轮 趣 科 技

串口通信控制与反馈

推荐关注我们的公众号获取更新资料

版本说明：

版本	日期	内容说明
V1.0	2024/4/17	第一次发布

目录

1. 通讯协议

这一章主要是对我司机器人运动底盘的通讯协议做一个详细的说明。我司的 ROS 教育机器人底盘和大型 ROS 科研机器人底盘使用的通讯协议是一样的，这种统一性不仅便于维护和升级，而且也为跨平台的协作提供了极大的便利。

文章中的数据上下行都是相对于机器人运动底盘的，上行数据指的是机器人底盘向上位机发送的状态数据，下行数据指的是上位机向机器人底盘发送的控制信息。

1.1 上行数据帧解析

机器人运动底盘通过串口发送的数据包格式，如下表所示：

数据内容	帧头(固定值 0x7B)		flag_stop		X 轴速度		Y 轴速度		Z 轴速度		X 轴加速度		Y 轴加速度	
数据类型	Uint8		Uint8		short		short		short		short		short	
占用字节	1		1		2		2		2		2		2	
高低位序					MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB
数组号	0		1		2	3	4	5	6	7	8	9	10	11
数据内容	Z 轴加速度		X 轴角速度		Y 轴角速度		Z 轴角速度		电池电压		数据校验位		帧尾(固定值 0x7D)	
数据类型	short		short		short		short		short		Uint8		Uint8	
占用字节	2		2		2		2		2		1		1	
高低位序	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB				
数组号	12	13	14	15	16	17	18	19	20	21	22		23	

表 1-1 上行数据包格式表

以下是数据帧各数据的含义：

帧头：固定值 0x7B，标识数据包的开始，占一个字节；

flag_stop：电机使能标志，0x00 时电机使能，其他值失能，占一个字节；

X 轴速度：机器人 X 轴上的速度，单位是 mm/s，占两个字节；

Y 轴速度：机器人 Y 轴上的速度，单位是 mm/s，占两个字节；

Z 轴速度：机器人 Z 轴上的速度放大 1000 倍，单位是 rad/s，占两个字节；

X 轴加速度：加速度计 X 轴上的加速度【原始数据】，占两个字节；

Y 轴加速度：加速度计 Y 轴上的加速度【原始数据】，占两个字节；

Z 轴加速度：加速度计 Z 轴上的加速度【原始数据】，占两个字节；

X 轴角速度：陀螺仪 X 轴上的角速度【原始数据】，占两个字节；

Y 轴角速度：陀螺仪 Y 轴上的角速度【原始数据】，占两个字节；

Z 轴角速度：陀螺仪 Z 轴上的角速度【原始数据】，占两个字节；

电池电压：机器人的电池电压大小，单位是 mv（毫伏），占两个字节；

数据校验位：前 22 个字节异或校验（BBC 校验）的结果，占一个字节；

帧尾：固定值 0x7D，标识数据包的结束，占一个字节；

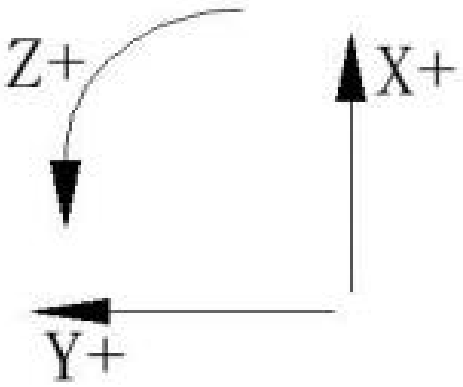


图 1-1 机器人 XYZ 三轴方向

需要注意的是数组号 8-19 的 12 个字节为三轴加速度计和角速度计的数据，注意这里的 XYZ 三轴方向为我们机器人 XYZ 轴方向，如图 1-1 所示，机器人的 X 轴正方向向前，Y 轴正方向向左，Z 轴正方向为向上。加速度计和陀螺仪计的速度皆为绕 XYZ 三轴旋转的速度，这些数据均是原始数据，需要进行转化才可使用。加速度计原始数据需要除以 1672 将单位转化 m/s²（米/平方秒）。角速度计原始数据需要除以 3753 将单位转化 rad/s（弧度/秒）。

1.2 下行数据帧解析

机器人运动底盘通过串口接收到的数据包格式，如下表所示：

数据内容	帧头（固定值 0x7B）	预留位	预留位	X 轴目标速度		Y 轴目标速度		Z 轴目标速度		数据校验位	帧尾（固定值 0x7D）
数据类型	Uint8	Uint8	Uint8	Short		Short		Short		Short	Uint8
占用字节	1	1	1	2		2		2		1	1
高低位序				LSB	MSB	LSB	MSB	LSB	MSB		
数组号	0	1	2	3	4	5	6	7	8	9	10

