轮 趣 科 技

STM32 运动底盘开发手册

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	2024/8/28	第一次发布

网址:www.wheeltec.net

序言

欢迎阅读《STM32 运动底盘开发手册》，本手册旨在为大型 ROS 科研机器人系列的开发提供全面而详细的指导。无论是机器人技术的爱好者，还是专业的教育研究人员，都能在本手册中找到宝贵的信息和资源。

本手册不仅阐述了 ROS 机器人运动底盘的基本原理和构成，还深入探讨了其运动学分析、通信协议和控制模式。通过本手册，读者将能够理解如何将理论应用于实践，设计和实现高效、可靠的机器人运动底盘。

本手册适用于以下几款大型科研机器人：两轮差速车、阿克曼车、麦克纳姆轮车、全向轮车和四驱车。每款机器人都有其独特的设计和运动特性，本手册将一一解析，确保读者能够根据具体需求选择合适的机器人底盘。

为了充分利用本手册，我们建议您具备以下基础知识：机器人学的基本原理、ROS 操作系统的基本概念、编程语言，尤其是 C 语言、基本的电子和电气工程知识。同时，本手册中的 ROS 部分的内容需要读者参考额外的教程和文档，如“ROS 相关的 ubuntu 基础教程”、“ROS 系列教程”和“ROS 使用与开发手册”等。

目录

序言 2
1. 引言 5
1.1 运动底盘与 ROS 的关系和作用 5
1.2 底盘坐标系说明 5
2. 运动底盘构成与说明 7
2.1 控制器、驱动器 7
2.2 编码器、电机和舵机 10
2.3 轮子 13
2.4 OLED 显示屏数据介绍 14
2.5 使能开关和按键功能说明 15
2.6 电位器功能说明 16
3. 不同类型底盘和运动学分析 17
3.1 运动学分析 18
4. 底盘电机的控制 19
4.1 轮子线速度计算 19
4.2 增量式 PI 控制器 20
5. 运动底盘控制模式 21
5.1 串口控制 21
5.2 CAN 控制 26
5.3 APP 遥控 27
5.4 PS2 手柄遥控 30
5.5 航模手柄遥控 30
6. 运动底盘状态数据反馈 33
6.1 串口控制的数据反馈 33
6.2 CAN 控制的数据反馈 37

6.3 里程计的作用与数据说明 39
6.4 IMU 的作用与数据说明 40
7. 运动底盘相关的流程图 42
7.1 机器人硬件框图 42
7.2 STM32 程序流程图 43
7.3 电机控制流程图 44
8. 程序烧录下载 46
8.1 串口下载 46
8.2 SWD 下载 48
9. 注意事项 50
9.1 关于代码 50
9.2 关于电控板上的电源接口 50
9.3 电机使用和保护 50
9.4 电池使用和维护 50

1. 引言

在本章中，我们将深入探讨机器人操作系统（ROS）与运动底盘之间的协同工作方式及其在机器人设计中的重要性，并且说明运动底盘使用的坐标系。

1.1 运动底盘与 ROS 的关系和作用

在机器人技术领域，ROS（机器人操作系统）提供了一个强大的中间件，它允许不同硬件和软件组件之间进行高效的通信和协同工作。在这种框架下，运动底盘不仅是机器人的移动平台，而且是一个关键的通信节点，通过串口与 ROS 环境进行数据交换。

运动底盘与 ROS 的关系和作用主要有以下内容：

1） **通讯节点**：运动底盘作为 ROS 的一个节点，运行在 ROS 网络中，负责发送和接收与机器人运动相关的消息。

2） **数据集成**：运动底盘节点集成了里程计、IMU（惯性测量单元）等传感器数据，提供了机器人状态的实时反馈。这些数据对于机器人的定位、导航和运动规划至关重要。

3） **控制执行**：运动底盘根据 ROS 中的指令（如速度命令、转向角度等）执行相应的控制动作。它将高级控制策略转化为具体的电机驱动信号，实现机器人的精确移动。

这种集成方式的优势在于运动底盘通过 ROS 的通信机制可以轻松地与其他 ROS 节点集成，提供高度灵活的机器人设计。并且随着项目的发展，可以简单地添加或替换 ROS 节点，以增强或修改机器人的功能，而无需对运动底盘进行大规模的重新设计。

在本手册中，将详细介绍我司设计的 STM32 运动底盘，这包括通信协议的定义、数据包的结构、以及如何在与上位机（ROS 等）的通信中解析和处理来自运动底盘的数据。

1.2 底盘坐标系说明

坐标系在机器人学中是描述机器人位置、方向和运动的基础。运动底盘使用的坐标系对于 ROS 系统中的数据处理和命令执行至关重要。如图 1-1 所示，为

坐标系示意图。对于我司的大型 ROS 科研机器人底盘均遵循该坐标系：

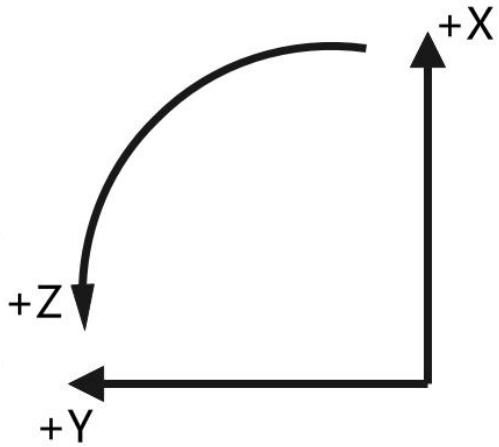


图 1-1 坐标系示意图

底盘的正前方为 X 轴的正半轴，底盘的正左方为 Y 轴的正半轴，而底盘逆时针方向为 Z 轴的正半轴。

2. 运动底盘构成与说明

机器人的构成通常分为三个关键部分：上层决策部分，它对算力有较高要求但对实时性要求不高，主要运行在电脑操作系统上，使用多种编程语言进行复杂决策的制定；环境感知部分，由激光雷达、深度相机、惯性导航等传感器组成，负责收集并反馈机器人周围环境的信息；底层执行部分，对算力要求较低但对实时性要求极高，通常在单片机上运行，使用 C 语言来执行上层决策的控制命令，并实时上传机器人的状态信息。本章要讲解的就是机器人的底层执行部分是如何构成的，如图 2-1 所示，即是大型 ROS 科研机器人底盘（阿克曼）的构成。

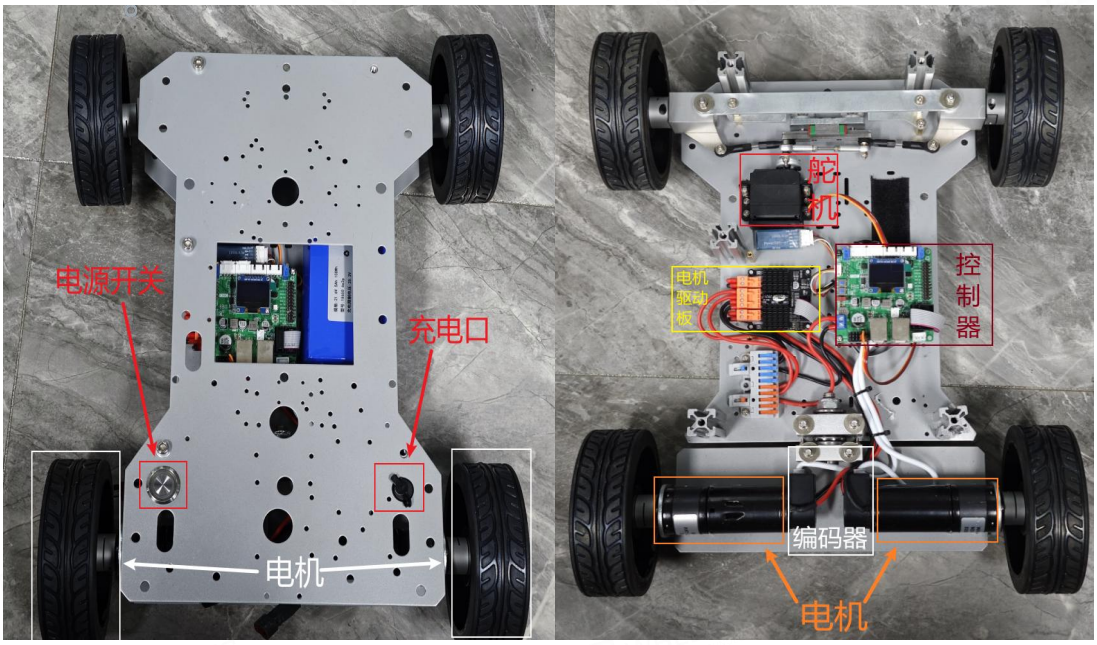


图 2-1 大型 ROS 科研机器人底盘（阿克曼）

底盘由电控板、舵机、编码器电机和轮子等主要部件组成。

2.1 控制器、驱动器

我司的大型 ROS 科研机器人运动底盘电控板上的 MCU 使用的是 STM32F407VET6 芯片如图 2-2 和图 2-3 所示，为运动底盘上电控板的接口详解图。

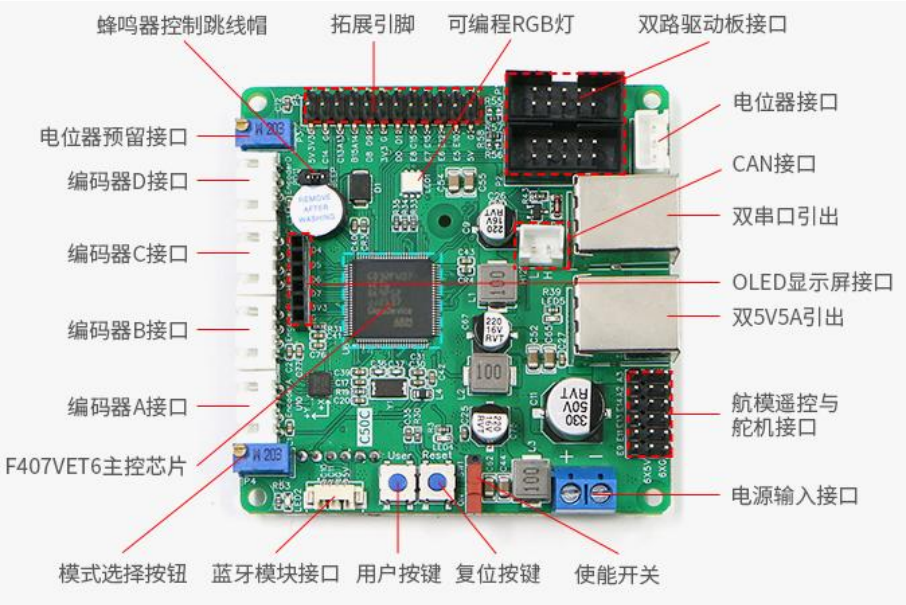


图 2-2 大型 ROS 科研机器人底盘控制板(正面)

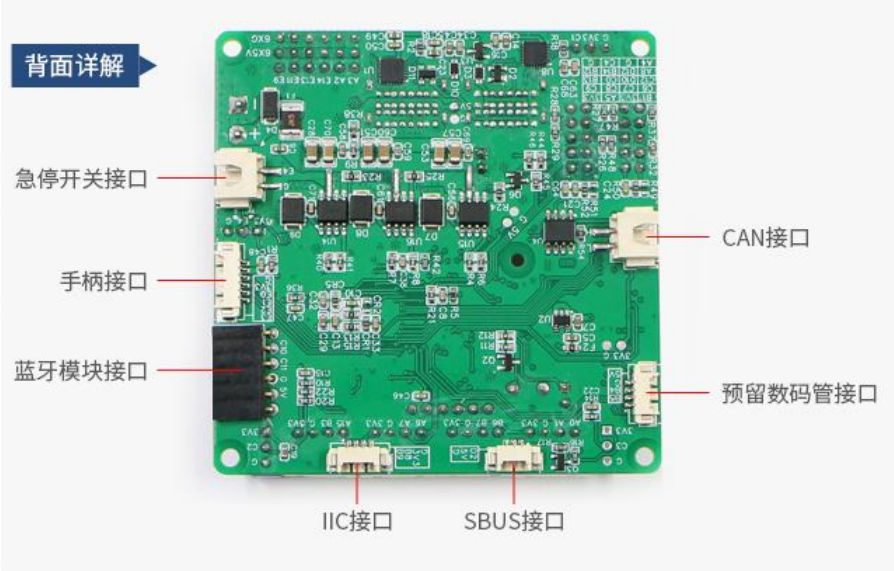


图 2-3 大型 ROS 科研机器人底盘控制板(背面)

