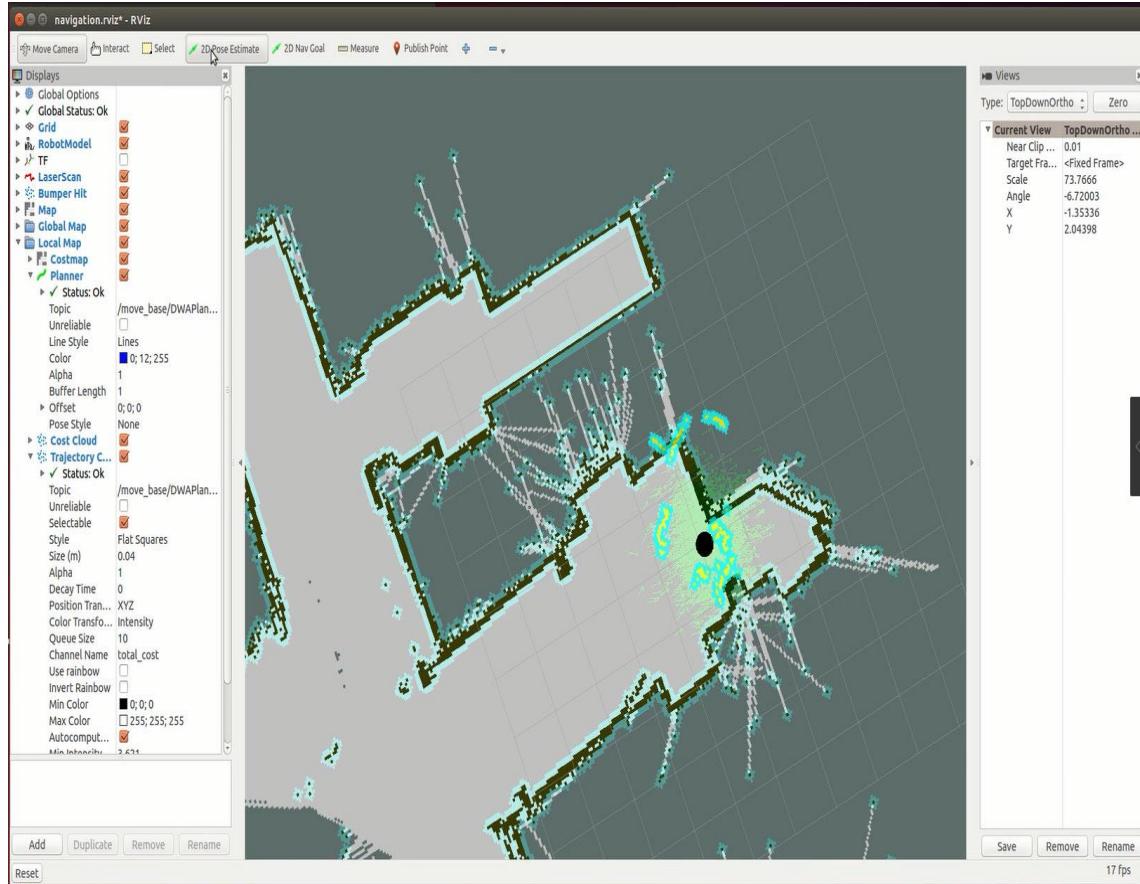
若实际中挡板的位置与 rviz 显示的位置不一致，则修改第 5 个参数，若当前数值为 0.0 ，则修改成 3.1415926 ；若当前数值为 3.1415926 ，则修改成 0.0 即可。

再运行 launch 文件与 rviz 核对效果。

**另一种方式：** Ctrl + Alt + T 打开终端，通过下面命令运行体量较大的 rviz

```
$ rosrun turtlebot_rviz_launchers view_navigation.launch
```

运行 rviz 显示的界面如下：



具体参数调整操作与前面操作一样。

## 3.7 D1 通过 ROS 建图

此节需要用到激光雷达 F4。

### 3. 7. 1 运行扫地图主程序

```
$ rosrun dashgo_nav gmapping_demo.launch
```

运行成功，如下面显示：

```
pi@dashgo-d1-web:~ $ rosrun dashgo_nav gmapping_demo.launch
... logging to
/home/pi/.ros/log/a05f59f0-a569-11e6-9a1a-b827eb8dfc88/rosrun-dashgo-d1-web-10624.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 rosrun server http://dashgo-d1-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!
```

