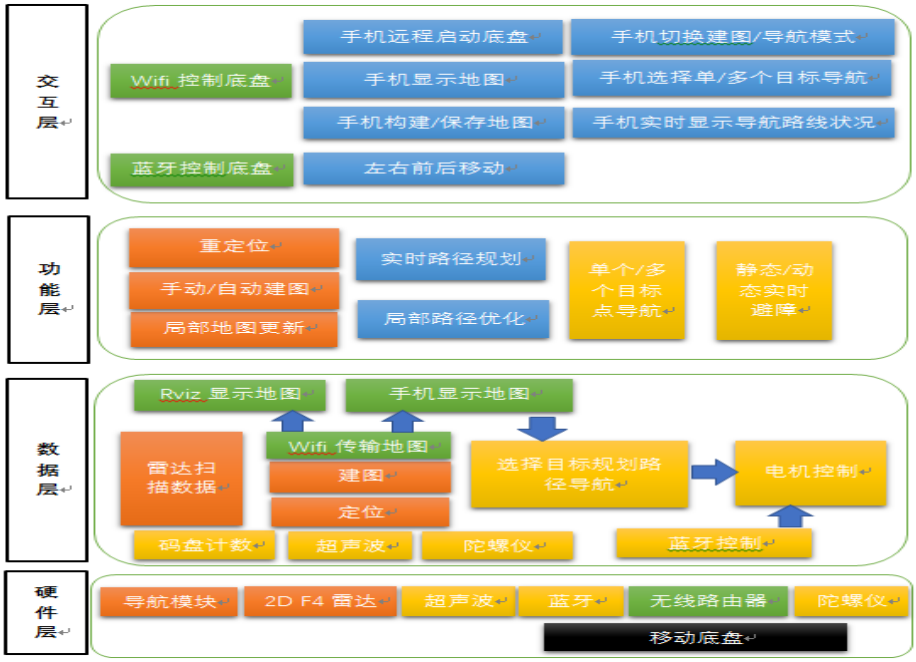
B1 移动平台调试开发指南

目录

一、B1 功能介绍

1.1 B1 系统框架介绍



1.2 B1 导航软件功能介绍

功能	支持功能	功能说明
建图功能	Gmapping 算法建图	适合一般室内环境建图（150*150 平方米以内）
导航功能	双雷达导航	增加安全性，有效避开矮的障碍物，防止压脚
	超声波导航避障	增加安全性，有效避开玻璃等透明障碍物
	深度摄像头导航避障	增加安全性，达到 3 维避障效果
	融合陀螺仪	提高导航精度，减小累计误差
	单点导航，多点巡逻导航	多种导航方式，提高场景适应性
客户端 APP	通过（网络）wifi 启动底盘建图，并显示地图，保存地图，然后导航	直接在 APP 上完成建图，导航使用，提高适用性，降低使用门槛
	通过 APP 查看底盘各种	

	传感器状态和日志信息	
	自动回充功能	增加智能性，有效续航，方便长时间使用

二、B1 导航系统主要参数详解

2.1B1 底盘基础参数介绍及调试方法

底盘基础参数主要是底盘的轮子直径，两轮子的间距，编码器值以及底盘移动速度控制，具体如下：

参数文件路径：dashgo_ws/src/dashgo/dashgo_driver/config/my_dashgo_params_imu.yaml		
参数名	目前值	说明
wheel_diameter	0.124	轮子直径，固定
wheel_track	0.31	两个轮子的间距，直接测量得到大概值，然后再通过转 360 度试验进行细微调整，出厂时会把参数调好，直接使用即可。
encoder_resolution	610	轮子转动一圈编码器输出的脉冲数，固定
gear_reduction	1.0	校准系数，主要用于校准走 1m 直线

通过测试底盘走 1m 直线，360 度转。

测试一：前进 1 米

远程进入导航模块，启动底盘驱动(带陀螺仪)，

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

然后远程进入导航模块另一个终端，启动移动脚本，

```
$ ssh eaibot@192.168.31.200
$ rosrun dashgo_tools check_linear_imu.py
```

测试完后，ctl+c 结束两个终端的程序。

测试二：原地转动 360 度

远程进入导航模块，启动底盘驱动(带陀螺仪)，

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

然后远程进入导航模块另一个终端，启动转动脚本。

```
$ ssh eaibot@192.168.31.200
$ rosrun dashgo_tools check_angular_imu.py
```

底盘参数调试方法

情况一:走 1m 直线和 360 度旋转参数调试情况

优先校准走 1m 直线，仅需修改 my_dashgo_params.yaml 文件中的 gear_reduction 参数，其他参数基本给定，超过 1m 时，就改小，否则就改大，误差控制在 1%左右。

在已校准 1m 直线的前提下，校准 360 度旋转，仅需细微修改两个轮子间距 wheel_track，转超 360 度就改小，否则就改大，误差控制在 1%左右。

如果底盘走成了斜直线，一般都是底盘摆放时，两轮子不在同一水平线引起的。Eai 底盘在出厂时，都会测试确认底盘能正常走直线。

如果底盘走 S 型（无法走直线），先确认底盘是否电量充足，若电量不足，无法拉动电机正常转动，就会行走异常，若充足但行走异常请找 eai 售后

情况二：提高/限制底盘移动速度

启动 driver.launch 或 driver_imu.launch 来驱动底盘行走时，默认都是有平缓控制速度的功能，所以主要是修改 yocs_velocity_smoother.yaml 配置文件中的最大线速度 speed_lim_v 和最大角速度 speed_lim_w 来控制底盘行走的。