① 控制器

控制器（STM32F407VET6）是机器人电控板的核心部件，负责接收上位机的指令并将其转换为电机的驱动信号。控制器（STM32F407VET6）的设计通常需要考虑信号的稳定性和实时性，确保机器人能够准确执行各种运动指令。在大型 ROS 科研机器人底盘中，电控板不仅负责运动控制，还集成了 MPU6050 六轴 IMU，能够实时监测并反馈机器人的运动状态和姿态信息，为上位机提供精确的数据支持。

② 驱动器

大型 ROS 科研机器人底盘的 STM32 控制器上没有板载电机驱动芯片，因此底盘上驱动电机使用的是我司配套的双路电机驱动板 D50A，如图 2-4 所示，为我司的双路电机驱动板。

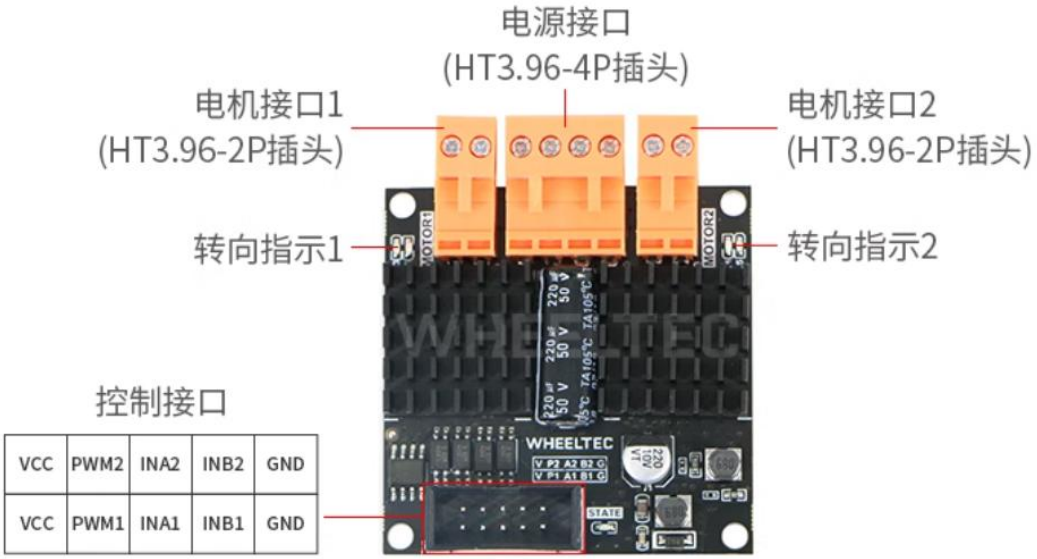


图 2-4 双路电机驱动板

驱动器是连接控制器和电机的桥梁，它将控制器输出的信号转换为电机所需的电流和电压，驱动电机按照预定的速度和方向运动。

我司配套的双路电机驱动板提供了防反接的牛角座信号接口，可以有效防止信号口接错导致驱动板烧毁，仅需使用我司配套的排线直接与 STM32 控制器上的双路驱动板接口连接即可。

接线说明：

在大型 ROS 科研机器人上，不同的车型会有搭配不同数目的驱动板，而驱动板不仅要与 STM32 控制器连接还要与电机相连接，接线较为复杂，如图 2-5 所示，为麦轮底盘上控制器与电机驱动器的连接方法。其他车型的连接方法用户可通过我司提供的资料包【WHEELTEC R550 PLUS 大型 ROS 科研机器人系列资料\1.硬件说明与底盘问题排查】进行相关的了解。

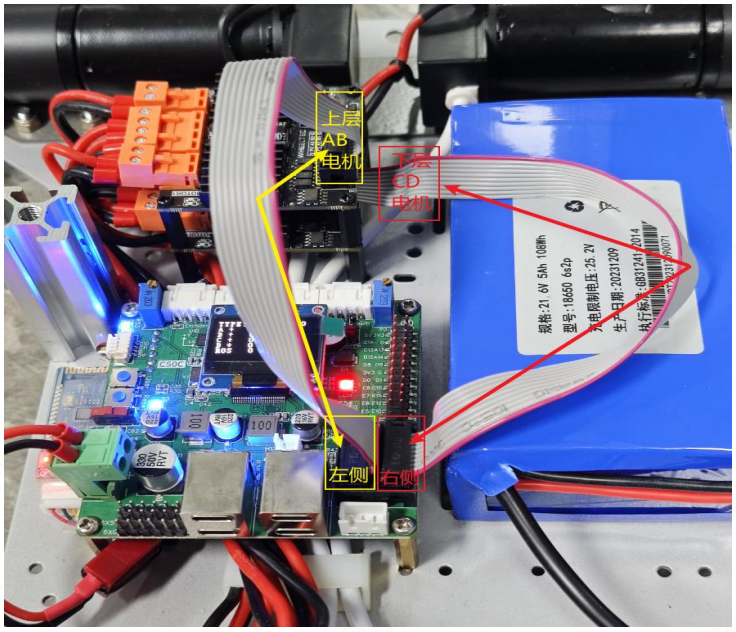


图 2-5 麦轮底盘控制板与驱动板连接示意

③ **注意事项**

我司的大型 ROS 科研机器人底盘电控板预留有非常丰富的接口、开关和外设，拥有极强的扩展性，板上还集成有型号为 MPU6050 的 6 轴 IMU，可以直接为上位机提供底盘的运动、姿态信息等，电控板上预留有 4 路编码器接口，接上任意电机上的编码器接口即可读取编码器数值。（使用非我司购买的电机时需确认引脚是否对应，可以通过查阅我司提供的资料包中的电控板原理图进行确认）

电机与双路电机驱动板的接线会因为车型的不太而有些许不同，用户可根据不同的车型查阅我司提供的资料包【WHEELTEC R550 PLUS 大型 ROS 科研机器人系列资料\1.硬件说明与底盘问题排查】进行接线内容的相关了解。

2.2 编码器、电机和舵机

本节旨在简单地介绍我们大型 ROS 科研机器人底盘中电机和舵机的基本工作原理，以便用户对底盘的运动机制有一个初步的认识。电机和舵机更加具体的使用方法和控制教程可查阅【WHEELTEC ROS 机器人通用资料\3.STM32 底层源码讲解教程\0-2.电机控制基础视频教程】进行相关的了解。

① **编码器**

编码器是机器人中的重要组成部分，用于测量并反馈电机的转速和位置信息。机器人底盘的电机编码器常见的有霍尔编码器和 GRM 编码器，编码器能够实时提供电机精确的速度和位置数据。这些数据对于闭环控制系统至关重要，它们使

得电机驱动器能够准确地调整电机的运行状态，以满足上位机发出的指令。

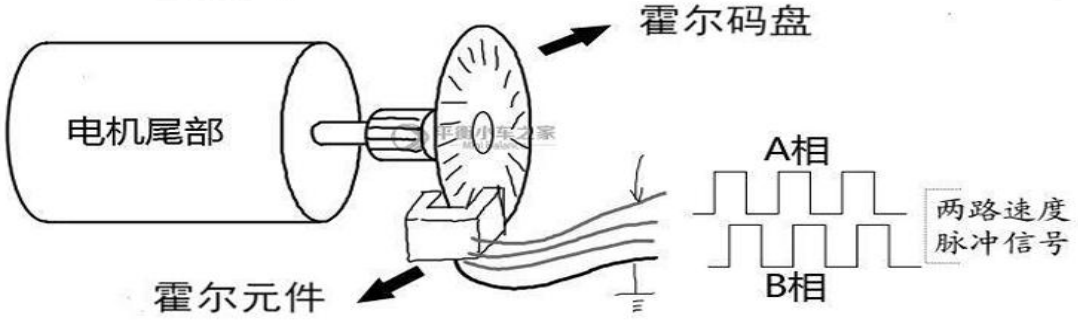


图 2-6 霍尔编码器工作原理

如图 2-6 所示，霍尔编码器是由霍尔码盘和霍尔元件组成。霍尔码盘是在一定直径的圆板上等分地布置有不同的磁极。霍尔码盘与电动机同轴，电动机旋转时，霍尔元件检测输出若干脉冲信号，为判断转向，一般输出 A 相和 B 相存在 一定相位差的方波信号。可以通过单片机采集 AB 相的脉冲数据，根据脉冲换算 得到电机位置数据。

GMR 编码器是一种使用磁阻效应进行测量的编码器。它利用了在磁性材料上施加外加磁场时，磁矩在自旋转动的过程中会产生电阻值变化的现象。利用这种效应，可以测量磁性标尺的位移，并输出相应的电信号。GMR 编码器一般有 A、B、Z 三根输出信号线。其中 A、B 是用来获取位置信息的正交编码信号，Z 信号则是用于标记旋转一周的起始位置。由于在实际使用中我们没有使用到 Z 信号，所以在设计时没有将其引出，只使用到了 A、B 两相信号。GMR 编码器具有高分辨率、高精度、低温漂移等特点。

编码器的分辨率是由编码器的线数决定的。例如：霍尔编码器的线数为 13 线，那么码盘旋转一圈会产生 13 个脉冲信号。GMR 编码器的线数为 500 线，这表示编码器的轴每旋转一圈，会输出 500 个脉冲信号。编码器的 AB 两相各有 13/500 个脉冲，通过 AB 两相的相位差可以判断旋转的方向，具体编码器的知识指引去看 ROS 机器人通用资料的对应内容【WHEELTEC ROS 机器人通用资料\3.STM32 底层源码讲解教程\0-2.电机控制基础视频教程\2.PID 基础入门：编码器的使用教程与测速原理】

② 电机

图 2-7 MD60 直流减速电机

电机是机器人运动的动力来源，负责将电能转换为机械能，驱动机器人底盘移动。在大型 ROS 科研机器人底盘中，电机采用高性能的直流电机，它们具有响应速度快、控制精度高的特点。电机安装编码器后与驱动器相连，编码器能够实时反馈电机的转速和位置信息，这些信息对于实现精确的速度控制和位置控制至关重要。

减速比是描述电机输出轴转速与输入轴转速之间关系的一个参数。具体来说，它是指电机内部通过齿轮机构实现的转速降低的比例。例如：一个常见的 1:30 减速比意味着电机的转子（输入轴）需要转动 30 圈，输出轴（负载轴）才会转动 1 圈。

图 2-8 MD36 电机详解图

如图 2-8 所示，为直流有刷减速电机 MD36 的组成，其中的编码器与电机的转子（输入轴）相连接，电机的输出轴与经过减速齿轮组后与电机的转子相连接，减速齿轮组可以提供较大的力矩和较低的速度。用户可以通过查阅我司一个的资料【WHEELTEC ROS 机器人通用资料\3.STM32 底层源码讲解教程\0-2.电机控制基础视频教程\1.PID 基础入门：直流电机与 TB6612】进行电机相关知识的学习。