若运行过程中，出现红色信息，说明运行出错了，可根据错误提示来寻找原因。

一般原因为下位机 arduino 连接不上，arduino 固件受损，激光雷达连接不上，激光雷达内部出错等等。

### 3.7.2 PC 端运行图形界面

```
$ roslaunch turtlebot_rviz_launchers view_navigation.launch
```

前面 launch 文件正常启动的情况下，打开的 rviz 只有 D1 控件并没有其他信息显示的情况下，可能是激光雷达没有正常启动。

判断激光雷达是否正常启动，主要看 launch 文件的启动信息有没有出现 Flash Lidar health status : 0 ，有这行信息则正常。

```
setting /run_id to a05f59f0-a569-11e6-9a1a-b827eb8dfc88  
process[rosout-1]: started with pid [10653]  
started core service [/rosout]  
process[arduino-2]: started with pid [10671]  
process[flashgo-3]: started with pid [10672]  
process[robot_state_publisher-4]: started with pid [10673]  
process[joint_state_publisher-5]: started with pid [10685]
```

```
Flash Lidar health status : 0
process[base_link_to_laser4-6]: started with pid [10707]
process[slam_gmapping-7]: started with pid [10720]
process[move_base-8]: started with pid [10732]
process[world_canvas_server-9]: started with pid [10758]
process[robot_pose_publisher-10]: started with pid [10768]
process[twist_marker_server-11]: started with pid [10780]
process[rosbridge_websocket-12]: started with pid [10799]
```