如果是在启动导航 navigation 时，此时底盘行走速度不单受平缓速度 yocs_velocity_smoother.yaml 的限制，还会受到局部路径规划 teb_local_planner_params.yaml 中的最大线速度限制，最终会取两者中最小的线速度。这点会在导航章节中详述。

如果是只用手机蓝牙来控制底盘时，速度是不受平缓控制的，所以会有急停（点头）和急速前冲（抬头）的现象。

底盘关键数据观察

以使用三一键盘控制底盘行走为例，主要观察底盘的里程计信息，线速度，角速度信息。具体如下：

odom—不带陀螺仪的里程计信息

.在启动 driver.launch 的情况下，在导航模块的另一个终端中输入指令，

```
rostopic echo /odom
```

odom_combined，imu，imu_angle—陀螺仪和带陀螺仪的里程计信息

在启动 driver_imu.launch 的情况下，会把/odom 的信息与陀螺仪/imu 的信息融合后得到新的里程计信息，并发出来给建图导航使用，在导航模块的另一个终端中分别输入指令。

```
rostopic echo /robot_pose_ekf/odom_combined    #查看带陀螺仪里程计信息
rostopic echo /imu
rostopic echo /imu_angle  #查看陀螺仪角度变化信息
```

/smoother_cmd_vel—经过平缓处理的底盘速度

在启动 driver.launch 或 driver_imu.launch 驱动底盘的情况下，底盘的最原始的速度信息是在/cmd_vel 中，经过平缓处理后，发布到新的主题/smoother_cmd_vel，建图导航等默认都使用经过平缓处理后的速度，在导航模块的另一个终端中分别输入指令。

```
rostopic echo /cmd_vel  #底盘原始的速度信息
rostopic echo /smoother_cmd_vel  #经过平缓处理后的速度，默认底盘驱动，建图，导航等都是用这里的线速度和角速度，然后在 dashgo_driver.py 中把线速度和角速度转换成点击的 pwd 值发给底盘从而控制底盘行走
```

2.2 B1 底盘雷达安装及参数调试

B1 雷达安装情况

B1 是倒装 F4 雷达，如下图所示，雷达 0 刻度超正前方，最大扫描角度为 360 度，且把扫描范围内的 6 根小柱子所在角度的数据剔除掉。雷达的具体参数，性能及单独使用请参照《雷达使用手册》

雷达具体参数介绍

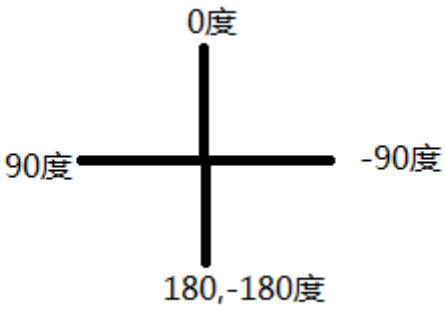
参数文件路径：dashgo_ws/src/dashgo/ydlidar-1.2.0/launch/ydlidar1_up.launch 1.3.1 为雷达驱动版本号，有可能变化, ydlidar1_up.launch 为上雷达启动文件，ydlidar2_down.launch 为下雷达启动文件。
`<node name="ydlidar_node" pkg="ydlidar" type="ydlidar_node" output="screen">`
雷达启动的节点名为 ydlidar_node
`<param name="port" type="string" value="/dev/port2"/>`
雷达与导航模块连接的串口，为 port2
`<param name="baudrate" type="int" value="115200"/>`
G4 雷达串口波特率 230400，如果是 F4 雷达则为 115200，X4 雷达为 128000
`<param name="angle_min" type="double" value="-180" />` `<param name="angle_max" type="double" value="180" />`
雷达扫描角度范围为-180~180 度，雷达角度范围设置具体参照下面说明
`<param name="range_min" type="double" value="0.08" />` `<param name="range_max" type="double" value="16.0" />`
雷达扫描距离范围为 0.08~16 m
`<param name="ignore_array" type="string" value="-163,-145,-107,-97,90,100,130,155,155,175" />`
雷达剔除的扫描范围，即不取该范围的数据，它与上面扫描角度参数结合，得到最终雷达有效的扫描角度范围
`<node pkg="tf" type="static_transform_publisher" name="base_link_to_laser4" args="0.215 0.0 0.28 3.053.14 0.0 /base_footprint /laser_frame 40" />`
这是倒装雷达，与 B1 底盘的 tf 转换参数，并剔除了 6 根柱子
参数文件路径：dashgo_ws/src/dashgo/dashgo_tools/conf/box_filter.yaml 其中 box_filter.yaml 表示上雷达安全范围，box_filter_2.yaml 为下雷达安全范围
max_x: 0.5 安全范围在 x 轴上离底盘重心最大距离 max_y: 0.22 安全范围在 y 轴左边离底盘重心距离

```
max_z: 0.5 暂时无用
min_x: 0.22 安全范围在 x 轴上离底盘重心最小距离
min_y: -0.22 安全范围在 y 轴右边离底盘重心距离
min_z: 0.05 暂时无用
参数具体意义见下面情况三详解
```