③ 舵机

大型 ROS 科研机器人底盘中，舵机应用于阿克曼底盘上，用于调整阿克曼底盘的转向角度，舵机的精确角度控制使阿克曼底盘拥有灵活的转向控制。

舵机简单的说就是集成了直流电机、电机控制器和减速器等，并封装在一个便于安装的外壳里的伺服单元。能够利用简单的输入信号比较精确的转动到给定角度的电机系统。

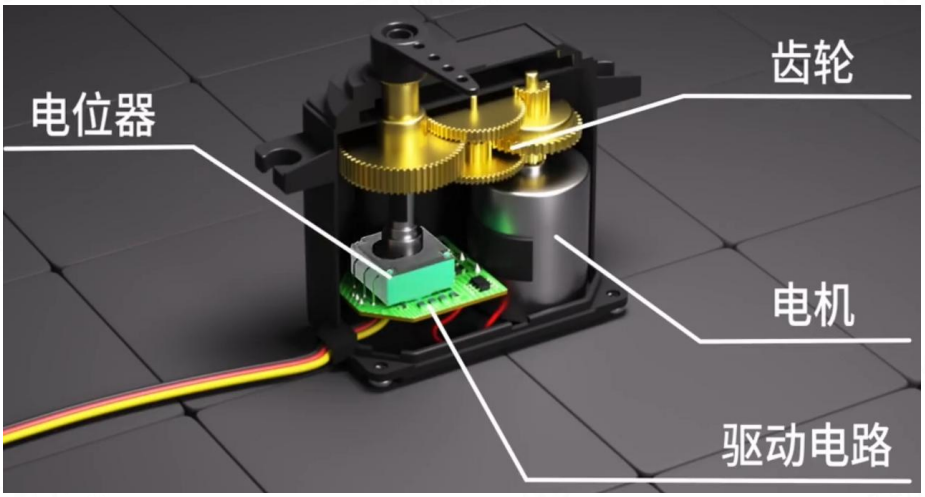


图 2-9 舵机详解图

舵机的主体结构如图 2-9 所示，主要有几个部分：电位器、减速齿轮组、电机、驱动电路。简单的工作原理是驱动电路接收信号源的控制信号，并驱动电机转动；齿轮组将电机的转速成大倍数缩小，并将电机的输出扭矩放大响应倍数，然后输出；电位器和齿轮组的末级一起转动，测量舵机轴转动角度；电路板检测并根据电位器判断舵机转动角度，然后控制舵机转动到目标角度或保持在目标角度。舵机的基本使用原理用户可通过我司提供的资料【WHEELTEC ROS 机器人通用资料\3.STM32 底层源码讲解教程\0-2.电机控制基础视频教程\8.PID 基础入门：舵机的基本原理】进行相关的学习。

2.3 轮子

我司设计的大型 ROS 科研机器人是轮式机器人，轮子作为大型 ROS 科研机器人与地面接触的唯一部分，对机器人的移动性能有着直接的影响。我们需要根据不同的应用需求选择不同的轮子。轮子的参数会影响底盘的性能和运动学分析时的部分参数，因此选择合适的轮子对于确保机器人的稳定性、负载能力和通过性至关重要。不同类型的轮子、底盘结构决定了运动底盘的运行场景和特点，也

决定了其运动学分析公式，因此对于轮子的线速度计算将留在第 4 章的底盘电机的控制进行讲解。

2.4 OLED 显示屏数据介绍

在我司的底盘电控板上预留有 OLED 显示屏接口，同时也配备了 OLED 显示屏用于显示底盘的当前的一些状态信息。

不同的底盘显示屏的内容大同小异，具体如文下所示：

① 两轮差速底盘

TYPE: 1 BIAS - 2	第一行：车型和陀螺仪零点偏差值
GYRO Z - 40	第二行：Z 轴角速度的值
L: + 0 + 0	第三行：左电机目标值和实际值（mm/s）
R: + 0 + 0	第四行：右电机目标值和实际值（mm/s）
MA: + 0 MB: + 0	第五行：左右电机的 PWM 值
ROS ON 22.03V	第六行：控制模式 使能开关 电池电压

图 2-10 两轮差速底盘 OLED 显示屏内容

② 阿克曼底盘

TYPE: 1 BIAS - 5	第一行：车型和陀螺仪零点偏差值
GYRO Z - 60	第二行：Z 轴角速度的值
L: + 0 + 0	第三行：左电机目标值和实际值（mm/s）
R: + 0 + 0	第四行：右电机目标值和实际值（mm/s）
SERVO: + 0	第五行：舵机 PWM 控制值
ROS ON 22.03V	第六行：控制模式 使能开关 电池电压

图 2-11 阿克曼底盘 OLED 显示屏内容

③ 全向轮底盘

TYPE: 1 BIAS - 2	第一行：车型和陀螺仪零点偏差值
+ 0 + 0	第二行：A 电机目标值和实际值（mm/s）
+ 0 + 0	第三行：B 电机目标值和实际值（mm/s）
+ 0 + 0	第四行：C 电机目标值和实际值（mm/s）
MoveZ: + 10	第五行：底盘 Z 轴方向的运动量
ROS ON 22.03V	第六行：控制模式 使能开关 电池电压

图 2-12 全向轮底盘 OLED 显示屏内容

④ 四驱/麦轮底盘

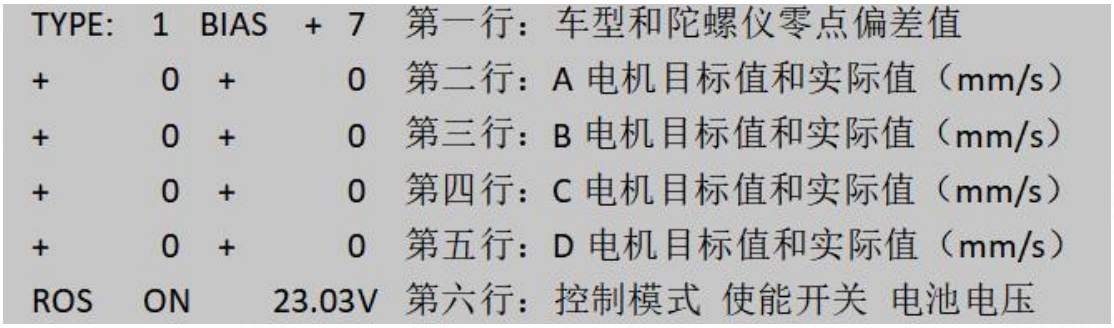


图 2-13 四驱/麦轮底盘 OLED 显示屏内容

⑤ 注意事项

底盘拥有多种控制模式，各个模式在的 LED 显示屏左下角显示的内容如下表 2-2 所示：

表 2-2 底盘 OLED 显示屏显示控制模式

控制模式	CAN	蓝牙 APP	PS2 手柄	航模遥控	串口 1	串口 3
显示内容	CAN	APP	PS2	R-C	USART	ROS

2.5 使能开关和按键功能说明

① 使能开关

如图 2-2 所示，使能开关在机器人 STM32 控制器的左上角，使能开关在“ON”状态下机器人的电机才使能控制。程序里面考虑到机器人的稳定性，STM32 控制器上电之后 10s 内电机是强制失能状态。这个时候即使您的使能开关是在“ON”状态， 也是显示“OFF”，10s 后就会更新成“ON”，需要注意的是在程序中设置了这 10s 内串口 1 和串口 3 不接收数据。

② 用户按键

如图 2-2 所示，用户按键位于 STM32 控制器的左下角，按键的旁边有丝印“USER”，大型 ROS 科研机器人的任意车型双击用户按键可以更新陀螺仪的零点。

③ 复位按键

如图 2-2 所示，复位按键位于 STM32 控制器的左下角，按键的旁边有丝印“RESET”，大型 ROS 科研机器人的任意车型单击复位按键可以重启 STM32 控制器。

④ 蓝牙模块接口

如图 2-2、2-3 所示，在 STM32 控制器的正反面均有一个蓝牙模块接口，这两个接口都是接的 STM32F407VET6 的串口 2，属于同一个外设接口。

⑤ **急停开关**

如图 2-3 所示，在 STM32 控制器的背面有一个急停开关接口，该接口接的是底盘表面的软件急停开关。如图 2-14 所示，为两轮差速底盘的软件急停开关，该开关由于底盘的机械结构限制仅部分车型拥有。

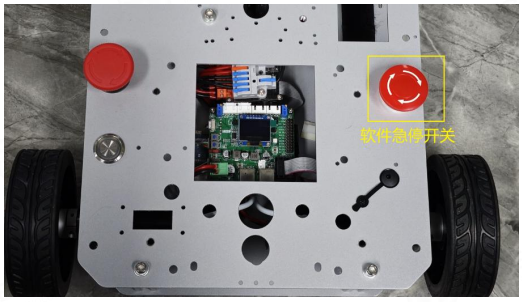


图 2-14 差速底盘软件急停开关

2.6 电位器功能说明

如图 2-2 所示，在大型 ROS 科研机器人电控板上有两个电位器，这两个电位器各有不同的作用。

① **车型选择电位器**

我司的大型 ROS 科研机器人同一种类型的底盘有不同型号的底盘，不同型号的底盘在硬件参数上会有不同，因此我们在电控板上设计了一个电位器用于车型选择。如图 2-2 所示，位于左下角的模式选择按钮即使底盘的车型选择电位器。可以通过旋转电位器进行车型选择，底盘当前的车型可以通过 OLED 显示屏的 TYPE 值进行确认（可参考 2.4 OLED 显示屏数据介绍进行了解），选择对应的 TYPE 值后需要给底盘重新上电才能生效。

② **舵机调整电位器**

如图 2-2 所示，位于电控板的左上角有一个预留的电位器，该电位器在大型 ROS 科研机器人阿克曼底盘上用于调节舵机的中值，当阿克曼底盘上电后舵机无法回正时，可以旋转该电位器进行中值调节，具体的调节方法可以参考我司提供的资料包【WHEELTEC R550 PLUS 大型 ROS 科研机器人系列资料\1.硬件说明与底盘问题排查\4.阿克曼底盘的常见问题处理】进行相关的了解。

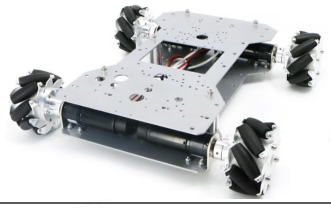
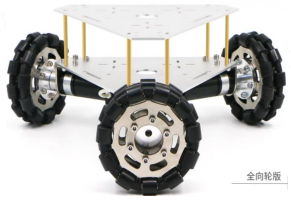
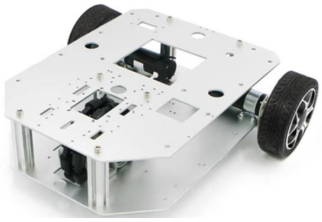
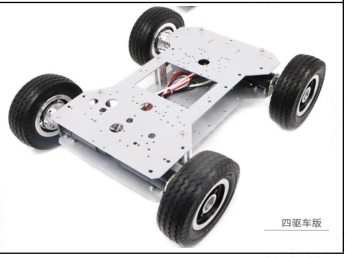
3. 不同类型底盘和运动学分析

通过前文的学习，我们已经了解了底盘的基本构成和使用方法。本章我们将学习不同类型底盘的运动学分析，这有助于我们理解底盘上各个轮子的线速度是如何与底盘的整体移动和旋转速度相关联的。

因为机器人的控制流程通常都是通过上位机给运动底盘下发三轴目标速度，底盘再通过运动学分析算出各个电机上轮子的线速度，从而驱动电机来使用底盘达到目标三轴速度，所以学习运动学分析是必要的。

首先，我司的大型 ROS 科研机器人底盘有以下几种类型。如表 3-1 所示，为不同类型的运动底盘。

表 3-1 不同类型的运动底盘表

底盘类型与特点	前后移动	左右横移	自旋转	负载能力	转向摩擦力	底盘实物
麦轮底盘	是	是	是	相对较低	相对较低	
全向轮底盘	是	是	是	相对较低	相对较低	
两轮差速底盘	是	否	是	相对较低	相对较低	
四驱底盘	是	否	是	相对较高	相对较高	

阿克曼底盘	是	否	否	相对较高	相对较低	

其次，将底盘的目标速度转换成电机的目标速度这个过程叫“运动学分析”，运动学分析又分为正解和逆解，在运动学分析之前先来分别解释一下什么是运动学正解和运动学逆解：

①运动学正解：通过底盘的各轮速度求出底盘 X 轴、Y 轴和 Z 轴方向的速度。

②运动学逆解：通过底盘 X 轴、Y 轴和 Z 轴方向的速度分别求出底盘各轮的速度。

明确运动学分析的含义和作用后，就可以开始学习不同类型底盘的运动学分析过程。

3.1 运动学分析

关于不同运动底盘的运动学分析推导，大家请到【1.WHEELTEC ROS 机器人通用资料\2.运动底盘的控制与运动学解析】下查看学习。

4. 底盘电机的控制

为了确保大型 ROS 科研机器人底盘能够精准响应速度指令并保持稳定运行，我们需要对底盘的电机进行闭环控制，闭环控制可以使电机的运行状态更加符合预期，提高整个机器人系统的稳定性和可靠性。

