重慶機電職業技術大學
Chongqing Vocational and Technical University of Mechatronics

毕业设计

基于 ROS 的智能网联小车

设计与实现

学　　院：车辆与交通学院

专业班级：汽车服务工程技术 3 班

层　　次：☑本科　□ 专科

学生姓名：谢正瑶

学生学号：12607120251835

指导教师：彭光雷

企业导师：陈政华

完成时间：2025 年 3 月 31 日

目　　录

摘　要

本设计以智能网联汽车技术验证平台构建为研究目标，针对全尺寸车辆研发成本高、测试风险大的技术痛点，提出基于机器人操作系统（ROS）的智能网联小车设计方案。研究通过构建"STM32 微控制器+Jetson Orin Nano 计算单元"的分层硬件架构，集成多模态传感器阵列（激光雷达/深度相机/鱼眼镜头），实现环境感知数据的多源融合。基于 ROS 框架的模块化软件系统设计，采用分布式通信机制实现硬件抽象与数据标准化，开发了包含环境感知、决策规划和控制执行的三层算法架构。针对车道保持系统（LKS）的核心功能，提出基于视觉感知与控制策略相结合的解决方案。实验验证表明，小车在直线和弯道工况下均能实现稳定车道保持证明了本设计的稳定性和适应性。研究成果为智能网联汽车技术与原型开发提供了可复用的技术框架。

关键字：智能网联小车 ROS 车道保持 硬件设计 软件开发

Abstract

This design takes the construction of intelligent connected vehicle technology v erification platform as the research goal, and puts forward the design scheme o f intelligent connected vehicle based on robot operating system (ROS) for the t echnical pain points of high research and development cost and high test risk of full-scale vehicles. By constructing a layered hardware architecture of"STM3 2 microcontroller+Jetson Orin nano computing unit", integrating multimodal sen sor array (LIDAR/depth camera/fisheye lens), the multi-source fusion of environ mental perception data is realized. The modular software system design based on ROS framework uses distributed communication mechanism to realize hardw are abstraction and data standardization,and develops a three-tier algorithm archi tecture including environmental perception,decision planning and control executi on. Aiming at the core functions of lane keeping system (LKS), a solution bas ed on the combination of visual perception and control strategy is proposed. T he experimental verification shows that the car can achieve stable lane keeping under straight-line and curve conditions, which proves the stability and adapta bility of the design.The research results provide a reusable technical framework for the technology and prototype development of intelligent connectedvehicles.

Keywords: Intelligent connected vehicle ROS Lane Keeping System (LKS) h ardware design; software development.

1 绪论

1.1 研究背景及意义

近年来随着汽车行业的快速发展以及科学技术的进步，智能网联汽车技术已经汽车领域发展的重要方向。智能网联汽车相关技术为提高交通效率，减少交通事故，减少能源消耗和提高旅行体验带来了新方向[1]，同时在全球范围带来了巨大的市场。根据中研普华产业研究院发布的《2025-2030 年中国智能汽车(智能网联汽车)行业深度调研及投资前景预测报告》，2023 年智能网联汽车行业的市场规模达到了 8276.5 亿元人民币，预计到 2025 年将突破 1.1 万亿元人民币，平均每年的增长率超过 28%[2]。

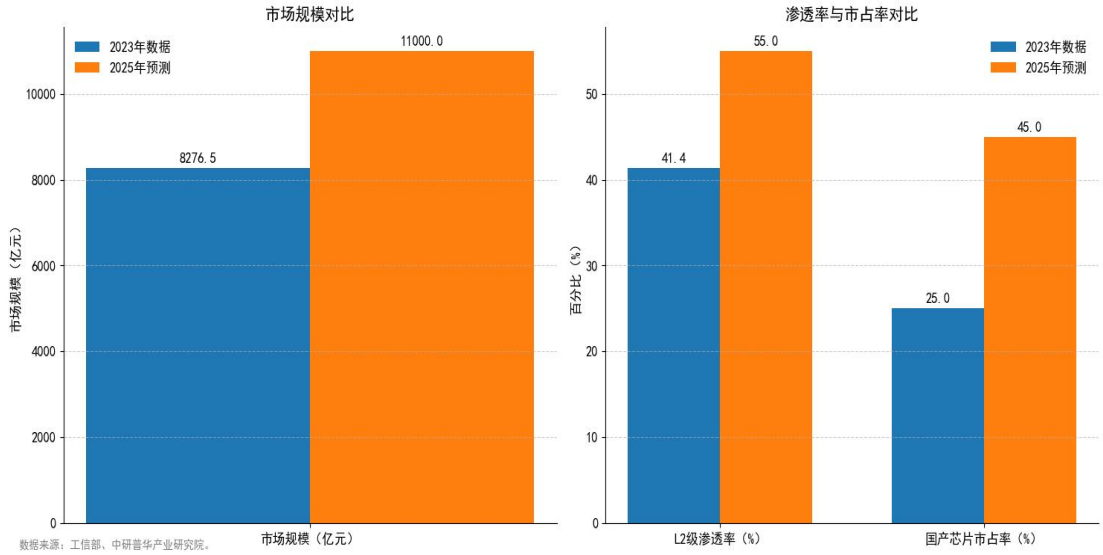

图 1-1 2023 年与 2025 年智能网联汽车市场数据对比

智能网联汽车通过多传感器设备，实现对四周环境的感知和基础设施之间的通信，从而提高驾驶安全和交通效率[3]。但是，在智能网联汽车的研发过程中，存在一些问题。比如全尺寸智能网联汽车的研发成本高，不仅需要大量人力物力进行车辆本身的设计和开发，还需要建设专门的测试场地，这些都极大影响了技术创新的速度和范围。智能网联汽车技术涉及多学科交叉，需要综合运用机械工程、电子信息工程、计算机科学、通信技术等[4]多方面的知识，而目前相关专业的学生在学习过程中往往缺乏一个直观、可操作的平台来深入理解和实践这些技

术，影响了人才的培养效果。

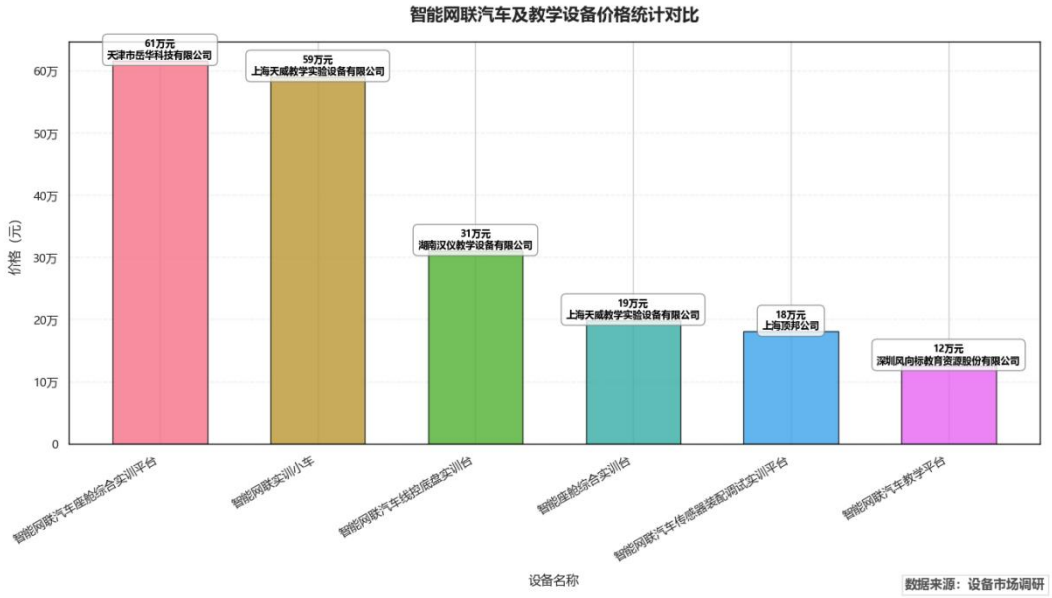

图 1-2 智能网联汽车及教学设备价格统计对比

ROS（机器人操作系统）作为一种灵活的框架，为机器人软件开发提供了许多便利。它的分布式架构可以将节点分布在多台计算机上，这方便了模块化开发和系统扩展，提高了系统的可维护性和可扩展性[5]。同时 ROS 拥有丰富的库和工具，包含了机器人感知、导航、控制等，大大减少了开发时间和工作量。ROS 还拥有活跃的开源社区，其有丰富的教程、文档和开源项目，方便开发人员获得支持和帮助。另外 ROS 的系统架构比较简单，容易学习和开发，适合多学科跨学科的教学和研究。已被广泛应用于各种机器人系统中。同时像 ROS 2 Prius、Autoware、Apollo 等[6]自动驾驶框架都是由 ROS 发展或借鉴参考 ROS 而来。

在这样的大背景下，本设计作为智能网联汽车的一种微缩模型，其在研究和应用中都具有重要价值。本设计成本相对较低，总体成本远远低于全尺寸智能网联汽车。这种成本效益可以让其成为在教育领域实施实践教学、在研究机构进行初步验证性实验的首选方案。降低了参与智能网联汽车的门槛，可以吸引和培养更多的人才，加快技术创新的步伐和知识的广泛传播。

本设计由于其体积较小，特别适合在室内环境中进行测试和调试。可以在有限的空间内移动，模拟各种交通场景，如避障、路径规划和多车辆协作，为验证相关自驾算法时提供高效安全的实验环境。同时本设计具有相对简单的系统架

构，简单易学，非常适合作为跨学科教育工具，帮助学生更直观地掌握智能网联汽车技术的基本原理和相关算法，提高学生的实践技能和创新能力，为行业培养更多的专业人才。

1.2 国内外研究现状

近年来，国内外学术界对智能网联汽车及其微缩模型——智能网联小车的研究热度持续攀升，相关技术已逐步从理论探索走向实际应用。

1.2.1 国内研究现状

在国内有许多高校和研究机构都在使用 ROS 框架进行智能小车开发。比如，清华大学的智能技术与系统国家重点实验室就通过融合多传感器数据，实现了高精度的环境感知和自主导航功能的小车，其成果被应用到了室内物流机器人中。哈尔滨工业大学则关注多车协同控制算法，利用 ROS 分布式架构搭，成功验证了车队编队行驶和动态避障。还有，北京理工大学的团队通过开源硬件和低成本传感器，开发了面向教学使用的低成本小车平台，降低了实验门槛。

在企业，如百度 Apollo 自动驾驶框架通过 ROS2 开发优化了系统的实时性和可靠性。还有小鹏汽车通过 ROS 生态系统开发了仿真测试工具链，可以在虚拟环境中进行大规模算法验证，为路测提供了基础。

1.2.2 国外研究现状

在国外相关研究更加关注前沿技术。比如，麻省理工学院的 CSAIL 实验室开发了一个基于 ROS 的“Duckietown”项目，这个项目通过微型的智能小车在模拟的城市环境中，完成了自动驾驶算法的复现和迭代。美国的斯坦福大学则在 ROS 下使用深度强化学习算法，使智能小车能够像全尺寸智能网联汽车一样通过端到端训练自主适应复杂动态环境的能力。

在工业界中，Waymo 通过 ROS 衍生的定制化系统，完成了多传感器融合与高精度定位技术的突破，部分技术方案也被应用于智能小车的多目标跟踪研究中。另外特斯拉的 Autopilot 系统虽没有直接采用 ROS 系统，可是其部分开源数据集都是由 ROS 开发者提供的训练样本。

1.2.3 对比与趋势分析

国内外的研究在核心目标上大致是一致的，都是通过低成本平台来加速算法

的验证和人才培养。可是，在具体实施策略上存在一些不同：国外研究更注重构建开源生态，并推动跨学科领域的整合，而国内则更加关注技术的实际应用和产学研一体化的发展模式。在当前阶段，本领域面临的主要挑战包括硬件性能程度不足和算法通用性受限等问题。在未来，ROS2 的广泛使用和边缘计算算力的提升，微缩智能联网车小车的实时响应能力和场景的适应能力都将得到提升；这让其在教育行业中的普及和应用也有望进一步增加。

1.3 本文主要研究内容与结构

1.3.1 主要研究内容

本设计以“基于 ROS 的智能网联小车设计与实现”为核心目标，围绕低成本、高灵活性的智能网联技术验证平台构建展开研究，具体内容包括以下方面：

（1）智能网联小车的硬件系统设计与集成

为了满足低成本和模块化要求，需要完成核心硬件选型如主控单元、传感器和通信模块等，另外为满足高灵活性需要设计较为紧凑的硬件架构。实现传感器的布局与数据接口的适配，解决不同硬件不同的供电需求等。

（2）基于 ROS 的软件系统开发与优化

配置 ROS 开发环境，设计各节点化功能模块，实现多模块协同运行。提升车道保持系统环境感知的精度；优化算法增强复杂光照下的程序的适应性。

（3）实验测试与性能评估关键功能

设计车道保持功能场景测试方案利用多场景进行对比验证。验证算法有效性，量化评估系统实时性、鲁棒性等核心指标，提出优化策略。

1.3.2 本文结构

本设计主要分为一下六章，各章节内容安排如表 1.1。

表 1.1 各章节内容安排

章节	内容
第一章 绪论	研究背景与意义、国内外研究现状、存在问题、研究目标与内容
第二章 系统总体设计	总体设计方案、硬件架构选型、整体布局、任务规划

第三章 硬件控制系统设计与实现	底层控制系统硬件电路设计、方案分析
第四章 硬件控制程序设计	底层开发环境、底层驱动开发、相关驱动程序
第五章 基于 ROS 的智能小车 LKS 功能实现	LKS 功能实现原理、相关算法、实验验证
第六章 全文总结与展望	研究成果总结、当前不足点、后续完善方向

1.4 本章小结

在这个章节全面介绍了本设计的背景、意义和国内外研究现状，并说明了本设计的核心内容和结构布局。由于汽车行业相关技术的快速发展，智能网联汽车成为全球相关领域研究的焦点，市场规模继续扩大。通过分析现状得出该领域也面临着研发成本高、测试风险高和跨学科人才培养不足等问题。在这个大背景下，基于 ROS 的智能网联小车孕育而生。

2 系统总体设计

2.1 系统总体方案概述

本设计采用自己设计的底层控制板，下位机底层控制采用 STM32 芯片进行控制，上位机平台采用 Jetson Orin Nano。传感器部分包括激光雷达、深度相机、四个 usb 鱼眼镜头等。通过这些传感器数据感知四周环境，实现包括车道保持（LKS）在内的相关高级驾驶辅助功能。

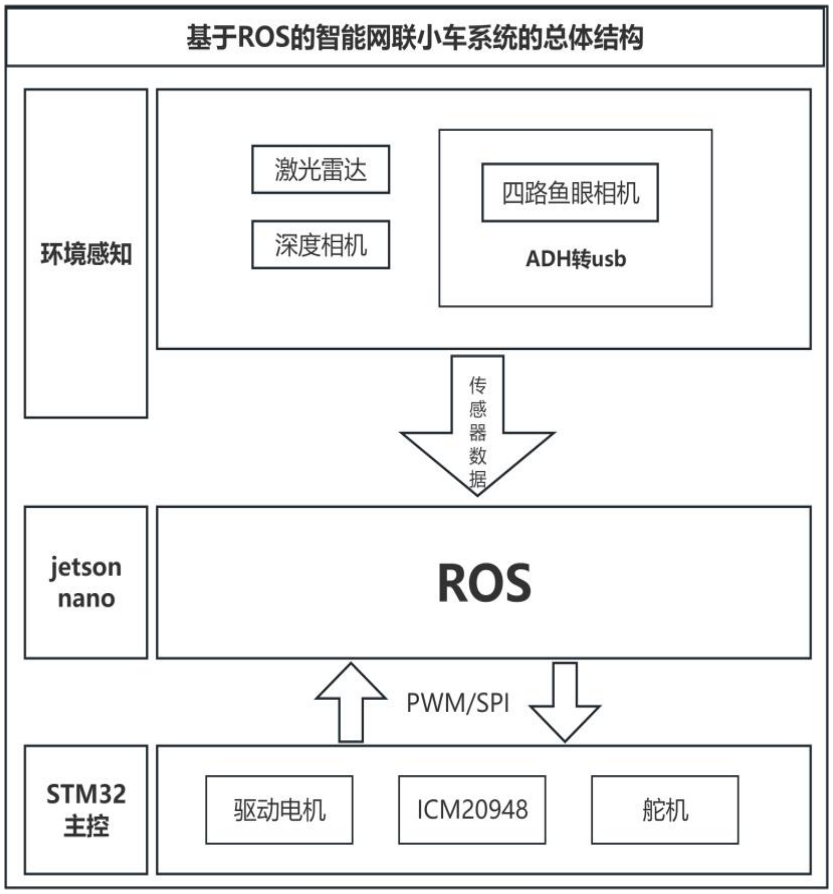

图 2-1 小车总体框架

2.2 小车结构设计

在设计移动平台结构和选择核心组件时，需建立完整的硬件体系认知。小车整体采用层成高强度铝合金框架分层架构设计如图 2-2，分为底盘层、计算层和传感器层。其核心机械设计特点如下：

（1）分层架构设计

底盘层：使用了铝合金框架，用以支撑整个小车采用铝合金框架不仅提高了强度还降低整体重量。小车采用阿克曼转向结构，使用高精度直流电机和舵机。底部预留多个 M4 螺丝孔位，用于支撑上层平台。

计算层：中段平台的中部位置用于放置 jetson nano 上位机。尾部位置用于安装 STM32 底盘控制板。同时设计了独立散热风扇，确保计算平台的长时间高负载运行。

传感器层：上层平台中部设计了用于安装激光雷达的位置，在前面也可安装可以调整俯仰角度的摄像头模组。

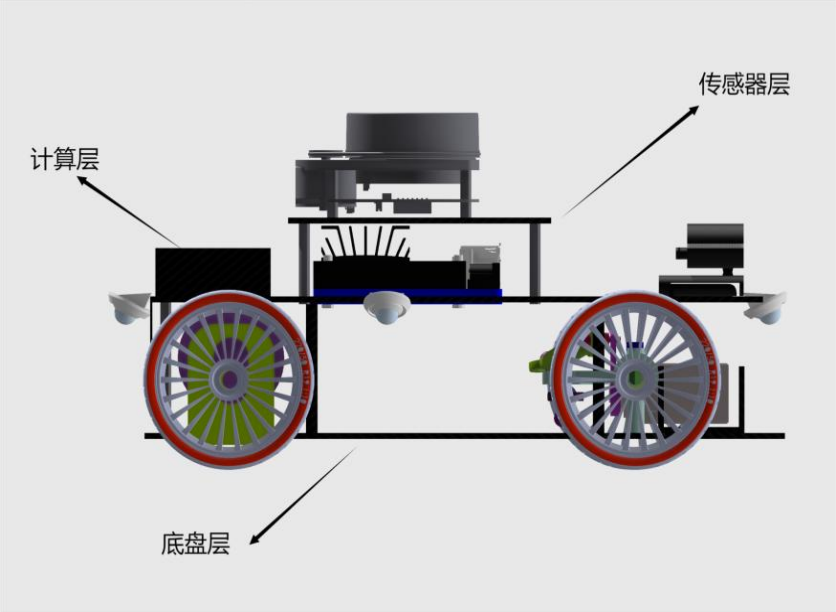

图 2-2 分层架构设计图

2.2.1 传感器选择

在智能网联汽车中非常重要的部分就是环境探测传感器，传感器也是智能小车感知周围环境的“眼睛”[7]，基于 ROS 的智能网联小车采用多传感器方案。各传感器的选型依据及技术参数如下：

(1)激光雷达

激光雷达是自驾中的重要传感器。通过发射激光脉冲并计算反射时间来精确测量距离，生成高分辨率的三维点云数据，为自动驾驶车辆提供周围环境的状况。本设计选用的是 RPLIDAR A1，2D 激光雷达。具体参数如表 2.1。

表 2.1 RPLIDAR A1 技术参数

参数名称	参数值
测距范围	0.15 米至 12 米
测距精度	实际距离的 1%（≤3 米），实际距离的 2%（3-5 米），实际距离的 2.5%（5-12 米）
扫描频率	5.5Hz，最高可达 10Hz
采样频率	8000Hz
角度分辨率	≤1°
激光波长	785nm
通信接口	UART
供电电压	5V
功耗	0.5W
工作温度范围	0℃至 40℃
尺寸	96.8mm×70.3mm×55.0mm
重量	248 克

