天津職業技術師範大學

Tianjin University of Technology and Education

硕 士 学 位 论 文

自动驾驶车辆车载控制器开发

Development of On-board Controllers for Autonomous Vehicles

作者姓名： 王曰浩

导　　师： 张军 教授

企业导师： 王金柱 高级工程师

类别/领域： 工程

分类号：　　学校代码：10066
密　级：　　学　　号：0422201029

自动驾驶车辆车载控制器开发
Development of On-board Controllers for Autonomous Vehicles

作 者 姓 名： 王曰浩
导师（职称）： 张军 教授
专 业 类 别： 工程硕士
专 业 领 域：
年　　级： 2020 级
提交论文日期： 2022 年 12 月
学位授予单位： 天津职业技术师范大学

摘要

当今社会信息技术的发展非常迅速，对车辆的研究也越来越深入，现代社会自动驾驶变得普及，同时也提高了对 VCU（车载控制器）的要求。VCU 通过采集传感器的信息和环境信息来制定行驶策略，制定车辆的能量管理策略，采集的车载传感器信息进行分析和处理并通过 CAN 总线向各个单元发送控制命令，保证车辆的安全行驶的同时达到更省电的效果。

本文以车载 VCU 为研究对象，对国内外自动驾驶与车载控制器的研究现状进行了分析和研究，主要针对车载传感器的数据传输和控制信息通信，车辆运行能量优化策略开展研究，并构建了车载 VCU 实验平台并基于 ROS 小车开展了仿真实验。主要研究内容包括：

针对车载传感器数据的特点和控制信息的要求，采用 CAN 总线进行数据传输。本文在分析了 CAN 总线的通信协议的基础上，分析了温度传感器、加速度传感器、霍尔传感器等车载传感器数据信息以及数据与车辆的行驶状态的关系，根据 CAN 的报文帧格式和 SAE J1939 协议制定了多传感器和控制节点的数据通信传输协议，实现了车载信息的可靠传输。

研究了车辆的能量优化策略问题。首先在分析了新欧洲驾驶循环工况 NEDC 和世界轻型汽车测试循环工况 WLTC 的基础上，构建了基于当地国内城市环境的车辆的行驶工况，构建时根据行驶过程中的行驶片段对工况进行提取，随后按照速度范围将提取的片段分类，使用 K-means 聚类算法进行行驶片段的优化组合，形成最终的车辆行驶工况仿真数据和工况曲线图。在此基础上，通过动态规划算法对车辆行驶工况进行能量优化仿真分析，分析电池 SOC（电池当前剩余电量）数值来判断车辆在同样路程和时间下的用电情况。并通过对比新欧洲驾驶循环工况 NEDC 和世界轻型汽车测试循环工况 WLTC 应用前后 SOC 的数值，通过仿真分析动态优化分析，证明了本文车辆的能量优化策略是可行的。

开展车辆 VCU 的软硬件设计工作，构建了 VCU 控制平台，选用树莓派做为 VCU 主控芯片、STM32 单片机做为传感器和控制部件的从控芯片，进行了树莓派外围电路的设计，完成了 VCU 主控平台的设计和搭建。应用激光雷达和其它传感器进行自动驾驶环境构建。构建了基于虚拟机系统车辆控制软件，并在 ROS 工作环境下使基于树莓派车辆的 ROS 与虚拟机 ROS 进行通信，通过 CAN 总线保持上位机与下位机通信，实时获取车辆位置坐标，实现了按照规定的路线自动驾驶的实验。

关键词:车载控制器，CAN总线，动态规划算法，自动驾驶

Abstract

Nowadays, with the rapid development of information technology, the research on vehicle is more and more in-depth. At the same time, the requirement of VCU (vehicle controller) is raised. VCU makes driving strategy and energy management strategy by collecting sensor information and environment information, the collected sensor information is analyzed and processed, and the control command is sent to each unit through CAN bus to ensure the safe driving of the vehicle and achieve more power-saving effect.

