# starrobot使用说明

本文主要介绍starrobot的使用，其中包括底层控制板说明、固件升级、代码调试、上层环境构建和功能包的使用等文档说明。下面将会逐一进行介绍。

# 开发板简介

# 构建AixBot项目的ROS包环境

## 为挂载设备重命名

在机器人开发过程中我们可能会使用到很多的USB设备，这些设备在linux中挂载的设备名称可能在每次开机后都会有变化，导致了我们无法分清哪一个设备挂载点对应哪一个设备，这样在程序中就可能因为挂载点的改变而导致打开设备不对，因此本节主要介绍怎么编写58-starrobot.rules脚本实现设备的重命名，首先我们使用lsusb指令查看当前系统挂载的usb设备如图7-1.1所示。

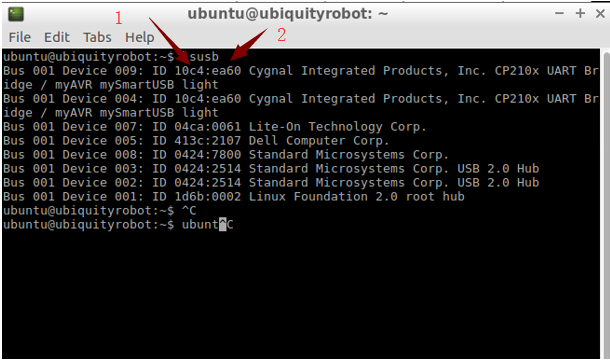


图7-1.1

当前只挂载了STM32控制板的串口和雷达的串口设备所以我们看到了有两个CP210x UART 设备，我们先拔掉一个USB设备再使用ls /dev 指令进行查看如图7-1.2所示。

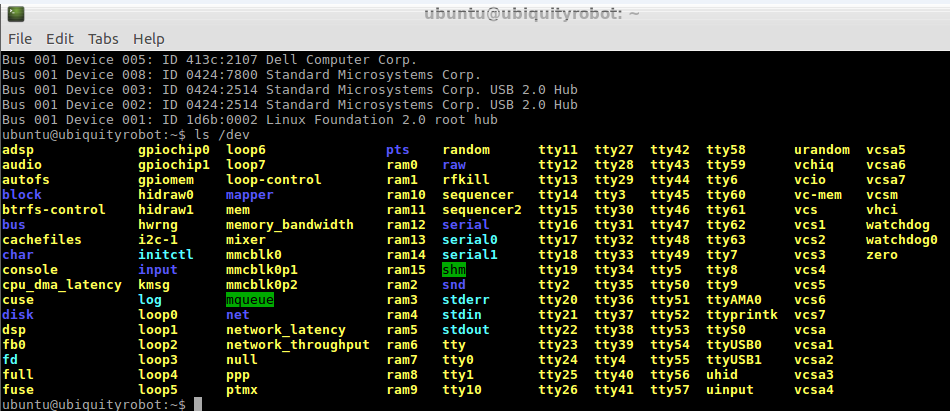


图7-1.2

然后我们再使用udevadm info --attribute-walk --name=/dev/ttyUSB1 | grep KERNELS 指令进行查看如图7-1.3所示。

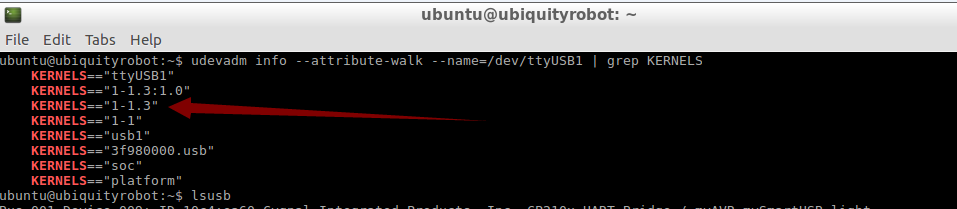


图7-1.3

当知道了这些数据后我们就可以开始编写.rules文件如图7-1.4所示

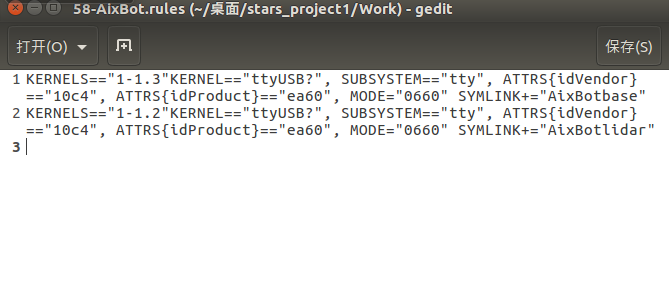


图7-1.4

KERNELS=="1-1.3"KERNEL=="ttyUSB?", SUBSYSTEM=="tty", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", MODE="0660" SYMLINK+="starrobotbase"