雷达参数调试的方法

情况一：设置雷达的扫描角度

设置雷达的扫描角度并剔除在扫描范围特定角度的数据（如只取雷达前方 270 度数据或者剔除扫描范围内的柱子等物体），雷达的数据获取符合右手定则（与雷达的转动方向没直接关系），具体如下图所示：



如果雷达只想扫描正前方 270 度，则需要把 ydlidar1_up.launch 的参数设置如下（以上雷达为例）：

```
<param name="angle_min"    type="double" value="-180" />
<param name="angle_max"    type="double" value="180" />
<param name="ignore_array" type="string" value="-163,-145,-107,-97,90,100,130,155,155,175"
/>
```

注意：雷达的扫描角度不能小于 180 度，否则会影响建图，导航避障等功能。

情况二：雷达坐标系与底盘坐标系的 tf 转换关系设置

该参数主要是在整套移动系统在建图导航前，进行雷达校准用到，单独雷达不需要用到此参数。

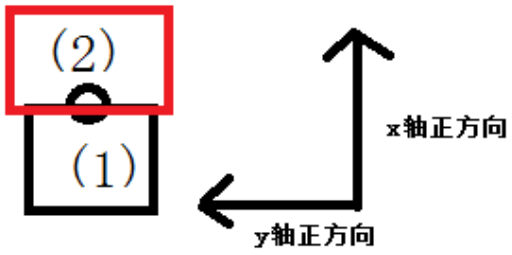
以上雷达为例，雷达正装，则 ydlidar1_up.launch 参数设置如下：（一般出厂时雷达参数都会设置好，可直接使用，但若移动，拆装后需要自己细微调整）

```
<node pkg="tf" type="static_transform_publisher" name="base_link_to_laser4"
args="0.215 0.0 0.28 3.05 3.14  0.0 /base_footprint /laser_frame 40" />
```

- args 第一个参数 0.2245 表示雷达中心距离底盘重心的 x 轴距离
- args 第二个参数 0.0 表示雷达中心距离底盘重心的 y 轴距离
- args 第二个参数 0.28 表示雷达中心距离底盘重心的 z 轴距离，该参数为虚拟的，不能改，因为会影响到导航的 costmap，（因为 G4 雷达为 2 维雷达，z 轴参数对雷达数据没影响，所以可以使用虚拟）
- args 第四个参数表示将雷达绕 z 轴左右偏转程度，为 yaw 偏航角
- args 第五个参数表示将雷达绕 y 轴前后翻滚程度，为 pitch 俯仰角
- args 第六个参数表示将雷达绕 x 轴左右侧滚，为 roll 侧滚角，该参数一般为 0.0，目前只能设为 0.0，-3.14 和 3.14

情况三：雷达滤波安全范围设置（仅在导航时使用）

如下图所示，雷达滤波安全范围是指，在雷达前方，画一个安全区域，一旦雷达突然发现前方有障碍物出现在安全区域内（例如底盘导航时，突然伸脚到底盘前面很近的地方），此时底盘优先停下然后再重新规划路径绕开，它认为离突然出现的障碍物太近，再往前就会撞到障碍物，这样可以有效防止底盘减速刹车不及时撞到障碍物的问题。



如图所示，红色部分为安全范围，它为矩形，根据 box_filter.yaml 参数，以底盘重心为原点，正常安全范围（红色部分）x 轴长度在 15cm 左右，y 轴宽度比底盘宽 2cm（左右两侧各宽 1cm），注意：该安全范围不能过大，否则会影响导航效果（例如通过狭窄的地方）。

雷达关键数据观察

主要是观察/scan 雷达节点是否有数据。

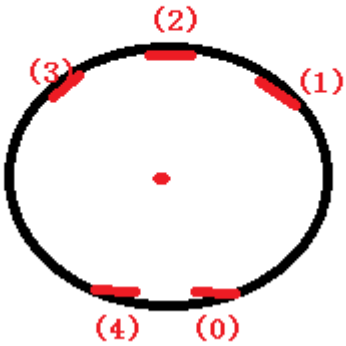
```
rostopic echo /scan
```

2.3 B1 超声波的安装及参数介绍

目前 B1 系统最多只支持 6 个超声波，安装在 stm32 控制板上（电机控制板），stm32 会实时把超声波数据传给导航模块，然后和导航避障算法进行融合避障。

超声波的位置安装（目前默认位置）

B1 超声波默认的安装情况如下：



0 号在正后方右侧，离中心坐标和偏角为（-0.23,-0.05），偏角为-3.14 弧度

1 号在前方右侧，离中心坐标和偏角为（0.23,-0.23），偏角为-0.785 弧度

2 号在正前方，离中心坐标和偏角为（0.23,0.0），偏角为 0 弧度

3 号在前方左侧，离中心坐标和偏角为（0.23,0.23），偏角为 0.785 弧度

4 号正后方左侧，离中心坐标和偏角为（-0.23,0.05），偏角为 3.14 弧度

超声波参数介绍

参数文件路径：dashgo_ws/src/dashgo/dashgo_driver/config/my_dashgo_params_imu.yaml
主要是超声波功能开关，坐标位置及偏移角初始化。

```
useSonar: True
```