In this paper, the vehicle-based VCU is taken as the research object, and the research status of automatic driving and vehicle-based controller at home and abroad is analyzed and studied, mainly aiming at the data transmission and control information communication of vehicle-based sensors, the optimization strategy of vehicle running energy is studied, and the VCU experimental platform is constructed and the simulation experiment is carried out based on ROS car. The main research contents include:

According to the characteristics of on-board sensor data and the requirements of control information, CAN bus is used for data transmission. Based on the analysis of the communication protocol of CAN bus, this paper analyzes the data information of vehicle sensors such as temperature sensors, accelerometer and Hall effect sensor, and the relationship between the data and the driving state of the vehicle, according to the frame format of Can and SAE J1939 protocol, the data communication protocol of multi-sensor and control node is established, which realizes the reliable transmission of vehicle information.

The energy optimization strategy of vehicle is studied. Firstly, based on the analysis of the new European driving cycle NEDC and the world light vehicle test cycle WLTC, the driving cycle of the vehicle based on the local and domestic urban environment is constructed, in the construction, the driving segments are extracted according to the driving segments in the driving process, and then the extracted segments are classified according to the speed range, and the optimal combination of driving segments is carried out by using K-means clustering algorithm, finally, the simulation data and the curve chart of vehicle driving mode are formed. On this basis, the dynamic programming algorithm is used to analyze the energy optimization simulation of the vehicle driving cycle, and the battery SOC (current battery residual) value is analyzed to judge the vehicle power consumption under the same distance and time. By comparing the SOC values of NEDC under new European driving cycle conditions and WLTC under World Light Vehicle Test cycle conditions before and after application, the dynamic optimization analysis is carried out through simulation analysis, it is proved that the energy optimization strategy of the vehicle is feasible.

The software and hardware design of VCU was carried out, and the VCU control platform was constructed. The Raspberry Pie was selected as the main control chip of VCU, and the STM32 single-chip microcomputer was selected as the slave control chip of sensors and control components, the design of raspberry pie peripheral circuit, completed the VCU main control platform design and build. Using lidar and other sensors to construct an autonomous driving environment. The vehicle control software based on virtual machine system (vms) is constructed, and the communication between the virtual machine and the raspberry-based vehicle is made in the ROS working environment, the vehicle position coordinates are obtained in real time, and the experiment of automatic driving according to the prescribed route is realized.

Key words: Vehicle Controller, Can Bus, Dynamic programming algorithm, automatic driving

目　录

第 1 章 绪论.....1
1.1 研究背景及意义.....1
1.2 国内外研究现状.....1
1.2.1 国外研究现状.....1
1.2.2 国内研究现状.....2
1.3 本文研究内容及章节安排.....3
第 2 章 车载通信网络设计与传输.....5
2.1 车载 CAN 通讯网络.....5
2.2 CAN 通信协议与报文帧格式.....5
2.3 基于 CAN 总线传感器数据通信.....6
2.3.1 车载传感器基本组成.....6
2.3.2 典型传感器工作原理与电路连接.....7
2.3.3 传感器数据采集与传输.....8
2.3.4 传感器数据传输协议的制定.....9
2.4 本章小结.....12
第 3 章 车辆能量管理控制策略设计与仿真.....13
3.1 能量管理控制的设计要求.....13
3.2 行驶工况的构建.....13
3.2.1 NEDC 行驶工况.....14
3.2.2 WLTC 行驶工况.....14
3.2.3 车载行驶工况构建.....15
3.3 传统能量管理策略.....18
3.3.1 恒温器型控制策略.....18
3.3.2 功率跟随型控制策略.....18
3.4 基于动态规划的能量管理策略.....18
3.4.1 动态规划算法原理.....18
3.4.2 动态规划算法的优化方法.....19
3.4.3 能量管理优化策略仿真分析.....21
3.5 本章小结.....22
第 4 章 车载控制器软硬件设计与实现.....24
4.1 车载控制器方案设计.....24
4.2 车载控制器的硬件设计.....25
4.2.1 主控芯片选择.....25
4.2.2 自动驾驶系统基本技术架构.....25

4.2.3 电源电路设计.....27
4.2.4 树莓派外围电路设计.....28
4.3 车载控制器的软件设计.....30
4.3.1 软件架构总体设计.....30
4.3.2 建立控制车载的软件系统.....31
4.4 车载自动驾驶的设计与实现.....32
4.4.1 自动驾驶车辆数据采集.....32
4.4.2 自动驾驶车辆定位解析.....32
4.4.3 自动驾驶车辆行走路径分析.....33
4.5 本章小结.....34
第 5 章 总结与展望.....35
5.1 总结.....35
5.2 展望.....36
参考文献.....37
附录.....40