底盘的三轴目标速度经过运动学分析后获得轮子的目标速度后还需要驱动电机使轮子的的线速度达到目标值。我司的大型 ROS 科研机器人底盘上的电机安装有编码器，可以通过读取编码器的数据来获得电机的转速，从而解算出轮子的实时线速度。

在底盘的程序中，我们对电机的闭环控制使用的是 PID 控制算法，PID 是一种广泛应用于工业和机器人领域的经典控制算法，它的名称来源于其三个主要调节环节的首字母：比例（Proportional）、积分（Integral）、微分（Derivative）。本章中我们主要了解如何计算底盘轮子的线速度和了解程序使用到的速度 PI 控制器。

4.1 轮子线速度计算

底盘的电机装有编码器，因此我们可以获取到编码器的脉冲信号。编码器的脉冲信号可以用来测量电机的转速，进而计算轮子的线速度。以下是轮子线速度计算方法：

线速度=每秒编码器计数*轮子周长/编码器线数/电机减速比

上述内容转化为底盘 STM32 控制器的 C 语言程序如下：

```
MOTOR_A.Encoder=Encoder_A_pr*CONTROL_FREQUENCY*Wheel_perimeter/Encoder_precision;
```

Encoder_A_pr 是编码器原始数据；

CONTROL_FREQUENCY 是控制频率（单位：HZ）；

Encoder_precision 是编码器精度（与电机机械结构和编码器芯片有关）；

wheel_perimeter 是轮子的周长（单位：米）；

MOTOR_A.Encoder 就是机器人的电机 A 上的轮子实时线速度（单位：米/秒）；

这里只展示 A 电机的轮子线速度计算，其余的 B、C、D 电机的轮子线速度计算同理。

4.2 增量式 PI 控制器

通过运动学分析得到电机的目标速度，又通对电机上编码器数据的读取和转化得到了电机的实时速度。我们需要把这个目标值以及实际值送入 PID 控制器进行速度闭环控制，使得电机的实际输出速度趋近于目标值。程序中的 PI 控制器源码如下所示：

```
/**************************************************************************
*  函数功能：增量 PI 控制器速度控制
*  入口参数：Encoder：编码器测量值、Target：目标速度
*  返 回 值：Pwm：电机 PWM
*  函数说明：pwm+=Kp[e（k）-e(k-1)]+Ki*e(k) 增量式 PI 控制
**************************************************************************/
int Incremental_PI_A (float Encoder,float Target)
{
     static float Bias,Pwm,Last_bias;
     Bias=Target-Encoder;                  //计算偏差
     Pwm+=Velocity_KP*(Bias-Last_bias)+Velocity_KI*Bias;   //增量式 PI 控制器
     if(Pwm>7200)Pwm=7200;
     if(Pwm<-7200)Pwm=-7200;
     Last_bias=Bias;                       //保存上一次偏差
     return Pwm;                           //增量输出
}
```

PID 算法的详细内容我们此处不进行扩展，仅做简单的介绍。对 PID 算法感兴趣的小伙伴可以通过查阅我司提供的资料包中【WHEELTEC ROS 机器人通用资料\3.STM32 底层源码讲解教程\0-2.电机控制基础视频教程】进行相关内容的学习。

5. 运动底盘控制模式

通过前文的学习，我们知道了底盘的工作原理以及如何将底盘的三轴目标速度转化为底盘各个电机的速度并使用 PID 进行闭环控制。接下来我们要学习的是大型 ROS 科研机器人底盘的各种控制模式。底盘支持 APP 遥控、PS2 遥控、航模手柄遥控、CAN 控制、串口控制，其中串口控制还可以区分为 ROS 控制模式和 USART 控制模式。底盘开机后控制模式会的 OLED 显示屏的左下角，默认开机使用 ROS 控制模式。

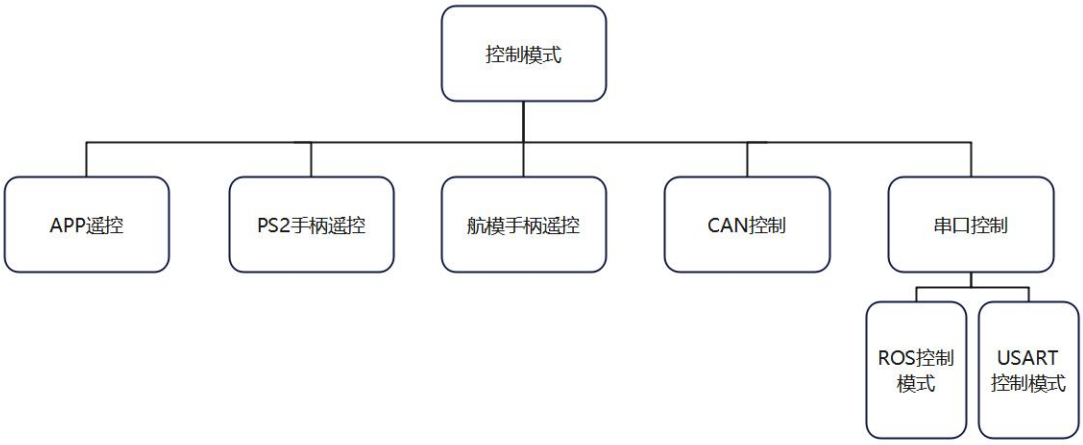


图 5-1 控制模式图

对于图中这几种控制模式，我司均有单独提供详细的视频教程以及相关资料，用户可以通过我司提供的【WHEELTEC ROS 机器人通用资料\2.运动底盘的控制使用】资料包获取相关资料。

本章仅对如何使用以上几种控制模式对底盘进行控制展开讲解，其中串口控制和 CAN 控制还包含有底盘的状态数据反馈，这一部分用户需通过第 6 章底盘状态数据反馈的学习才能完全了解，本章中底盘的状态数据反馈的内容不进行赘述。

5.1 串口控制

串口控制在机器人中是一种常见的控制方式，它允许通过串行通信接口发送指令来控制底盘。这种方式在 ROS 系统中尤为常见，因为它可以方便地与 ROS 节点进行集成。而在大型 ROS 科研机器人底盘上，串口控制中就包含了 ROS 控制模式和 USART 控制模式，这两者其实都是串口通信且波特率为 115200，差别在于使用的串口和应用途径不同。

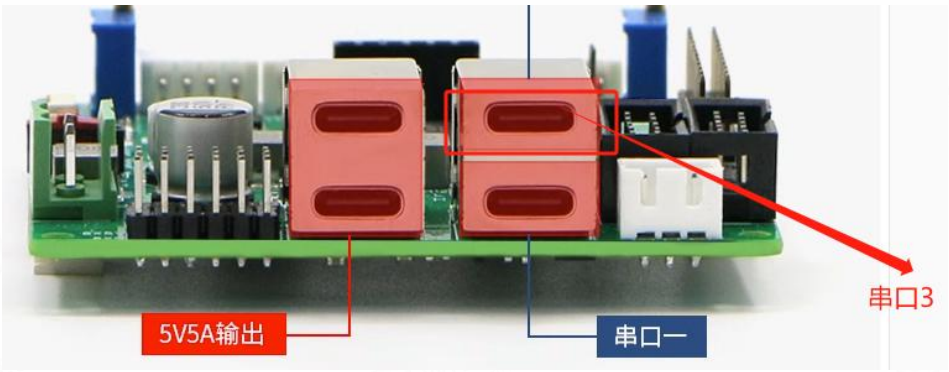


图 5-1　大型 ROS 科研机器人串口位置图

如图 5-1 所示，ROS 控制模式使用的是串口 3，而 USART 控制模式使用的是串口 1。

① ROS 控制模式

ROS 控制模式是上位机通过电控板的串口 3 下发速度指令来控制底盘运动。首先我们需要了解的就是串口通信时下发的数据帧（下行数据帧）格式，只有了解该数据帧格式我们才能向底盘下发我们期望的速度控制指令去控制底盘运动。

大型 ROS 科研机器人运动底盘通过串口 3 接收到的数据包格式，如下表所示：

表 5-1 下行数据包格式表

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		Uint8	Uint8
占用字节	1	1	1	2		2		2		1	1
高低位序				MSB	LSB	MSB	LSB	MSB	LSB		
数组号	0	1	2	3	4	5	6	7	8	9	10

1）数据帧含义

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（BCC 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；

2）数据帧解析

通过上文的内容我们了解了 ROS 控制时上位机下发的速度控制命令，该命令由 11 个字节组成，每个字节包含 8 位数据。接下来我们将对一帧数据进行解析。假设该数据帧内容为：【7B 00 00 00 64 00 00 00 00 1F 7D】

第 1 个字节和第 11 字节为固定的帧头帧尾，它们的作用是当串口接收到一个 16 进制数据 0X7B 后，如果 0X7B 其后的第 10 个字节数据为 0X7D，那么就认为 0X7B 及其后的 10 个字节组成了一个完整的速度控制命令。接着会对这 11 个字节数据进行校验，校验成功了才会认为这个命令是有效的并执行对应运动。

第 2、3 个字节：均为 0x00，预留位，此处数据无意义；

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(rad/s)；

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

大家可以使用 window10/11 自带的计算器验证一下，需要选择【HEX】即 16 进制，异或的运算符号为【XOR】，如图 4-2 所示。

图 5-2 异或校验方法

3）验证方法

我司提供的资料包中有 WHEELTEC—串口助手，我们可以在 Windows 系统下使用串口助手模拟上位机给底盘下发数据帧，进而验证底盘的 ROS 控制模式以及学习串口下行数据帧。

使用串口助手前我们需要先给电脑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在资料包【WHEELTEC ROS 机器人通用资料\9.软件与驱动】中为用户进行提供。用户可以参考资料包相关的教程进行安装以及验证，此处我们不进行赘述。

首先我们打开 WHEELTEC—串口助手，将波特率设置为 115200，停止位设置设置为 1，数据位设置为 8，校验位设置为无校验，并选择对应的端口号后，点击打开串口。

接着再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。

Tips：发送指令进行控制前，建议将底盘架起，使底盘的电机轮子悬空，防止指令下发后底盘乱跑发送碰撞造成不必要的损失。

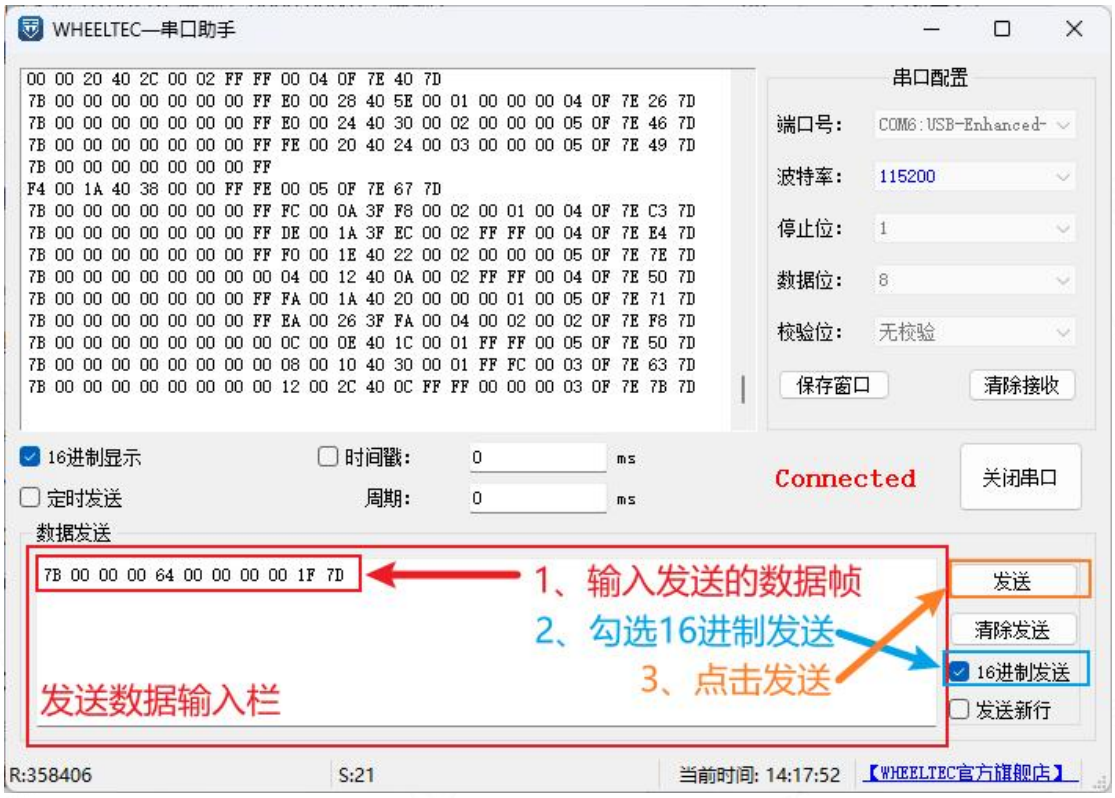


图 5-3 串口助手发送数据图

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，即使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，如图 5-4，为阿克曼底盘接收到数据帧后的 OLED 显示内容。