超声波功能开关，True 表示打开，False 表示关闭。

```
sonar0_offset_yaw: 3.14   5 个超声波的位置和偏移角初始化
sonar0_offset_x: -0.091
sonar0_offset_y: -0.11

sonar1_offset_yaw: -0.7
sonar1_offset_x: 0.265
sonar1_offset_y: -0.128

sonar2_offset_yaw: 0.0
sonar2_offset_x: 0.27
sonar2_offset_y: 0.0
```

```
sonar3_offset_yaw: 0.7
sonar3_offset_x: 0.265
sonar3_offset_y: 0.128

sonar4_offset_yaw: 3.14
sonar4_offset_x: -0.091
sonar4_offset_y: 0.11
```

参数文件路径：dashgo_ws/src/dashgo/dashgo_driver/launch/driver_imu.launch

```
<node pkg="tf" type="static_transform_publisher" name="base_link_to_sonar0"
args="-0.091 -0.11  0.15 3.1416 0.0  0.0 /base_footprint /sonar0 40" />
<node pkg="tf" type="static_transform_publisher" name="base_link_to_sonar1"
args="0.265 -0.128 0.15 -0.70 0.0 0.0 /base_footprint /sonar1 40" />
<node pkg="tf" type="static_transform_publisher" name="base_link_to_sonar2"
args="0.27 0.0  0.15 0.0 0.0 0.0 /base_footprint /sonar2 40" />
<node pkg="tf" type="static_transform_publisher" name="base_link_to_sonar3"
args="0.265 0.128 0.15 0.70 0.0 0.0 /base_footprint /sonar3 40" />
<node pkg="tf" type="static_transform_publisher" name="base_link_to_sonar4"
args="-0.091 0.11  0.15 3.1416 0.0  0.0 /base_footprint /sonar4 40" />
```

这分别是超声波与底盘的坐标 tf 转换关系，同时会把相应位置显示到 rviz 上，如果 rviz 上看到超声波位置不对，请查看此参数是否正确。

超声波参数调试方法

步骤一:要使用超声波避障功能，必须先打开超声波功能开关。

```
useSonar: True
```

超声波功能开关，True 表示打开，False 表示关闭。

步骤 2:运行导航 launch,观察超声波数据变化，验证每一个超声波工作正常具体如下：

在导航模块中运行导航 launch（以不带陀螺仪单点导航为例）；

```
roslaunch dashgo_nav navigation_imu.launch
```

在导航模块另一个终端中，分别监听每一个超声波主题，如下与 0 号超声波为例；

```
rostopic echo /sonar0
```

然后再 0 号超声波前面放一个障碍物并来回移动，观察 0 号超声波数据变化是否正常，以此验证其他四个超声波是否都正常。

步骤 3：在 rviz 中添加超声波的显示，并观察超声波看到障碍物时，是否会停止。

保持运行导航 navigation.launch 程序，在电脑终端中运行 rviz 显示地图，add->by Topic 然后选择 sonar0 的 Range,点击 ok 就会把超声波的锥形范围显示在 rviz 中，最后 ctrl+s 保存 rviz 配置，类似地把其他超声波加入到 rviz 中，然后再把障碍物放到超声波前面（障碍物最好是玻璃等透明物体，只有雷达才可以看到），观察导航时是否避开它。

超声波关键数据观察

主要是观察/sonar0~4 超声波节点是否有数据。

```
rostopic echo /sonar0  观察 0 号超声波数据，默认情况只取 0.8m 以内的有效数据
rostopic echo /sonar1
rostopic echo /sonar2
rostopic echo /sonar3
rostopic echo /sonar4
```

2.4 建图与导航参数介绍及调试

参数文件路径：dashgo_ws/src/dashgo/dashgo_nav/launch/include/imu/gmapping_base.launch Gmapping 扫图算法参数
<param name="maxUrange" value="8.0"/> <param name="maxRange" value="8.0"/> 雷达最远扫描距离设置，正常 G4 能扫到 16m，由于越远激光数据点越少且不稳定，因此只取 10m 内的数据
参数文件路径：dashgo_ws/src/dashgo/dashgo_nav/config/imu/teb_local_planner_params.yaml Teb 局部路径规划配置
max_vel_x: 0.30 #机器人导航时最大线速度，与 1.2.4 章节中情况三控制底盘平缓行走的参数一起控制底盘导航，最终取两者最小的线速度。 max_vel_x_backwards: 0.15 #机器人后退速度，不能改小 max_vel_theta: 0.35 #最大角速度 acc_lim_x: 0.15 #线加速度

acc_lim_theta: 0.25 #角加速度，不能过大，否则行走可能左右摆动
min_turning_radius: 0.0
footprint_model: # types: "point", "circular", "two_circles", "line", "polygon"
vertices: [[-0.085, -0.20], [-0.085, 0.2],[0.315, 0.2], [0.315, -0.2]]
#底盘模型
GoalTolerance

xy_goal_tolerance: 0.2 #导航里目标点最大距离误差为 20cm
yaw_goal_tolerance: 0.5 #最大角度误差为 0.5 *6=30 度
free_goal_vel: False

Obstacles

min_obstacle_dist: 0.22 #距离障碍物的最小距离

weight_kinematics_forward_drive: 100 #机器人前进的权重，增大时，机器人后退几率，后退距离都会减小，但不能过大，具体要根据实际情况调试。