表 1-2 下行数据包格式表

以下是数据帧各数据的含义：

帧头：固定值 0x7B，标识数据包的开始，占一个字节；

预留位：第 2、3 个字节均为预留位，无意义，各占一个字节。

X 轴目标速度：机器人 X 轴上的目标速度，单位 mm/s，占两个字节；

Y 轴目标速度：机器人 Y 轴上的目标速度，单位 mm/s，占两个字节；

Z 轴目标速度：机器人 Z 轴上的目标速度放大 1000 倍，单位 rad/s，占两个字节；

数据校验位：前 9 个字节异或校验（BBC 校验）的结果，占一个字节；

帧尾：固定值 0x7D，标识数据包的结束，占一个字节；

机器人 XYZ 方向与章节 1.1 中介绍的保持一致。

2. 串口接口介绍

我司的机器人底盘 STM32 控制器上集成有 USB 转串口芯片，用户不仅可以使用串口对 STM32 的固件进行一键烧录，还可以通过串口通信与上位机进行通讯。本章主要介绍串口的位置以及应用。

2.1 机器人串口接口的位置

① ROS 教育机器人

以 ROS 教育机器人为例：

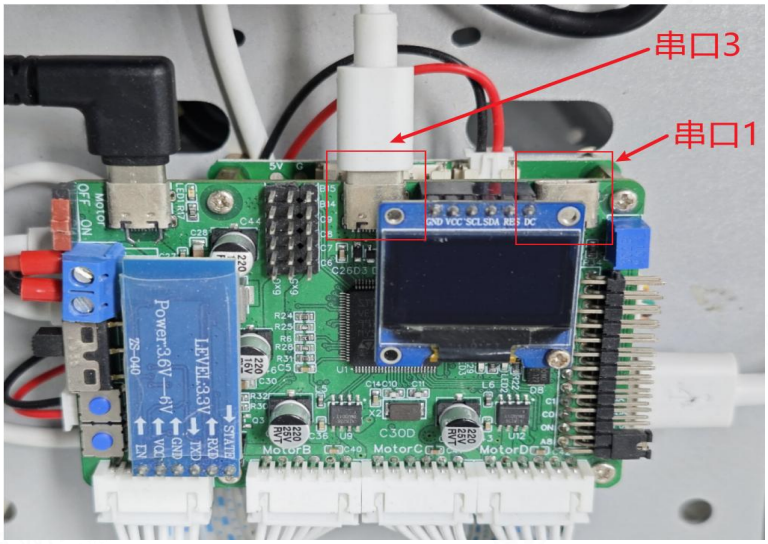


图 2-1　ROS 教育机器人串口位置图

② 大型 ROS 科研机器人

以大型 ROS 科研机器人为例：

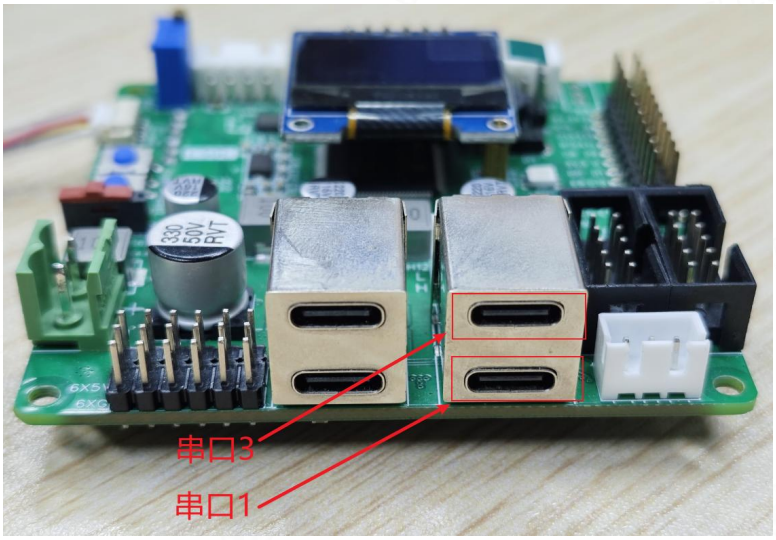


图 2-2　大型 ROS 科研机器人串口位置图

2.2 串口应用

机器人底盘上的串口主要用来连接上位机以及进行 ros 通信使用，串口 1 通常用来烧录底盘的固件以及调试使用，串口 3 通常用来与 ROS 上位机进行通信。

① ROS（串口 3）控制

ROS 和 STM32 控制器（机器人运动底盘）之间通过串口实现通信，STM32 控制器使用的是串口 3，波特率 115200，通信协议详情见章节 1.通讯协议。