图 5-4 ROS 教育机器人阿克曼底盘图 OLED 显示

② USART 控制模式

USART 控制模式也是通过串口通信来实现的，USART 控制模式与 ROS 控制模式的差别在于使用的串口不同，串口接收使用的下行数据帧是完全一样的，对于数据帧的格式、含义和解析可参考上文中的 ROS 控制模式的内容。

USART 控制模式的验证方法也可以参考上文中 ROS 控制模式的内容，仅需

要把数据线接着串口 1 的位置即可，串口助手下发的数据帧以及使用方法是一样的。

底盘接收到完整的数据帧后进入 USART 控制模式，底盘会以 100mm/s 的速度向前移动，即使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，与 ROS 控制下的现象差别仅在 OLED 显示屏的左下角控制模式字样上，USART 控制下该控制模式字样为 USART。

③ **注意事项**

1）底盘解析收到的控制指令数据帧时，X、Y 和 Z 轴的目标速度方向均遵循章节 1.1 中的坐标系说明。

2）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。

3）虽然串口 1 和串口 3 的串口通信处理基本一样，但不建议使用大型 ROS 科研机器人的串口 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，重新上电后收到的上位机下发的数据帧 STM32 将其会当成烧录程序用的数据包。

4）对于串口控制的讲解我司有提供更加详细的资料以及教程，用户可通过查阅资料包中的【WHEELTEC ROS 机器人通用资料\2.运动底盘的控制使用\4.串口控制讲解】进行相关的学习。

5.2 CAN 控制

大型 ROS 科研机器人底盘支持 CAN 通信，CAN 通信的接线方式是 CANL 对 CANL,CANH 对 CANH，波特率是 1M。控制指令的帧 ID 为 0X181，底盘通过解析其中的数据内容获得下发的控制指令。

① **帧 ID：0X181**

数据帧说明如下：

帧 ID：0X181

帧类型：标准帧

帧格式：DATA

DLC：8 字节

帧 ID0X181 的下行数据帧内容如下表：

表 5-2 帧 ID 为 0X181 的下行数据帧

数据域	Rx[0]	Rx[1]	Rx[2]	Rx[3]	Rx[4]	Rx[5]	Rx[6]	Rx[7]
内容	X 轴目标速度		Y 轴目标速度		Z 轴目标速度		预留位	预留位
数据类型	Short		Short		Short			
高低位序	MSB	LSB	MSB	LSB	MSB	LSB		

1） 数据帧含义

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；

预留位：Rx[6]、Rx[7]均为预留位，各占一个字节，用户可根据需求自定义。

对于 CAN 控制的详细讲解，用户可通过查阅资料包中的【WHEELTEC ROS 机器人通用资料\2.运动底盘的控制使用\5.CAN 控制讲解】进行相关的学习，资料中包含对 CAN 控制的数据帧解析和验证方法，此处就不进行赘述。

2） 注意事项

CAN 通信接收到数据后，CAN 控制使能，之后控制器不再接收其他控制模式的指令。CAN 控制使能后可以看到 OLED 显示屏下角显示“CAN”,接着我们可以发送 CAN 指令对小车进行控制。

rx[6]、rx[7]是预留的数据位，可以供我们添加自己需要传输的数据，CAN 通信自带 BCC 校验所以这里不需要数据校验位。

差速底盘、阿克曼底盘不支持 Y 轴移动控制，因此 Y 轴方向的控制量默认是 0。

5.3 APP 遥控

机器人支持 APP 蓝牙控制以及在线调参。在 APP 模式中，直接使用摇杆控制机器人在空间中的运动。APP 控制机器人运动的速度单位是 mm/s(毫米/秒)，摇杆的斜上方的加速和减速按键，每按下一次机器人的速度增加/降低 100（mm/s）。

APP 在控制机器人的同时，机器人会通过蓝牙(APP 同时支持 WIFI 和蓝牙通

信)发送数据给手机，在 APP 的 Debug 栏里面可以看到显示发送的数据，具体的发送内容可以查看代码的 show.c 文件中的 APP_Show()函数。APP 控制模式其原理是通过蓝牙(或 WIFI)实现串口通信控制，该机器人的 APP 控制用的是串口 2，波特率设置为 9600，其控制指令在串口 2 接收中断服务函数中。

1） 在线调参

将最新版的 WHEELTEC APP 安装到安卓手机上，然后根据相应的视频教程操作即可遥控机器人或者进行参数在线调节等操作。在“参数”界面中每个通道的名称可以点击“参数 x”可以自定义修改名称，具体效果见图 5-8 和图 5-9。另外，在调节 PID 参数之前，我们需要点击[获取设备参数](右上角菜单按钮调出)，把机器人的 PID 参数更新到 APP 上面，然后拖动滑块，当我们松手的时候，APP 即可把参数发送到机器人上面。

参数 0 为小车的速度参数，调整参数 0 可以调节小车的速度，单位为 mm/s。

参数 1、2 是小车运动控制速度环的 PI 参数。

需要注意的是，假设当前机器人控制速度是 40，当我们想把速度参数调到 150（超过范围）时，需要在第一次调到 80 后，再点击一次[获取设备参数]，这时调整范围就改变成 0-160 了。机器人设定了速度范围是 20—200。

图 5-8 默认调参界面

图 5-9 获取设备参数后的界面

（需要自行修改参数名）

2） APP 控制机器人

图 5-10 APP 控制界面

APP 操作界面对应的每一个操作是实际上向机器人发送不同的指令信息（切换控制方式及界面也同样发送指令），机器人接收到指令信息后作出响应，详细的 APP 操作界面的每个操作对应发送的信息见表 5-3：

表 5-3 APP 界面操作指令说明

APP 摇杆	↑	↗	→	↘	↓	↙	←	↖
机器人接收到的数据	0x41	0x42	0x43	0x44	0x45	0x46	0x47	0x48
机器人实现效果	前进运动	右前运动	原地右转	右后运动	后退运动	左后运动	原地左转	左前运动
按键	重力		摇杆		按键		减速	加速
机器人接收到的数据	0x49		0X4a		0x4b		0x59	0x58

注意：手机成功连接上小车蓝牙后，需要往前推摇杆 0.5 秒后才能正式控制小车。

5.4 PS2 手柄遥控

PS2 模式使用的是 PS2 无线手柄，PS2 模式控制的顺序是：上电前先插好手柄，然后接通电源，此时手柄上的指示灯红灯常亮表明正常工作，若指示灯不亮则按一下指示灯上方的按键，按下 START 按键后，进入手柄模式，此时可以看到显示屏的左下角显示“PS2”。

在 PS2 模式中，使用左边遥控控制机器人前后移动，右边的摇杆控制机器人的左右转向(全向运动小车则为左边遥控控制小车在空间中 8 个方位的运动，右边的摇杆控制小车原地自旋转运动)。这样设计的原因是如果前后和左后的控制都使用同一个摇杆容易出现误触转向的情况。左上角两个按键为加速、减速按键。

注意：在 PS2 模式下，要插好 PS2 手柄后再上电，否者容易烧坏手柄以及出现电机乱转的现象。PS2 手柄模式在开机后,需要按下 PS2 手柄上面的 START 按键后才进入 PS2 手柄模式。

图 5-11 PS2 手柄实物图

5.5 航模手柄遥控

航模遥控的正确使用步骤是：小车先接上航模遥控接收器，小车上电，航模遥控右摇杆拉至最低后打开航模遥控的电源，航模遥控接收器指示灯绿色常亮说明已经连接上，接着向前推动左摇杆可以看到显示屏左下角显示“R-C”。航模遥控的控制方式是：航模遥控的左边摇杆控制小车前进后退，右边摇杆控制小车左右转向(全向运动小车则为左边遥控控制小车在空间中 8 个方位的运动，右边

的摇杆控制小车原地自旋转运动)，同时右边摇杆前后拨动可以控制小车车速即相当于油门。显示屏面板的右上角的大小舵量按键控制小车选择正常模式和低速模式。

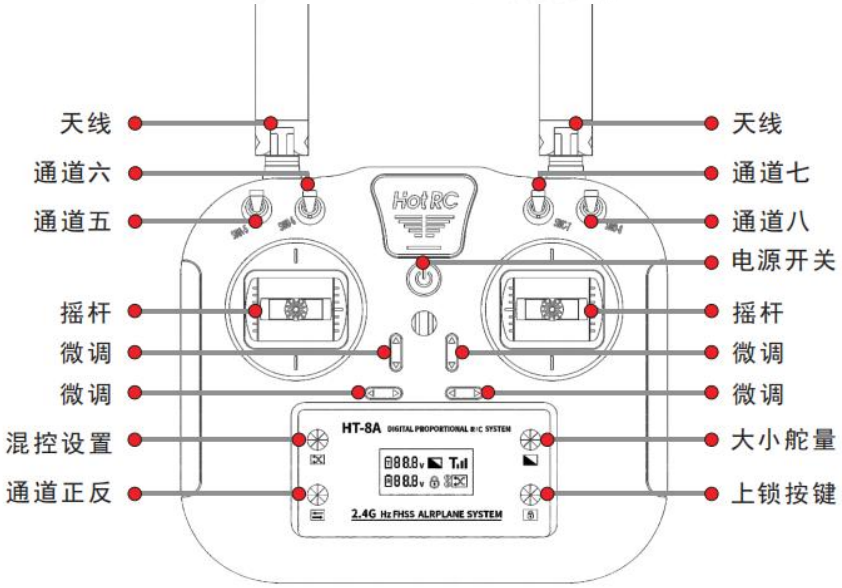


图 5-12 航模遥控实物图

需要注意的是航模遥控的通道需要正确配置的，如果遥控正常连接后无法使用可以查阅我司提供的资料进行排查纠正，视频教程在【ROS 机器人通用资料】中。

下面讲解一下航模遥控接收器该如何连接转接板。

图 5-13 航模接收器实物图

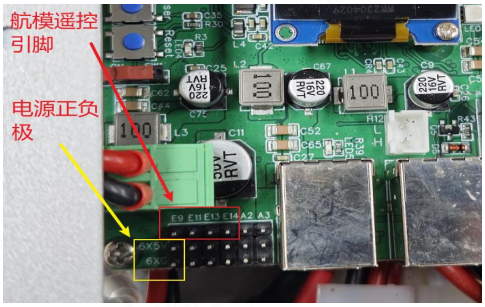


图 5-14 转接板上的航模接口

表 5-4 航模遥控引脚对应通道

航模接收器	GND	5V	CH1	CH2	CH3	CH4
转接板	-	+	E9	E11	E13	E14

如图 5-13、图 5-14、表 5-4 所示，航模接收器有三列，分别是 GND、5V 和信号线，我们使用的时候 GND 和 5V 接一对即可，然后信号线的 CH1、CH2、