参数文件路径：dashgo_ws/src/dashgo/dashgo_nav/config/imu/ move_base_params.yaml
Move_base 算法参数

planner_frequency: 1.0 #路径规划频率
oscillation_timeout: 5.0 #超时时间为 5.0*2=10s
oscillation_distance: 0.2 #如果在 10s（超时时间）内，机器人没有行走超过 0.2m，则认为机器人在来回挪动（震荡），此时取消该次导航

参数文件路径：dashgo_ws/src/dashgo/dashgo_nav/config/imu/ costmap_common_params.yaml
代价地图 costmap 基础参数

```
footprint: [[-0.085, -0.20], [-0.085, 0.2],[0.315, 0.2], [0.315, -0.2]]  #底盘的模型
obstacle_layer:  #动态层 costmap
  enabled: true
max_obstacle_height: 1.2   #costmap 的最大高度
min_obstacle_height: 0.0
obstacle_range: 2.0        #2m 内有障碍物就加入 costmap 中
raytrace_range: 5.0
inflation_radius: 0.30        #障碍物膨胀系数
combination_method: 1
 #非常重要，这里表明 costmap 是由传感器雷达 laser_scan_sensor，超声波
 #sonar_scan_sensor 数据组成，数据来源具体下面会介绍
observation_sources: laser_scan_sensorsonar_scan_sensor
track_unknown_space: true  #是否往未知区域规划路径

origin_z: 0.0   #costmap 高度从 0m 开始
z_resolution: 0.1  #costmap 立体分成，每一层为 0.1m
```

```
   z_voxels: 10     #costmap 立体一共分 10 层数据
   unknown_threshold: 15
   mark_threshold: 0
   publish_voxel_map: true
   footprint_clearing_enabled: true  #是否清楚底盘脚下的 costmap

   laser_scan_sensor:  #表明是从/scan 主题中获取雷达数据构成 costmap
   data_type: LaserScan  #雷达数据类型
       topic: /scan          #雷达数据主题
       marking: true
       clearing: true
   expected_update_rate: 0
   min_obstacle_height: 0.20  #雷达数据在 costmap 中的高度范围
   max_obstacle_height: 0.30

     laser_scan_sensor_2:  #第二个雷达（底下），预留双雷达功能使用
   data_type: LaserScan
       topic: /scan_2
       marking: true
       clearing: true
   expected_update_rate: 0
   min_obstacle_height: 0.01
   max_obstacle_height: 0.1

   sonar_scan_sensor:  #超声波点云数据
   data_type: PointCloud2
       topic: /sonar_cloudpoint
       marking: true
       clearing: true
   min_obstacle_height: 0.11  #超声波点云数据在 costmap 中高度范围
   max_obstacle_height: 0.2
   observation_persistence: 0.0

   inflation_layer: #静态层 costmap
     enabled:            true
   cost_scaling_factor:  10.0  # exponential rate at which the obstacle cost drops off (default:
10)
   inflation_radius:     0.30 # 障碍物膨胀系数
   static_layer:
     enabled:            true
   map_topic:          "/map"

   sonar_layer:    #超声波数据
     enabled:          true
```

```
clear_threshold:    0.2
mark_threshold:     0.8
  topics: ["/sonar0", "/sonar1", "/sonar2", "/sonar3", "/sonar4"]
clear_on_max_reading: true
```

建图与导航使用

步骤 1：在导航模块中，启动建图 launch

```
$ ssh eaibot@192.168.31.200   #远程进导航模块
$ roslaunch dashgo_nav gmapping_imu.launch
```

步骤 2：在电脑 ubuntu 系统中，启动 rviz 工具（注意该命令是在电脑上运行，而不是导航模块中，之后启动 rviz 的操作都是在电脑上，导航模块中没有安装 rviz 工具）

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

步骤 3：手机 app wifi 控制底盘行走（注意此时不能用蓝牙控制，会导致控制冲突）

步骤 4：建完地图后，保持建图程序运行，进行如下操作保存好地图

```
$ ssh eaibot@192.168.31.200        #远程进入导航模块
$ roscd dashgo_nav/maps            #进入地图目录
$ rosrun map_server map_saver -f eai_map_imu
#保存地图，名为 eai_map_imu,然后会在 maps 目录下生成 eai_map_imu.yaml 和 eai_map_imu.pgn 文件（即保存的地图为 pgn 格式），之后带陀螺仪导航时，默认会导入名为 eai_map_imu 的地图，
```

地图保存好后，ctl+c 关闭建图程序

步骤 5：在导航模块中，启动单点导航的 launch

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

步骤 6：在电脑 ubuntu 系统中，启动 rviz 工具

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

步骤 7：设置机器人起点位置，然后设置单个目标位置，开始导航

建图与导航参数调节的情况

情况一：修改障碍物膨胀系数，防止规划的路径贴近障碍物（沿边规划）

主要修改 costmap_common_params.yaml 文件中的 inflation_radius 参数，该文件中有两个这样的参数，必须同时该

```
inflation_radius: 0.30        #障碍物膨胀系数


obstacle cost drops off (default: 10)
inflation_radius:     0.30 # 障碍物膨胀系数
```

情况二：限制机器人导航行走的速度

主要修改局部路径规划 teb_local_planner_params.yaml 的参数，它与 1.2.4 章节情况三—平缓速度限制一起控制底盘，要想限制机器人行走速度，需要同时修改两个配置文件，最终会取两者中最小的线速度和角速度

```
# Robot
max_vel_x: 0.3 #机器人导航时最大线速度，与 1.2.4 章节中情况三控制底盘平缓行走的参数一起控制底盘导航，最终取两者最小的线速度。
max_vel_x_backwards: 0.15  #机器人后退速度
max_vel_theta: 0.5   #最大角速度
acc_lim_x: 0.15      #线加速度
acc_lim_theta: 0.25  #角加速度，不能过大，否则行走可能左右摆动
min_turning_radius: 0.0
```