**注意：在建图前要先对激光雷达 F4 与 D1 的坐标进行对齐，请看“激光雷达 F4 与 D1 的坐标校正”这章节。**

### 3.7.3 控制 D1 移动扫描地图

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：



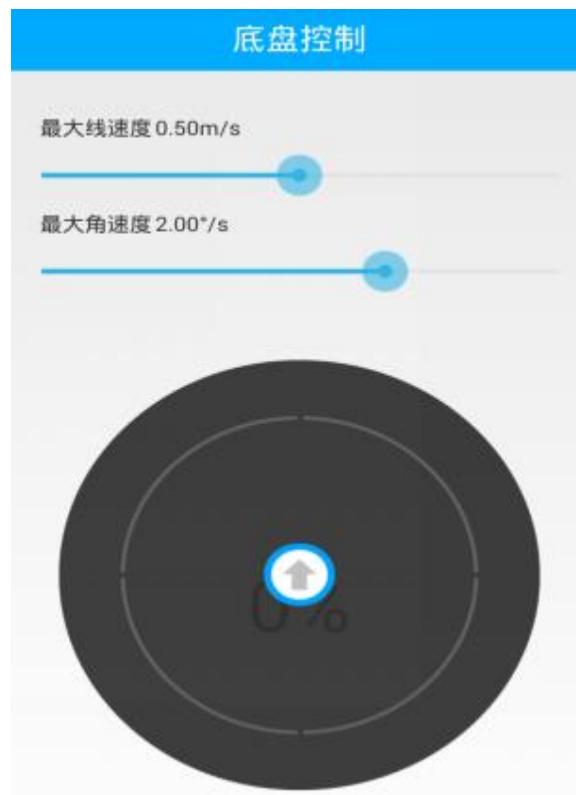
需要输入的 Master IP 是 PC 的 IP 地址。

查看 IP 地址，Ctrl + Alt + T 打开终端，输入 ifconfig，然后回车

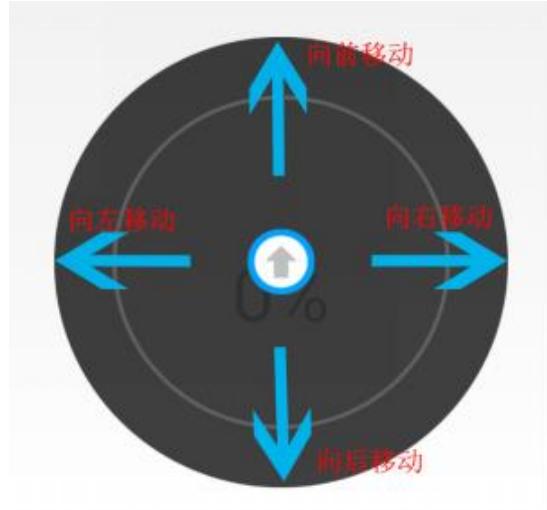
```
pi@eaibot:~$ ifconfig
eth0 Link encap:以太网 硬件地址 44:37:e6:3d:e0:fc
inet 地址:192.168.11.100 广播:192.168.11.255 掩码:255.255.255.0 //有线网络的 IP 地址
inet6 地址: fe80::4637:e6ff:fe3d:e0fc/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST MTU:1500 跳点数:1
      接收数据包:350 错误:0 丢弃:0 过载:0 帧数:0
      发送数据包:134 错误:0 丢弃:0 过载:0 载波:0
      碰撞:0 发送队列长度:1000
      接收字节:27790 (27.7 KB) 发送字节:18843 (18.8 KB)
      中断:17
```

```
lo Link encap:本地环回  
inet 地址:127.0.0.1 掩码:255.0.0.0 //本地的 IP 地址  
inet6 地址: ::1/128 Scope:Host  
UP LOOPBACK RUNNING MTU:65536 跃点数:1  
接收数据包:167 错误:0 丢弃:0 过载:0 帧数:0  
发送数据包:167 错误:0 丢弃:0 过载:0 载波:0  
碰撞:0 发送队列长度:1  
接收字节:12903 (12.9 KB) 发送字节:12903 (12.9 KB)
```

连接成功后，界面如下：



方向的操控，如下图所示：



---

**注意：**

- 1、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。
  - 2、建议最大线速度为 0.2m/s，最大角速度为 0.5° /s，扫描出来的地图效果最佳。
- 

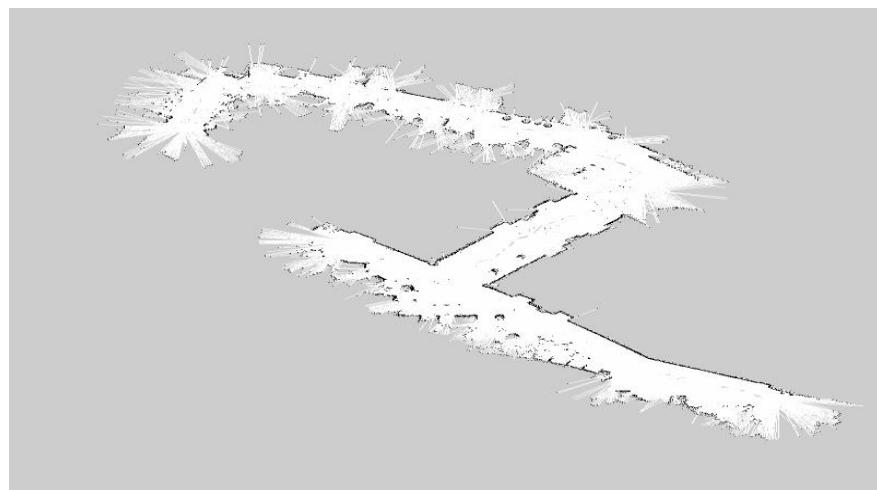
### 3.7.4 扫描完地图后，保存地图

Ctrl + Alt + T 打开新的终端，运行下面命令

```
$ roscd dashgo_nav/maps  
$ rosrun map_server map_saver -f my_map
```