CH3、CH4 分别接转接板上的排针 E9、E11、E13、E14。E14 引脚只有全向移动小车才需要用到，用于左右横移。

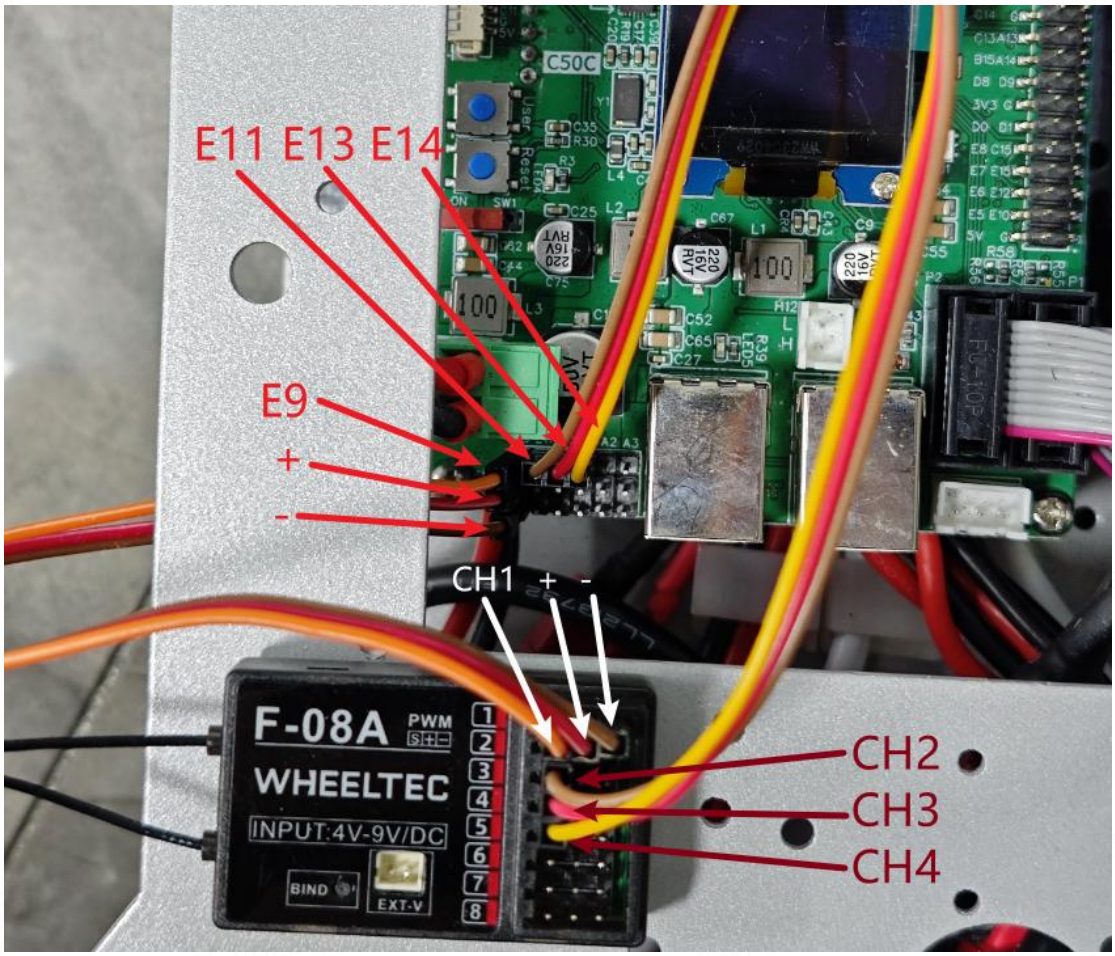


图 5-15 航模接线示例

6. 运动底盘状态数据反馈

通过第五章的学习，我们了解到在各种模式是如何控制底盘的，在机器人中不仅要实现对底盘的控制，同时对于底盘的状态数据获取也至关重要，所以在本章中主要学习在 CAN 控制和串口控制时如何获取底盘发出来的状态数据。

大型 ROS 科研机器人底盘状态数据可以通过 OLED 显示屏进行实时观测，也可以通过上位机（PC 或 ROS）连接 CAN 接口和串口接口接收底盘上发的状态数据进行监测。前者仅能观测到部分状态数据，具体内容参考章节 2.4，后者可以接收到底盘当前的所以状态数据。需要注意的是底盘上发的状态数据无论是通过那种接口和协议上发的，数据帧内容都是一样的。

6.1 串口控制的数据反馈

对于底盘完整的状态数据，需要通过串口接口和 CAN 接口获取，底盘通过串口（ROS 控制模式和 USART 控制模式）上发的状态数据的数据帧如下表 6-1 所示。用户可以通过我司提供的【WHEELTEC ROS 机器人通用资料\2.运动底盘的控制使用】的文档和视频教程进行更深入的学习。

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	
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	
数组号	12	13	14	15	16	17	18	19	20	21	22		23	

表 6-1 串口上行状态数据表

① **数据帧含义**

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（BCC 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；

② **数据帧接收验证**

我们可以在 Windows 系统下用 type-c 数据线连接底盘的串口 1 或串口 3，然后打开 WHEELTEC—串口助手软件，按照下图 6-1 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 6-1　串口助手接收数据图

通过上文的内容我们了解并接收到了底盘上发的状态数据帧，该数据帧由 24 个字节组成，每个字节包含 8 位数据。接下来我们将对底盘上发的一帧数据进行解析。该数据帧内容为:【7B 00 00 9B 00 00 FF DF 00 60 00 0C 40 A8 FF FD 00 06 00 1E 5B 87 82 7D】

第 1 个字节：0x7B，串口通信时为帧头；

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：BCC 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；大家可以使用 window10/11 自带的计算器验证一下，需要选择【HEX】即 16 进制，异或的运算符号为【XOR】，如图 6-2 所示。

图 6-2　异或校验方法

第 24 个字节：0x7D，串口通信时为帧尾；

③ **注意事项**

对于上文中的三轴加速度和角速度的数据解析涉及了 IMU 数据的量程转化，具体的量程转化方法可以参考章节 6.6 IMU 的作用与数据说明进行解析。

6.2 CAN 控制的数据反馈

机器人底盘 CAN 上行的数据跟串口上行的数据内容是相同的，机器人底盘的上行数据较多共有 24 字节，发送数据时是一次发送 8 个字节，分 3 次发送，即机器人底盘的上行数据帧共有 3 帧，每一帧都有机器人底盘上不同的信息。接收端可以通过标识符 ID 确认当前接收的是第几组数据，发送第一组数据的标识符是 0X101，发送第二组数据的标识符是 0X102，发送第三组数据的标识符是 0X103。

① **帧 ID：0X101**

数据帧说明如下：

帧 ID：0X101

帧类型：标准帧

帧格式：DATA

DLC：8 字节

帧 ID0X101 的上行数据帧内容如下表：

表 6-2 帧 ID 为 0X101 的上行数据帧

数据域	Rx[0]	Rx[1]	Rx[2]	Rx[3]	Rx[4]	Rx[5]	Rx[6]	Rx[7]
内容	0X7B	flag_stop	X 轴速度		Y 轴速度		Z 轴速度	
数据类型	uint8	uint8	short		short		short	
高低位序			MSB	LSB	MSB	LSB	MSB	LSB

以下是数据帧内容的含义:

0X7B：固定内容，0X7B，CAN 通信时无意义，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；

② **帧 ID：0X102**

数据帧说明如下：

帧 ID：0X102

帧类型：标准帧

帧格式：DATA

DLC：8 字节

帧 ID0X102 的上行数据帧内容如下表：

表 6-3 帧 ID 为 0X102 的上行数据帧

数据域	Rx[0]	Rx[1]	Rx[2]	Rx[3]	Rx[4]	Rx[5]	Rx[6]	Rx[7]
内容	X 轴加速度		Y 轴加速度		Z 轴加速度		X 轴角速度	
数据类型	Short		Short		Short		Short	
高低位序	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB

以下是数据帧内容的含义:

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；

③ **帧 ID：0X103**

数据帧说明如下：

帧 ID：0X103

帧类型：标准帧

帧格式：DATA

DLC：8 字节

帧 ID0X103 的上行数据帧内容如下表：

表 6-4 帧 ID 为 0X103 的上行数据帧

数据域	Rx[0]	Rx[1]	Rx[2]	Rx[3]	Rx[4]	Rx[5]	Rx[6]	Rx[7]
内容	Y 轴角速度		Z 轴角速度		电池电压		校验位	0X7D
数据类型	Short		Short		Short		uint8	uint8
高低位序	MSB	LSB	MSB	LSB	MSB	LSB		

以下是数据帧内容的含义:

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（BCC 校验）的结果，CAN 通信自带校验，可以不再进行异或位校验，占一个字节；

0X7D：固定内容，0X7D，CAN 通信时无意义，占一个字节；

④ **注意事项**

由于 CAN 通信需要设置帧 ID 以及 CAN 通讯是自带 CRC 校验的，因此在 CAN 通讯时数据帧中的帧头、帧尾以及校验位都是没有任何意义的，这几位数据仅在串口通讯时起作用。

对于数据帧的解析可以参考串口控制时的数据解析方法或者通过我司提供的【WHEELTEC ROS 机器人通用资料\2.运动底盘的控制使用\5.CAN 控制讲解】的文档和视频教程进行更深入的学习，此处不再进行赘述。

6.3 里程计的作用与数据说明

里程计是一种测量和记录车辆或机器人移动距离和方向的仪器。在机器人技术中，里程计通常用于确定机器人相对于起始点的位置和朝向，是实现机器人自主导航和定位的关键组件之一。它通过测量机器人轮子的转动来估算机器人的移动距离和方向变化。里程计数据通常包括机器人的线性速度、角速度等。

对于我司底盘状态数据中的三轴速度由 STM32 控制器程序上实现的里程计提供，且底盘的三轴速度方向遵循 图 1-1 坐标系示意图的方向。

底盘通过测量每个轮子的速度，然后通过运动学正解算法，结合底盘的几何参数，计算出底盘在三个方向上的速度。而轮子的速度是由电机的编码器数据计算得到的，具体的解算过程可以根据车型通过章节 3.不同类型底盘和运动学分析进行了解，此处只举例麦轮小车进行简单介绍。

以下代码段为麦轮小车的三轴速度解算：

```
    X_speed=((MOTOR_A.Encoder+MOTOR_B.Encoder+MOTOR_C.Encoder+MOTOR_D.Encoder)/4)*1000;
    Y_speed=((MOTOR_A.Encoder-MOTOR_B.Encoder+MOTOR_C.Encoder-MOTOR_D.Encoder)/4)*1000;
    Z_speed=((-MOTOR_A.Encoder-MOTOR_B.Encoder+MOTOR_C.Encoder+MOTOR_D.Encoder)/4/(Axle_spacing+Wheel_spacing))*1000;
```

代码中 MOTOR_A.Encoder 为电机 A 的速度（单位：m/s），该变量具体计算方法在章节 4.1 轮子线速度计算有进行讲解，Axle_spacing 为小车的前后轮轴

距（单位：m），Wheel_spacing 为底盘的主动轮轮距（单位：m），通过对底盘各轮子的转速、轮距和轴距进行运动学正解获得 X 和 Y 的速度并将单位转化为 mm/s,Z 轴的解算结果单位是 1000*rad/s。

6.4 IMU 的作用与数据说明

IMU，即惯性测量单元，是一种用于测量和报告一个物体的特定力、角速度以及在某些情况下，磁场周围物体的方向的设备。

底盘的 STM32 控制器上集成有 IMU，而 IMU 使用的是 MPU6050 集成有加速度计以及角速度计，由底盘上发的状态数据中的三轴加速度和角速度均来自底盘控制板上 IMU 的原始数据。板子上的丝印代表的是 IMU 的原始方向，并非状态数据中的数据方向，状态数据中的加速度以及角速度经过方向重定义处理后再进行上发，而重定义后的三轴加速度和角速度的方向遵循图 6-3 ROS 机器人姿态数据正方向。这里的 X、Y、Z 与 ROS 机器人运动正方向的设置是一致的，X 轴前进为正，Y 轴向左为正，Z 轴逆时针为正。

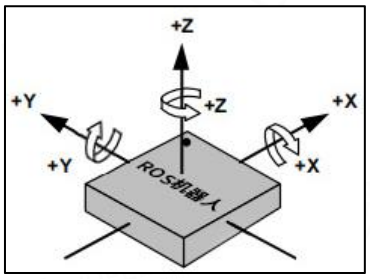


图 6-3 ROS 机器人姿态数据正方向

需要注意的是，机器人正常放置在水平地面时，Z 轴的加速度计数据即为重力加速度 G，这是由 IMU 的加速度计算原理决定的，相当于此时 IMU 认为机器人在无重力环境下以重力加速度的大小向上(Z 轴正方向)运动。如果把机器人绕 Y 轴旋转 180°(旋转方向为正方向)，此时 X 轴的加速度计数据即为重力加速度 G，Z 轴加速度计数据则在 0 附近波动。根据这一特性，把加速度计数据传给 ROS 主控后，ROS 主控是可以计算出机器人此时的姿态的，通过角速度积分也可以计算出机器人的实时姿态，处理加速度和角速度时会有一定的权重设置。

