目录

二维码自动充电 ros 驱动包配置手册

1. 概述

摄像头通过二维码可以精确计算出两者之间的相对姿态，利用这个特性蓝鲸机器人开发了一套二维码自动对接充电解决方案。本方案主要由车载端电路铜片、充电桩、摄像头三部分构成。基本电气参数如下，rs232 串口通信，支持的充电电压范围：12v 到 59v，充电电流:最大 10A。

2. 硬件安装

2.1 充电桩

水平靠墙放置，充电桩的高度可以上下调节。充电桩的供电是电池充电器，注意正负极，因为本方案需要靠充电电压判断铜片是否对接完好，所以电池充电器需要没有接电池也一直有输出电压的特性。

2.2 车载端电路铜片

水平固定在车尾部中间位置，铜片高度要和充电桩一致，电路板需要外接直流 5V 供电，模块工作电流大概 100 毫安。电路板配的 usb 转 rs232 串口线需要插入 ros 工控机。

2.3 摄像头

水平固定在车尾部，注意摄像头端子所在那一侧对应的是图像下方，保证图像上下没有颠倒，镜头要在铜片中轴线上。高度需要和充电桩二维码平齐，高度可以上下 15cm 偏差，但是要保证视野不会被其他东西挡住。摄像头 usb 接口插入 ros 工控机。

3. 软件包下载与编译

3.1 先安装 sophus 依赖包

在任意位置下载源代码包

```
git clone  -b  noetic  http://git.bwbot.org/publish/sophus.git
cd sophus
mkdir build
cd build
cmake  ..
```

注意不需要 make 编译，这个包就是一些头文件， cmake 命令会自动添加相关文件到 cmake 环境中完成关联。

3.2 还需要 6 个 ros 软件包：

xiaoqiang_log、galileo_msg、usb_cam、galileo_serial_server、ar_track_alver、bw_auto_dock 在 ros 工作空间 src 目录，执行下列命令下载和编译软件包。

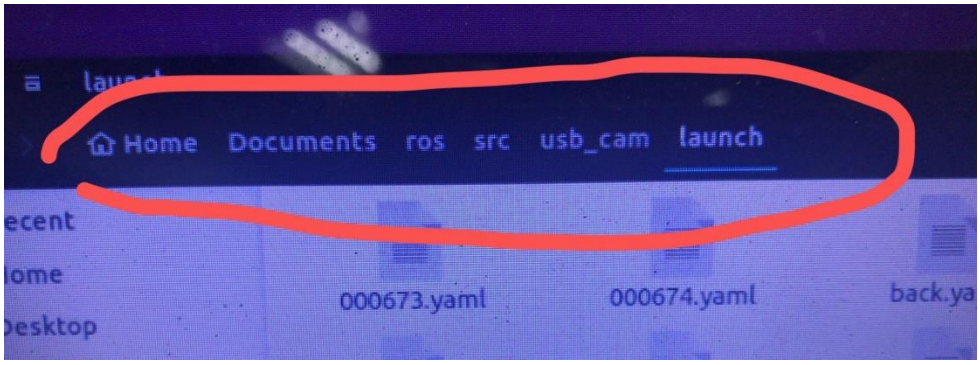
```
git clone  -b  noetic  http://git.bwbot.org/publish/xiaoqiang_log.git
git clone  -b  master  http://git.bwbot.org/publish/galileo_msg.git
git clone  -b  artag-retail  http://git.bwbot.org/publish/usb_cam.git
git clone  -b  noetic  http://git.bwbot.org/publish/galileo_serial_server
git clone  -b  artag-retail  http://git.bwbot.org/publish/ar_track_alvar.git
git clone  -b  artag-retail  http://git.bwbot.org/publish/bw_auto_dock.git
cd ..
catkin_make  -DCMAKE_BUILD_TYPE=Release  -DCATKIN_WHITELIST_PACKAGES=""
```

4. 系统环境参数配置

4.1 摄像头内参设置

先确定新摄像头的编号，比如下图中的 000693，准备好对应编号的标定文件（购买时会提供），比如 000693.yaml。

根据编号将对应文件拷贝到上文安装的 usb_cam 包 launch 文件夹内，完整路径例子在下图。



用新拷贝的文件替换 back.yaml 文件内容。

4.2 摄像头安装位置设置

用卷尺测量摄像头镜头在车 base_footprint 坐标系下的 x、 y、 z 坐标，精度 0.5 厘米左右就行。然后设置 usb_cam 包中的 back.launch 文件内的 tf 参数。只用修改 x 、y 、z 参数。static_transform_publisher x y z qx qy qz qw frame_id child_frame_id period_in_ms。

4.3 设置 usb 串口和 usb 摄像头 udev 规则

ros 主控通常外接了很多设备，为了区分可以使用 udev 规则进行别名映射。我们需要把 rs232 串口映射成 ttyUSB004，把 usb 摄像头映射成 back_camera。在 bw_auto_dock 软件包中有一个“配置 udev 规则脚本”的文件夹，执行 sudo ./initenv.sh 指令即可完成 udev 规则配置，然后重启系统生效。