**my\_map** 是保存地图的名称，根据实际情况命名。

保存的地图效果如下图：



## 3.8 D1 通过 ROS 实现自主导航

### 3.8.1 修改 launch 文件，引用已经保存好的地图

```
$ roscl dashgo_nav/launch
$ vim teb_amcl_demo.launch
```

```
<launch>
<node name="arduino" pkg="dashgo_bringup" type="dashgo_driver.py" output="screen">
<rosparam file="$(find dashgo_bringup)/config/my_dashgo_params.yaml" command="load"
/>
</node>

<node name="flashgo" pkg="flashgo" type="flashgo_node" output="screen">
<param name="serial_port" type="string" value="/dev/flashlidar"/>
<param name="serial_baudrate" type="int" value="230400"/>
<param name="frame_id" type="string" value="laser_frame"/>
<param name="inverted" type="bool" value="false"/>
<param name="angle_compensate" type="bool" value="true"/>
</node>
<include file="$(find dashgo_description)/launch/dashgo_description.launch"/>
<node pkg="tf" type="static_transform_publisher" name="base_link_to_laser4"
args="0.0 0.0 0.0 0.0 3.1415926 0.0 /base_link /laser_frame 40" />
<!-- Map server -->
<arg name="map_file" default="$(find dashgo_nav)/maps/my_pi_map.yaml"//修改
my_pi_map.yaml
<node name="map_server" pkg="map_server" type="map_server" args="$(arg map_file)"
/>

<arg name="initial_pose_x" default="0.0"<!-- Use 17.0 for willow's map in simulation --&gt;
&lt;arg name="initial_pose_y" default="0.0"<!-- Use 17.0 for willow's map in simulation --&gt;
&lt;arg name="initial_pose_a" default="0.0"/&gt;
&lt;include file="$(find dashgo_nav)/launch/amcl.launch.xml"&gt;
&lt;arg name="initial_pose_x" value="$(arg initial_pose_x)"/&gt;
&lt;arg name="initial_pose_y" value="$(arg initial_pose_y)"/&gt;
&lt;arg name="initial_pose_a" value="$(arg initial_pose_a)"/&gt;
&lt;/include&gt;
&lt;include file="$(find dashgo_nav)/launch/teb_move_base.launch"/&gt;</pre>

```

```
</launch>
```

将 my\_pi\_map 改成已经保存好的地图名称 my\_map

## 3.8.2 运行地图自主导航节点

Ctrl + Alt + T 打开命令窗口，运行命令

```
$ rosrun dashgo_nav tef_amcl_demo.launch
```

## 3.8.3 运行 rviz 图形界面

Ctrl + Alt + T 打开新的命令窗口，运行命令

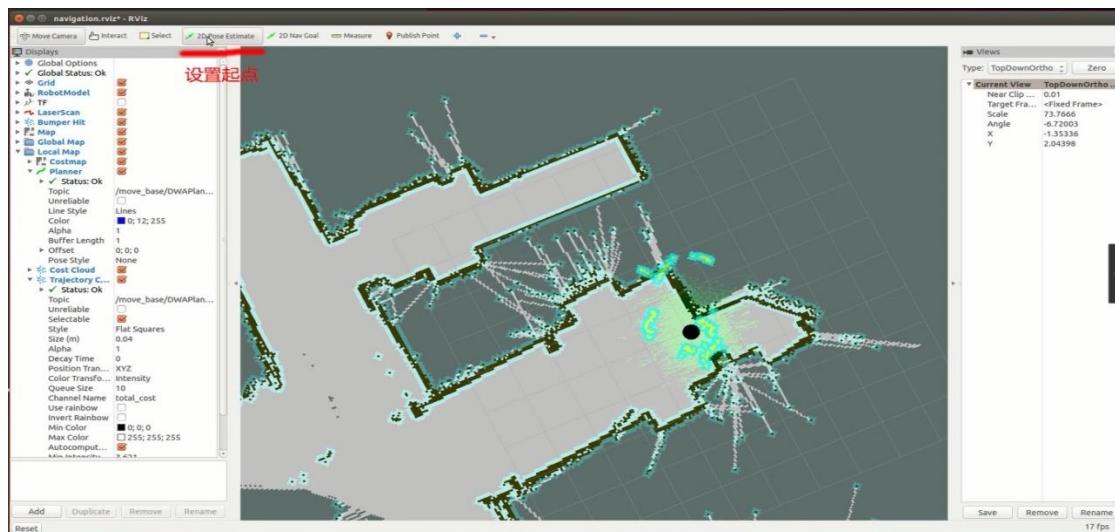
```
$ rosrun turtlebot_rviz_launchers view_navigation.launch
```