图 2-3 RPLIDAR A1 实物

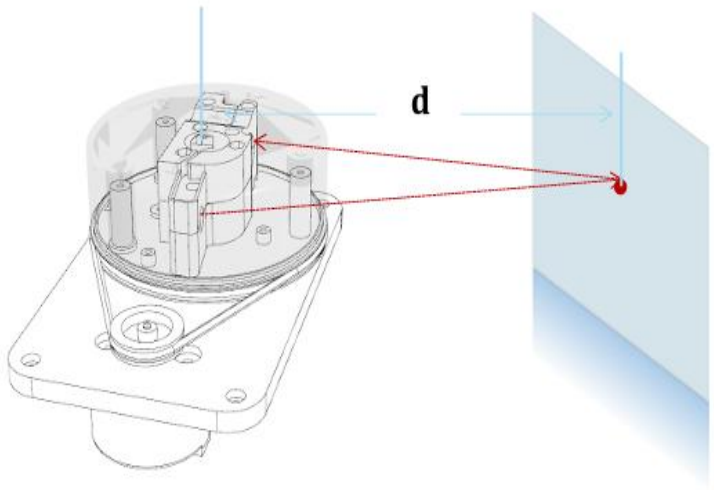

图 2-4 三角测距原理

（2）深度相机

深度相机也被称为 3D 相机，它和普通 2D 相机的区别在于可以获取物体到相机的距离信息，加之 2D 平面的 X，Y 坐标，可计算出每个点的三维坐标[8]，以此我们可推断深度相机的应用，如三维重建、目标定位、导航避障等。

本小车选用的深度相机的型号为 Astra Pro，相机采用结构光技术，通过投射红外光图案并分析其在物体表面的变形来计算深度信息。其内置的 ASIC 芯片能够高效处理深度数据，确保高精度和高帧率的深度图像输出。具体技术参数如表 2.2。

表 2.2 Astra Pro 技术参数

参数名称	参数值
测距范围	0.6 米至 8 米
测距精度	±1-3 毫米@1 米
深度图像分辨率	640x480 @30fps
RGB 图像分辨率	1280x960 @7fps,640x480@30fps
视场角	水平 60°，垂直 49.5°，对角线 73°

激光波长	850nm
通信接口	USB2.0
供电电压	5V
功耗	最大 2.5W
工作温度范围	10℃至 40℃
尺寸	165mm×30mm×40mm
重量	145 克

图 2-5 Astra Pro 深度相机

（3）鱼眼相机

鱼眼相机作为智能网联汽车中基础的传感器。其能够提供超广的视角，四个鱼眼相机就能覆盖车辆周围的范围，捕捉整个近距离区域。在复杂的城市交通环境中，它能够有效识别行人、车辆、自行车等交通参与者，帮助智能网联汽车准确判断周围环境状况，为后续的决策和控制提供可靠的数据支持，同时在自动泊车系统中发挥着重要作用[9]。通过捕捉车辆周围的全景影像，系统可以实时监测车辆与障碍物之间的距离和位置关系，帮助车辆准确地停泊在车位中。这不但提高升泊车的效率、安全和驾驶体验。

本小车选用的鱼眼镜头为科霸 180° 广角镜头，具体技术参数如表 2.3。

表 2.3 科霸 180° 广角镜头技术参数

参数名称	参数值
品牌	科霸
电源电压	5V(V)
分辨率	1920*1080P
抗震强度	6.8G
视频信号	AHD
重量	200g(g)
材质	塑料+金属
调整角度	180°
功率	0.1(W)(W)
扫描频率	1920*1080P
外壳材质	金属壳
正像/镜像	镜像

图 2-6 科霸 180° 广角镜头

2.3 底层主控制器选择

本设计选用了 STM32F103RCT6 微控制器，其采用了 90 纳米的制造工艺，在实时控制和数据处理中其性能与效率比较优秀。在价格较低的同时依然满足设计需求。

其技术特性可划分为表 2.4。

表 2.4 STM32F103RCT6 技术参数

技术维度	特性描述
核心架构	ARM Cortex-M3 内核
存储配置	256KB Flash（支持扇区擦写）
通信接口	CAN 2.0B ×1、USB 2.0 全速×1、USART×3（支持 ISO7816）、SPI×2（18Mbps）、I²C×2（兼容 SMBus/PMBus）
控制模块	高级 PWM 定时器×2（6 通道互补输出）、12 位 ADC×2（1μs 转换/16 通道）、独立/窗口看门狗双保护机制
时钟系统	主时钟：4-16MHz 晶振
电气特性	工作电压：2.0-3.6VDC
开发支持	调试接口：SWD/JTAG

图 2-7 STM32F103RCT6

通过 STM32F103RCT6 微控制器技术特性可以发现选用此微控制器作为本设计底层控制核心完全满足使用需求。ARM Cortex-M3 内核的设计让该控制器具有较为强大的实时处理能力，能够高效执行本设计相关的运动控制任务。这些特性保证了系统能够在毫秒级别上做出响应，满足了本设计对于实时数据同步的要求。另外此款微控制器还具有多种通信接口，内置了 CAN2.0B、多路 USART/SPI/I²C 等多种标准协议。利用 STM32CubeMX 等工具可以快速开发底层控制程序。

2.4 机器人操作系统（ROS）

ROS 可以为智能小车上位机程序的设计和开发提供一套标准化的框架，ROS 的核心价值就是于通过模块化设计实现复杂系统的快速集成与迭代。通过硬件抽象可以将激光雷达、深度相机、鱼眼镜头这些传感器统一封装为标准化数据格式，降低了不同传感器的协同开发难度[10]。同样 ROS 的分布式通信架构如图 2-8 非常适合本设计的分层控制需求，利用 STM32 等微控制器处理底层实时运动控制如电机 PWM 驱动、舵机角度调节，同时 Jetson Nano 通过 ROS 节点运行 LKS 算法，形成“感知-决策-执行”的闭环逻辑。

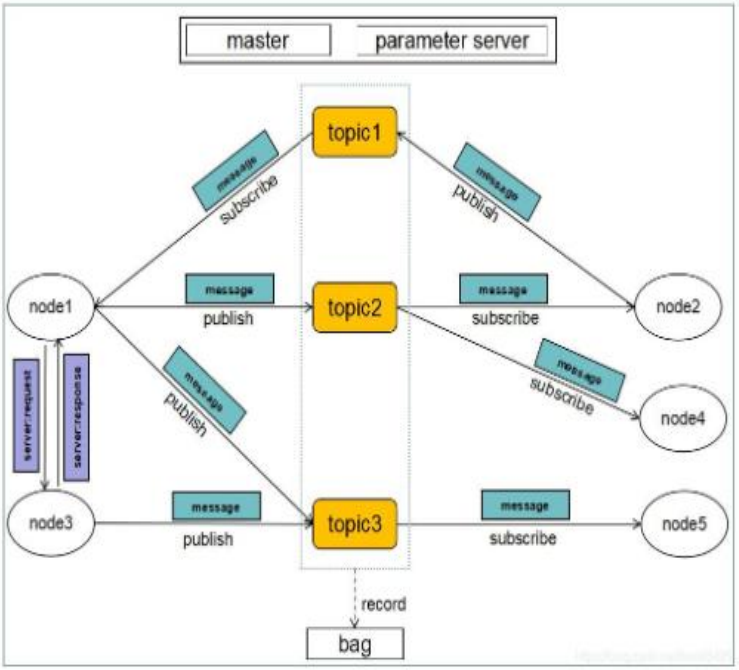

图 2-8 分布式进程

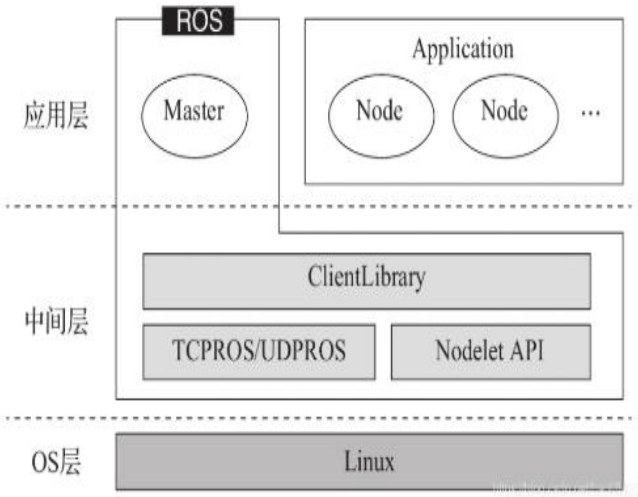

图 2-9 ROS 的分层架构图

2.5 Jetson Orin Nano 介绍

小车选用的是 Jetson Orin Nano8G 版本。从计算资源数量来看，Orin Nano 8GB 的规格为 AGX Orin 64GB 版的一半，不过频率更低，因此功率也更低。Nano 模组的功率只有 15W。在极低的功率下，Orin Nano 依然能提供极强的性能。

由图 2-9 可以发现,Orin Nano 达到了每秒 40 万亿次运算(TOPS)的 AI 性能，是之前 nano 的 80 倍。

Models	Jetson Nano (FPS)	Jetson Orin Nano 8GB (FPS)
PeopleNet (v2.5 unpruned)	2	116
Action Recognition 2D	32	368
Action Recognition 3D	1	26
LPR	47	979
Dashcam Net	11	398
Bodypose Net	3	136
Resnet50	36	1144

图 2-10 orin nano 和 nano 之间的性能对比

小车整体视图如图 2-11。

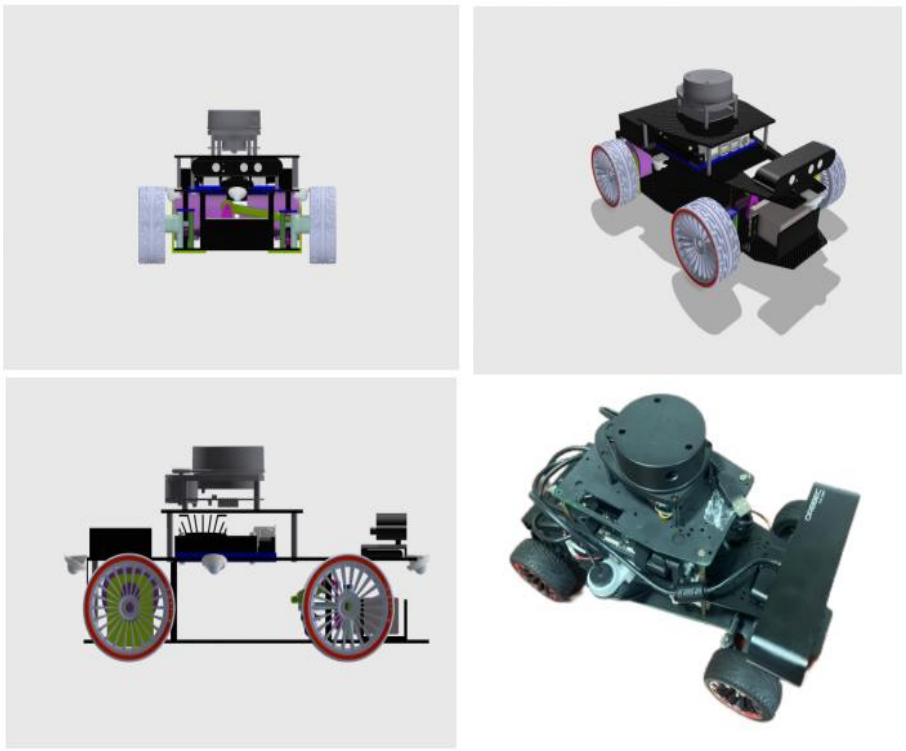

图 2-11 小车整体视图

2.6 本章小结

在一章主要介绍了本设计的总体系统设计方案，硬件安装平台搭建、传感器选型和底层控制器核心选型。介绍了机器人操作系统（ROS）和 Jetson Orin Nano。和它们的系统任务分工。

3　硬件控制系统设计与实现

本设计采用 STM32F103RCT6 作为底层控制芯片，底层控制系统主要包括电源，AM2857 电机驱动模块、PWW 舵机模块等模块等，还包括两个编码直流电机和一个舵机，本章将主要介绍底层控制板的硬件设计与实现。

3.1 Altium Designer 介绍

Altium Designer 是一款电子设计自动化软件，该软件集成了多种功能，包括元件库管理、原理图设计和 PCB 设计被广泛应用于工业自动化、通讯设备、医疗设备等领域[11]。在 PCB 设计上比 PROTEUS 有明显的优势。它提供了一体化设计平台，有强大的布线功能和自动布局优化工具，拥有 3DPCB 视图和实时设计检查功能，以及丰富的部件库和完善的项目管理功能。本设计在 PCB 板设计时选择了该平台。