KERNELS=="1-1.2"KERNEL=="ttyUSB?", SUBSYSTEM=="tty", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", MODE="0660" SYMLINK+="starrobotlidar"

下面就讲解一下对应的参数怎么填写，KERNELS 参数

就是我们使用udevadm info --attribute-walk --name=/dev/ttyUSB1 | grep KERNELS指令得到如图7-1.3箭头所示的值，KERNEL参数不需要改

变、SUBSYSTEM参数也不需要改变，ATTRS{idVendor}参数就是我们使用lsusb得到如图7-1.1箭头1所示的值，ATTRS{idProduct}参数就是如图7-1.1箭头2所示的值，MODE参数是给设备权限不需要进行修改，SYMLINK参数就是我们想要重命名的名称，如果是底盘设备则需要和aixbot\_bringup.launch里面的名称一致。

## 7.2用户怎么修改.rules文件

如果您购买了我们的设备，需要修改挂载设备名称时请详细阅读7.1和7.2的内容。首先我们先使用sudo vim /etc/udev/rules.d/58-starrobot.rules指令打开我们已经加载到udev下面的脚本文件，然后参考7.1的内容进行参数的修改，修改完成后保存退出，再使用sudo reboot指令进行重启，重启后使用ls /dev指令就可以查看到starrobotbase、starrobotlidar这两个名称了。

## 7.3怎么使用脚本文件构建starrobot开发环境

首先我们打开命令终端然后输入 git clone <https://github.com/StarsFu/AixBot_Work.git> 命令把我们的脚本工程下载到本地，如图7-3.1所示。

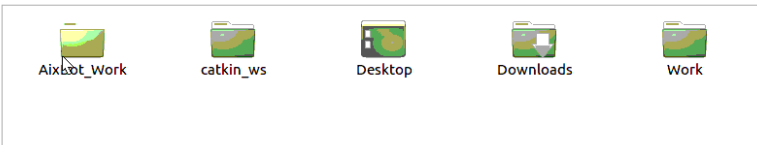


图7-3.1

里面有的内容如图7-3.2所示。

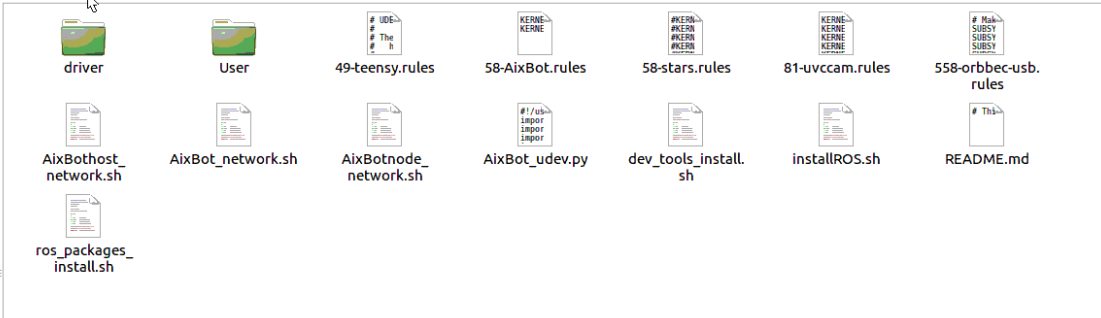


图7-3.2

其中58-AixBot.rules文件是重命名设备的文件，dev\_tools\_install.sh是把我们编写重命名设备的文件拷贝到 /etc/udev/rules.d/文件夹里，ros\_packages\_install.sh是创建工程和下载一些开源包到工程里面，所以我们先使用sudo chmod 777 dev\_tools\_install.sh sudo chmod 777 ros\_packages\_install.sh

命令给这两个文件赋权限，然后再执行./ dev\_tools\_install.sh ./ ros\_packages\_install.sh 即可完成环境的构建。因为目前AixBot的代码还没有开源，所以没有在脚本中编写，可以使用 1：cd ~/catkin\_ws/src 2：git clone <https://github.com/StarsFu/AixBot_Project.git> 命令下载AixBot的工程代码。具体的工程目录如图7-3.3所示。