第 1 章 绪论

1.1 研究背景及意义

进入到 21 世纪，我机动车和驾驶员数量正在急剧增加。根据交通管理部门的数据，到 2020 年底，中国将有 3.72 亿辆机动车和 4.56 亿名注册司机，这都达到了前所未有的水平[1]。任何事物都有两面性，机动车和驾驶员数量的增加，在方便人们出行的同时，也带来了一些问题，如拥堵、道路事故和环境污染等，这些问题日益严重[2]。如何提高交通效率和安全已经成为一个需要解决的全球性问题[3]，为了避免因驾驶员技能不足和驾驶失误而造成的交通拥堵和事故，自 d 动驾驶技术和无人驾驶汽车应运而生，它们能够通过严格执行车内程序，自动规划和自我管理车辆的行驶状态、道路、速度等状态指标，无需驾驶员干预[4]。

在经济高速发展的今天，汽车已经成为大多数家庭必备的交通工具，人们对汽车的需求也越来越大[5]。自动驾驶技术的发展也促进了驾驶技术和人工智能技术的进步，两者相辅相成，在汽车市场上得到了广泛的应用，有效地缓解和改善了交通安全、拥堵、资源消耗和空气污染[6, 7]。整车控制器主要分为两部分，一部分是硬件设计，由 VCU 和配套的控制与采集板组成。信号处理模块包括数字和模拟电路，通信模块主要是 CAN 通信电路以及继电器控制电路。另一部分是软件设计，指的是控制策略的开发，包括输入输出的控制、发动机控制算法、加速踏板分析、各种传感器传输信息的分析和处理，以及通过 CAN 网络将信号传输给各个控制单元。车辆控制策略的设计是通过制定合理的逻辑并遵循开发、实验、然后优化的顺序进行的。车辆控制器的控制逻辑必须进行优化，以处理复杂的环境，使机车即使在不同的环境中也能保持高度的稳定性和对干扰的稳健性[8]。

1.2 国内外研究现状

1.2.1 国外研究现状

自动驾驶在 1940 年首次被提出，随着人们的不断探索，技术也随之发展，国外发达国家开始展开了对自动驾驶的研究。1979 年，日本首当其冲设计开发了全世界第一辆自动驾驶汽车，随后进行了一系列的实际实验和测试，汽车最终能够以 30 公里/小时的速度自动行驶[9]。1990 年起，美国卡耐基梅隆大学开发了 NavLab 和 NavLab-2 智能汽车并在相应道路上测试，结果就是确实可以完成自动驾驶[10]。1994 年，在德国对开发的机器人汽车进行了测试，速度达到了 130 公里/小时，并实现了自动驾驶，经过一年的实验，它已经行驶了 1000 多公里[11]。自 2004 年起，美国国防部高级研究计划局组织了无人驾驶汽车挑战赛，对自动驾驶技术的发展和推广起到了关键作用[12]。2005 年，该挑战赛由斯坦福大学赢得，该大学驾驶使用一辆改装的大众途锐完成了整个障碍赛[13]。谷歌自 2009 年以来一直在开发自动驾驶汽车，并在 2018 年 12 月推出了世界上第一个商业自动驾驶服务[14]。Novariant、

Trrimble、Topcon 和 AgLeader 等几家国外商业公司已经测试了一整套自动驾驶系统产品，包括 RTK-GPS 定位系统、自动驾驶控制器和导航显示控制终端等核心模块，并在多年的技术积累中成为自动驾驶系统商业化和运营管理的领导者[15, 16]。