4.4 设置 bw_auto_dock 软件包参数

bw_auto_dock 包 launch 文件夹内有两个 launch 文件，区别只在于全局坐标系的选择。以 xiaoqiang_local.launch 为例，需要设置里程计话题名字、充电桩保存位置这两个参数。里程计话题名字需要 remap 成你们自己小车发布的里程计，要是标准的 Odomery 话题类型。

bw_auto_dock 包 launch 文件夹内还有一个 default.yaml 文件，里面是一些参数，请仔细阅读注释。现在需要调整的参数是 power_threshold 和 crash_distance。

crash_distance 表示铜片和充电桩接触上时，二维码离 base_footprint 坐标系原点的距离，单位是毫米。有两种办法获取这个值，先把车手动对接上充电桩，启动 xiaoqiang_local.launch，听到开始充电后，可以通过 rqt_console 来查看 dock_driver 节点的 debug 输出，里面有 distance 开头的输出，后面跟着的 x 值就是当前二维码离 base_footprint 坐标系原点的距离，注意单位换算。第二种办法就是用卷尺测量，精度 0.5 厘米左右就行。

5. 可执行节点 nodes 与 launch 文件

bw_auto_dock 包 launch 文件夹内有两个 launch 文件，区别只在于全局坐标系的选择。xiaoqiang_local.launch 表示充电桩位置保存在 odom 坐标系，因为 odom 坐标系是会漂移和重置的，因此这个 launch 一般用来快速验证自动对接充电算法。xiaoqiang_global.launch 表示充电桩位置保存在 map 坐标系，可以结合 slam 导航完成实际的部署使用。

两个 launch 都会启动三个 node 节点：ar_track_alvar 的二维码识别节点、usb_cam 的摄像头驱动节点、bw_auto_dock 的 dock_driver 节点。我们只介绍 dock_driver 节点，因为它是核心的控制节点。

5.1 订阅的 tf 数据

根据 launch 文件中设置的全局坐标系名字，dock_driver 会订阅全局坐标系到 odom 坐标系的 tf 关系。还有 odom 坐标系到 base_footprint 坐标系的 tf 关系。

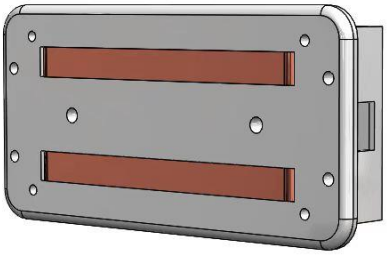
5.2 订阅的话题数据

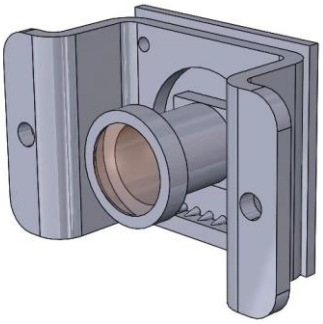
/odom (nav_msgs/Odometry)
里程计话题，需要 remap 成自己小车发布的里程计话题名字。
/ar_pose_marker (ar_track_alvar_msgs/AlvarMarkers)
二维码姿态话题，用来计算车相对充电桩的位置。
/bw_auto_dock/dockposition_save (std_msgs/Bool)
设为 true 则触发充电桩位置保存动作。
/bw_auto_dock/EnableCharge (std_msgs/Bool)
设为 true 则触发自动对接充电动作。