情况三：限制机器人只能前进，不能后退

主要修改局部路径规划 teb_local_planner_params.yaml 中的机器人前进的权重参数，减小机器人后退几率和后退距离

```
weight_kinematics_forward_drive: 100
```

```
#机器人前进的权重，增大时，机器人后退几率，后退距离都会减小，但不能过大，否则导航起步时可能会停止不动，具体要根据实际情况调试。修改该参数还没能使底盘完全不会后退的情况（尤其是在转 180 度时有可能会稍微后退调整），后续会继续优化
```

2.5 建图与导航主要数据观察

在启动导航 launch 情况下（例如 roslaunch dashgo_nav navigation_imu.launch）,然后 rostopic list 列出所有的主题，如下分析常用关键的主题信息作用：

```
eaibot@PathGoE1:~$ rostopic list
/Lencoder   #左轮编码器值变化
/Lvel        #左轮速度
/Rencoder    #右轮编码器值
/Rvel        #右轮速度
/amcl_pose            #amcl 算法定位得到底盘所处的地图位置
/cmd_vel                #下发给机器人的线速度和角速度
/emergencybt_status      #急停开关状态主题，1——按下，0——未按下
/imu                #陀螺仪信息
/imu_angle           #陀螺仪的角度变化
/initialpose         #导航时，默认的起点位置和方向
/is_passed           #显示在雷达滤波安全范围内是否有障碍物，>1 表示有障碍物，底盘线速度设为 0，否则不影响导航
/is_passed_2         #显示在第二个（下雷达）滤波安全范围内是否有障碍物，只用在双雷达导航
/joint_states        #导航时，添加目标点是否成功状态反馈
#全局规划的路径，需要在 rviz 上才能直观地看到
/move_base/TebLocalPlannerROS/global_plan
#局部规划的路径，需要在 rviz 上才能直观地看到
/move_base/cancel              #取消当前导航
/move_base/current_goal     #当前导航要去的目标点坐标
/move_base/goal          #当前导航要去的目标点坐标
/move_base/result             #导航结果反馈
/move_base/status                  #导航实时状态反馈
/move_base_simple/goal         #获取在 rviz 上点击设置的目标点坐标及方向
/odom                              #里程计信息
/robot_cmd_vel
/robot_pose                        #机器人导航过程中的实时坐标及位姿信息
/robot_pose_ekf/odom_combined #融合陀螺仪后，新的里程计信息
/scan                              #雷达数据
/scan_2                                 #第二个雷达（下雷达）数据
/smoother_cmd_vel                  #经过平缓处理后，发给底盘的速度信息
/sonar0                                 #0~4 号超声波的数据
/sonar1
/sonar2
```

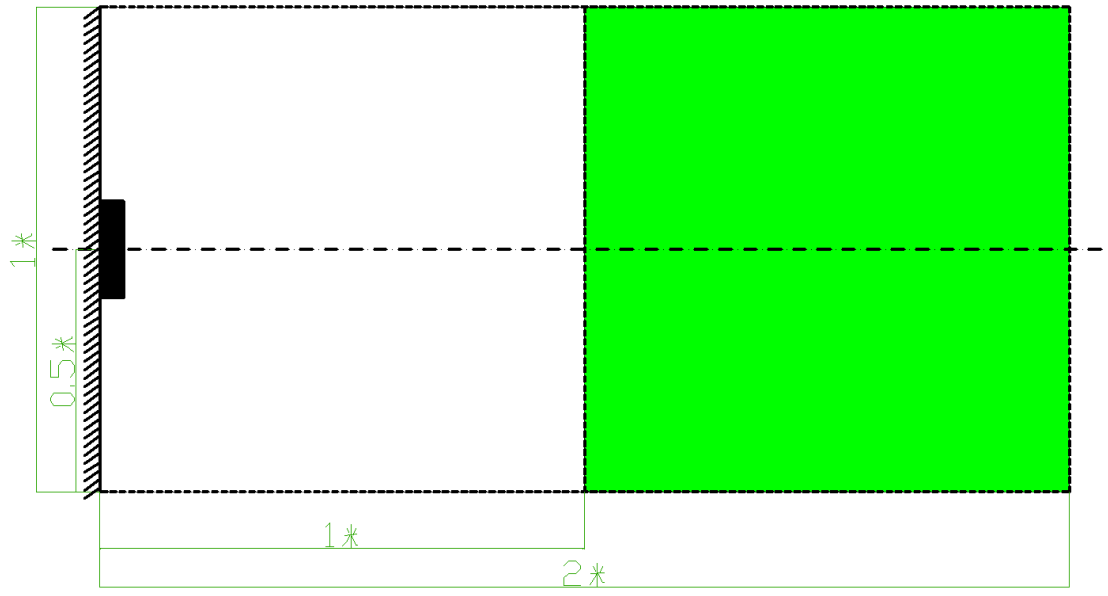
```
/sonar3
/sonar4
/sonar_cloudpoint                    #0~4 号超声波的点云数据
/voltage_value                       #电量显示主题
```