## 3.8.4 设置起点

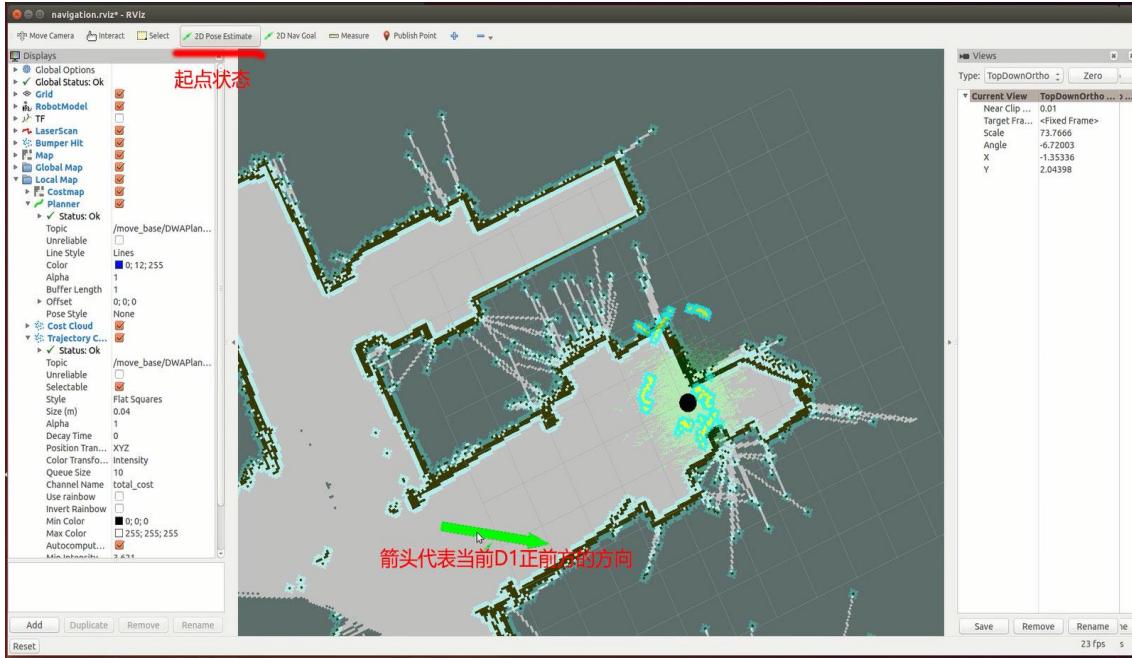
rviz 打开后显示 D1 默认所在的位置是栅格的中心点，不一定是 D1 实际所在的位置