机器人上电后默认是使用 ROS 控制模式，并且 ROS 控制在众多控制中拥有最高优先级。在 STM32 控制器的程序里，通过串口 3 向 ROS 发送机器人运动状态，并通过串口 3 接收 ROS 向机器人底盘发送的运动控制指令。

② 串口 1 控制

串口传输时需要注意双方波特率的设置需要一致，程序中串口 1 设置的波特率是 115200。接收到串口 1 的数据后机器人进入串口控制模式，显示屏左下角显示“USART”。

注意：对于大型 ROS 科研机器人，由于航模遥控与串口 1 共用了引脚 PA9（串口 1TX 引脚），所以默认只能通过串口 1 进行小车控制，但是小车不发送自身的数据（即 24 字节数据帧）。若通过串口 1 发送小车数据，需手动注释航模初始化函数 TIM1_Cap_Init(0XFFFF,72-1);(底盘源码 system.c 的第 191 行附近)，以及取消串口 1 发送数据函数 USART1_SEND();的注释(底盘源码 usart.c 的第 30 行附近)。

同时我们不建议使用大型 ROS 科研机器人和 ROS 教育机器人的串口 1 进行 ROS 通信，原因是机器人的底盘使用串口 1 与 ROS 进行通信时 STM32 控制器重新上电后会出现卡死的情况，原因是串口 1 主要是用于底盘程序的烧录，重新上电后收到的数据帧会被当成烧录程序用的数据包。