2.6 底盘自动回充功能

环境要求：

为确保自动回充功能正常使用，请您按要求设置充电环境：

1）回充站必须与底盘在同一个平面内。
2）回充站周围无类似镜面的反光物体。
3）回充站需放置在左右分别有 0.5 米无障碍物空间的墙面。前方需有 2 米的无障碍物空间供底盘进行回充使用。其具体的要求如下图所示。



注：绿色区域为最佳回充区域。

方法一：导航过程中，导航模块给底盘下发指令让底盘开始回充

例如，导航模块启动导航 launch，

```
$ ssh eaibot@192.168.31.200
```

```
$ roslaunch dashgo_nav navigation_imu.launch
```

打开另一个终端，在 ubuntu pc 中运行 rviz 观察地图

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

在 rviz 上设置好起点和适当的目标点（让底盘导航进入回充范围），等机器人导航到回充范围后，此时在导航模块的另外一个终端中，下发启动回充指令

```
$rostopic pub -1 /recharge_handle std_msgs/Int16 1
```

此时会暂停导航，原地旋转进行红外对接，然后开始自动回充

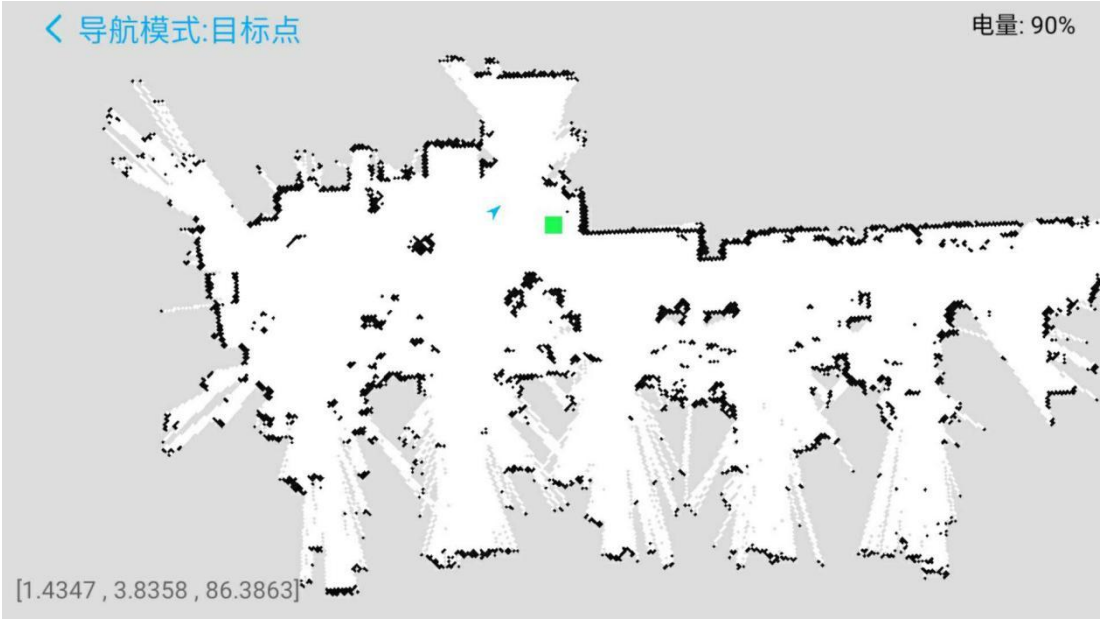
若想停止，输入停止回充指令

```
$ rostopic pub -1 /recharge_handle std_msgs/Int16 0
```

此时会退出回充状态，进入到导航状态，再设一目标点后，就会继续导航

方法二：用手机建图导航时，直接在手机上启动回充

用最新的手机 app EAIGO5x 建好地图后，切换到导航，如下图所示



正常情况，切换到导航时，会自动设置好起点（如果发现起点位置不对，就需要手动设置起点，具体操作看 app 使用手册），此时，先设定好充电位置（其他任何时候都可以设

置）；

点击  按钮，底盘会先导航到充电点，然后红外自动寻找充电桩，找到后开始充电对接；

点击  按钮，取消回充，继续导航；

点击  按钮，删除已经设置的充电点位置。

注意：1.如果底盘上在绿色最佳的区域内启动回充，成功率达 95%以上，如果不在绿色区域内，失败概率会增大（尤其是比较近充电桩时，失败率非常高，主要是没有足够距离来调整姿态）。

2.正常回充成功会先滴一声表示回充到位，然后再滴两声表示正式开始充电，此时回充站绿灯闪烁。如果有其他情况，就表示不成功，需要放到合适位置，先发指令停止回充，若要继续回充，再要发启动回充指令。(例如电池比较满时，先滴一声表示回充到位，然后再滴两声表示正式开始充电，再滴一声表示电池满了，如果是滴 6 声，表示充电错误)