国外汽车工业在车载控制器的研究和开发以及更深层次的技术发展方面有着悠久的历史，特别是在欧洲、美国和日本等国家[[17]。博世、大陆和德尔福作为汽车零部件的供应商，在这方面最为突出，其产品范围包括全球主要的汽车 OEM[18]。在混合动力汽车中，那些复杂的控制策略能够对内燃机和电动机进行协调，从而提高车辆的燃油效率[19]。日产 LEAF 是最成功的全电动汽车之一，其车载控制器接收来自速度传感器和加速踏板传感器的信息，以完成车辆加速和加速控制，然后应用精确的发动机管理技术来实现接近传统燃料动力汽车规格的良好动力性能。此外，车载控制器还集成了车窗控制、除霜控制和空调控制，将这些功能进行整合确保了安全和舒适的驾驶体验[20]。丰田汽车公司的某款将车载控制器运用在两个后轮进行驱动，车辆的车载控制器接收来自车厢的控制信号和来自车载传感器的信号，如加速/减速踏板信号、换档信号和转角传感器信号[21]。车载控制器接收这些信号，随后对策略进行计算后将其传送给驱动器。由于有两个发动机，车载控制器还充当左右发动机控制器输出之间的协调者。欧洲历来重视节能减排，欧盟国家通过行政法规来鼓励电动汽车的发展和使用[22]。英国政府规定，如果电动汽车达到每百公里二氧化碳排放量低于 7.5 公斤的要求，购买者将获得最高 5000 英镑的补助，并免除燃油税和道路税等税费[23]。在 1997 年的法兰克福国际车展上，德国公司梅赛德斯-奔驰展示了 NECAR 3，这是一款以替代燃料甲醇为动力的燃料电池汽车，基于 A 级车，甲醇容量为 38 升，续航里程约为 300 公里[24]。法国政府与雷诺、标致和其他汽车制造商一起，计划在 20 个城市推广电动汽车，10 多个城市已经有了相对完善的充电设施[25]。进入 21 世纪，传统汽车因特斯拉在美国的强势崛起而受到重创，欧洲传统汽车公司也将先进的电动汽车推向市场。2018 年，英国捷豹公司推出了 I-PACE，这是一款电动 SUV，由前后轴的两个电机驱动，系统总输出功率为 294kW，最大系统扭矩为 696N/m。9 月，德国奥迪推出 e-tron，电池组总功率为 95kW-h，NEDC 续航里程为 470km，两个异步电机最大输出功率为 300kW[26]。2019 年 9 月，保时捷推出 Taycan，一款电动跑车。Taycan 由安装在前后轴上的两个永磁同步电机驱动，系统总输出功率为 560kW，最大扭矩为 761N/m，百公里加速时间仅为 2.8s。保时捷采用 800V 高压系统，可提供 250kW 的充电功率，只需 15 分钟就能将电池充电到 80%[27]。

1.2.2 国内研究现状

相比于国外，国内对于自动驾驶技术[28]的研究起步可以说是较晚，最开始大部分都是一些高校进行研究。1992 年，中国第一辆真正意义上的自动驾驶汽车在国防科技大学研制成功[29]。2001 年，在贺汉根教授的领导下，经过不懈努力最终研制出时速在 76km 自动驾驶汽车[30]。之后，百度也开始对自动驾驶汽车项目进行研究，并于 2021 年在北京和加州

的道路上测试，最后在乌镇试运行[31]。2016 年，小马智行等公司开始进入市场，进行自动驾驶汽车的研究。2021 年以来，随着对自动驾驶技术的研究国家也出台了相关政策，给研究机构和企业提供了重要支撑[32]。

田军辉[33]重点研究了纯电动客车的车载控制器驱动系统的控制策略，利用台架实验全面分析了电动客车使用的永磁同步电机和动力电池。在不同的模式下，执行相应的控制策略，发出最合适的扭矩指令，从而实现动力系统的最佳性能。朱晓琪[34]研究了电动汽车的电机控制器，认为稳定可靠的控制器可以对运行起到良好的作用。此外，该文详细介绍了基于 SAE J1939 协议的 CAN 网络通信的发展。张志鹏[35]研究了插入式混合动力汽车的换挡控制策略。由于混合动力汽车有两个动力系统，如何让它们协调工作是控制策略的关键点，需要同时考虑内燃机、电动机和电池这三者的状态和特性，以合理地分配输出扭矩。在升档过程中，根据不同的模式，有一个最佳动力换档策略和一个最佳经济换档策略。许同盟[36]使用飞思卡尔的 MPC5604A 作为核心搭建了 VCU 的硬件平台并详细描述了控制策略。这根据路况划分了驾驶控制区域，导致了更好的初始车辆体积响应。对于反向制动，发动机映射被指定为确定能量回收量的方式。此外，本文还开发了不同的操作模式以适应不同的操作情况。最后，作者还设计了一个车载高低电压控制、防滑系统和故障检测。王炜[37]介绍了与 ISO 26262 相关的功能安全，并在车载控制器的设计中加入了功能安全。选择 MPC5744 双核主控制器芯片来满足基本的功能安全要求。对三个功能进行了详细的功能安全分析：扭矩控制、高压升压和切断以及负载控制，并确定了 ASIL 级别 D、C 和 C。