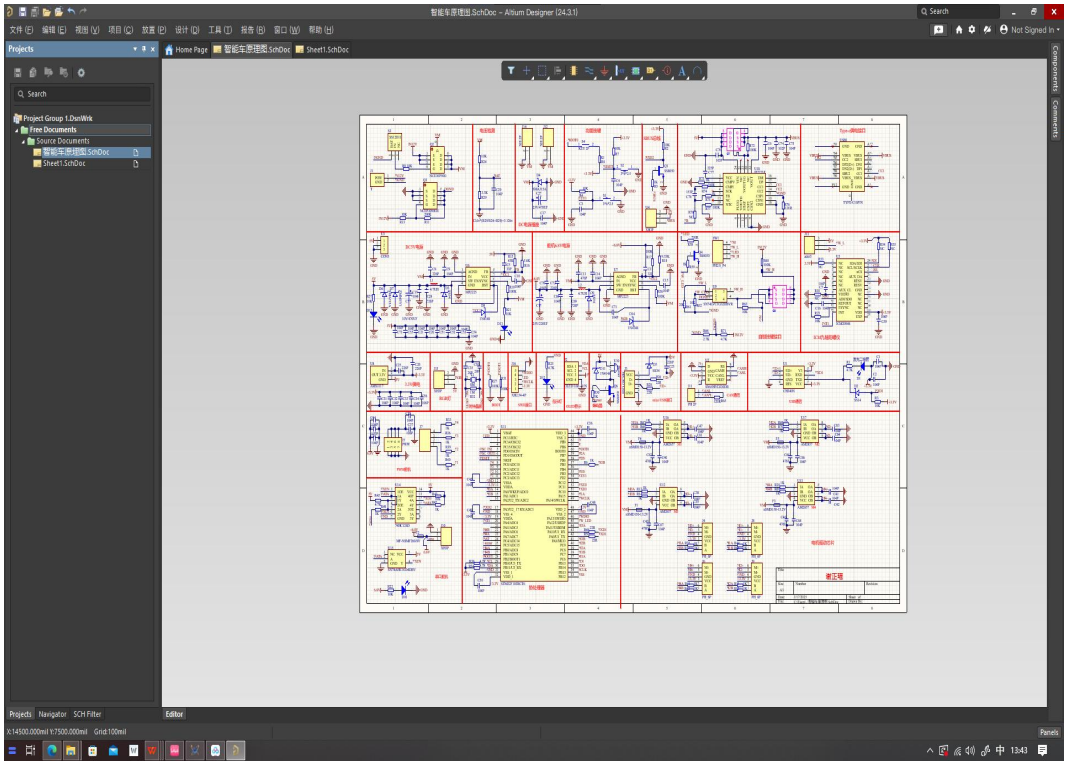

图 3-1 Altium Designer

3.2 底层控制板系统设计

底层控制板原理图如图 3-2，PCB 板设计图如图 3-3。

3.3 STM32 最小系统

本设计的最小系统原理图如图 3-4。

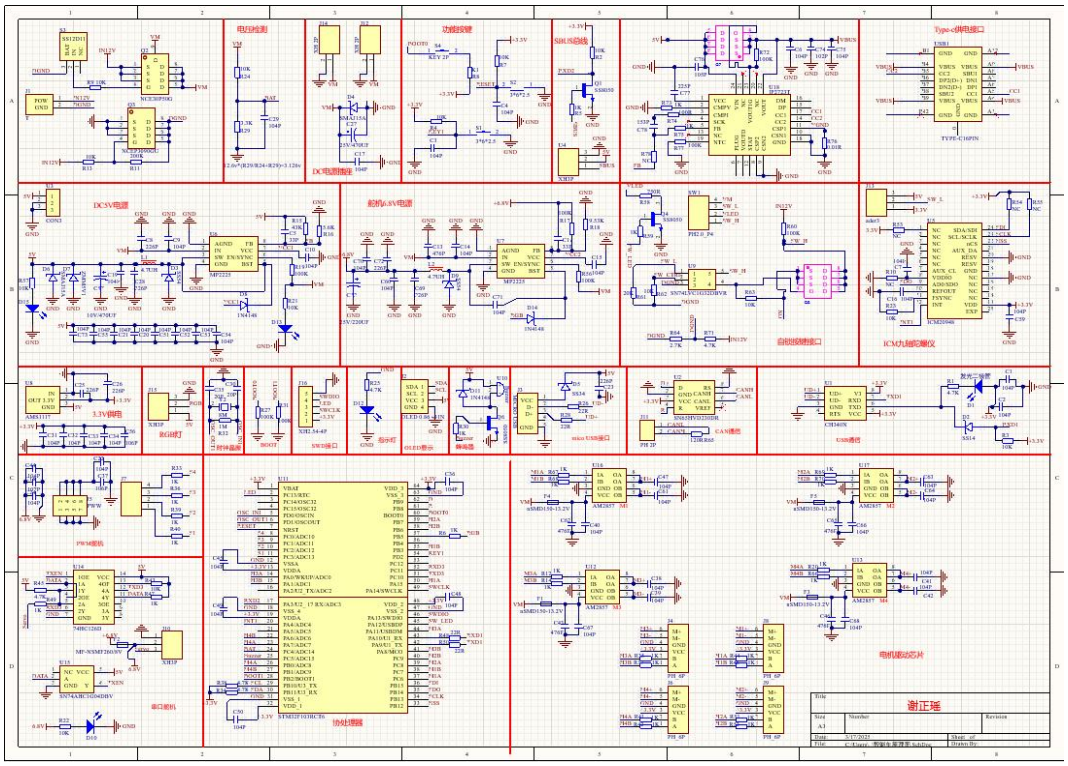

图 3-2 底层控制板原理图

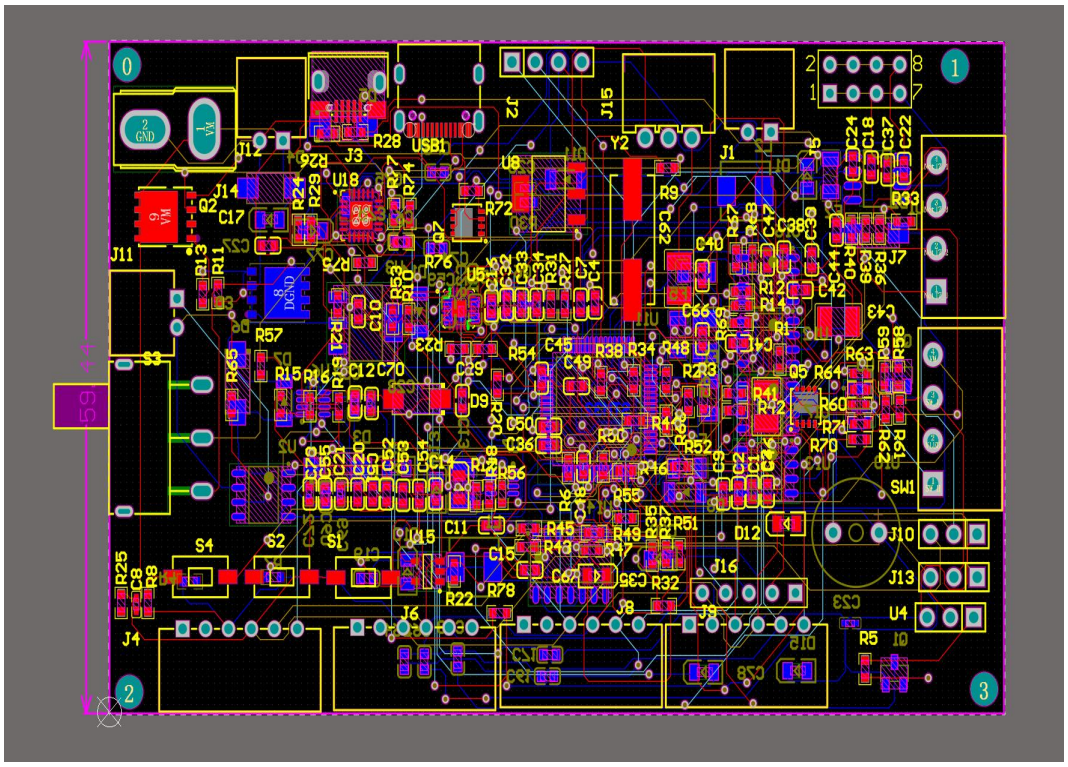

图 3-3 PCB 板设计图

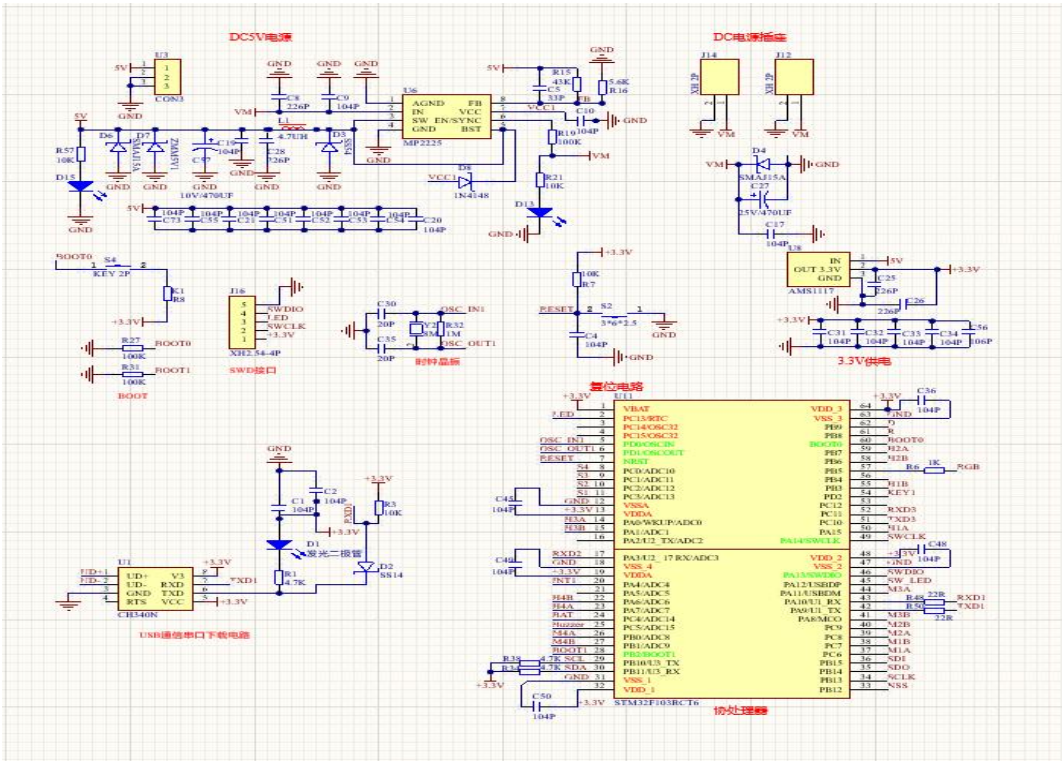

图 3-4 最小系统原理图

3.3.1 电源电路

根据小车搭载的各传感器和计算平台的电器特性，结合驱动板本身。确定输入电源为直流 12V。

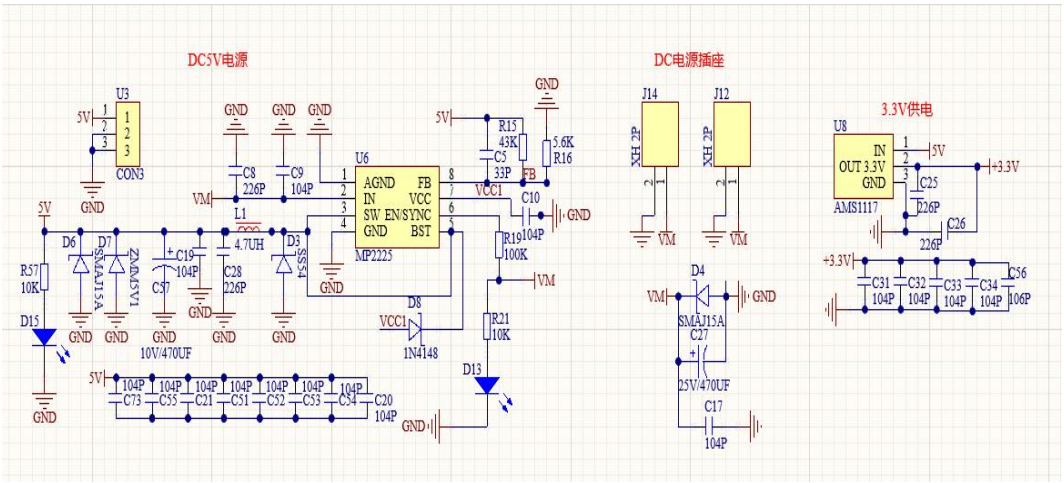

图 3-5 电源电路

其中 DC5V 电路利用开关原理实现将输入的 12V 直流电源转换为 5V 电压输出。电路中，MP2225 芯片作为开关控制器进行高频开关动作，将输入的电压斩波。其中 SM 为开关引脚，FB 为反馈引脚，BST 为内部 MOSFET 引脚。输入电源先经过

电容 C8、C9 进行滤波和储能，确保输入到 MP2225 芯片的电压稳定。MOSFET 导通后，电流经过电感 L1 流向负载，同时电感储能量。当 MOSFET 截止时，电感 L1 储存的能量通过 D3 释放，输出稳定的电压。同样舵机输入的 6V 电源也通过相同电路变换。另用一个 AMS1117 模块将输入的 5V 电压转换为 3.3V 输出。

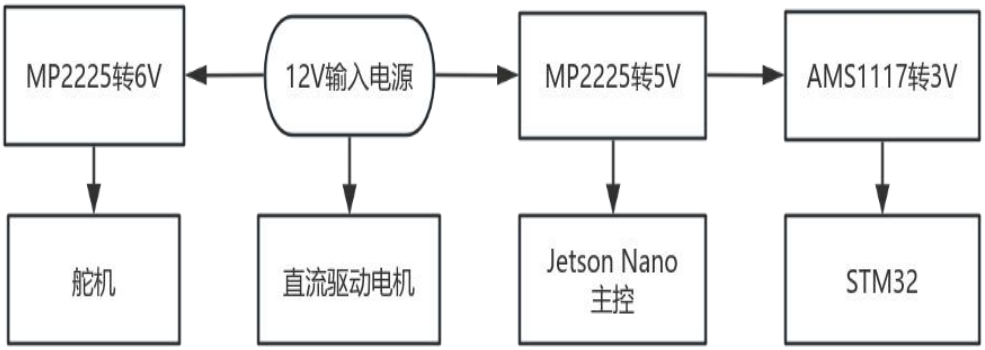

图 3-6 电源供电模块结构图

3.3.2 晶振电路

当 STM32 上电时，晶振电路开始工作。晶振与电容器 C30 和 C35 构成振荡电路。在振荡电路帮助下开始振荡。这个振荡信号被送入微控制器的内部振荡放大器对信号进行放大，输出为方波信号。这个信号就是微控制器的时钟信号，它决定了 STM32 的运行速度和定时精度。本系统的晶振电路如图 3-7。

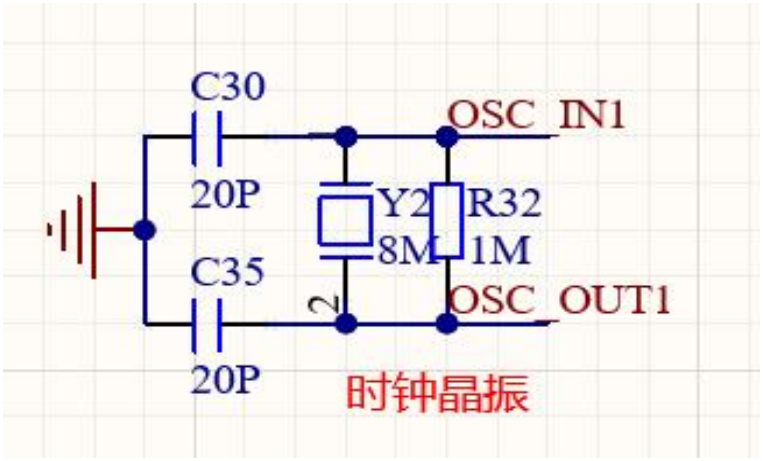

图 3-7 晶振电路

3.3.3 SWD 接口电路

在嵌入式微控制器的设计中程序下载，通常有 SWD 和 JTAG 两种接口，但它们在引脚数量、信号功能方面存在一些区别。SWD 接口通常使用两个信号线，SWCLK 和 SWDIO 还有电源和地线。而 JTAG 接口使用多个信号线，包括 TCK、TMS、TDI、TDO、TRST 等，另外还需要电源和地线。为了使在 PCB 板设计时能更少的占用空间，所有本设计采用 SWD 接口下载和调试程序。SWD 接口电路如图 3-8。

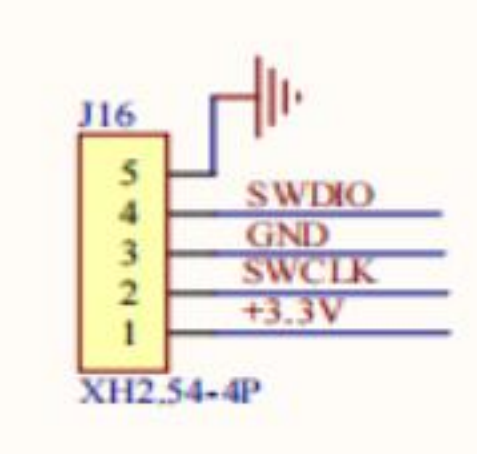

图 3-8 SWD 接口电路

3.3.4 复位电路

复位电路也被叫做初始化电路，它作用是让微控制器恢复到初始状态，主要是在电源接通时的短暂上升期内保证所有模块初始化成功后才允许为控制器发出指令，或在调试程序出现问题后通过复位电路进行恢复。

本设计的复位电路如图 3-9。

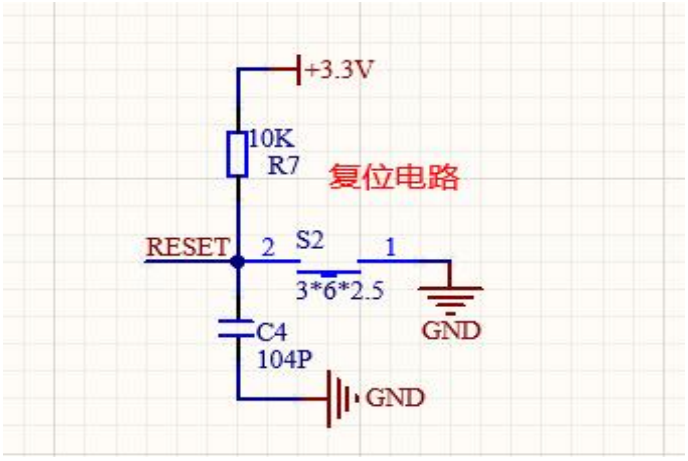

图 3-9 复位电路

3.3.5 USB 通信电路

为了实现 Jetson Orin Nano 和底层控制板的通信，Jetson Orin Nano 负责初始化通信和处理数据。底层控制板响应 Jetson Orin Nano 的请求，按照指定的协议进行数据的接收、发送和执行相关动作。本设计通过 mico USB 接口与 CH340H 芯片实现串口通信。原理图如图 3-10。

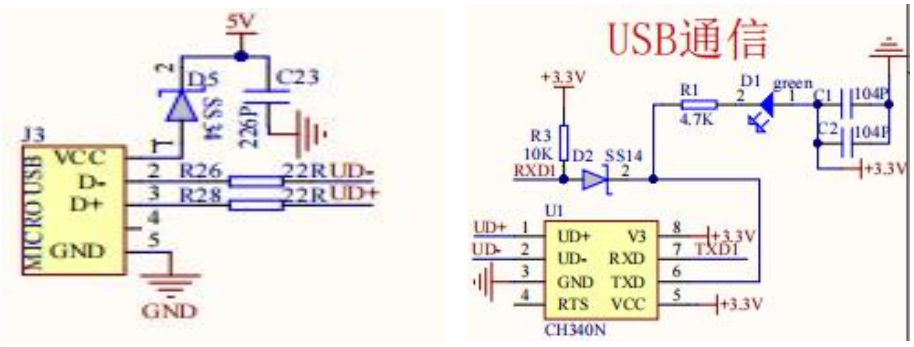

图 3-10 USB 通信电路

3.4 电机驱动电路

本设计的电机驱动芯片选用 AM2857，12V 电源通过 VM 和 nSMD150-13.2V 保险输入给电机。微控制器生成控制信号，由引脚输出至驱动芯片，芯片根据接收到的信号控制电机的正反转和启停。AM2857 引脚 1 输出高电平，引脚 2 输出低电平时，电机正转，反之电机反转。当引脚 1 和引脚 2 引脚都为低电平或高电平时，电机不转。通过读取 H1A 和 H2A 的电平状态来判断电机的当前状态，并根据需要调整控制信号。电机驱动电路如图 3-11。

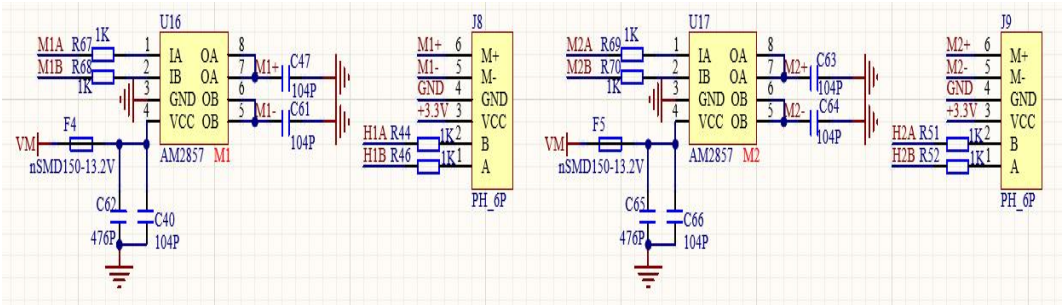

图 3-11 电机驱动电路

3.5 舵机电路

本设计实现 PWM 控制舵机什么的是 STM32 的基础定时器中断功能，模拟输出 PWM 信号，控制 PWM 舵。舵机有三条线分别是电源线（VCC）、地线（GND）、信号线（PWM）。其中信号线可以接在 S1、S2、S3、S4 中任意一端上。

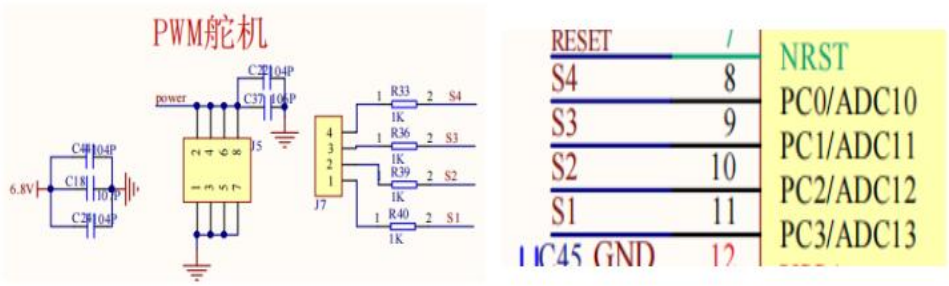

图 3-12 PWM 控制舵机电路

3.6 本章小结

本章节详细介绍以 STM32F103RCT6 微控制器作为底层控制核心的小车硬件电路示例。为了满足本设计的灵活性与紧凑使用 AD 软件完成了控制板的原理图设计与 PCB 布局设计，详细介绍了 STM32 最小系统、电机驱动电路和舵机驱动电路，为实现对小车运动的精准控制和后续软件开发提供了支持。

4 硬件控制程序设计

本章节将设计硬件驱动控制程序，包括电机驱动程序、舵机驱动程序串口通信程序设计和 PID 算法实现，通过 STM32CubeMX+Keil 联合进行程序设计。最后为了能实现 ROS 开发编写了通信驱动库。

4.1 开发环境介绍

STM32CubeMX 是 ST 意法半导体近几年来大力推荐的 STM32 芯片图形化配置工具，通过自己对硬件的需要，进行选择，而后可以快速生成 C 代码，CubeMX 还提供了自动识别引脚冲突、设置时钟树、功耗预测等功能[12]。

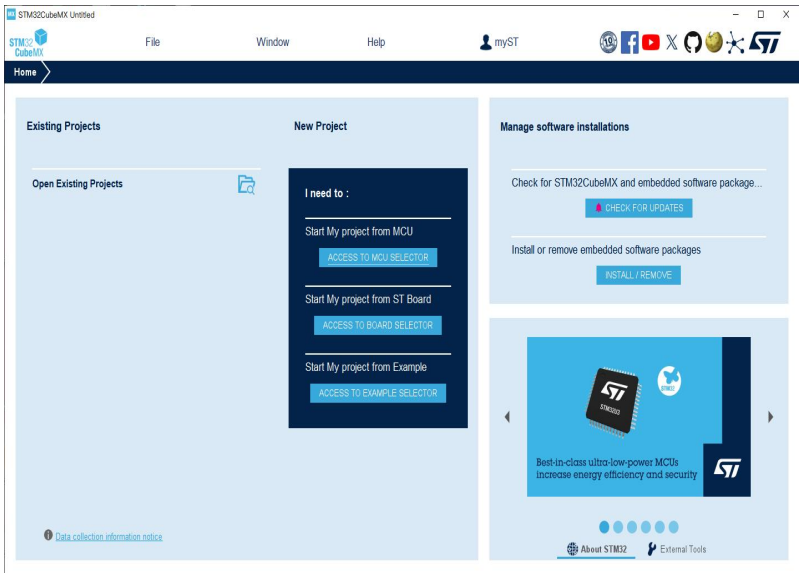

图 4-1 STM32CubeMX

Keil 主要用于对嵌 stm32 进行编程。它的功能主要包括源代码编辑器、项目管理、调试。它支持常用的各种基于 ARM 的微控制器，为软件开发提供了完善的环境，允许开发人员在不需要实际硬件的情况下测试代码。从而减少了开发时间和工作量。

4.2 电机驱动程序设计

使用 STM32 的定时器功能，驱动电机驱动芯片 STM32 的定时器功能，从而控制电机正转、反转和停止[13]。根据原理图总共有四个 AM2857 电机驱动模块，一个电机驱动模块控制一个电机，使用 STM32CubeMX 设置定时器 1 和设置定时器 8。