3. 控制与反馈

本章主要介绍在 Windows 和 Ubuntu 系统下如何使用串口接收机器人底盘发出来数据以及对机器人底盘进行控制。

下文均使用 ROS 教育机器人底盘和大型 ROS 科研机器人底盘的 STM32 控制板上的串口 3 来演示不同系统下的上位机使用。

在使用上位机对机器人进行控制前，我们建议将机器人的电机使能开关关闭，以防出现上位机发送指令后小车出现乱跑的情况，照成不必要的损失。

ROS 教育机器人的电机使能开关处于 OFF 时电机失能。

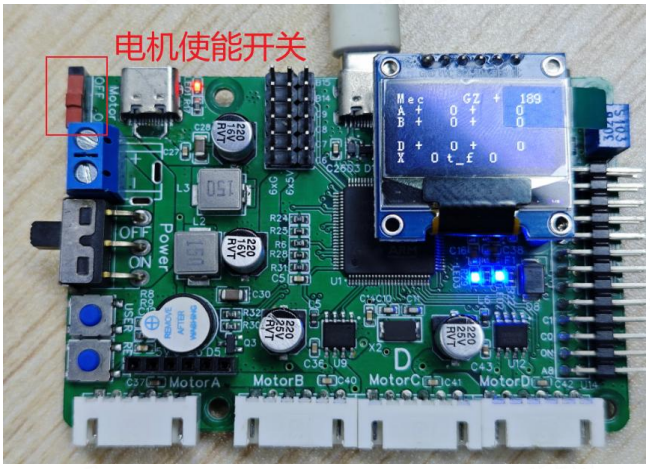


图 3-1　ROS 教育机器人电机使能开关

大型 ROS 科研机器人的电机使能开关处于 SW1 时电机失能。

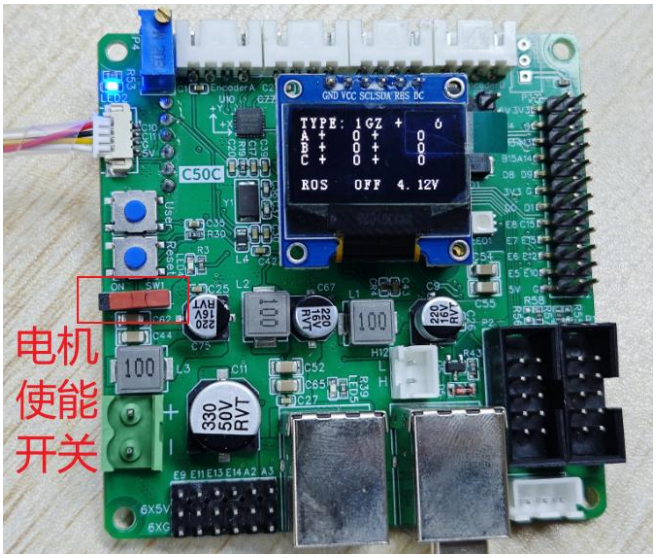


图 3-2　大型 ROS 科研机器人电机使能开关

3. 1 Windows 系统

① WHEELTEC—串口助手获取

在 Windows 系统下，用户可以使用我们提供的上位机：WHEELTEC—串口助手，来对机器人进行串口控制以及数据接收。

在您购买了我们的机器人产品之后，可以通过我们提供的资料来下载 WHEELTEC—串口助手软件。

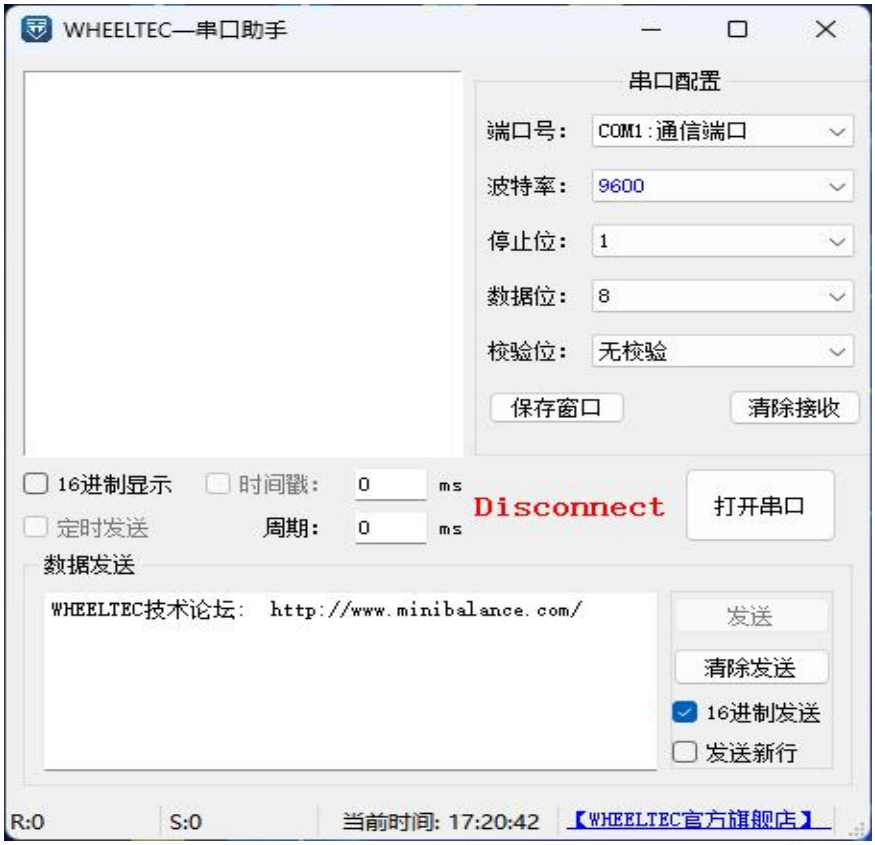


图 3-3 WHEELTEC—串口助手

② 串口助手接收数据

使用串口助手前，我们需要先安装对应的串口驱动，我们的机器人控制板是 USB 转 TTL 串口芯片是 CH9102 或 CH2102,这两款芯片对应的驱动我们也会在资料中为用户进行提供。用户安装完驱动后，使用数据线连接机器人控制板上的串口 3 和电脑的 USB 接口后可以在电脑控制面板的设备管理器中查询到对应的端口号。

图 3-4　设备管理器

接下来我们就可以打开 WHEELTEC—串口助手软件，按照下图 3-5 设置好串口助手后，点击打开串口就可以接收到小车发出的数据帧了。

图 3-5　串口助手接收数据图

对于上图中接收到的这一帧数据帧进行解析，含义如下：

完整的数据帧为 7B 00 00 9B 00 00 FF DF 00 60 00 0C 40 AB FF FD 00 06 00 1E 5B 87 82 7D 共 24 字节。

第 1 个字节：0x7B，帧头；

第 2 个字节：0x00，电机处于非停止状态；

第 3、4 个字节：X 轴速度，高 8 位 0x00(16 进制)=0000 0000(2 进制)、低 8 位 0x9B(16 进制)=1001 1011(2 进制),最高位为 0，正数(前进)，速度大小为

0*256+155=155(mm/s)，底盘当前 X 轴速度为 155mm/s；

第 5、6 个字节：Y 轴速度，高 8 位 0x00(16 进制)=0000 0000(2 进制)、低 8 位 0x00(16 进制)=0000 0000(2 进制),最高位为 0，正数(左移)，速度大小为 0*256+0=0(mm/s)，底盘当前 Y 轴速度为 0mm/s；

第 7、8 个字节：Z 轴速度，高 8 位 0xFF(16 进制)=1111 1111(2 进制)、低 8 位 0xDF(16 进制)=1101 1111(2 进制),最高位为 1，负数(顺时针旋转)，速度大小为 2^16(FF FF+1)-FF DF=00 21(16 进制)=(0*256+33)/1000=0.033(rad/s)，底盘当前 Z 轴速度为 0.033rad/s；