每次打开 rviz 都要设置起点

点击 2D Pose Estimate

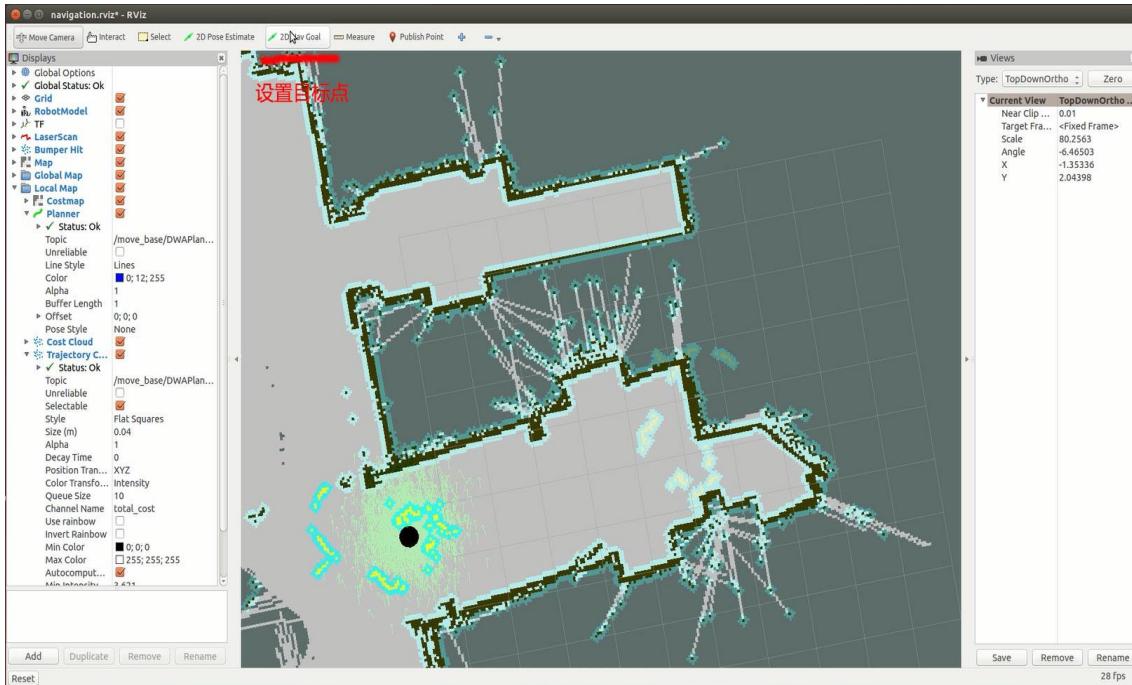


根据当前 D1 实际位置，在地图上选择正确的位置，并调整好 D1 的正前方

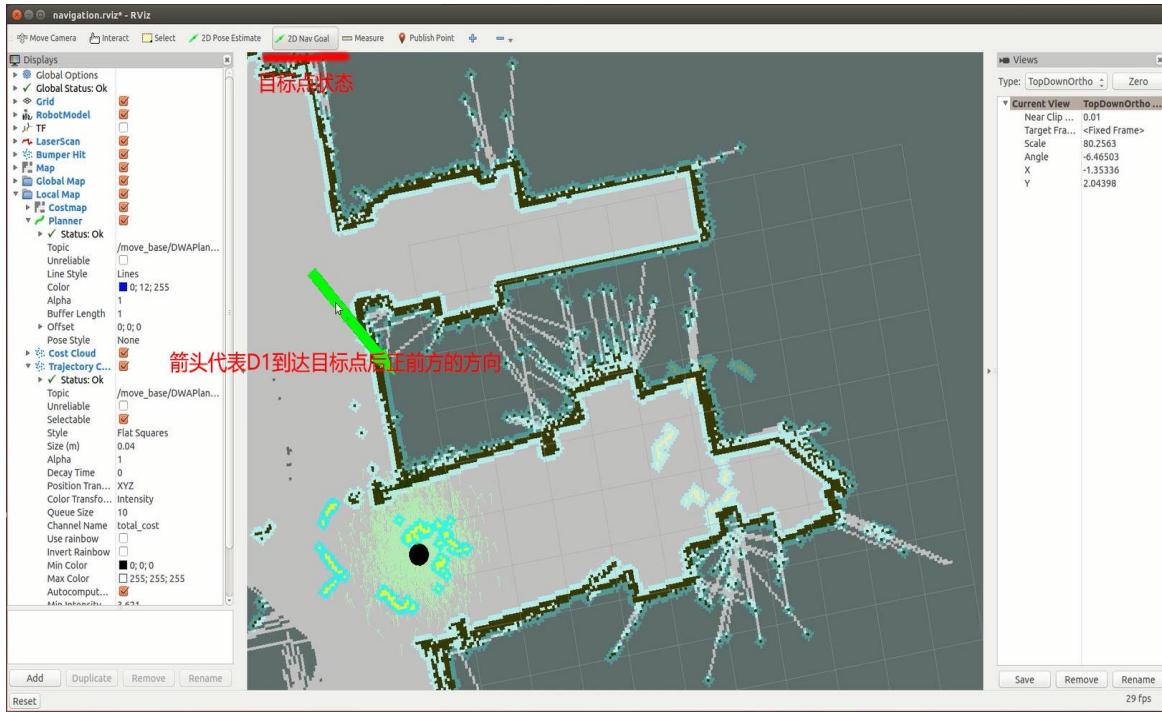


### 3.8.5 设置目标点

点击 2D Nav Goal



在地图上选择要到达的目标点位置，并调整好 D1 停止时的正前方方向



设置好目标点后，Dashgo 便自主计算路径并控制 D1 向目标点移动