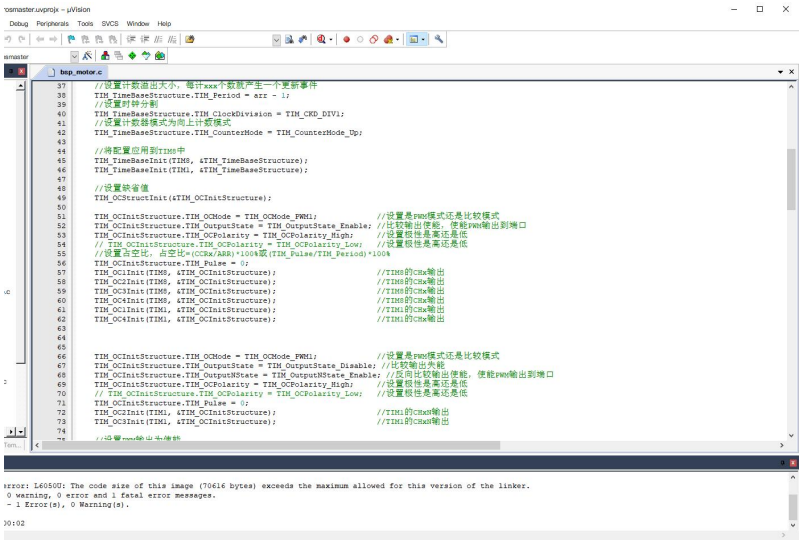

图 4-2 Keil 集成开发环境

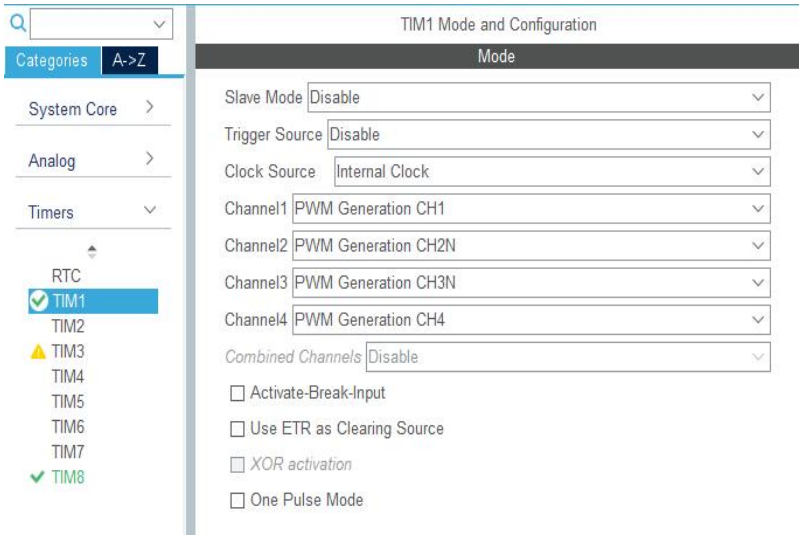

图 4-3 定时器 1 设置

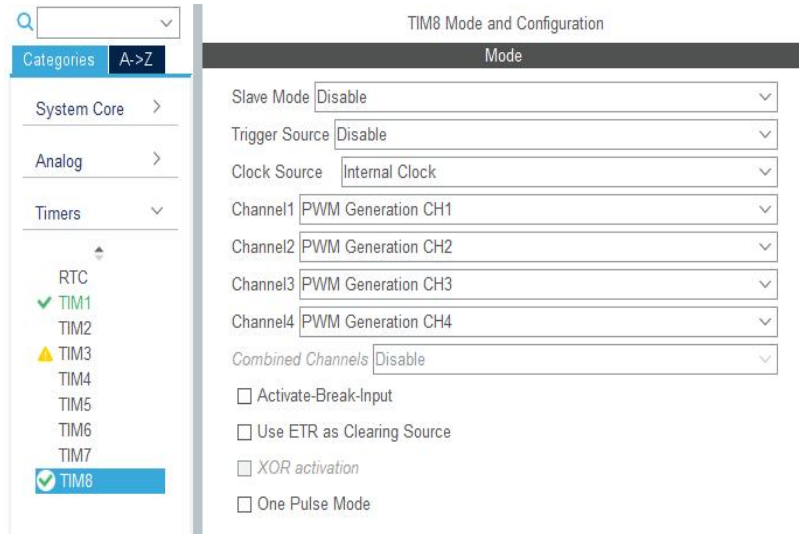

图 4-4 定时器 8 设置

最后芯片配置引脚如图 4-5 所示。

导出工程文件用 Keil 打开设计程序。

电机驱动函数伪代码如图 4-6。

程序执行流程如图 4-7。

程序设计流程如图 4-8。

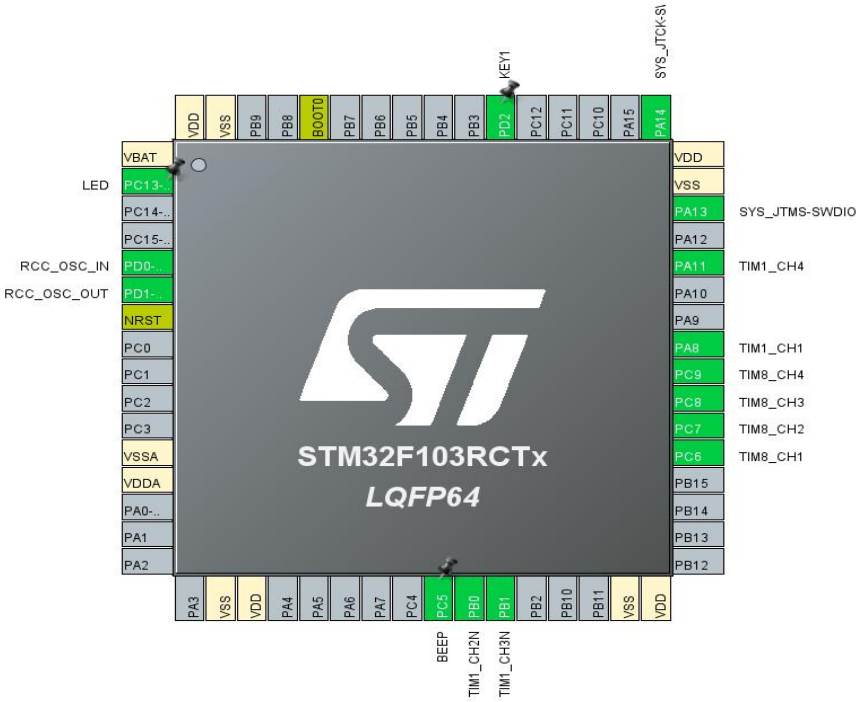

图 4-5 芯片配置引脚

Algorithm 1: Optimized Motor PWM Control

Input: id, speed
Output: PWM register update
begin
  speed $\leftarrow \max(\min(speed, \text{MAX}), -\text{MAX})$;
  speed $\leftarrow$ speed $+ \operatorname{sgn}(speed) \cdot \text{DEADZONE}$;
  switch *id* **do**
    case *M1/M2* **do**

$$(\text{ccrA}, \text{ccrB}) \leftarrow \begin{cases} (1,2) & \text{if M1} \\ (3,4) & \text{if M2} \end{cases};$$

      if $speed \geq 0$ **then**
        $\text{TIM1} \triangleright \text{CCR}[A] \leftarrow speed$;
        $\text{TIM1} \triangleright \text{CCR}[B] \leftarrow 0$;
      else
        $\text{TIM1} \triangleright \text{CCR}[B] \leftarrow -speed$;
        $\text{TIM1} \triangleright \text{CCR}[A] \leftarrow 0$;
      end
    end
    otherwise do
      Motors handled similarly
    end
  end
end

图 4-6 电机驱动函数伪代码

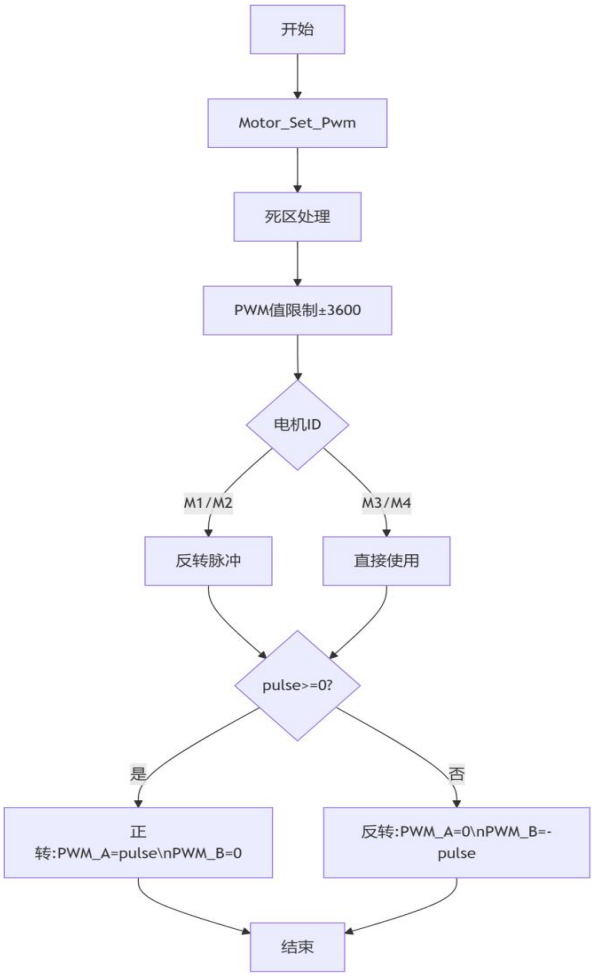

图 4-7 程序执行流程

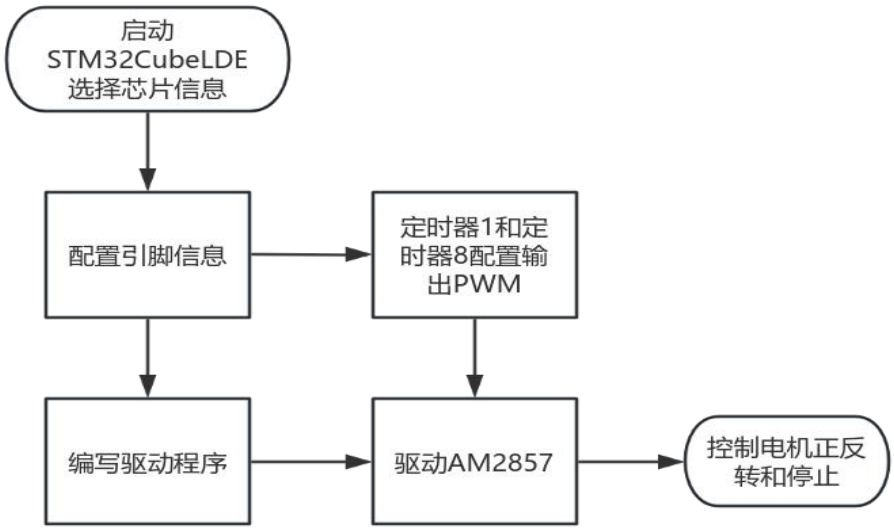

图 4-8 程序设计流程

4.3 舵机驱动程序设计

本设计使用 STM32 的基础定时器中断功能，模拟输出 PWM 信号，控制 PWM 舵机。根据原理图可知，舵机 S1 S2 S3 S4 分别连接到 STM32 的 PC3 PC2 PC1 PC0 引脚。设置 PC0 PC1 PC2 PC3 引脚为输出模式，具体参数如图 4-9 所示：

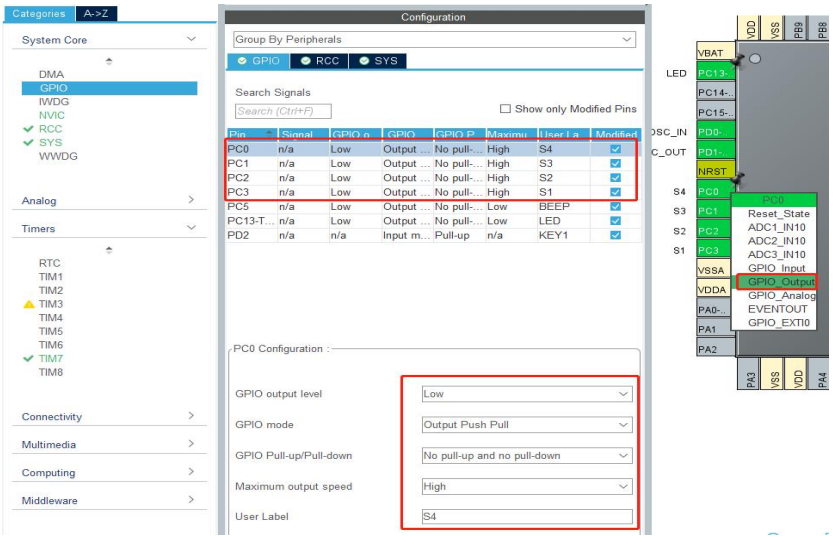

图 4-9 具体参数

配置定时器 7，具体配置参数如图 4-10 所示。

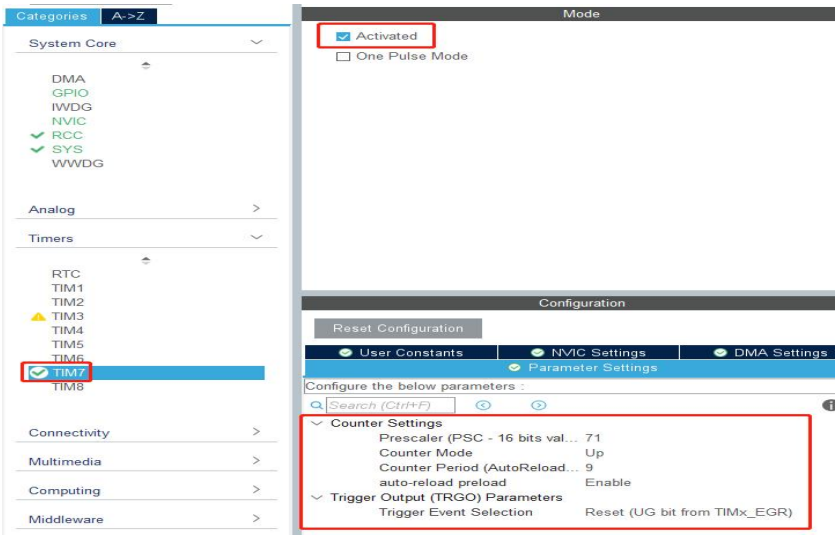

图 4-10 配置定时器 7

打开定时器全局中断设置如图 4-11。

图 4-11 打开定时器全局中断设置

舵机驱动函数伪代码如图 4-12。

Algorithm 1: Optimized Servo Control Protocol

Input: id ∈[1,6], angle ∈[0,180]
Output: RS485 command frame
begin
  if *id invalid* ***or*** *angle* > *180* **then**
    return
  pulse ← $\lfloor \frac{angle \times 1000}{180} \rfloor$ + FlashRead(id);
  pulse ← min(pulse, ARM_MAX);
  cmd ← [0x55,0x55,id,0x07,0x01,(pulse≫8)&FF,pulse&FF,0x00];
  for $i \leftarrow 0$ ***to*** *7* **do**
    USART_Send(cmd[i]);
    while ¬*USART_Ready()* **do**
      ; // Spinlock
    end
  end
end

图 4-12 舵机驱动函数伪代码

程序执行流程如图 4-13。

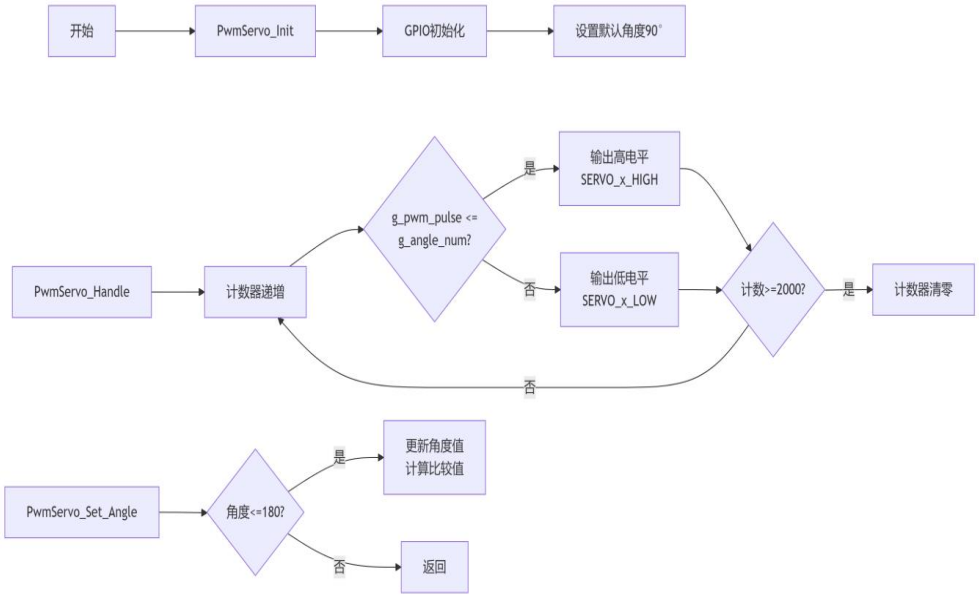

图 4-13 程序执行流程

程序设计流程如图 4-14。

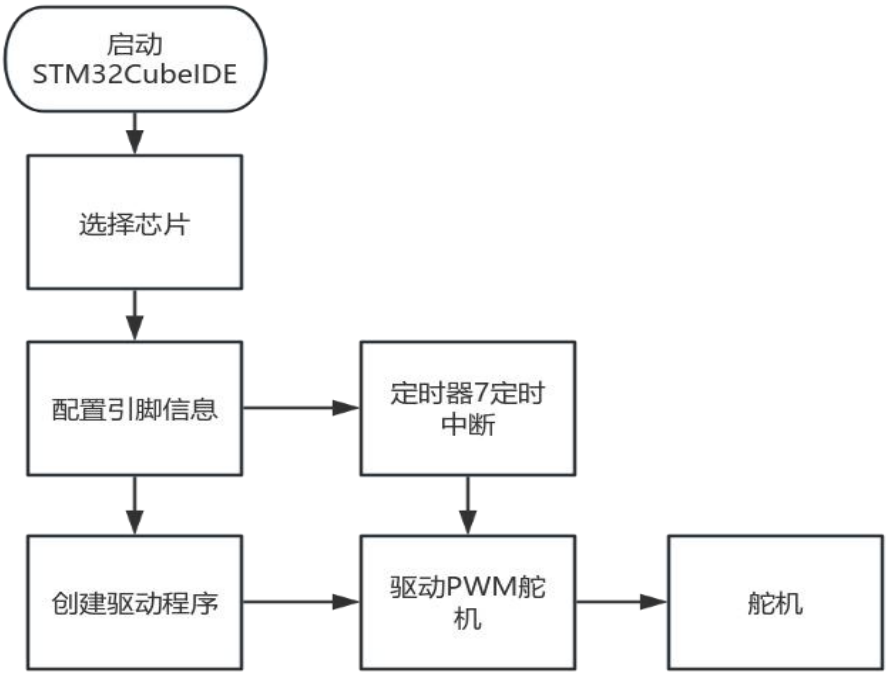

图 4-14 程序设计流程

4.4 烧录程序

由于控制板的 USB 通讯使用了 CH340N 芯片，所以需要安装 CH340N 芯片的驱动。

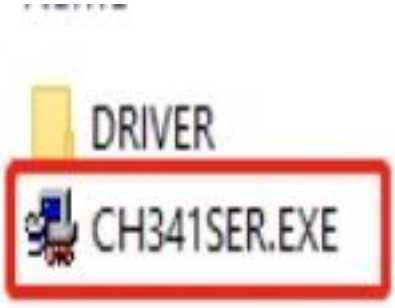

图 4-15 CH340N 驱动

本设计烧录固件需要用到 mcuisp（或 flymcu）烧录软件，将 USB 数据线的一端插入电脑 USB 接口，另一端插入控制板的 Micro USB 接口。点击【开始编程】，mcuisp 烧录软件会将我们上一步选择的固件烧录到控制板上的单片机上。当右边出现如图 4-17 的提示就表示下载成功。

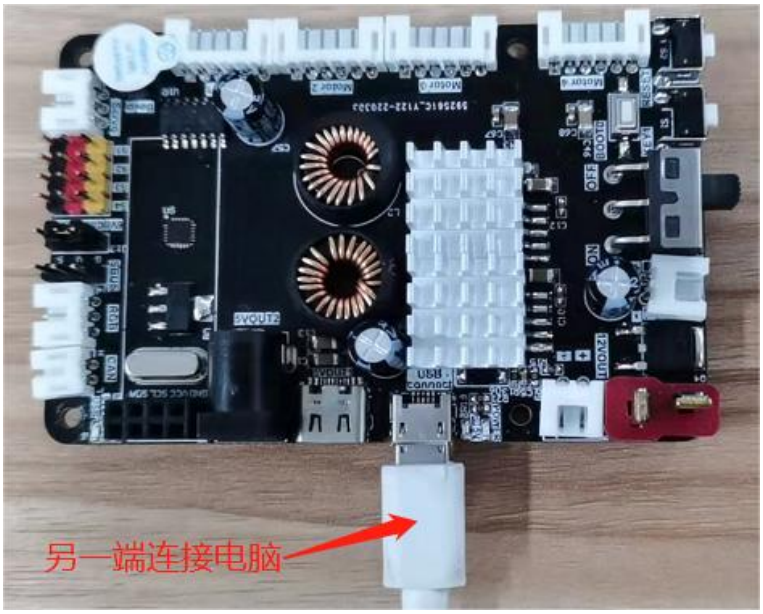

图 4-16 连接控制板

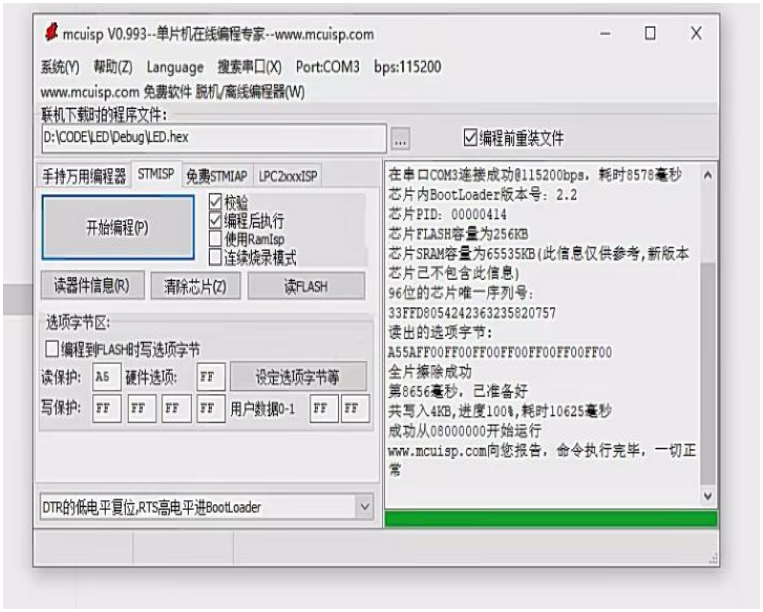

图 4-17 固件烧录

4.5 通信系统设计

通常复杂的智能车系统会存在多个控制器，不会是一个控制器完成所有的任务，这样会造成计算资源的过度集中，并且无法实现模块化的调试。以简单的智能车为例，一般会存在两个控制器，一个是运行 ROS 的主控，另一个是运行电机控制和传感器信息采集的单片机 STM32[14]。由于存在多个控制器，完成一个具体任务，那么这多个控制器间则需要建立通信。