第 9、10 个字节：X 轴加速度，高八位 0x00(16 进制)=0000 0000(2 进制)，低 8 位 0x60(16 进制)=0110 0000(2 进制),最高位为 0，正数，加速度大小为 0*256+96=96，转化为 X 轴加速度 96/1672=0.0574（m/s^2）；

第 11、12 个字节：Y 轴加速度，高八位 0x00(16 进制)=0000 0000(2 进制)，低 8 位 0x0C(16 进制)=0000 1100(2 进制),最高位为 0，正数，加速度大小为 0*256+12=12，转化为 Y 轴加速度 12/1672=0.0071（m/s^2）；

第 13、14 个字节：Z 轴加速度，高八位 0x40(16 进制)=0100 0000(2 进制)，低 8 位 0xA8(16 进制)=1010 1000(2 进制),最高位为 0，正数，加速度大小为 64*256+168=16552，转化为 Z 轴加速度 16552/1672=9.8995（m/s^2）；

第 15、16 个字节：X 轴角速度，高八位 0xFF(16 进制)=1111 1111(2 进制)，低 8 位 0xFD(16 进制)=1111 1101(2 进制),最高位为 1，负数，角速度大小为 2^16(FFFF+1)-FFFD=00 02(16 进制)=0*256+2=2，转化为 X 轴角速度 2/3753=0.0004（rad/s）；

第 17、18 个字节：Y 轴角速度，高八位 0x00(16 进制)=0000 0000(2 进制)，低 8 位 0x06(16 进制)=0000 0110(2 进制),最高位为 0，正数，角速度大小为 0*256+6=6，转化为 Y 轴角速度 6/3753=0.0015（rad/s）；

第 19、20 个字节：Z 轴角速度，高八位 0x00(16 进制)=0000 0000(2 进制)，低 8 位 0x1E(16 进制)=0001 1110(2 进制),最高位为 0，正数，角速度大小为 0*256+30=30，转化为 Z 轴加速度 30/3753=0.0079（rad/s）；

第 21、22 个字节：电池电压，高八位 0x5B(16 进制)=0101 1011(2 进制)，低 8 位 0x87(16 进制)=1000 0111(2 进制),最高位为 0，正数，电池电压大小为

91*256+135=23431(mV)，转化为加速度 23431/1000=23.431V；

第 23 个字节：BBC 校验(前 22 字节异或校验)，0x82=0x7B^0x00^0x00^0xFF^0xDF^0x00^0x60^0x00^0x0C^0x40^0xA8^0xFF^0xFD^0x00^0x06^0x00^0x1E^0x5B^0x87；