2.7 Sick 雷达建图与导航

Sick 雷达参数介绍

参数文件路径:/opt/ros/kinetic/share/sick_tim/launch/sick_eai.launch
<param name="min_ang" type="double" value="-1.57" /> <param name="max_ang" type="double" value="1.57" />
雷达的扫描角度为-1.57~1.57 弧度（即-90~90 度）共 180 度，根据实际情况，最大只支持 270 度扫描角度
<param name="range_max" type="double" value="25.0" /> sick 雷达最大扫描距离
<param name="hostname" type="string" value="192.168.31.201" /> <param name="port" type="string" value="2112" /> <param name="timelimit" type="int" value="5" />
设置 sick 雷达通过网口把数据传给导航模块，并指定雷达 ip 为 192.168.31.200，与导航模块 ip 必须在同一个网段,sick 雷达的 ip 地址需要在 windows 下用 sick 官网软件设置，具体请参考 sick 官网资料
<node pkg="tf" type="static_transform_publisher" name="base_link_to_laser_sick" args="0.2245 0.0 0.28 0.08 0.0 0.0 /base_footprint /sick_laser 40" />
设置 sick 雷达与底盘坐标系的 tf 转换关系，校准方法与 eai 雷达一样

参数文件路径： dashgo_ws/src/dashgo/dashgo_nav/launch/include/sick/imu/gmapping_base.launch Sick 雷达的建图与导航参数
`<param name="maxUrange" value="20.0"/>` `<param name="maxRange" value="22.0"/>`
Sick 雷达 gmapping 建图的扫描距离，不能都设置成 25.0

Sick 雷达建图导航使用

注意：1.sick 雷达使用前，必须进行 tf 坐标系校准（方法与 F4 雷达校准一样）

2.默认 sick 雷达使用带陀螺仪建图与导航，根据情况可修改是否带陀螺仪

进入导航模块中，运行 sick 雷达建图 launch，该 launch 默认启动带陀螺仪建图

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

打开另一个终端，在 ubuntu pc 中运行 rviz 观察地图

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

手机控制底盘行走，建好地图后，保持以上程序正常运行，打开另外一个终端，远程进导航模块，把地图保存在导航模块的 dashgo_nav/maps 目录中，

```
$ ssh eaibot@192.168.31.200
$ roscd dashgo_nav/maps/
$ rosrun map_server map_saver -f eai_map_imu
```

地图保存好后，ctrl+c 退出建图程序，在导航模块中启动 sick 导航 launch

```
$roslaunch dashgo_nav sick_navigation_imu.launch
```

打开另一个终端，在 ubuntu pc 中运行 rviz 观察地图

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

设置好起点位置，然后设置目标点开始导航.

2.8 深度摄像头导航避障

深度摄像头参数及校准

目前 B1 支持奥比中光深度摄像头导航避障，具体参数调试如下:

参数文件路径: ~/package_ws/src/astra_ros/ros_astra_launch-master/launch

```
<node pkg="tf" type="static_transform_publisher" name="base_link_to_camera"
args="0.15 0.0 0.75 0.08 0.26 0.0 /base_footprint /camera_link 40" />
```

摄像头与底盘的 tf 转换关系

在导航模块中，启动单独摄像头的 launch：

```
roslaunch astra_launch astra.launch
```

然后在 rviz 中显示彩色点云，添加类型为 PointClound2 的 topic，参数 topic 设置为 /camera/depth_registered/points，color transformer 设置为 rgb8，根据 rviz 中现实的图像可判断上述 tf 是否修改准确。

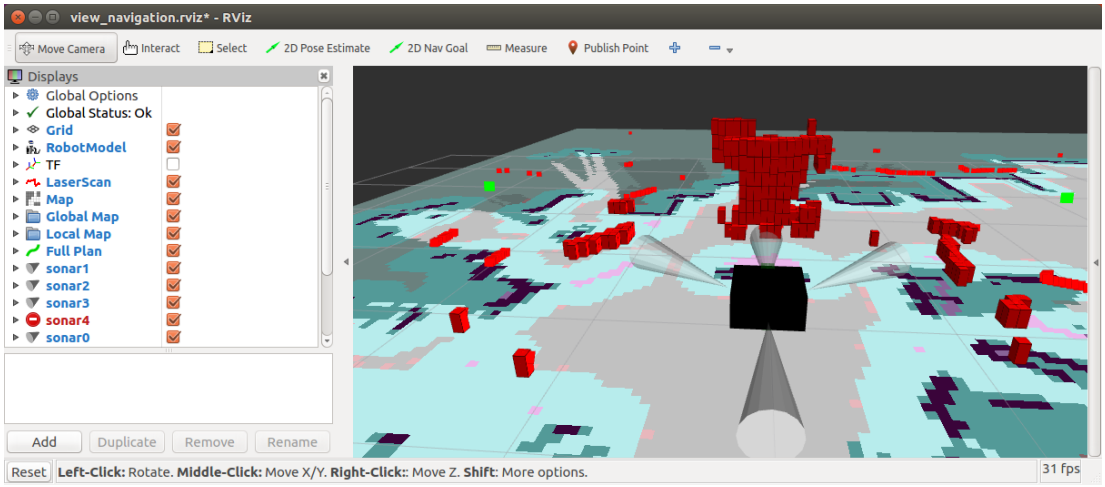
深度摄像头导航避障

在导航模块中，启动带有摄像头导航的 launch：

```
roslaunch dashgo_nav  navigation_camera_imu.launch
```

在电脑 ubuntu 中运行 rviz，并添加 PointCloub2 选择为/camera/depth_registered/points

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



如图所示，摄像头会把看到的障碍物设置成多层的 costmap，并让机器人导航时，避开这些障碍物。

修订历史

日期	内容
2018-08-13	V1.1
2018-09-13	V1.2