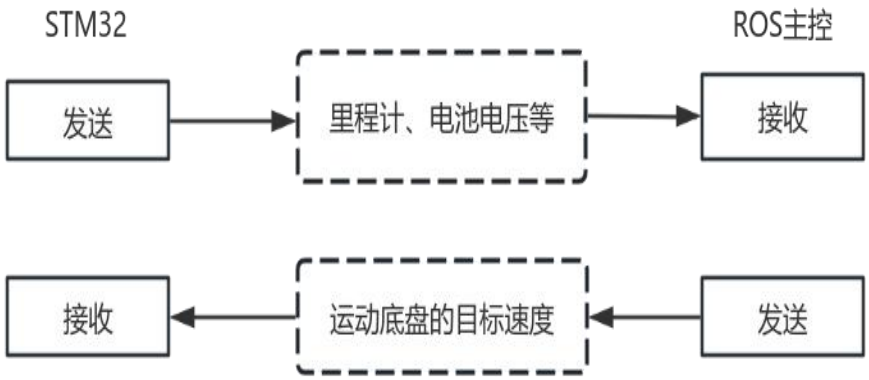

图 4-18 通信系统框架

ROS 主控与 STM32 之间需要做到一个双向的数据传输，这里就涉及到了两个控制器之间的通信问题，本设计的通信系统框架图 4-18 下面则介绍本设计如何实现两者之间的通信。

4.5.1 硬件连接

ROS 主控通过 usb 线连接到 mico USB 接口，再由 CH340H 芯片将 USB 信号转换为 TTL 电平信号，通过 TXD1 和 RXD1 引脚与 STM32F103RCT6 的串口引脚相连，实现串口通信图 4-19。

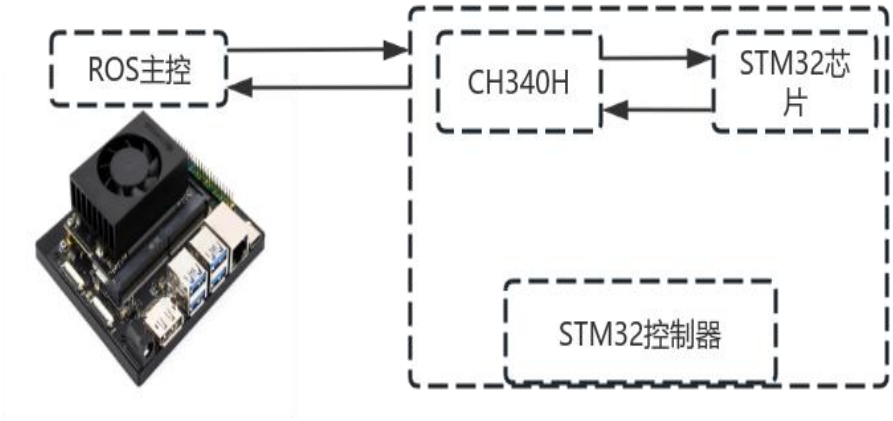

图 4-19 串口通信

4.5.2 软件设计

（1）通信协议

设计一份通信协议需要先明确应用场景、设备类型、通信方式和功能需求等，再划分功能模块，定义数据格式，设计校验机制，最后撰写文档并优化。此小车的通信协议设计，选择串行通信，根据功能需求，划分出自动发送数据、PWM 舵机控制、小车运动控制等模块，定义了数据包结构、字段说明、数据类型和字节顺序。接着设计校验位计算和错误处理机制。本设计撰写的协议控制的核心部分文档如图 4-20。

功能	包头1	包头2	长度	功能字	开关	永久保存	checksum	备注：默认为开启状态，四包数据轮流发送，每包数据间隔10毫秒，相当于一包数据每40毫秒更新一次。如果关闭后，不会自动返回数据，会影响Python驱动库读取线速度、角速度、陀螺仪、加速度计、电池电压等数据。
设置自动发送数据	0xFF	0xFC	0x04	0x01	0x00|0x01	0x00|0x5F	XX	
下标	0	1	2	3	4	5	6	

功能	包头1	包头2	长度	功能字	ID（1~4）	角度(0-180	checksum	备注：ID对应四路PWM舵机，角度控制范围：0~180。
PWM舵机控制	0xFF	0xFC	0x05	0x03	XX	XX	XX	
下标	0	1	2	3	4	5	6	

功能	包头1	包头2	长度	功能字	电机（±10	电机（±10	电机（±10	电机（±10	checksum	备注：用于测试电机或者电机驱动是否正常。正数为前进，负数为后退。
控制电机PWM速度，未使用编码器	0xFF	0xFC	0x07	0x10	XX	XX	XX	XX	XX	
下标	0	1	2	3	4	5	6	7	8	

功能	包头1	包头2	长度	功能字	小车类型	状态（0~7）	度（0~100	checksum	备注：小车类型：保留功能。 状态：=0停车，=1前进，=2后退，=3左平移，=4右平移，=5左旋转，=6右旋转，=7刹车停止。 速度：0~100，速度越大，速度越快。
小车状态控制	0xFF	0xFC	0x06	0x11	XX	XX	XX	XX	
下标	0	1	2	3	4	5	6	7	

功能	包头1	包头2	长度	功能字	小车类型	X轴速度（±1000）		Y轴速度（±1000）		Z轴速度（±5000）		checksum	备注：小车运动控制，X轴控制小车前进后退，Y轴速度控制小车左右平移，Z轴速度控制小车左右旋转。由于速度是小数，所以都放大了1000倍。
小车运动控制	0xFF	0xFC	0x0A	0x12	XX	低位	高位	低位	高位	低位	高位	XX	
下标	0	1	2	3	4	5	6	7	8	9	10	11	

功能	包头1	包头2	长度	功能字	P(0-10000)		I(0-10000)		D(0-10000)		永久保存	checksum	备注：由于PID都是小数，所以都放大了1000倍，实际范围为0-10。
小车速度PID调节	0xFF	0xFC	0x0A	0x13	低位	高位	低位	高位	低位	高位	0x00|0x5F	XX	
下标	0	1	2	3	4	5	6	7	8	9	10	11	

功能	包头1	包头2	长度	功能字	确认	checksum
清空flash数据	0xFF	0xFC	0x04	0xA0	0x5F	XX
下标	0	1	2	3	4	5

图 4-20 通信协议

（2）底层通信控制程序设计

FreeRTOS（Free Real-Time Operating System）是一个开源的实时操作系统，专门设计用于嵌入式系统和实时应用程序[15]。它提供了一套简单、可移植、可扩展的内核功能，帮助开发者管理任务调度、内存管理、中断处理、通信和同步等操作，以便在资源受限的嵌入式环境下实现可靠的实时任务调度和协作如图 4-21。本设计控制板采用 FreeRTOS 实现上位机（Jetson Orin Nano）和下位机（控制板）的通信链接和数据收发。

Algorithm 1: FreeRTOS Task Initialization

Input: System configuration flags
Output: Active real-time tasks
begin
  // Create tasks with priorities
  foreach *task* **in** *[("Speed", 512, 10, vTask_Speed), ("Control", 128, 9, vTask_Control), ("KEY", 128, 8, vTask_Key), ("Report", 512, 7, vTask_Auto_Report), ("App_Handle",128,4, vTask_App_Handle), ("MPU", 1024, 3, vTask_IMU)]* **do**
    xTaskCreate(task.func, task.name, task.stack, NULL, task.prio, NULL);
  end
  if *ENABLE_OLED* **then**
    xTaskCreate(vTask_OLED, "OLED", 512, NULL, 1, NULL);
  vTaskStartScheduler();
  ; // Never returns
end

Task Priorities

10 (Highest)	Speed Measurement
9	Control System
8	Key Input
7	Data Reporting
4	Application Logic
3	IMU Processing
1	OLED Display (Conditional)

图 4-21 FreeRTOS 任务列表

解析函数为 vTask_Control()，核心程序如图 4-22。

Algorithm 1: Compact Data Frame Parser

Input: buf[0..len-1]
begin
  sum $\leftarrow \sum_{i=2}^{len-2} buf[i]$; // Checksum calculation
  if *sum* $\neq$ *buf[len-1]* **then**
    return
  end
  switch *buf[3]* **do**
    case *AUTO_REPORT* **do**
      ▷ Process sensor data
    end
    case *MOTION_CTRL* **do**
      target $\leftarrow$ buf[4:7]; // Extract 4-byte float
    end
    case *PID_TUNE* **do**
      Kp $\leftarrow$ buf[4], Ki $\leftarrow$ buf[5], Kd $\leftarrow$ buf[6]
    end
    otherwise do
      Log unknown function code
    end
  end
end

图 4-22 解析函数

数据发送函数为 vTask_Auto_Report(),其中数据发送核心程序如图 4-23。

Algorithm 1: Optimized Telemetry Transmission

Input: Vx, Vy, Vz ∈ [-32768,32767]

begin

buf ← [HEAD, DEV_ID, LEN, FUNC_REPORT]; // Initialize header

// Pack 16-bit velocities

buf[4:9] ← (Vx&FF, Vx≫8, Vy&FF, Vy≫8, Vz&FF, Vz≫8);

buf[10] ← ReadBattery();

buf[-1] ← $\sum_{i=2}^{10} buf[i]$; // Checksum calculation

USART_Send(buf); // Atomic transmission

end

图 4-23 发送函数

程序主要函数与执行流程如图 4-24。

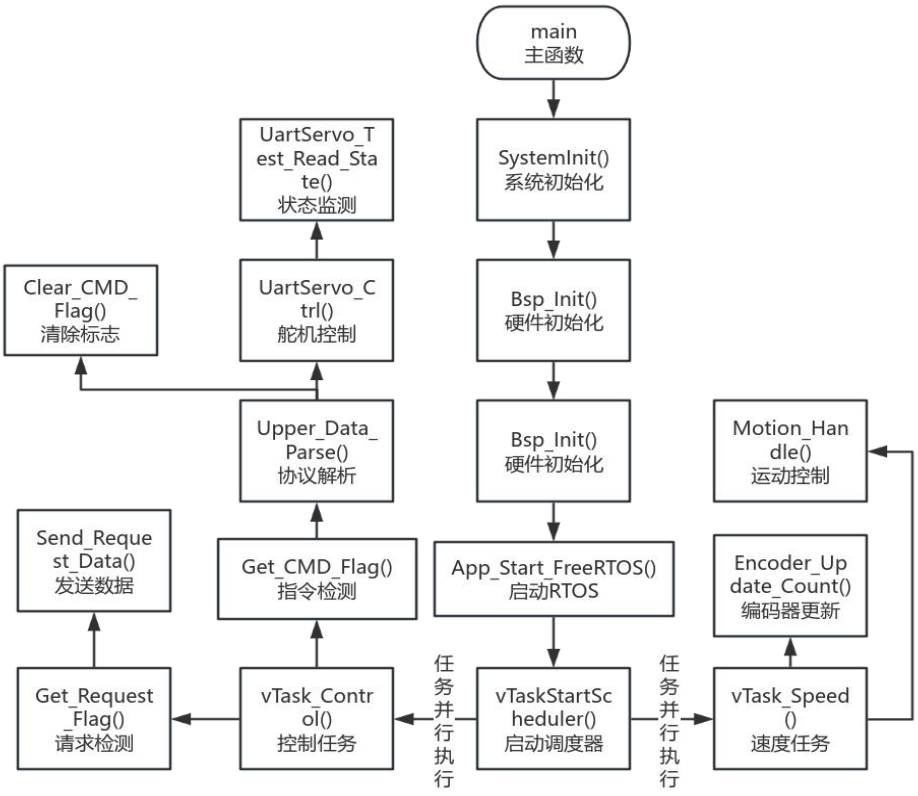

图 4-24 程序主要函数与执行流程

同样的根据协议上位机可以通过串口通信实现与小车底层硬件的通信和控制，

Python 程序相关接收与发送程序如数据帧的发送与接收遵循如表 4.1 格式。上位机数据解析 python 程序如图 4-25。向底层单片机请求数据如图 4-26。

表 4.1 数据帧的发送与接收

字段名称	描述	示例值
帧头（__HEAD)	数据帧的起始标识，固定值为 0xFF。	0xFF
设备 ID(__DEVICE_ID)	表示发送或接收数据的设备标识，固定值为 0xFC。	0xFC
数据长度 (ext_len)	表示整个数据包的长度（包括类型和数据部分）。	10
功能字 (FUNC_XXX)	指定具体的功能或操作，例如数据采集、控制指令等。	0x01
数据内容 (ext_data)	根据功能字的不同，包含不同的参数或数据内容。	采集数据值
校验值	数据帧的校验值，用于确保数据的完整性和正确性。	0xA5

Algorithm 1: Serial Frame Processor

```
begin
    while stream_active do
        if ReadByte() = HEAD then
            if ReadByte() = DEV_ID then
                len ← ReadByte();
                type ← ReadByte();
                payload ← [ReadByte() — i ∈ 0..(len − 3)];
                checksum ← ReadByte();
                if (sum([len, type] + sum(payload)) % 256 = checksum
                 then
                    Dispatch(type, payload);
                end
            end
        end
    end
end
```

图 4-25 上位机数据解析

Algorithm 1: Command Frame Construction

```
begin
    cmd ← [HEAD, DEV_ID, 0x05, FUNC_REQ, func&FF,
     param&FF];
    cmd ← cmd ∪ (sum(cmd) % 256);
    SerialSend(cmd);          // Atomic transmission with checksum
end
```

图 4-26 向底层请求数据

4.6 本章小结

本章节主要介绍了本设计的底层驱动程序开发和通信协议配置。通过图形化配置工具生成硬件的初始代码，并使用 Keil 完成功能模块的编程。使用 STM32 的定时器功能产生对直流电机进行控制并通过函数封装了控制逻辑。舵机驱动程序通过STM32定时器中断机制模拟PWM信号实现角度闭调节。在通信系统设计中，使用串口通信实现上位机（Jetson Orin Nano）与下位机（STM32）数据交互。使用 CH340H 芯片完成 USB-TTL 电平转换，另外，开发了 python 驱动库，通过协议实现底层硬件与上层算法的通信，为后续功能的开发奠定基础。

5 基于 ROS 的智能小车 LKS 功能实现

5.1 ROS 环境搭建和概念介绍

（1）ROS 诞生背景

随着技术的进步，机器人产业的逐渐呈多层次化发展。许多核心元器件都是由不同的厂家进行生产。对于一个复杂的系统来说一般都搭载这多个不同的传感器，那么如何将这些不同传感器整合起来，这就需要一个统一的软件平台来协调不同硬件之间的数据通信，ROS 在这样的背景下但是了。

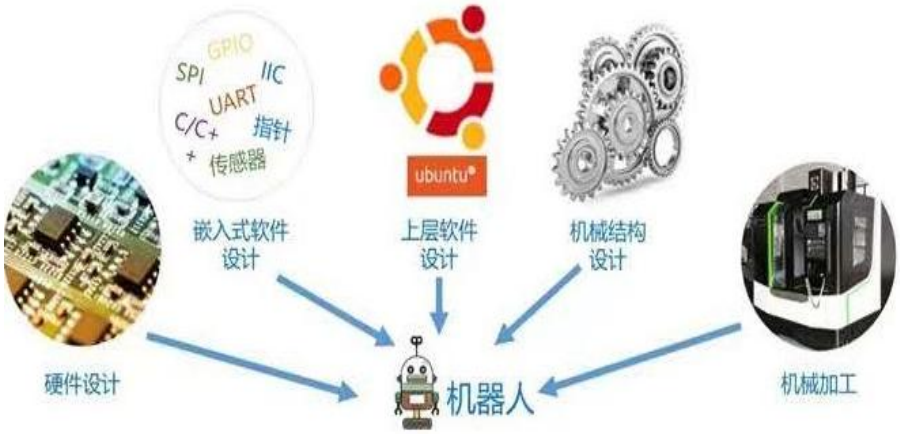

图 5-1 ROS 系统生态

（2）ROS 文件系统

ROS 文件系统其结构大致可以如图 5-2。

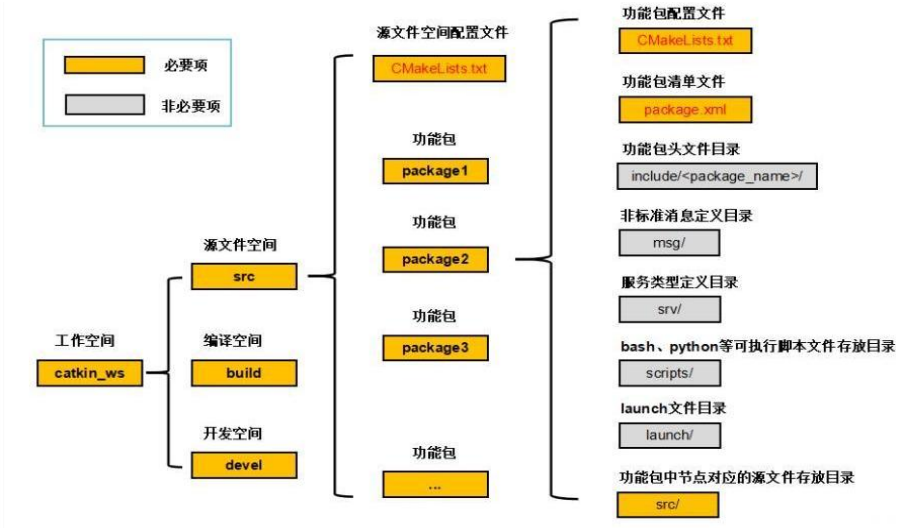

图 5-2 ROS 文件系统

（3）ROS 环境搭建

由于 ros 环境安装的大部分依赖文件都在国外的服务器，对于国内用户来说传统的安装方法经常因为网络问题安装失败，所以本设计使用了由鱼香 ROS 社区提供的一键安装命令，简化的安装过程。

鱼香ROS一键安装工具通过自动化脚本整合了ROS核心、依赖库及常用工具，大幅简化传统安装中繁琐的配置流程（如手动添加源、逐项解决依赖冲突），尤其针对国内用户优化下载源，突破网络限制实现高速部署。

打开 ubuntu 终端，输入：

```
wget http://fishros.com/install -O fishros && . fishros
```