第 24 个字节：0x7D,帧尾；

③ 串口助手发送数据

串口助手不仅可以通过串口接收数据，还可以发送数据。如下图 3-6，对串口助手的数据发送功能进行介绍。

首先我们将要发送的数据帧输入发送数据栏中，在将 16 进制发送的选项勾选上，点击发送就可以将数据发送出去了。

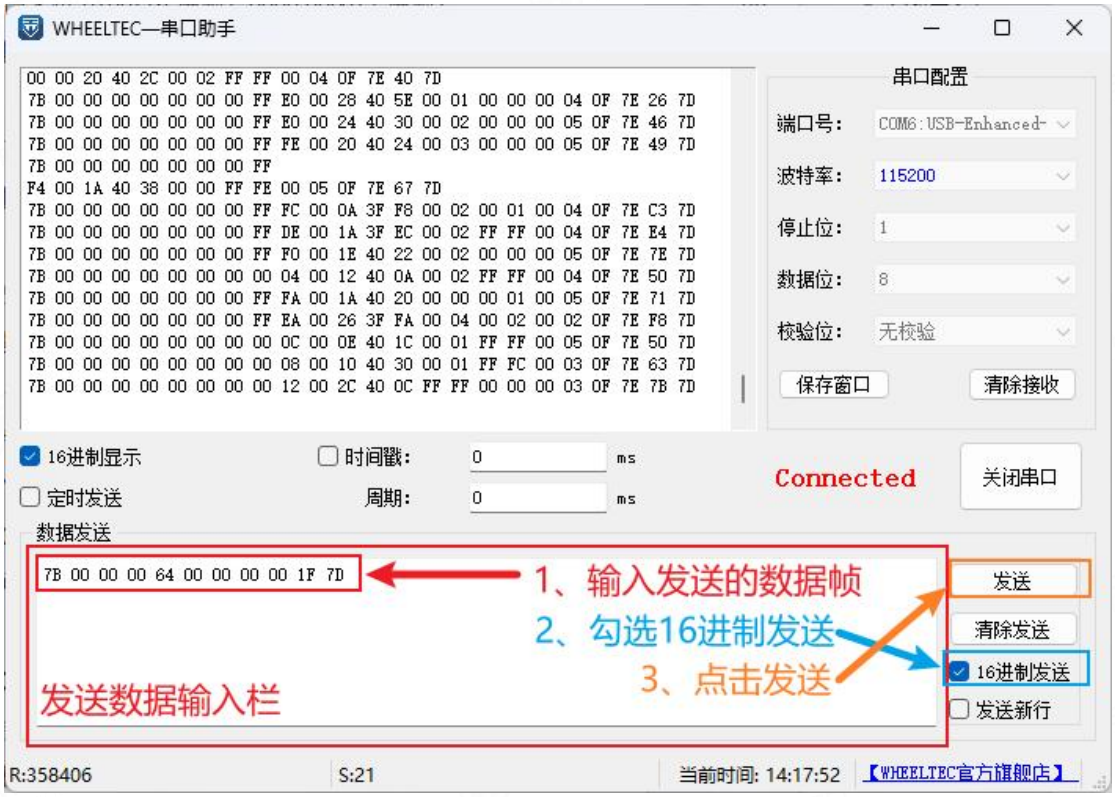


图 3-6　串口助手发送数据图

对于上图中发送到的这一帧数据帧进行解析，含义如下：

第 1 个字节：0x7B，帧头；

第 2、3 个字节：均为 0x00，预留位，数据无意义；

第 4、5 个字节：X 轴目标速度，高 8 位 0x00(16 进制)=0000 0000(2 进制)、低 8 位 0x64(16 进制)=0110 0100(2 进制),最高位为 0，正数(前进)，速度大小为 0*256+100=1(mm/s)，设置底盘当前 X 轴目标速度为 100mm/s；

第 6、7 个字节：Y 轴目标速度，高低位均为 0x00，设置底盘当前 Y 轴目标速度为 0(mm/s);

第 8、9 个字节：Z 轴目标速度，高低位均为 0x00，设置底盘当前 Z 轴目标速度为 0(mm/s);

第 10 个字节：BBC 校验(前 9 位字节异或校验)，0x1F=0x7B^0x00^0x00^0x00^0x64^0x00^0x00^0x00^0x00;

第 11 个字节：0x7D，帧尾。

这个数据帧的控制效果是机器人底盘会以 100mm/s 的速度向前移动，如果将机器人的电机使能开关关闭了，也可以从机器人上的 OLED 显示屏上观察到控制现象。如下图 3-7，以我们大型 ROS 科研机器人(差速车型)上的 OLED 显示屏为例，可以观察到显示屏上的电机目标速度变成了 100mm/s。

图 3-6　大型 ROS 科研机器人控制现象图

3.2 Ubuntu 系统

① cutecom 获取方法

在 Linux 系统下，用户可以通过安装 cutecom 来进行机器人底盘的串口收发数据。cutecom 是 Linux 系统下的图形界面形式的串口工具。用户可以在 Ubuntu 系统下使用终端来进行安装。