通过第五章的学习，我们知道 6 个姿态数据（三轴加速度和角速度）都是由两个字节组成的 16 位数据，由于是带符号的数据，那么可以知道这 6 个数据的大小范围为±2^15=±32768。此时引出 IMU 的量程设置，在 STM32 程序里面设

置了 IMU 的加速度计量程为±2G=±2*9.8m/s^2=±19.6m/s^2，角速度计的量程为±500°/s=±8.72665rad/s。即例如读取计算得到 X 轴的加速度计数据为+32768，则代表此时机器人的前进加速度为+2G=+19.6m/s^2。量程转化为国际主单位的方法是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）。

7. 运动底盘相关的流程图

通过前文对底盘的构成以及控制功能实现的了解，我们可以将在底盘的硬件组成、软件实现等方面归纳成对应的流程图，以此来方便我们理解底盘的工作原理。

7.1 机器人硬件框图

在机器人中用到很多控制器和外设，包括：树莓派（Orin Nano）、激光雷达，STM32 控制器，电机、编码器、双路驱动、蓝牙、PS2 手柄、航模遥控、陀螺仪等。而在底盘的 STM32 控制器上我们还预留了串口 1 和 CAN 接口方便用户扩展控制，这些控制器与外设，控制器与控制器之间的连接，如图 7-1 机器人硬件框图所示，我们可以了解到运动底盘的各个外设、控制器间的连接方式以及通讯协议。

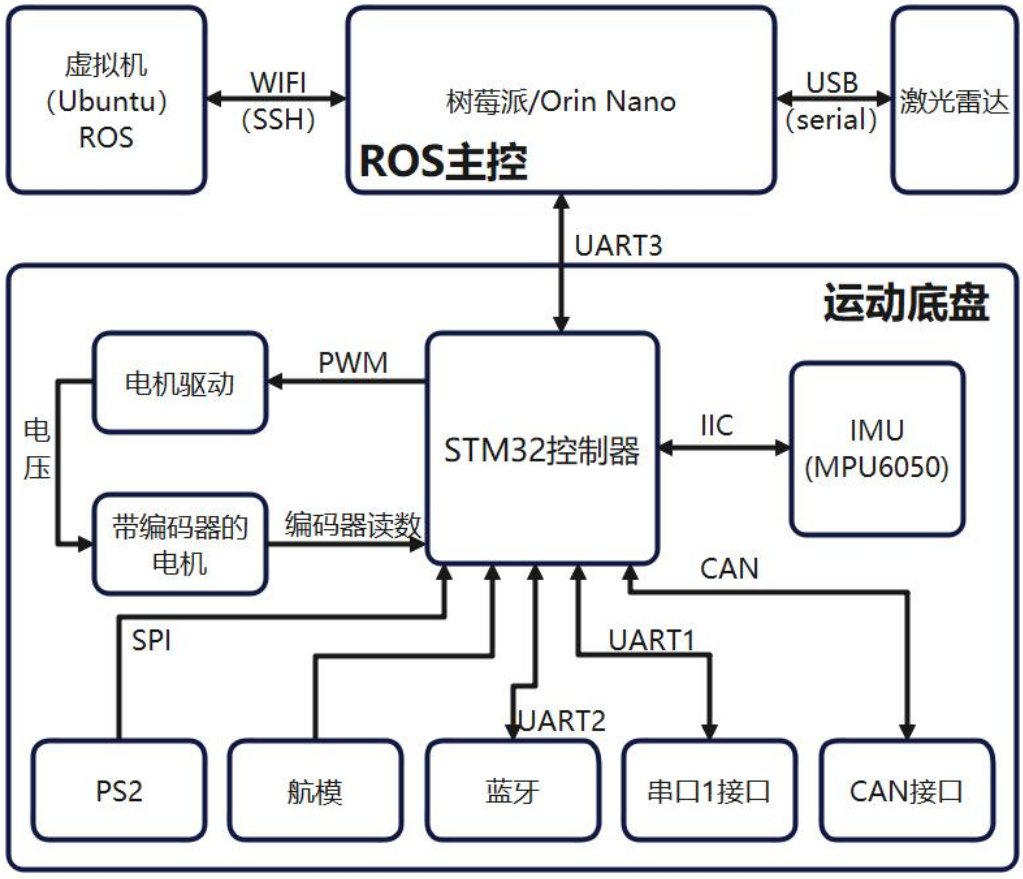


图 7-1 机器人硬件框图

在图 7-1 中提到的与 ROS 主控连接的激光雷达是一种关键的传感器设备，

它在机器人的导航和环境感知中扮演着重要角色。激光雷达（LIDAR）通过发射激光束并测量反射回来的光线来探测周围物体的距离和形状。这种设备能够为机器人提供精确的环境地图，帮助机器人进行路径规划和避障。在机器人的身上，激光雷达相当与人身上的眼睛，它对于机器人的设计与实现及其重要。

7.2 STM32 程序流程图

接下来，我们将深入了解大型 ROS 科研机器人底盘的源码架构。我们的底盘电控板搭载的是 STM32F407VET6 主控芯片，运行的源码程序是在 Keil MDK-ARM（Keil5）开发环境中，采用 C 语言编写而成。为了增强程序的实时性和响应速度，我们引入了 FreeRTOS 这一免费的实时操作系统。

FreeRTOS 的使用允许系统通过多任务处理来并行执行多个操作，这对于需要同时处理传感器数据收集、运动控制和通信的任务至关重要。因此我们的大型 ROS 科研机器人底盘不仅能够实现高效的资源管理和任务调度，还能保证在多变的工作环境和需求下，保持高度的稳定性和可靠性。

如图 7-2 所示，为 STM32 程序流程图。

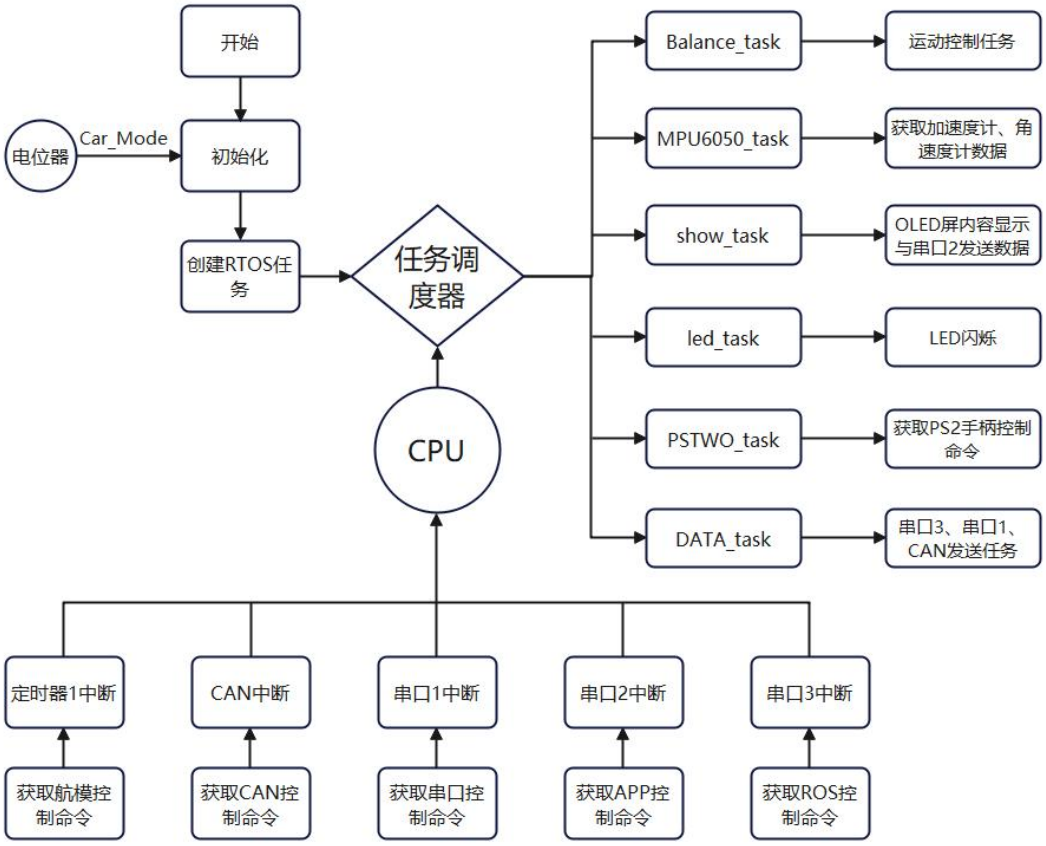


图 7-2 STM32 程序流程图

FreeRTOS 的任务调度器通过任务的优先级来动态安排任务执行顺序。请注意，图 7-2 中展示的任务顺序并不代表实际的优先级顺序；具体的优先级设置需参考程序代码中的配置。由于每个任务的执行时间设计得非常短暂，这使得从宏观上看，所有任务似乎在同时进行。此外，系统设计允许在任何时间点，一旦检测到中断信号，如串口 2 的 APP 蓝牙控制中断或串口 3 接收到来自 ROS 上位机的信息中断，调度器将立即响应并处理这些中断，确保控制系统的实时性。

程序在上电后会进行初始化，而初始化的时候会根据采样电位器数值后计算得到的 Car_Mode 来选择不同的底盘的参数进行程序的初始化。如图 7-3 所示，为 Car_Mode 影响到的程序部分。

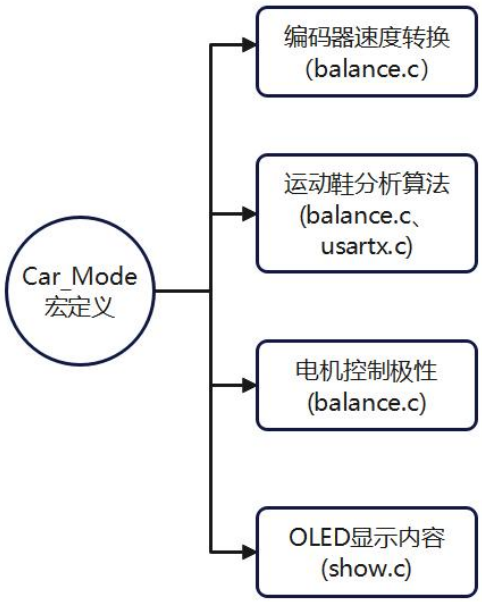


图 7-3 Car_Mode 影响到的程序部分

用户可以通过我司提供的资料包【WHEELTEC ROS 机器人通用资料\3.STM32 底层源码讲解教程】中的相关资料底盘源码进一步深入学习。

7.3 电机控制流程图

机器人支持多种控制模式，这几种控制模式实现控制的原理都是通过改变机器人的目标速度来实现控制的。目标速度经过运动学分析函数得出每个电机的实际输出，最后通过 PID 控制器（PID 速度控制函数）来实现电机的速度控制。