然后会出现如图 5-3 的界面。

输入“1”一键安装->不更换源安装->选择 ubuntu 版本对应的 ros 版本->进行安装。

安装完成后新建一个终端，打开输入启动 rosmaster。

```
roscore
```

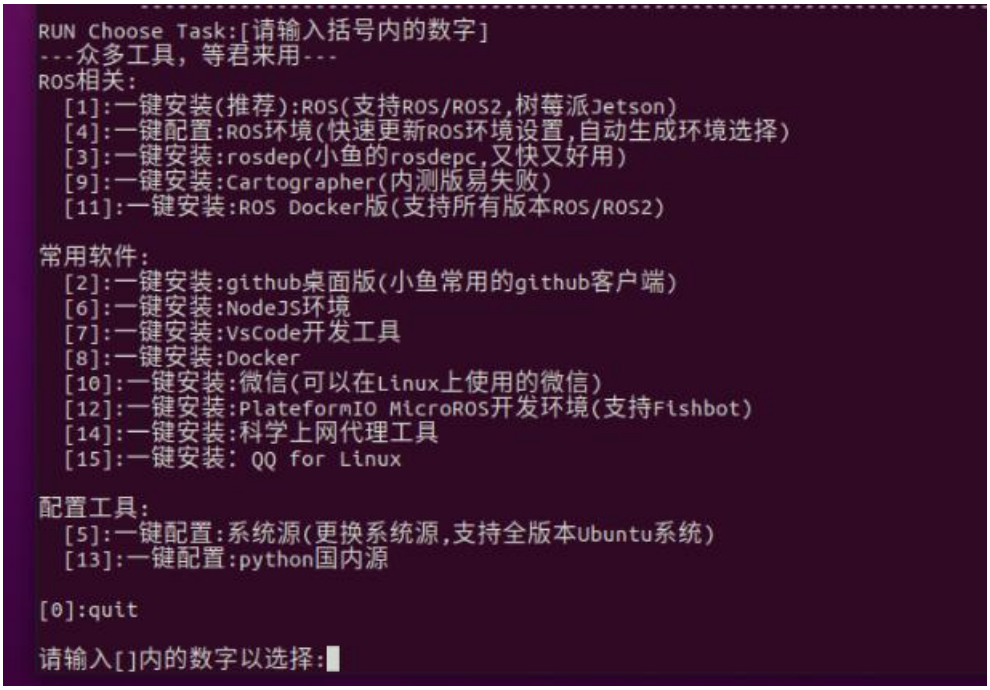

图 5-3 鱼香 ROS 一键安装命令

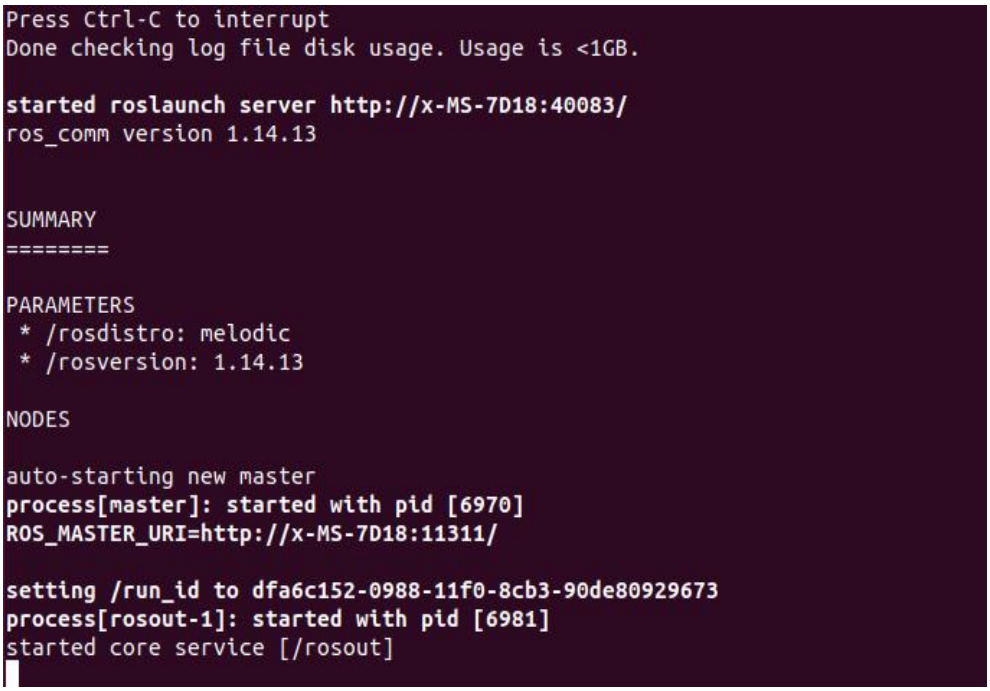

图 5-4 启动 rosmaster

终端完成后会显示[/rosout]图 5-4，再新建第二个终端，输入

```
rosrun turtlesim turtlesim_node
```

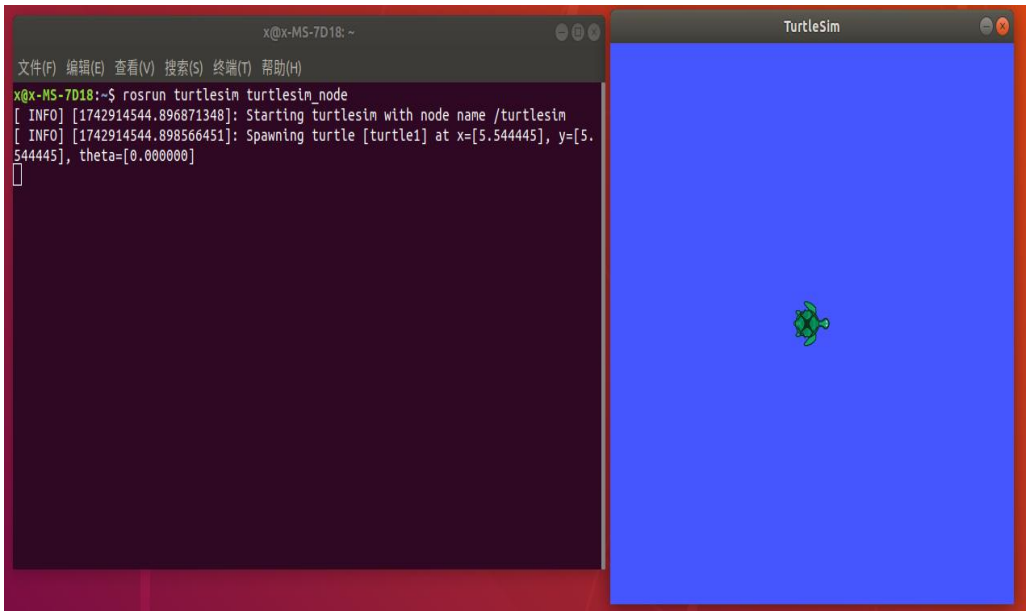

图 5-5 ROS 测试小海龟

它会打开一个屏幕的小乌龟在中间图 5-5，再建第三个终端，输入：

```
rosrun turtlesim  turtle_teleop_key
```

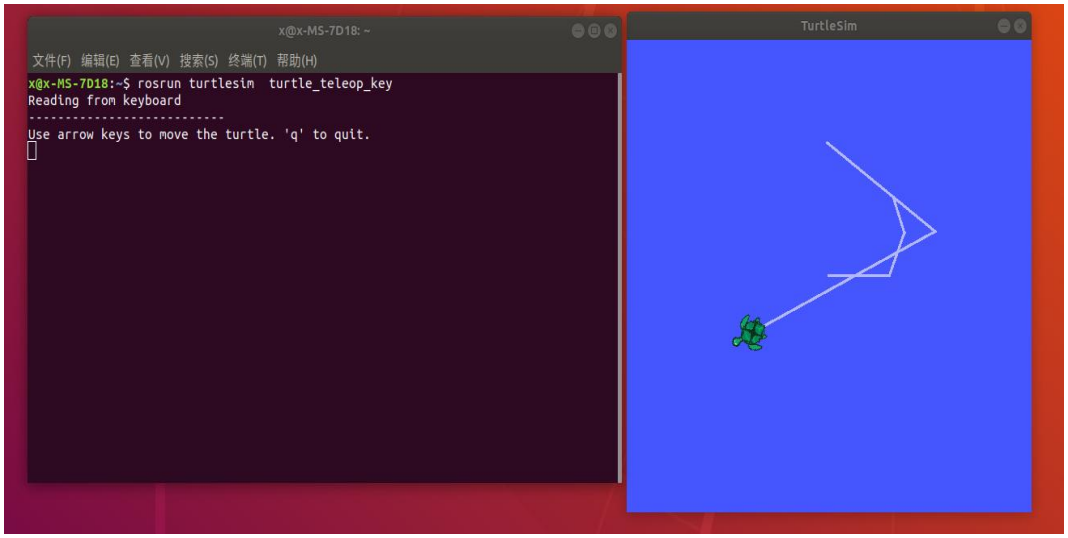

图 5-6 启动小海龟键盘控制节点

鼠标放在第三个终端，就可以通过按下键盘的上下左右键来对小海龟进行运动控制了图 5-6。到这里说明 ROS 安装成功了。

（4）LKS 功能包创建

创建 ros 空间

```
mkdir -p LKS_ws/src
cd LKS_ws
catkin_make
```

进入 src 创建 ros 包并添加依赖

```
cd src
catkin_create_pkg lks roscpp rospy std_msgs
```

上述命令，会在工作空间下生成一个名为 lks 的功能包，后续车道保持相关程序将在这个功能包中实现。

（5）节点功能与通信关系

通过编写个节点程序实现系统各功能实现，各节点和如表 5.1。

表 5.1 节点功能与通信关系

节点	功能	输入话题	输出话题
camera_node	摄像头数据采集节点，实时捕获车辆前	无	/camera/image_raw（原始图像）

	方道路的原始图像。		
image_processor_node	图像处理节点，接收原始图像进行车道线检测与偏差计算。	/camera/image_raw	/processed_image（处理后的图像） /lane_error（车道偏差值）
controller_node	控制决策节点，根据车道偏差值生成横向控制指令（舵机转向角度）。	/lane_error	控制指令（如/steering_angle）
visualization_node	可视化节点，显示处理后的图像及车道偏差信息，用于实时调试与监控。	/processed_image/lane_error	/visualization/image（可视化结果）

5.2 LKS 功能介绍

车道保持系统（Lane Keeping System，简称 LKS）是智能驾驶中的关键模块之一，它通过感知和控制技术实现车辆在行驶过程中自动保持在道路的正确车道上[16]。

LKS 原理和基本工作机制是基于先进的传感技术和实时数据处理算法的智能驾驶功能之一。通过摄像头获取车车道信息，然后利用计算机视觉算法识别关键信息。一旦系统检测到车辆偏离当前车道，LKS 模块即通过车辆的电控系统对车辆进行主动的横向控制，确保车辆重新回到正确的行驶轨道上。如图 5-7。

5.3 基于 OPENCV 的车道保持

5.3.1 车道线识别

本设计将使用传统的车道线检测方法实现车道保持功能利用摄像头获取道路的图像信息，然后通过 OpenCV 对图像进行分析和处理，识别出车道线。例如，通过颜色空间转换、阈值化、边缘检测等操作，将图像中的车道线从复杂的背景中提取出来，再利用霍夫变换等算法检测出车道线的直线或曲线模型[17]。

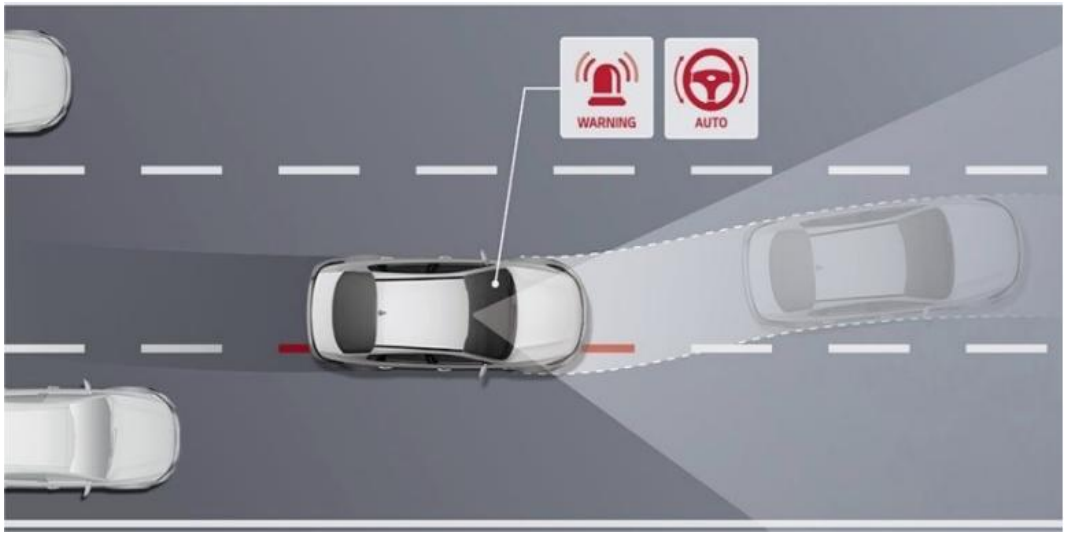

图 5-7 车道保持示意图

（1）Hough（霍夫）直线检测原理

我们看见直角坐标系中有两个点 A、B，它们构成了一条直线 y = kx + b。知道了斜率 k 和截距 b 就可以知道一条直线。将其放在笛卡尔空间中如图 5-8，可以知道当知道一个 x 时，就一定知道直线上对于的点。

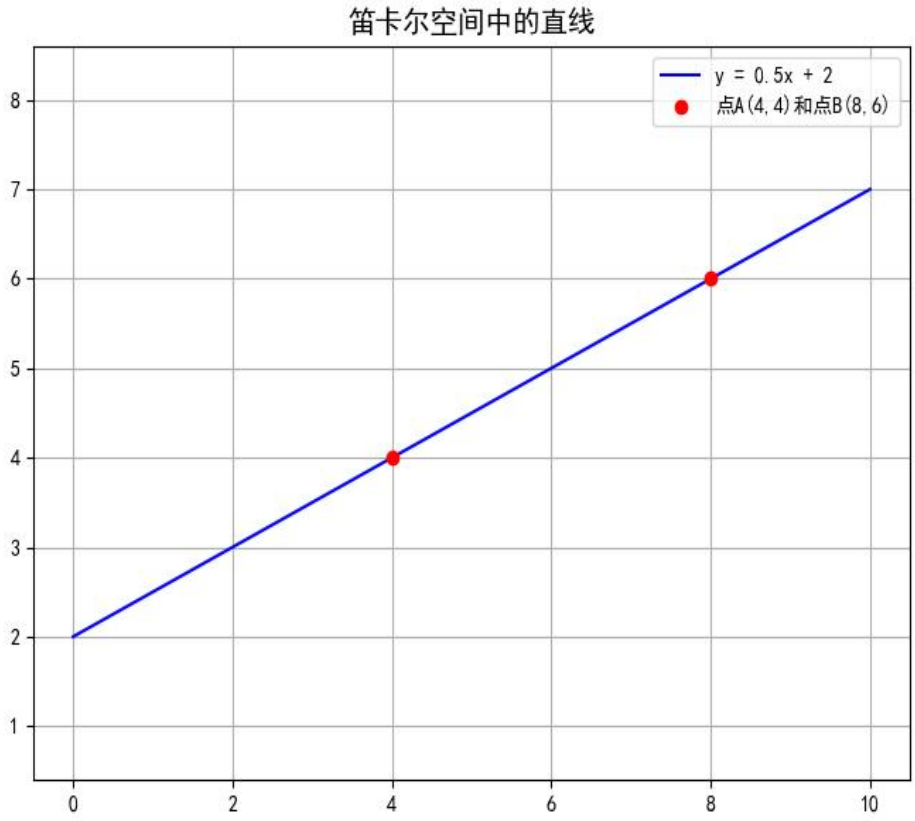

图 5-8 笛卡尔空间直线

然而，该模型存在两个问题：1.垂直直线问题：当直线垂直于 x 轴时，斜率 k 为无穷大。2.参数范围无限：k 和 b 的取值范围为 $(-\infty,+\infty)$，导致参数空间难以离散化。

为解决上述问题，霍夫变换采用极坐标参数化直线：

$$\rho = x\cos\theta + y\sin\theta \tag{5.1}$$

其中：

-ρ：直线到原点的垂直距离（$\rho \in [-D, D]$，D 为图像对角线长度）。

-θ：直线法线与 x 轴的夹角（$\theta \in [0, \pi)$）。

设直线 L 的法线方向为θ，原点到直线的垂直距离为ρ。对直线上任意一点(x, y)，其到直线 L 的投影应满足几何关系（如图 59 所示）：

$$\rho = x\cos\theta + y\sin\theta \tag{5.2}$$

推导过程：

直线法线方向单位向量为 $n = (\cos\theta, \sin\theta)$。原点 O 到直线的向量投影为$\rho n$，即直线方程为：

$$r \cdot n = \rho \tag{5.3}$$

展开后得到：

$$x\cos\theta + y\sin\theta = \rho \tag{5.4}$$

图像空间到参数空间的映射关系，图像空间中的一个点 (x_i, y_i) 对应参数空间中的一条正弦曲线：

$$\rho = x_i\cos\theta + y_i\sin\theta \quad (\theta \in [0, \pi)) \tag{5.5}$$

图像空间中的一条直线对应参数空间中的一个点(ρ_0, θ_0)。

证明：

设图像空间中存在一条直线 $L: \rho_0 = x\cos\theta_0 + y\sin\theta_0$，其上的所有点 $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$均满足方程：

$$\rho_0 = x_i\cos\theta_0 + y_i\sin\theta_0 \quad (i = 1,2,\ldots,n) \tag{5.6}$$

每个点在参数空间中生成的正弦曲线均通过点(ρ_0, θ_0)，因此该点为所有曲线的交点。

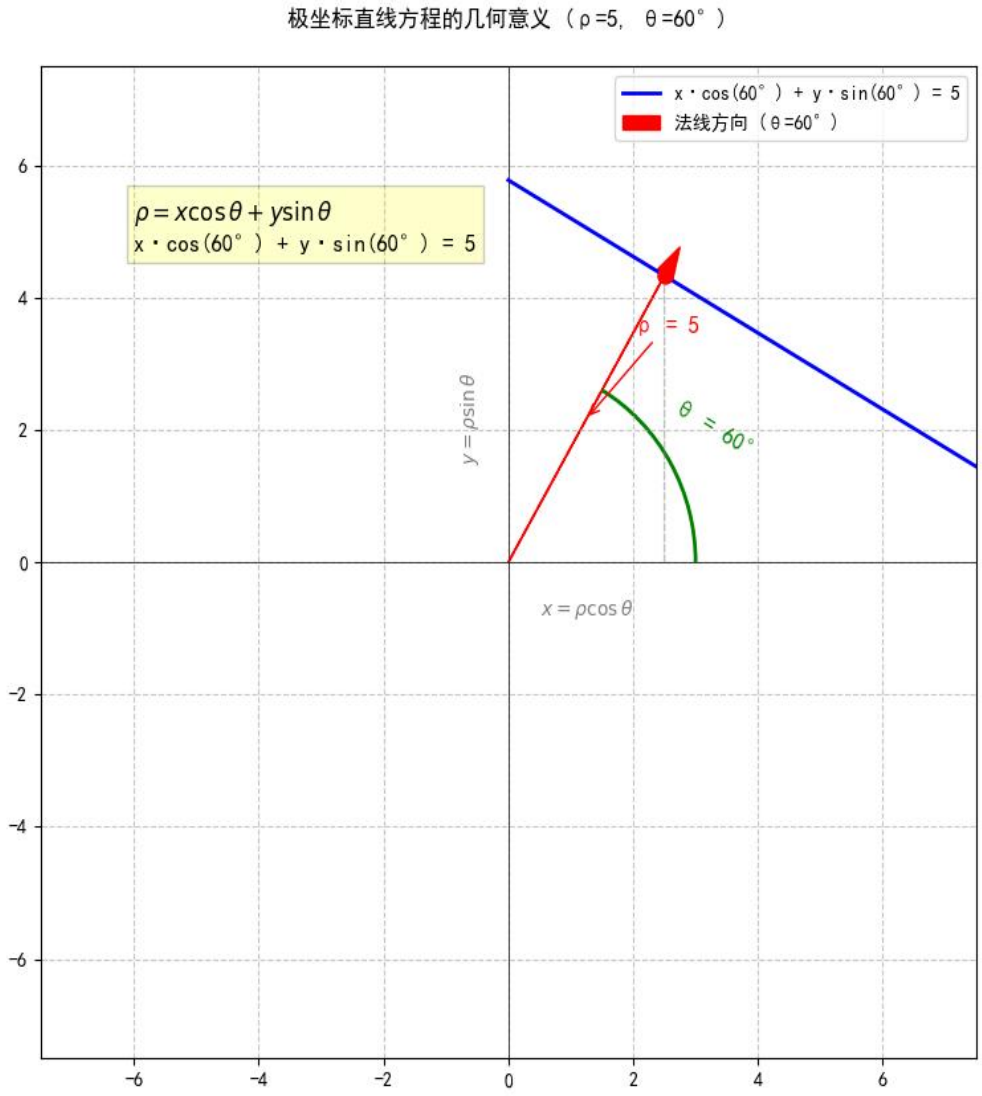

图 5-9 极坐标直线方程的几何意义

通过图 5-9 所示的笛卡尔空间与霍夫空间映射实验，验证了以下结论：

点 A(4,4)和 B(8,6) 在笛卡尔空间中确定的直线 $y = 0.5x + 2$ 对应的霍夫空间交点为 $(\rho = 5.66, \theta = 135°)$。将参数 $\rho = 5.66$ 和 $\theta = 135°$ 代入极坐标方程，可还原为笛卡尔空间中的直线方程：

$$x\cos(135°) + y\sin(135°) = 5.66 \Longrightarrow y = -x + 8 \qquad (5.7)$$

该直线为原直线 $y = 0.5x + 2$ 的法线方向，验证了参数计算的正确性。

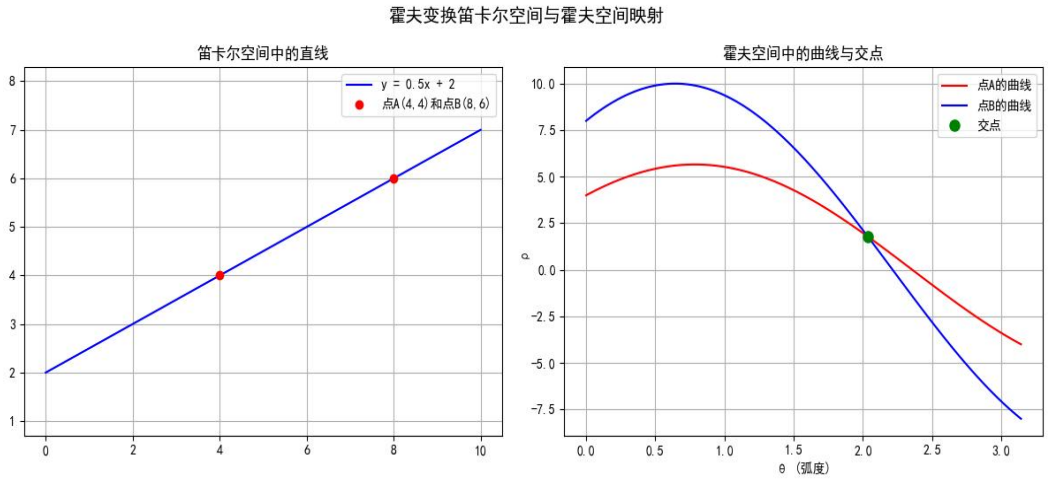

图 5-10 笛卡尔空间与霍夫空

（2）车道线识别算法实现

车道线的信息主要包括实车道线和虚车道线，在程序中使用霍夫变换的最小线段长度参数 min_line_length 和最大间隙参数 max_line_gap，可以有效对实线车道和虚线车道进行动态调节与分类。高的 min_line_length 阈值可过滤路面裂纹和阴影噪声，而极小的 max_line_gap 容忍值能确保即使存在轻微边缘断裂也能被识别为完整线段。这种方法可以使实线检测在复杂路况下依然可以保持较高的识别准确度，同时调整角度滤波参数可以有效排除斑马线等干扰线条。