首先打开终端(ctrl+alt+T 键)，输入安装指令：

```
sudo apt-get insatll cutecom
```

此时会从 Ubuntu 的软件仓库中下载并安装 CuteCom。

图 3-7　cutecom 安装

接下来我们可以通过终端来打开 cutecom，输入以下指令：

```
Sduo cutecom
```

此时会有如下图 3-8 的一个窗口弹出，就说明我们的 cutecom 安装成功了。

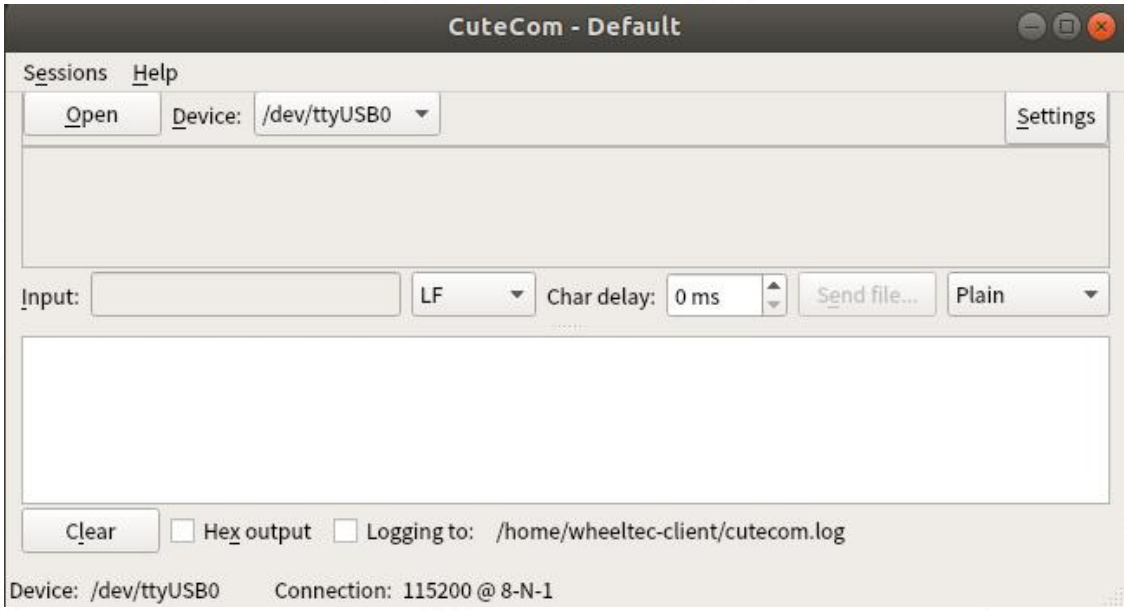


图 3-8　cutecom 界面图

② cutecom 接收数据

首先点击 cutecom 界面图右上角处的 Settings 按键打开串口设置界面，并按照下图 3-9 进行配置。配置好后点击 Open 按键与机器人底盘进行串口通信。