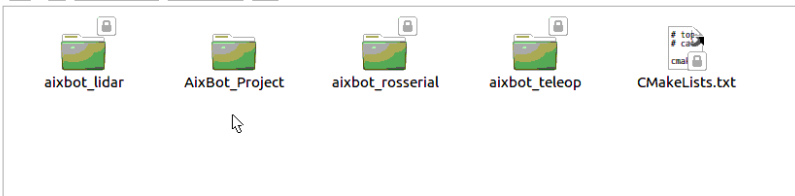


图7-3.3

## 7.4编译AixBot工程

首先我们使用cd ~/catkin\_ws命令去到catkin\_ws文件夹下，首次编译需要使用catkin\_make –j1命令进行编译生成工程依赖的头文件，如果直接使用catkin\_make 进行编译可能会提示找不到某一些头文件，编译完成后我们使用rospack profile命令把我们的工程刷新一下，就可以完成工程的代码编译。

## 7.5运行launch文件与底盘进行通信

在编译完成后我们进行ROS与底盘通信的测试，首先我们运行roslaunch aixbot\_bringup aixbot\_bringup.launch指令，出现如图7-5.1所示图怎说明通信成功。

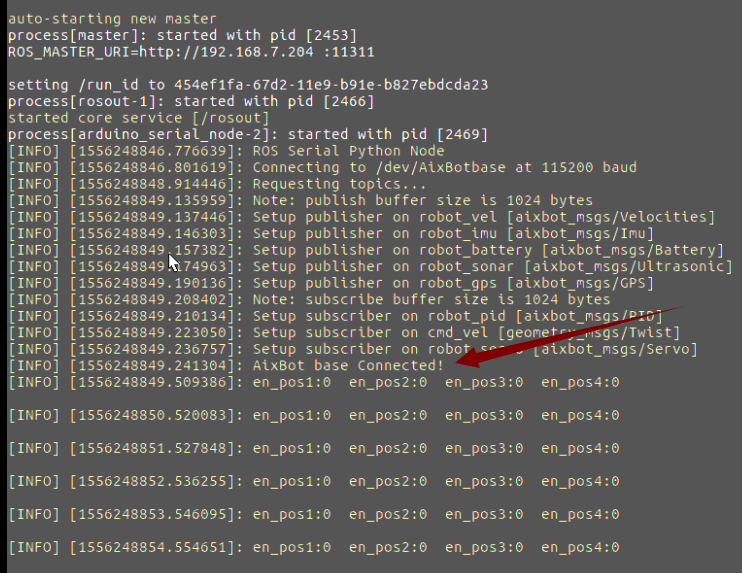
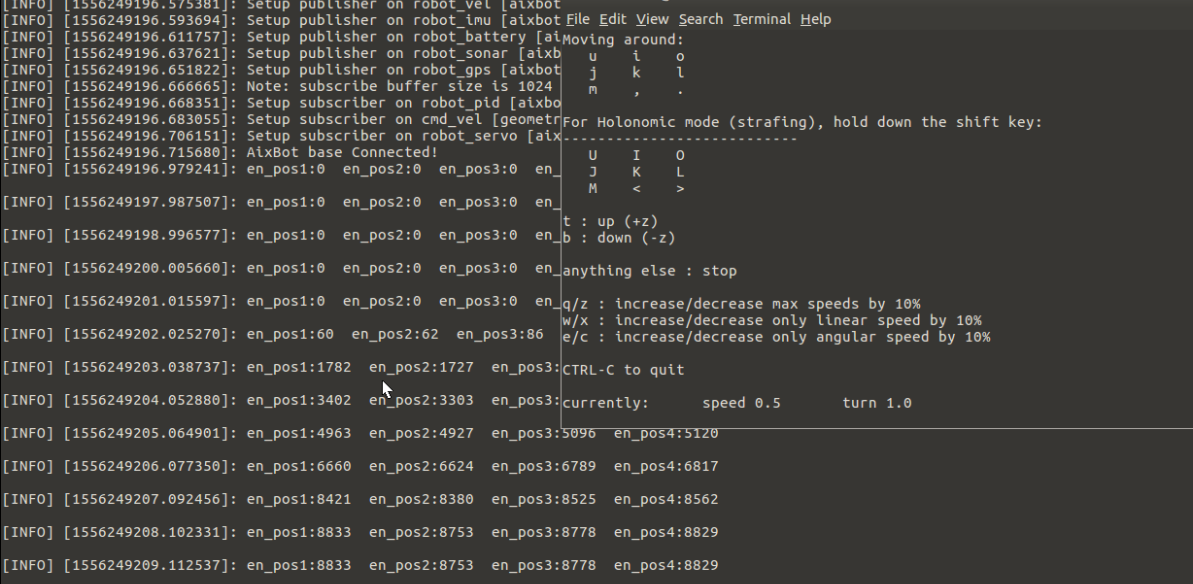