对于虚线车道线检测，可以通过动态调节霍夫参数。降低可检测的最小线段长度，放宽最大间隙阈值允许跨越虚线间隔。在配合自适应 Canny 阈值，可以在保留微弱虚线边缘的同时降低纹理干扰。在适应弯道场景下的虚线弧度变化效果比较号。

主流程伪代码，程序执行流程如图 5-11，车道线检测程序流程如图 5-12。

5.3.2 车道线拟合

最小二乘法在工程中得到了广泛的应用，应用最多的是直线拟合（线性拟合），本设计将采用此方法进行车道线拟合[18]。

（1）原理介绍

给定一组数据x_i、$y_i(i=1,2,\cdots,N)$，假定它们具有线性关系，即可以用 $y=kx+b$ 的方式进行拟合，我们的任务是找到最优参数 k 和 b，使得直线 $y=kx+b$ 能最大程度接近，这个时候最小二乘法就派上用场了。

Algorithm 1: Visual Processing Workflow

begin

frame ← ImageAcquire(src);

edges ← frame ▷ GrayConvert() ▷ GaussianBlur(σ) ▷ CannyEdge(low, high);

roi ← edges ∩ PolyMask(vertices);

lines ← HoughTransform(roi, ρ, θ, thresh, min_len, max_gap);

valid ← {l ∈ lines — $\theta_{min} < \angle(l) < \theta_{max}$};

Visualize(frame, valid);

end

图 5-11 程序执行流程图

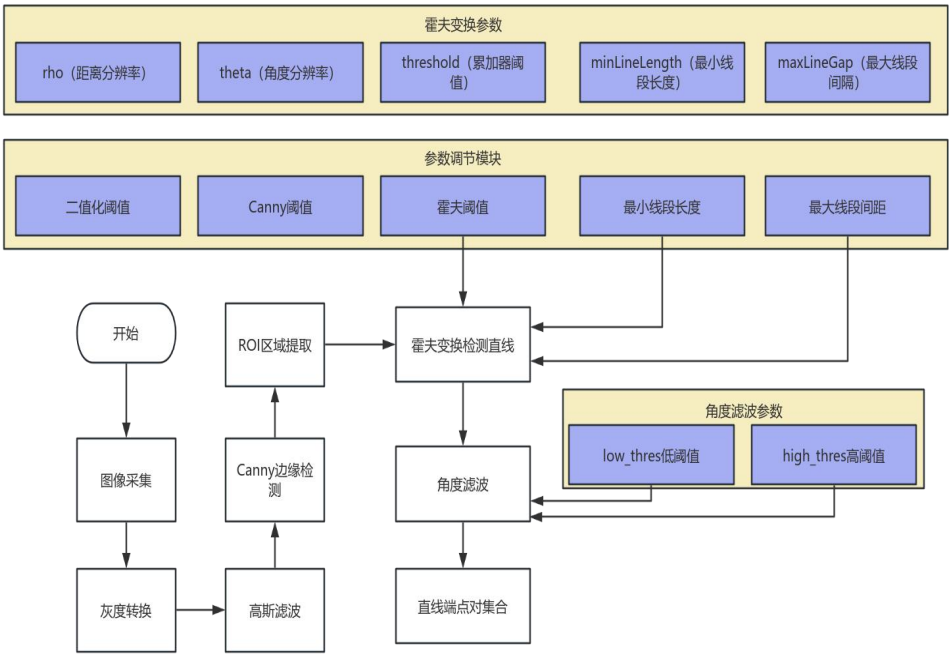

图 5-12 车道线检测程序流程图

为了描述 最小二乘拟合 参数 k 和 b 的效果，定义一个函数，使用残差平方和来定义：

$$f = \sum(y_i - kx_i - b)^2 \tag{5.7}$$

当拟合效果最好时，残差平方和最小，残差平方和最小是显然的，问题转化为求解 f 为极值点时，k 和 b 为多少？

为了找到极值点，我们需要对 k 和 b 分别求偏导数并使其为 0：

$$\begin{aligned} \frac{\partial f}{\partial k} &= -2\sum(y_i - kx_i - b)x_i \\ &= -2\left(\sum y_i x_i - k\sum x_i^2 - b\sum x_i\right) = 0 \end{aligned} \tag{5.8}$$

$$\begin{aligned}\frac{\partial f}{\partial b} &= -2\sum(y_i - kx_i - b)\\ &= -2(\sum y_i - k\sum x_i - Nb) = 0\end{aligned} \tag{5.9}$$

由式(5.9)括号内为 0，则有 b 的表达式：

$$\begin{aligned}b &= \frac{\sum y_i}{N} - k\frac{\sum x_i}{N}\\ &= \bar{y} - k\bar{x}\end{aligned} \tag{5.10}$$

将式(5.10)代入到式(5.8)中得到式(5.11)：

$$\sum y_i x_i - k\sum x_i^2 - (\bar{y} - k\bar{x})\sum x_i = 0 \tag{5.11}$$

整理式(5.11)我们可以得到 k 的表达式：

$$\begin{aligned}k &= \frac{\sum y_i x_i - N\bar{x}\bar{y}}{\sum x_i^2 - N\bar{x}^2}\\ &= \frac{\sum y_i x_i - \frac{(\sum x_i)(\sum y_i)}{N}}{\sum x_i^2 - \frac{(\sum x_i)^2}{N}}\end{aligned} \tag{5.12}$$

所以最终 k 和 b 的表达式可以表示为：

$$\begin{aligned}k &= \frac{\sum x_i y_i - N\bar{x}\bar{y}}{\sum x_i^2 - N\bar{x}^2}\\ b &= \bar{y} - k\bar{x}\end{aligned} \tag{5.13}$$

线性代数表达，寻找最佳的 k 和 b：

$$\begin{cases} y_1 = kx_1 + b \\ y_2 = kx_2 + b \\ \vdots \\ y_N = kx_N + b \end{cases} \tag{5.14}$$

写成矩阵的形式：

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix} \begin{bmatrix} k \\ b \end{bmatrix} \tag{5.15}$$

令(5.16)

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, K = \begin{bmatrix} k \\ b \end{bmatrix}, X = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix} \tag{5.16}$$

我们就可以把它写成矩阵形式：

$$Y = XK \tag{5.17}$$

我们要解 K，这时候 X 如果是方阵那么自然的就可以左右两边同时乘 X 的

逆，但是 X 不是方阵（行数大于列数），我们可以左右同乘以 X 的转置，转换为方阵再求解（当然能求逆的前提是行列式不为零），具体如下，求得的逆叫做伪逆：

$$\begin{aligned} &\Rightarrow Y = XK \\ &\Rightarrow X^T Y = X^T XK \\ &\Rightarrow (X^T X)^{-1} X^T Y = (X^T X)^{-1} X^T XK \end{aligned} \tag{5.18}$$

所以解得 K 的值：

$$K = (X^T X)^{-1} X^T Y \tag{5.19}$$

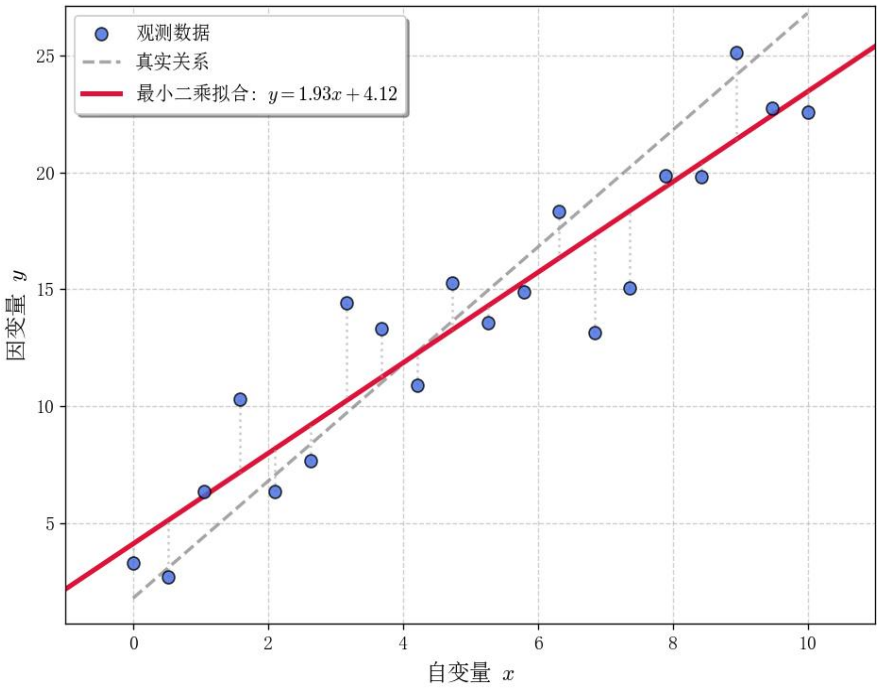

图 5-13 最小二乘拟合示意图

（2）车道线拟合与偏差计算算法实现

在程序执行程中，先对检测到的线段进行分类，斜率为正的为右车道线，为负归为左车道线，分别存储它们的坐标、斜率和截距信息。对于左右两侧车道线，通过寻找最高点来确定线段端点，利用中位数计算平均斜率和截距，来实现对车道线的拟合。并在图像上绘制绿色线段标记检测到的车道线

在偏差计算时本设计采用了自适应的偏差计算策略。当同时检测到左右车道线时，通过计算两条车道线的中点位置与图像中心的偏差来确定转向角度；如果只检测到单侧车道线，则通过该车道线的角度信息来计算偏差。

程序伪代码如图 5-14。

Algorithm 1: Lane Deviation Calculator

begin

(left, right) ← Partition(lines, λl: slope(l) 0);

if $left \neq \emptyset \wedge right \neq \emptyset$ **then**

$p_{mid} \leftarrow \frac{1}{2}(left_{base} + right_{base})$;

error ← $\|p_{mid} - p_{center}\|$;

else if $left \neq \emptyset$ **then**

error ← $-\arctan(mean\_slope(left))$;

else if $right \neq \emptyset$ **then**

error ← $\arctan(mean\_slope(right))$;

return error;

图 5-14 车道线拟合程序伪代码

控制策略

函数 control_servo(error)：

限幅 error 在[-40, 40]范围内

计算舵机角度

发送控制指令

程序执行流程图如图 5-15。

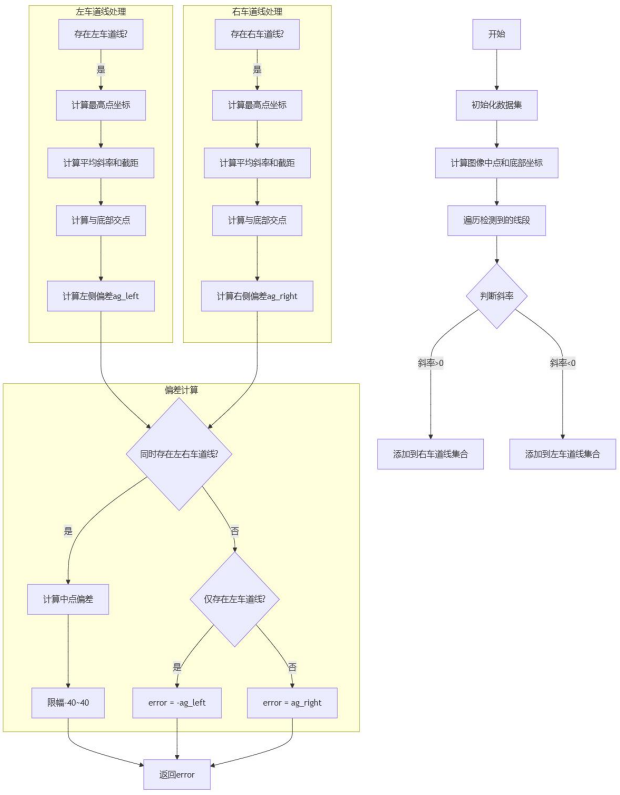

图 5-15 程序执行流程图

5.4 测试 LKS 功能

5.4.1 实验道路设计

道路设计主要设计了两种道路不同光照来进行实验，分别为直道和弯道，根据车辆宽度将道路宽度设计为 50cm，如图 5-16 所示为两种道路。

5.4.2 测试结果分析

根据相机检测到的左右车道线的结果，通过计算车道中心和图像中心的像素偏差来纠正转向角度。如图 5-17 所示，在行驶过程中检测到了正常光照条件下直道车道线。如图 5-18 所示，在行驶过程中也可以检测到正常弯道车道线。图 5-17 和图 5-18 显示，车道线已成功检测到。图 5-19 和图 5-20 为低光照下的两种工况，同样成功的检测出车道线，并输出车道线与偏差。图 5-21 和图 5-22，在高曝光下的两种工况，同样成功的检测出车道线，并输出车道线与车辆的偏差。