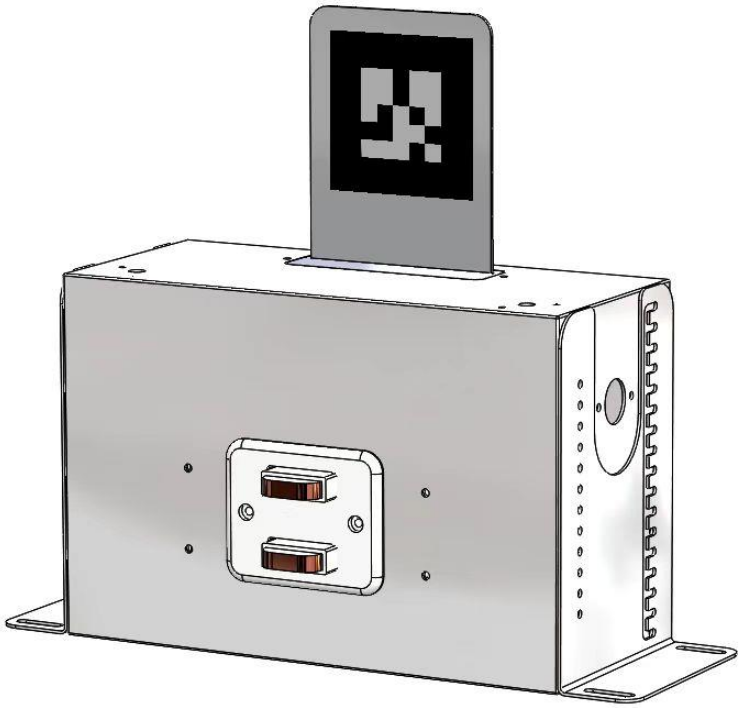
5.3 发布的话题数据

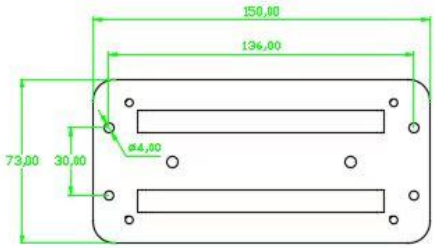
/cmd_vel (geometry_msgs/Twist)
输出的速度控制指令。
/bw_auto_dock/Chargecurrent (std_msgs/Float32)
当前测量的充电电流，单位 A
bw_auto_dock/Chargepower (std_msgs/Float32)
当前测量的充电铜片电压，单位 V
bw_auto_dock/Batterypower (std_msgs/Float32)
当前测量的电池电压，单位 V
bw_auto_dock/Chargestatus (std_msgs::Int32)
当前充电状态，0 表示空闲，1 表示正在充电，2 表示充电完成，>=3 表示在执行对接过程中。

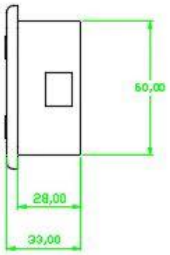
6. 硬件尺寸图纸

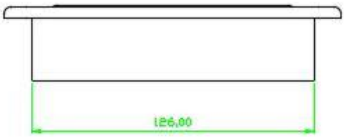


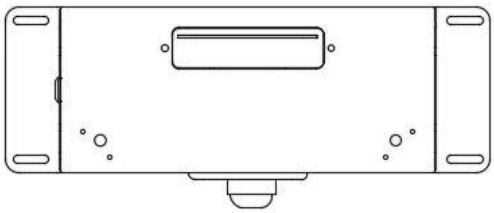


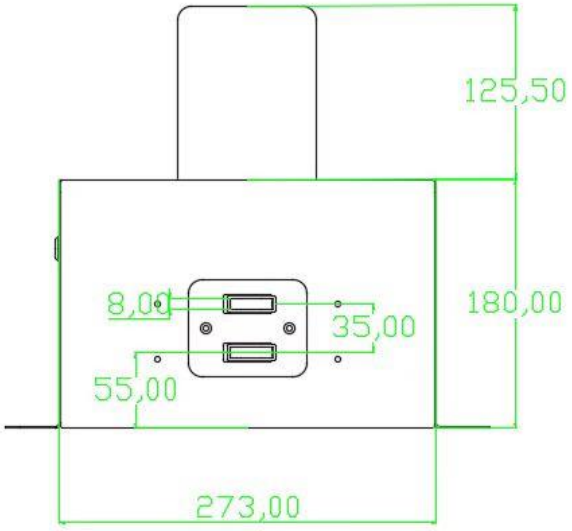


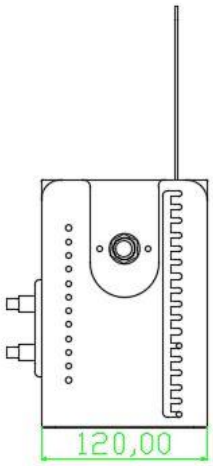

150,00
136,00
73,00
30,00
Ø4,00

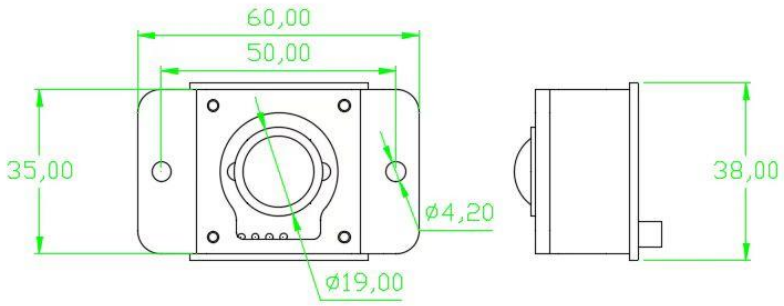

60,00
28,00
39,00

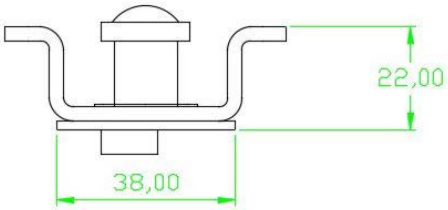

126,00




125,50
180,00
8,00
35,00
55,00
273,00


120,00


60,00
50,00
35,00
ø4,20
38,00
ø19,00


22,00
38,00