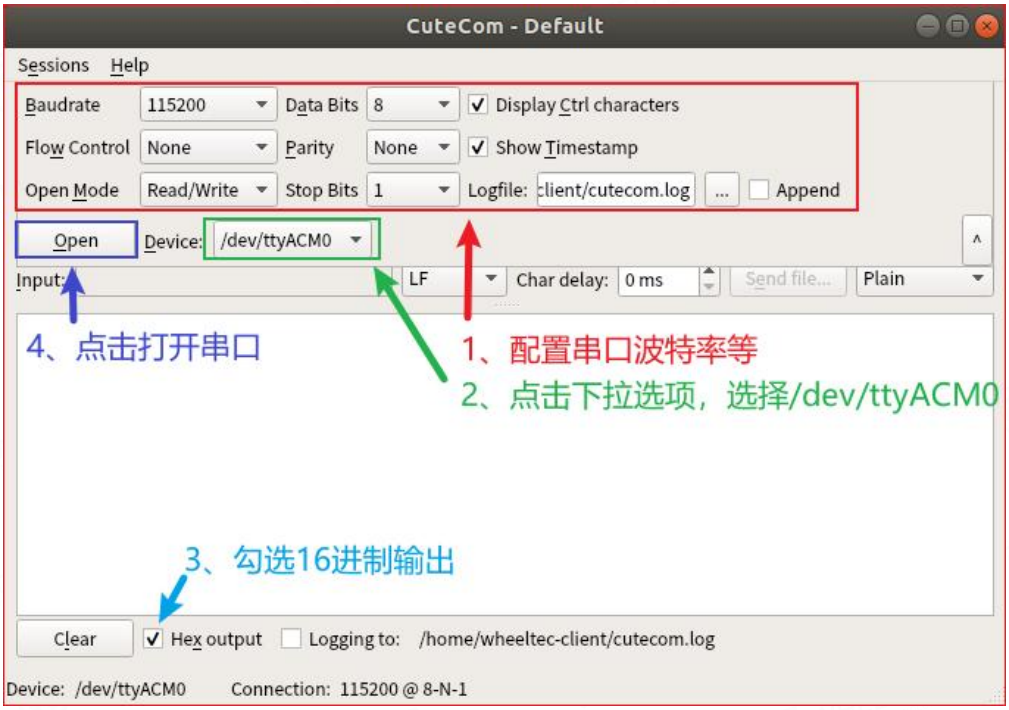


图 3-8 cutecom 配置图

打开串口后 cute 就能收到机器人底盘上发的数据了。

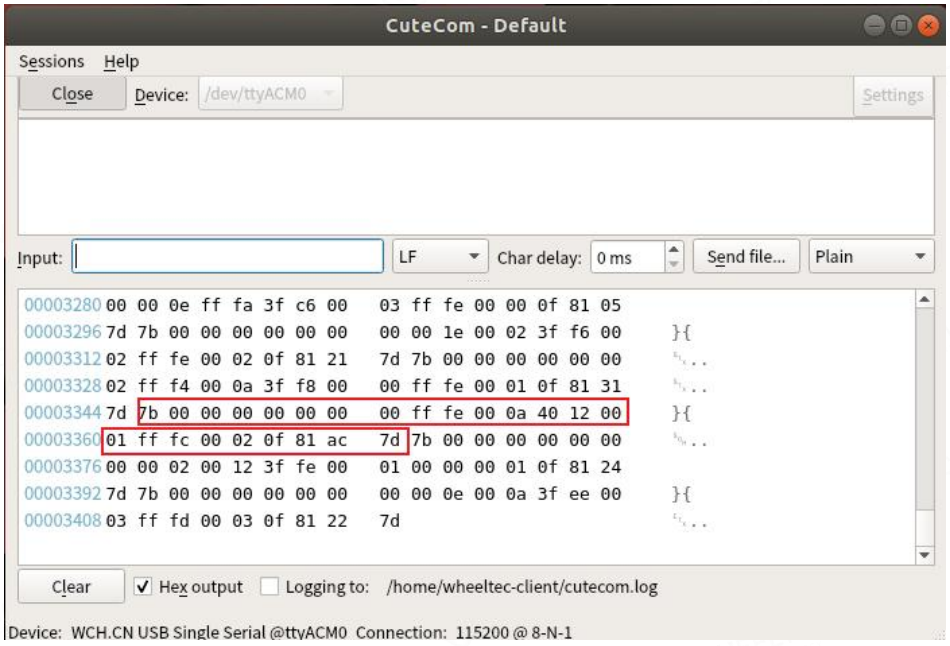
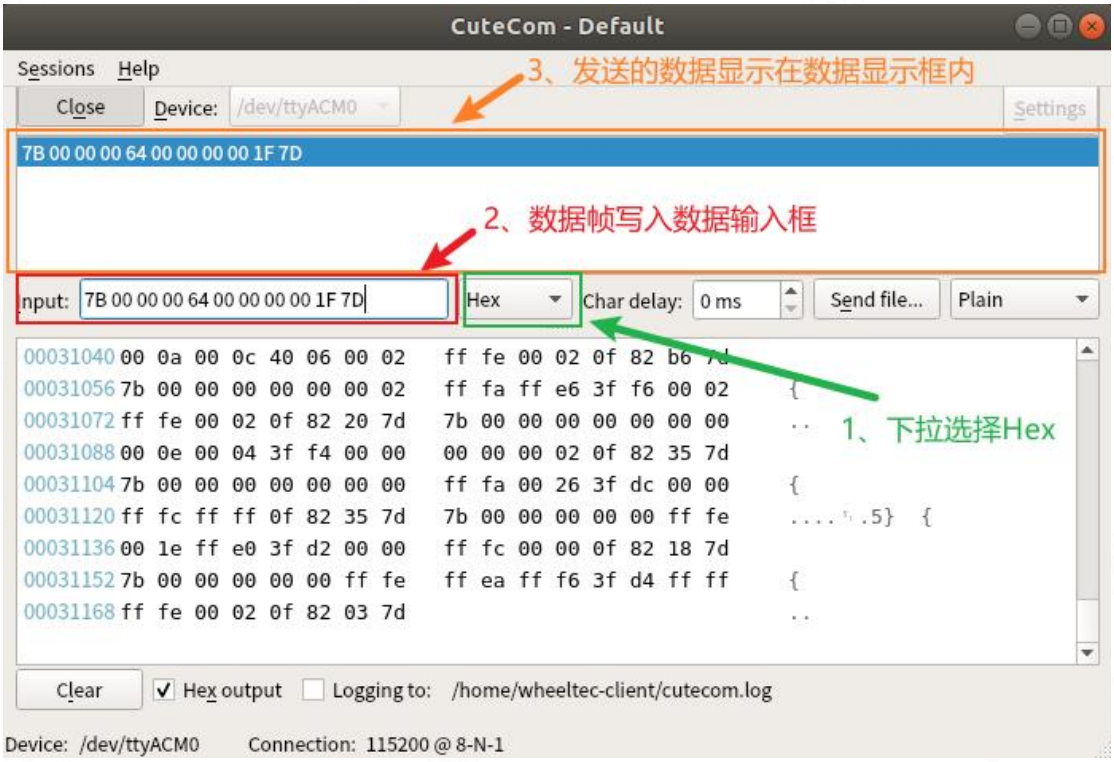


图 3-9 cutecom 接收数据

图 3-9 中的红框为收到的一帧数据帧。数据的具体含义这里不在进行相关介绍，具体的解算过程可以在章节 3.1 中串口助手接收数据有做详细的介绍。

③ cutecom 发送数据

cutecom 也可通过串口向机器人发送数据，将发送的数据格式选择为 Hex，接着把数据帧写入数据输入框内，敲下回车发送数据，数据会显示在数据显示框内。



数据发送成功的现象与串口助手发送数据成功的现象一样，详情可参考章节 3.1 中的串口助手发送数据。