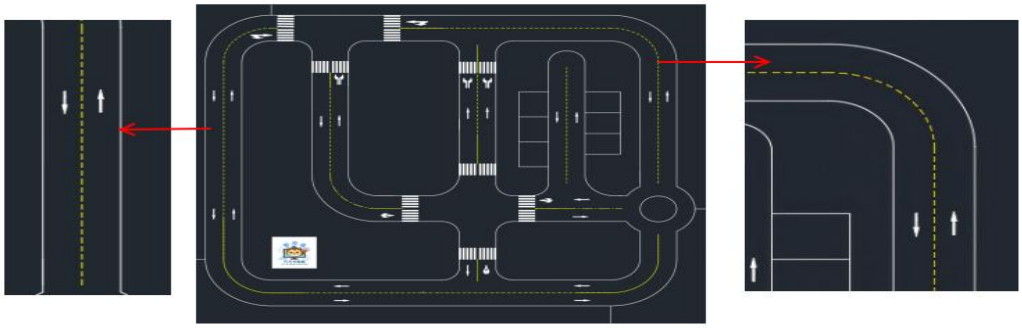

图 5-16 直道和弯道示意图

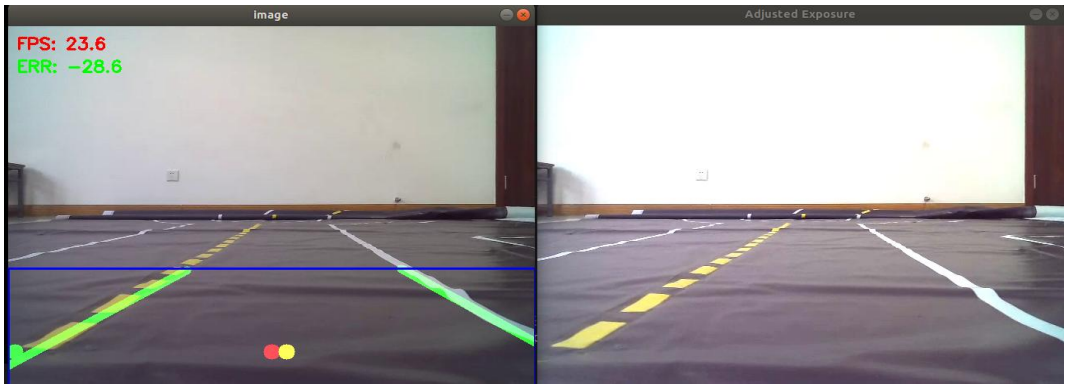

图 5-17 正常直线行驶检测结果

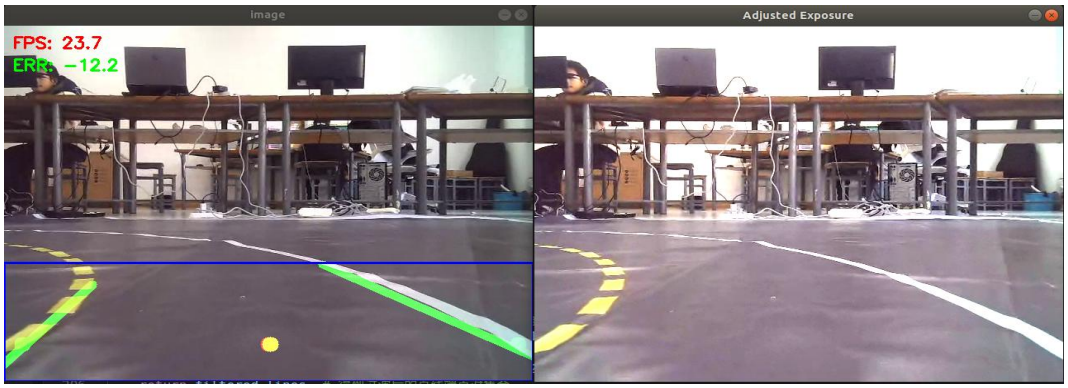

图 5-18 正常弯道行驶检测结果

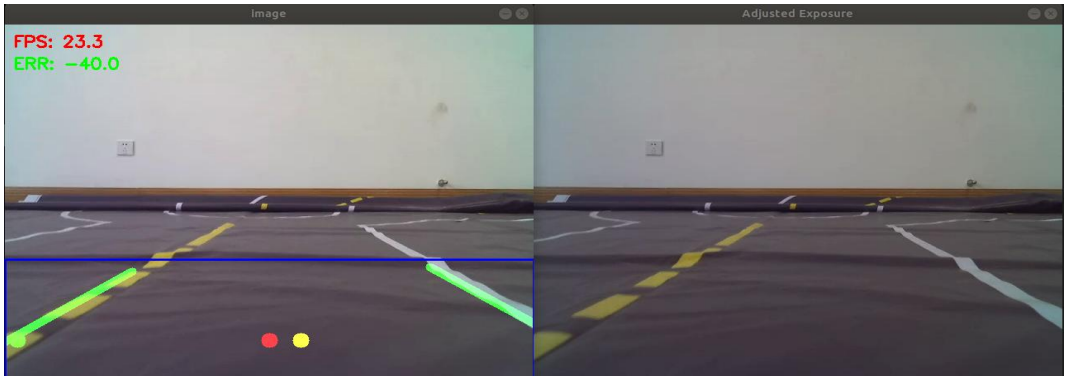

图 5-19 低光照直线行驶检测结果

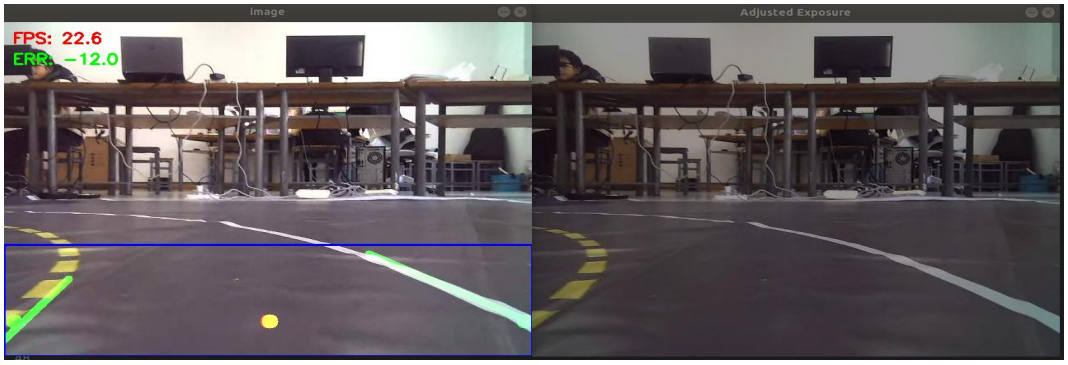

图 5-20 低光照弯道行驶检测结果

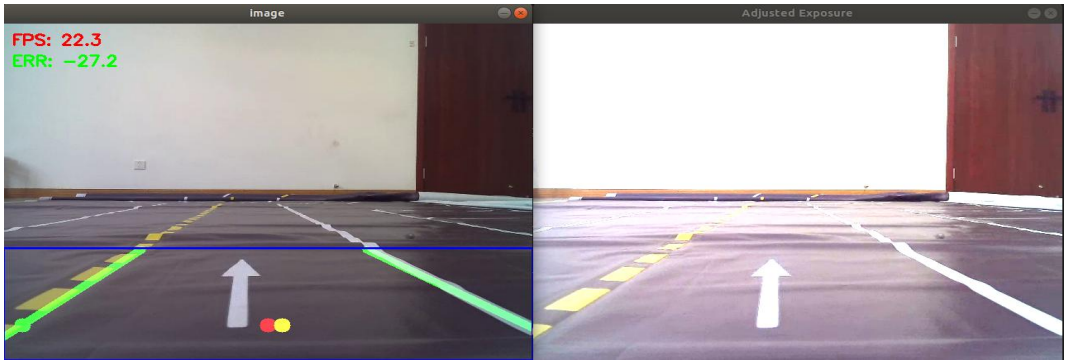

图 5-21 高曝光直道行驶检测结果

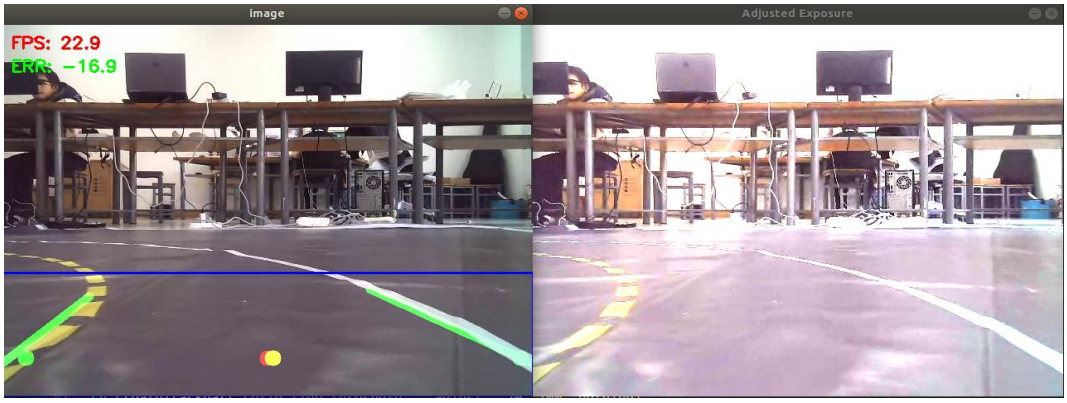

图 5-22 高曝光弯道行驶检测结果

动态测试不同光照条件车道线识别算法对光照的适应性。启动小车 LKS 功能，动态调节曝光值，记录相关车道线检测质量数据，分析数据生成如图 5-23 时间序列分析图、图 5-24 相关性分析图和图 5-25 滑动窗口分析图。

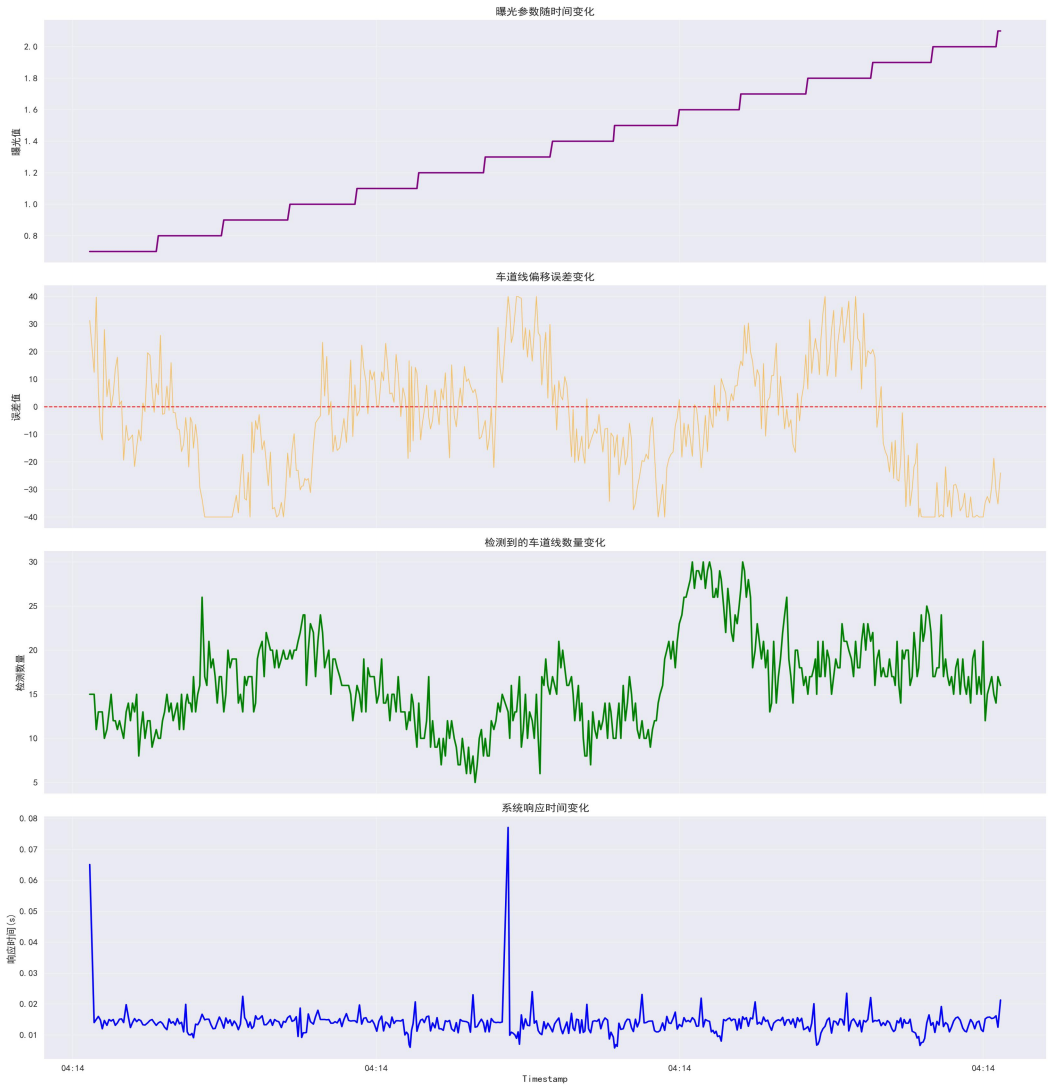

图 5-23 时间序列分析图

由时间序列分析图分析，当光度增强时车道线中心偏移误差（单位：像素）的误差值曲线与曝光值曲线的相位关系，高曝光时伴随误差增大。但在正负 40 标准范围内，符合设计标准。由滑动窗口分析图可知，橙色区域为误差值的标准差范围，反映控制稳定性。红色曲线为 5 分钟窗口内的平均误差，识别长期趋势。绿色曲线为检测数量中位数，评估系统可靠性 。蓝色虚线为最大响应时间，定位性能瓶颈。由数据可以看出本设计的 LKS 功能在复杂光线条件下仍能保持较高的识别精度和适应力。

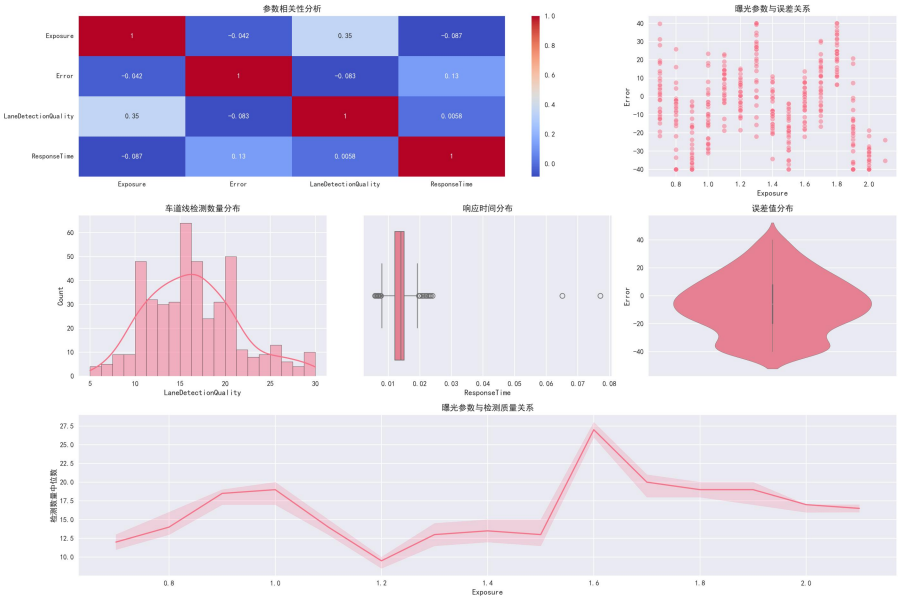

图 5-24 相关性分析图

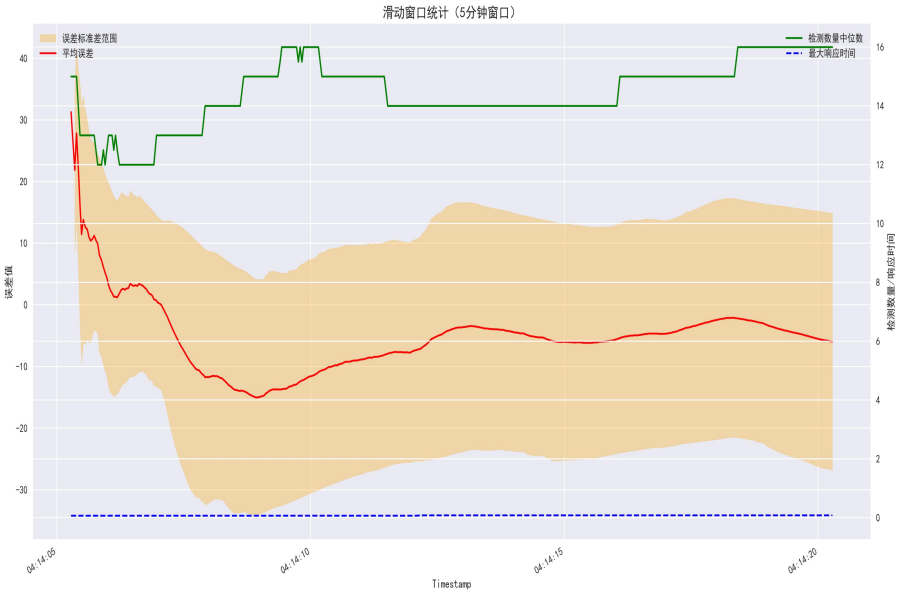

图 5-25 滑动窗口分析图

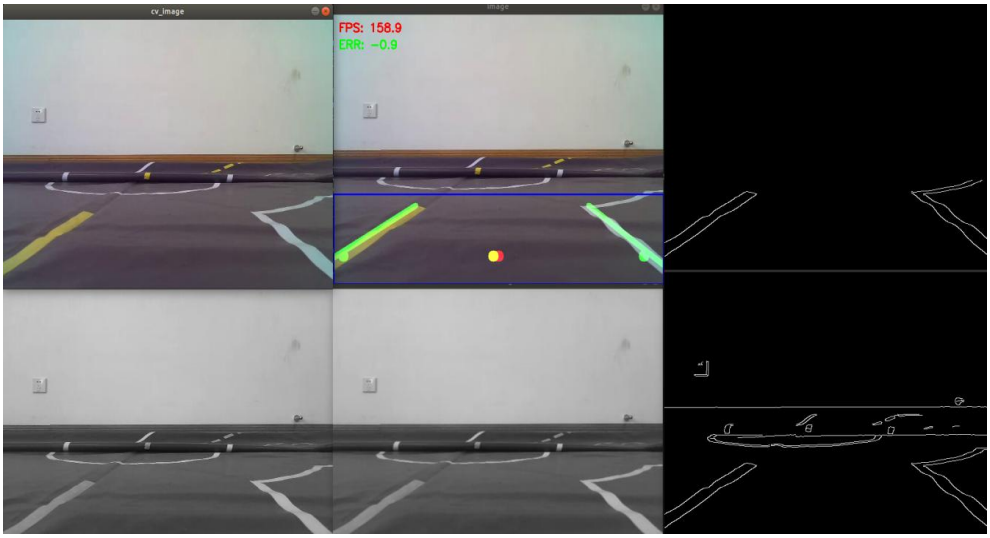

图 5-26 车道线识别可视化

5.5 本章小结

本章节主要绍了 ROS 的诞生背景和它的优势，阐述了本设计的车道保持功能。利用摄像头识别车道图像，通过算法技术提取车道线。针对实线和虚线的不同特点，采取不同的霍夫参数设置检测效果。在车道线拟合时，采用最小二乘法对检测到的线段进行分类和拟合，计算出左右车道线的中点偏差来修正转向角度。

在最后，通过直线和弯道两种工况不同光照条件的实验，验证了系统的有效性。实验结果表明，在不同道路条件下，小车均能成功检测车道线并保持在车道内行驶，证明了依托本设计所设计的 LKS 功能的稳定性和适应性。

6 全文总结与展望

6.1 总结

本设计为设计一种较低成本、高灵活性的智能网联汽车技术验证平台。通过设计，成功实现了智能网联小车的硬件系统设计、软件系统开发以及 LKS 功能开发。

底层控制以 STM32F103RCT6 微控制器为控制核心，Jetson Orin Nano 作为上层计算单元，设计了稳固高效的控制架构。选用激光雷达、深度相机、鱼眼镜头等传感器，实现了小车对周围环境的感知。并对底层控制板主电路进行了设计。对底层各驱动部分进行了程序设计，利用 ROS 的分布式通信与硬件抽象，实现了底层控制与上层计算平台决策算法的高效协调。使用本设计实现了车道保持（LKS）高级驾驶辅助功能。

通过直线和弯道两种工况不同光照条件下的实验，验证了基于本设计开发的车道保持系统（LKS）。实验结果表明，小车在不同道路条件下均能成功检测车道线并保持在车道内行驶，证明了所设计的 LKS 功能的有效性和适应性。

6.2 存在问题与不足

虽然本设计虽取得了一些的成果，但在设计过程中也发现了一些问题和不足之处，比如在硬件方面，部分传感器的数据传输方式有待提升，比如说本设计有多个 usb 相机，硬件接口不足同时 usb 数据带宽较低。如后面开发 360 环视功能时可以会出现较高延迟或卡顿。在 LKS 功能设计中使用了较为传统的方法进行车道线识别，没有发挥出 Jetson Orin Nano 计算能力。

6.3 展望

针对上述问题，对未来的本设计提出以下展望：

对应 usb 接口不足和数据带宽较低的问题可以使用以太网进行数据传输。进一步开发小车的功能，如实现自动紧急刹车与自主停车和 360 环视等功能，实现车与车，车与路基设施间的通讯。

致 谢

行文致此，落笔为终。这一年我 23 岁，告别 18 岁的青春，在我黄金的青春时代和自己的本科阶段告别，是一个阶段的告别，又是新一阶段的启程。回顾我的本科阶段，是青春，是友情，激励着我向上向前，目光所及皆为回忆。

这段故事开始于 2021 年秋，终于 2025 年夏，一路披荆斩棘，却顾所来径，苍苍横翠微。再回头，轻舟已过万重山。总以为四年很长，现在却已经到站；以为来日方长，但却时光匆匆。四年的点滴化为了宝贵的回忆，纵有不舍，更多是感怀。

草春晖，难以回报。四年时间，异地求学，一路见证了我的成长。他们很平凡，却让我踩在他们的手上，站上他们的肩，见识更加广阔的世界。他们托起我的全部，永远在我的背后支持我的每个决定，一直是我坚强的后盾。家人一直是我勇往直前、无所畏惧的力量源泉，感恩有你们。

桃李不言，下自成蹊。衷心感激老师们的细致入微的教诲，您们的鼓励与支持对我至关重要，在您们的耐心引导下，我深刻领悟到了严谨与规范的学术价值。对您们不倦的指导与宝贵意见，我满怀敬意与感激。感谢每位老师，在我人生道路上的指引，让那个迷茫的少年再次坚定了自己的向往。承蒙教诲，心存感激，虽未至臻，终有所悟。

人生海海，幸得遇见。感谢我遇见的朋友，你们的包容与理解让我不断完善自己，接纳我的缺点，给予我鼓励和帮助。感谢朝夕相处的兄弟们，四年的陪伴带来了无数欢乐与温情；也感谢在我困境中伸出援手的朋友，每一次交谈都给予我无限关心与建议。还有昔日旧友，虽未再见，友情不减。愿我们在各自的领域里闪闪发光，成为更好的自己，砥砺前行，不被现实磨平棱角，将理想照进现实。

道阻且长，行则将至。最后感谢平凡的自己，虽不甘于平庸，纵有诸多不如意，依旧保持向上的动力与勇气。感谢自己变得勇敢自信，从“我不行”变成“一定可以”。18 岁的少年，书桌上没有荒废的青春，无数日夜执笔伏案，感谢你的坚持。山不见你，你自去见山。愿未来的你永怀赤诚，坚定不移地走下去，去见更广阔的世界，去听更遥远的声音。千言万语，终究词不达意。感谢遇见，感

谢经历。于我而言，所有经历皆是馈赠，所有相遇皆是幸运。山水一程，终有一别，再见。母校校训铭记于心，落笔致此，但人生不止于此，请继续书写未来新的篇章。

参 考 文 献

[1] 王硕,刘洋.新能源汽车技术专业《智能网联汽车技术》课程教学探究[J].汽车与配件,2025,(06):68-70.

[2] 中研普华产业研究院《2025-2030 年中国智能汽车(智能网联汽车)行业深度调研及投资前景预测报告》

[3] 王飞飞,陈炫萧,金浩.高速电动机控制系统反馈信号调理电路实验方案设计[J].实验室研究与探索,2025。

[4] 张栋,张潇云.评估不同激光 SLAM 算法在真实环境下的定位性能[J].激光与红外,2025。

[5] 田振芳.汽车智能网联系统中自动寻车技术的应用研究[J].内燃机与配件。

[6] 朱冰,范天昕,赵文博,等.自动驾驶汽车连续测试场景复杂度评估方法[J/OL].吉林大学学报(工学版),1-12[2025-03-29].

[7] 王源隆,赵万忠.智能网联汽车多学科融合协同创新人才培养模式研究[J].工业和信息化教育,2024。

[8] 巴宝莲.3D 打印逆向工程技术在产品开发与设计中的应用研究[J].造纸装备及材料,2024,53(12):81-83.

[9] 关恬恬,徐世伟,吴卓昆,等.基于深度学习的轻量级车道线检测算法[J].光电技术应用,2024,39(06):54-62.

[10] Živković,Aleksandar.Development of Autonomous Driving using ROS.University of Oulu, Faculty of Information Technology and Electrical Engineering,M3S,1 June 2018.

[11] 李一铎.基于多传感器融合的校园无人接驳车自主导航系统研究[D].浙江科技大学,2024。

[12] 薛硕硕.自动驾驶车辆局部避障轨迹规划及跟踪控制方法研究，长安大学,2024。

[13] 梁家硕.基于深度学习的自动泊车车位线检测[D].电子科技大学,2024。

[14] 童佳乐.基于改进实例分割的煤矿电机车障碍物检测技术研究[D].安徽理工大学,2023。

[15] 陈锦.顶杆输送带式自动取苗系统设计与试验[D].中国农业机械化科学研究院,2023。

[16] 郑星.FreeRTOS 在 ARM 平台的 SMP 系统方案研究与实现[D].北京邮电大学,2021。

[17] 康渴楠.基于滑动窗多项式拟合的车道检测研究[J].汽车科技,2019。