图7-5.1

## 7.6使用键盘控制小车

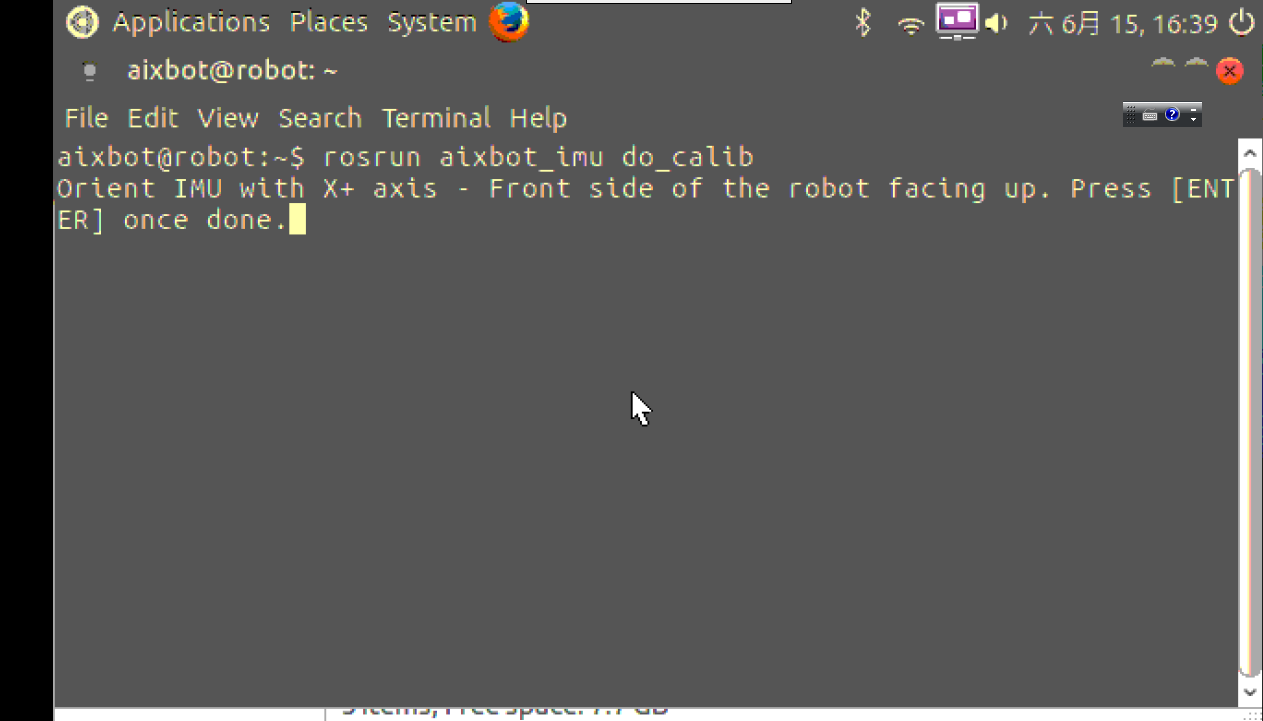
在启动了aixbot\_bringup.launch文件后我们再打开一个终端，输入rosrun teleop\_twist\_keyboard teleop\_twist\_keyboard.py指令启动键盘控制，看到如图7-6.1所示图并且小车可以运动，说明底盘的接线是没有问题的。