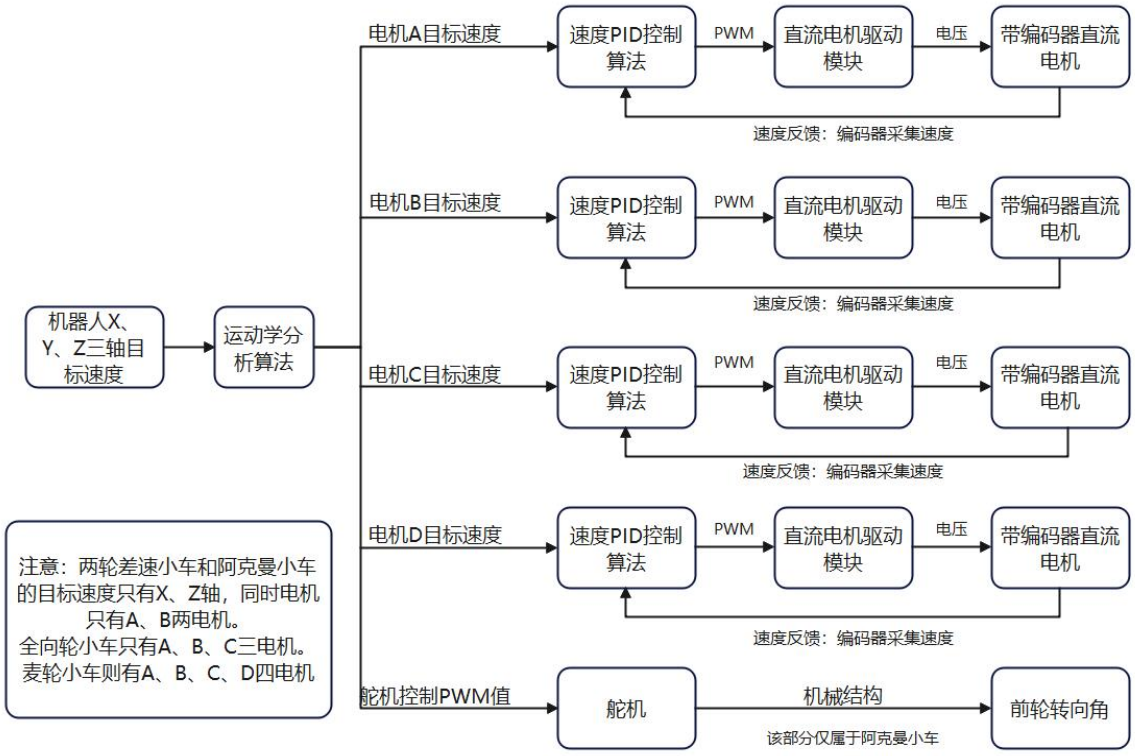


图 7-4 电机控制流程图

如图 7-4 所示，为电机控制流程图，由于大型 ROS 科研机器人电控板采用的是驱控一体的设计所以直流电机驱动模块已经集成在电控板上了，仅需将电机接到电控板上对应的电机接口即可使用。

8. 程序烧录下载

STM32 控制器可以通过串口或者 SWD 接口下载程序。串口是通过 USB 数据线下载，默认有赠送，SWD 接口建议使用金属外壳的 STlink 下载，需要自备。

对于底盘固件的烧录我司有提供相应的资料包以及教程，用户可通过【WHEELTEC ROS 机器人通用资料\3.STM32 底层源码讲解教程】和【WHEELTEC ROS 机器人通用资料\9.软件与驱动】获取相关资料

8.1 串口下载

STM32 控制器上设计有一键下载电路，下载程序非常方便。只需一根 MicroUSB 手机数据线就行了。

① **硬件准备与接线**

硬件：STM32 控制器、TypeC 数据线

接线如下图所示，注意 TypeC 接口不能更换。

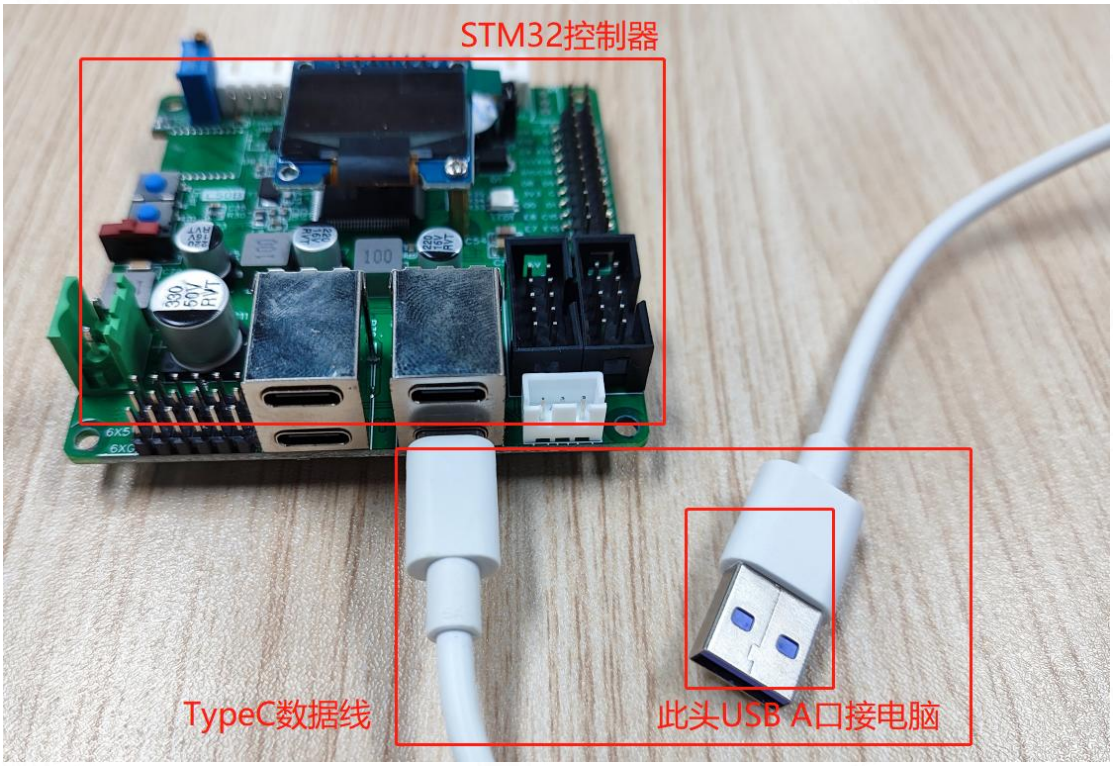


图 8-1 串口下载的接线

② 软件准备

软件：FlyMCU 烧录软件（附送的资料中有），相应的 USB 转 TTL 模块 CH9102 的驱动，附送的资料里面有对应驱动安装文件【 ROS 机器人通用资料/9.软件与驱动】。

安装成功后可以打开设备管理器看看，可以看到驱动已经安装成功，否则会有红色的感叹号。

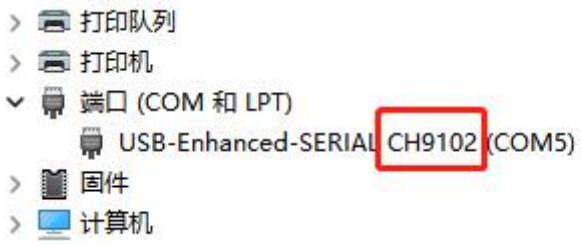


图 8-2 设备管理器查看 CP210x 驱动

③ 下载（烧录）程序

打开资料包中【 ROS 机器人通用资料/9.软件与驱动】文件夹里面的 FlyMcu 软件，根据图 8-3 中的操作顺序设置即可。

图 8-3 FlyMcu 下载程序配置说明

OK,一切准备就绪，然后点击开始编程，程序就可以下载了。因为勾选了编程后执行，所以程序下载完后，会自动运行。

8.2 SWD 下载

STM32 控制器可以通过 SWD 接口下载程序，在电板上面有标识，分别是 PA13 和 PA14，具体位置可以参考图 2-1，相关的资料跟教程可通过【WHEELTEC ROS 机器人通用资料/3.STM32 底层源码讲解教程】进行了解。

① **硬件准备**

1.STM32 控制器

2.STlink

② **软件准备**

对应的 STlink 或者 Jlink 驱动的安装。

安装成功后可以打开设备管理器查看是否又显示 STLink 设备。

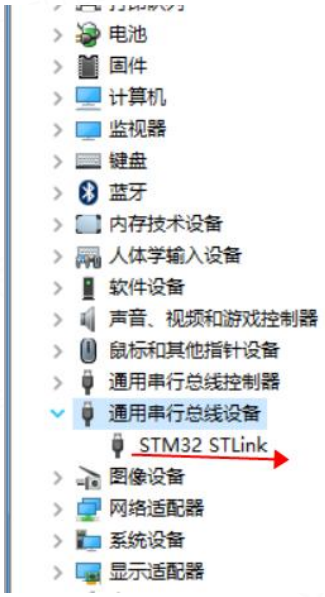


图 8-4 设备管理器查看 STlink 驱动

可以看到驱动已经安装成功！

③ **接线**

STlink ---------- STM32 控制器

SWDIO----------PA13

SWCLK----------PA14

GND----------GND

OK，一切准备就绪。

④ 下载程序

点击图 8-5 中的箭头所指的按钮，程序就可以下载了！因为勾选了编程后执行，所以程序下载完后，会自动运行。默认程序的配置是针对 STlink 的，如需配置 Jlink 下载，需要修改 MDK 设置。

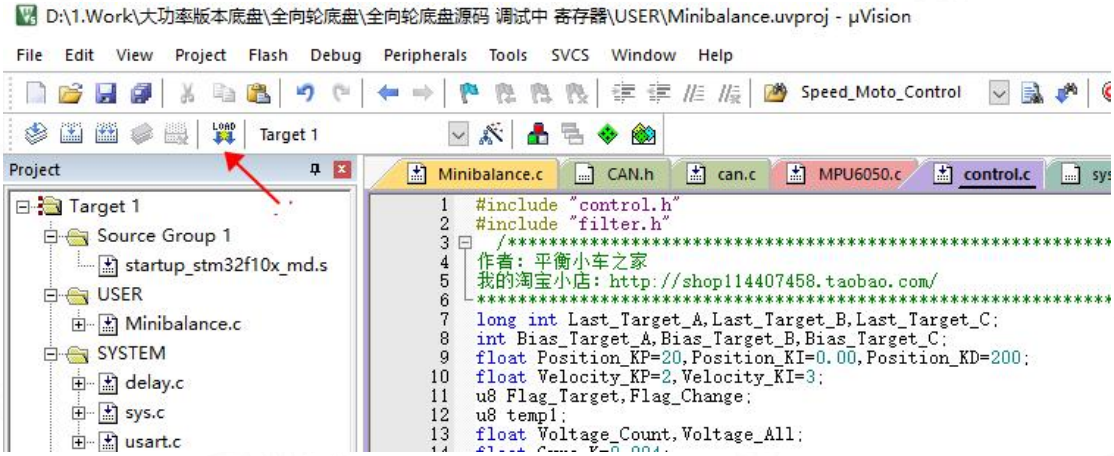


图 8-5 STlink 下载程序界面

9. 注意事项

9.1 关于代码

机器人 STM32 控制器上的程序写法是基于 RTOS 系统，与中断控制的方式不同，RTOS 是任务形式轮流执行，优先级高的任务执行优先级更高（中断优先级高于任务优先级）。需要说明的是，如果某个任务有执行逻辑的错误的话，程序会卡在该错误处无法继续执行下去。例如：假如程序中有一个串口 3 的发送任务，但是串口 3 没有初始化或者是初始化的代码出错，那么在执行串口 3 发送任务时，程序就会卡住。因此如果遇到调试过程中程序卡住无法正常执行，需要逐个任务排查是否有 bug 卡住了。

9.2 关于电控板上的电源接口

STM32 控制器上面的 5V 和 3.3V 引脚可以对外供电，但是不可带太大电流的负载，其中 5V 输出不建议带超过 1.5A 的负载，3.3V 输出不建议带超过 200mA 的负载。在我们提供的原理图上可以看到，转接板上配备了两路 5V 电源电路，其中一路给基础外设供电，另一路单独给树莓派（Orin Nano）供电（USB 母座）。接口位置可以参考图 2-1 大型 ROS 科研机器人控制板左上角的 5V 供电接口。

9.3 电机使用和保护

使用的过程中要避免电机堵转，否则很容易烧毁主板。在机器人没有完全测试通过之前，请把机器人架起来，让电机悬空，避免误操作导致电机乱转引起不必要的损坏。

9.4 电池使用和维护

显示屏上显示了电池的电压，电量低时（低于 20V）请及时充电。电池自带过放和过充保护。电池充电的时候请不要使用电池，电池充电接线如图 9-1 所示。

图 9-1　机器人电池与充电器接线图

电池充电结束后，应该将充电口的盖子合上，避免误触导致电池断路，具体如图 9-2 所示。

图 8-2　机器人电池充电线接口