目录

ROS 底盘驱动 xqserial_server 包配置手册

1. 概述

ROS 底盘驱动 xqserial_server 包负责将订阅的/cmd_vel 话题进行差速解算后，转换成串口命令下方给电机驱动器。同时处理驱动器上传的串口包信息，然后整理发布成里程计和传感器数据话题。

2. 硬件连接

电机驱动器和主机通过 USB 转 rs232 模块连接（对应“电机串口”），串口要通过 udev 规则进行 USB 端口绑定，映射成 ttyUSB001。串口参数是波特率为 115200，8 个数据位，1 个停止位，无奇偶校验。

驱动器的接线请根据驱动器手册来操作，在调试 xqserial_server 包之前需要保证用 mtools 软件能正常闭环控制电机的正反转，否则可能会烧毁驱动器。

3. Ubuntu 主机串口权限设置和设备号映射

我们使用的是 usb 转 rs232 模块，ubuntu 默认会识别成 ttyUSB0 设备。这个可以通过 ls /dev 指令查询。识别出的设备号是随机的，为了将串口号映射成 ttyUSB001，同时设置设备用户读写权限。我们可以通过 udev 规则来实现。

具体参考后面这篇帖子的步骤三，https://community.bwbot.org/topic/501/小强 ros 机器人教程-27-___bw_auto_dock 自动充电功能包的使用和实现原理

4. 软件包编译

涉及到 ros 工作空间、ros 包、catkin_make、ros 工作空间 source 指令、git 这些基础知识，如果不清楚这些请百度学习一下。

先 git clone 下载 xqserial_server 包放入 ros 工作空间 src 文件夹内，切换到 lungu 分支，最后 catkin_make 编译。

假设你的 ros 工作空间在~/catkin_ws 内,下面是下载编译全部指令。

```
cd  ~/catkin_ws/src/
git  clone  https://github.com/BluewhaleRobot/xqserial_server.git
cd  xqserial_server
git  checkout  lungu
cd  ~/catkin_ws/
catkin_make
```

5. 可执行节点 nodes

catkin_make 编译后会生成可执行节点 xqserial_server，launch 文件夹中已经提供了一份可直接使用的 launch 文件 xqserial.launch。script 文件夹中的 visualization.py 是一个 python 脚本文件，可以用来可视化 xqserial_server 发布的 IMU 数据。

5.1 xqserial_server 节点

xqserial_server 节点是包主节点，涉及差速解算、里程计、传感器、tf 等数据的处理和发布。硬件上，这个节点会访问驱动器的串口设备。下文将详细介绍这个节点的 ROS 配置信息。

5.2 订阅的话题数据

/cmd_vel （geometry_msgs/Twist）
速度话题，订阅后解算成电机控制指令。

/imu_cal （std_msgs/Bool）
IMU 自动标定指令，设为 true 会触发自动标定，此时机器人需要静止等待 2 分钟，2 分钟内不能重复触发。

/globalMoveFlag （std_msgs/Bool）
设为 true，/cmd_vel 指令生效；设为 false，则禁用/cmd_vel。

/barDetectFlag （std_msgs/Bool）
设为 true 则使能超声波避障，设为 false 则关闭超声波和红外避障。

5.3 发布的话题数据，50hz 频率

/xqserial_server/Odom （nav_msgs/Odometry）
底盘里程计，角度部分已经融合了 IMU。

/xqserial_server/StatusFlag （std_msgs/Int32）
值为 0 表示处理状态正常，值为 1 表示 IMU 还在初始化或处于标定状态中。值为 2 则表示红外被触发。

/xqserial_server/Twist （geometry_msgs/Twist）

底盘反馈的速度，线速度是编码器反馈值，角速度是 IMU 返回值。

/xqserial_server/Power (std_msgs/Float64)
电池电压，单位为 V。

/xqserial_server/Pose2D (geometry_msgs/Pose2D)
二维的 pose 数据，可以方便查看里程计坐标和角度。

/xqserial_server/IMU (sensor_msgs/Imu)
IMU 话题数据，IMU 的 tf 关系需要自己根据下文在 launch 中设置调整。

/xqserial_server/Sonar1 (sensor_msgs/Range)
超声波模块 S1 话题数据，tf 关系需要自己根据下文在 launch 中设置调整。

/xqserial_server/Sonar2 (sensor_msgs/Range)
超声波模块 S2 话题数据，tf 关系需要自己根据下文在 launch 中设置调整。

5.4 节点参数

~port (string)
电机驱动器串口名字，默认/dev/ttyUSB001

~wheel_separation (double)
两侧动力轮或履带中心间距，单位米，默认 0.36。

~wheel_radius (double)
轮半径或者履带动力轮半径，单位米，默认 0.0825。

~max_speed (double)
车轮最大转速或者履带动力轮最大转速，单位转每秒，默认 5.0。

5.5 发布的 tf 变换关系

odom→base_footprint
50hz 发布频率，base_footprint 在车正中心。

6. 校准 IMU

把小车水平静止放置好，发布 ros 话题启动 IMU 自标定，2 分钟左右的标定过程种不能移动、碰撞小车。

```
rostopic pub /imu_cal std_msgs/Bool '{data: true}' -l
```

校准过程中，/xqserial_server/IMU 话题数据一直是零值，校准完成后恢复正常值，通过这个话题数据可以判断校准进度和校准结果。

7. 校准机器人参数

执行下面操作之前需要先完成步骤 6 的 IMU 校准工作，因为 IMU 会影响里程计反馈精度。

7.1 校准轮半径参数 wheel_radius

校准原理：

卷尺可以测量小车实际前进距离(设为 L)，/xqserial_server/Pose2D 这个话题可以查看小车里程计反馈距离(设为 l)，这样我们可以得到 L/l 这个比值（校准之后的值应该为 1），这个比值就是 launch 文件中轮半径参数 wheel_radius 的缩放因子。用这个比值乘以现在的值可以得到校准后的值。

校准后 wheel_radius = 校准前 wheel_radius * L /l

校准步骤：

将车遥控到起始线，车角度要摆正对齐，然后重启 xqserial_server 包 launch 文件，使里程计归零。

遥控车直线前进 2 米左右，记录用卷尺测量的小车前进距离 L，和话题反馈值 l，计算 L/l 比值。

根据校准原理得到校准后的 wheel_radius 值，将 launch 文件中的轮半径参数修改成这个新值，然后重启 launch 文件，半径就校准完成了。

7.2 校准轮间距参数 wheel_separation

注意：校准前需要完成轮半径的校准工作。

校准原理：

/xqserial_server/Twist 话题里面的角速度反馈值是通过 IMU 计算出来的，因此它可以作为一个基准值 W，/cmd_vel 话题里面的角速度值是设定值 w。计算得到的 W/w 比值，就是轮间距参数的缩放因子。

这个值可以不用很精准，因为它只会影响角速度控制的精确性，不会影响里程计角度部分的精度。如果要获取精确值，推荐自己写个 python 节点订阅这两个话题，通过计算平均值来得到更精确的比值。

校准后 wheel_separation = 校准前 wheel_separation * W /w

校准步骤：

遥控车原地旋转，根据校准原理得到校准后的 wheel_separation 值，将 launch 文件中的轮间距参数修改成这个新值，然后重启 launch 文件，轮间距参数就校准完成了。