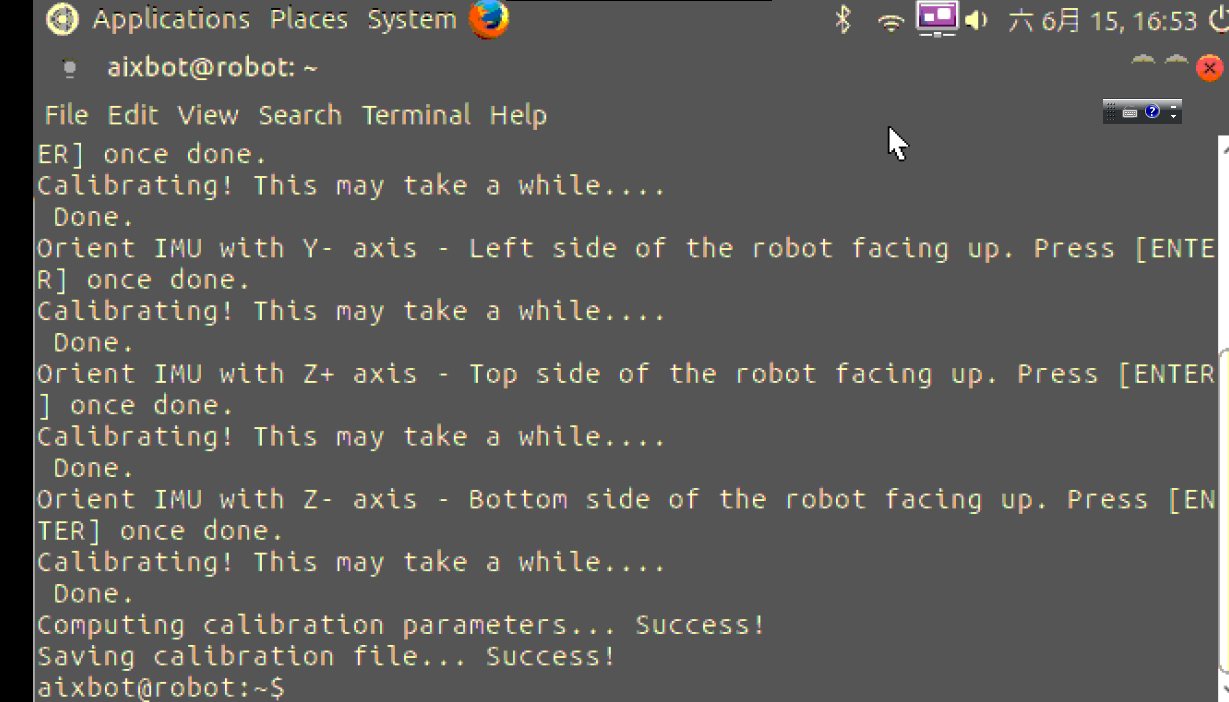


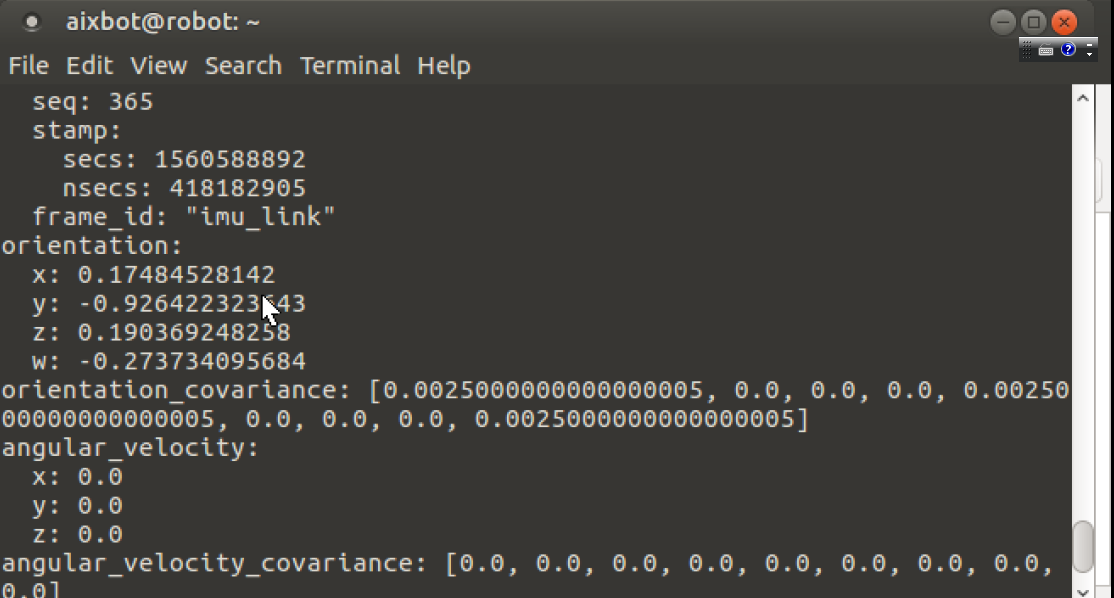
如果编码器的值偏差很大或者方向不对的话请检查接线是否正确。

# AixBot机器IMU校准说明

首先需要远程登录到AixBot机器人主控上运行机器人底盘节点，指令如下（roslaunch aixbot\_bring aixbot\_bring.launch）启动机器人节点后，运行IMU校准节点，指令如下（rosrun aixbot\_imu do\_callib）然后根据提示按Enter键进行校准，总共有X+,X-,Y+,Y-，Z+,Z-九个 方向。







# AixBot机器线速度校准

首先登陆aixbot机器端运行启动底层节点指令：

然后继续运行线速度校准节点指令：

最后在PC（远程操控的电脑）上运行指令：rqt\_re

# AixBot机器角速度校准

# AixBot机器创建地图

# 12.AixBot机器自主导航