学术研究者和汽车公司进行了大量的研究，以设计车载控制器的软件和硬件，并对其进行模拟和测试。闫龙涛等人[38]选用英飞凌 XC2268N 的 16 位芯片作为 VCU 主控芯片，设计并完成了不同功能模块的硬件电路。目的就是提高控制器的稳定性和准确性，通过使用较快运行速度和较多的芯片外围接口，来解决现有硬件一系列问题。宋晓妍等人[39]通过分析车载控制器的功能，进行全面解析处理，选择了 16 位 MC9S12XEP100 作为车载控制器的主控芯片，并采用了高效的冗余设计方案，满足了硬件资源的可扩展性。周美兰等人[40]通过分析车载控制系统组成，提出了一种复合电源组成结构，设计了基于逻辑门约束、CAN 通信网络和能量控制策略的复合电源工作模式，并通过 CRUISE 仿真和车载测试平台验证了设计的合理性，但是最后没有对实车进行测试。张庆等人[41]提出了一种带有驾驶模式检测的扭矩优化控制方法，以解决纯电动汽车扭矩控制中经济性降低等问题。对车载驾驶模式进行了仔细分类，并使用模糊控制器优化扭矩控制，并通过仿真分析证明了这个方法对改善车辆经济性有很大帮助。费为伟等人[42]通过 ConfigurationDesk 配置了车载控制器接口资源，并设计了基于 ControlDesk 的测试管理系统，通过结合测试用例、车载仿真模型、车载仿真模型，搭建了硬件测试平台，完成了电路中的实时测试。

1.3 本文研究内容及章节安排

本文以车载 VCU 为研究对象，对国内外自动驾驶与车载控制器的研究现状进行了分

析和研究，主要针对车载传感器的数据传输和控制信息通信，车辆运行能量优化策略开展研究，并构建了车载 VCU 实验平台并基于 ROS 小车开展了仿真实验。

第一章绪论部分主要分析了当前自动驾驶和车载控制器的发展现状和主要研究情况。

第二章车载通信网络设计与数据传输针对车载传感器数据的特点和控制信息的要求，采用 CAN 总线进行数据传输。本文在分析了 CAN 总线的通信协议的基础上，分析了温度传感器、加速度传感器、霍尔传感器等车载传感器数据信息以及数据与车辆的行驶状态的关系，根据 CAN 的报文帧格式和 SAE J1939 协议制定了多传感器和控制节点的数据通信传输协议，实现了车载信息的可靠传输。

第三章车辆能量管理控制策略设计与仿真研究了车辆的能量优化策略问题。首先在分析了新欧洲驾驶循环工况 NEDC 和世界轻型汽车测试循环工况 WLTC 的基础上，构建了基于当前国内城市环境的车辆的行驶工况，构建时根据行驶过程中的行驶片段对工况进行提取，随后按照速度范围将提取的片段分类，使用 K-means 聚类算法进行行驶片段的优化组合，形成最终的车辆行驶工况仿真数据和工况曲线图。在此基础上，通过动态规划算法对车辆行驶工况进行能量优化仿真分析，分析电池 SOC（电池当前剩余电量）数值来判断车辆在同样路程和时间下的用电情况。并通过对比新欧洲驾驶循环工况 NEDC 和世界轻型汽车测试循环工况 WLTC 应用前后 SOC 的数值，通过仿真分析动态优化分析，证明了本文车辆的能量优化策略是可行的。

第四章车载控制器软硬件设计与实现开展车辆 VCU 的软硬件设计工作，构建了 VCU 控制平台，选用树莓派做为 VCU 主控芯片，STM32 单片机做为传感器和控制部件的从控芯片，进行了树莓派电路外围的设计，完成了 VCU 主控平台的设计和搭建。应用激光雷达和其它传感器进行自动驾驶环境构建。构建了基于基于虚拟机系统车辆控制软件，并在 ROS 工作环境下使基于树莓派车辆的 ROS 与虚拟机 ROS 进行通信，通过 CAN 总线保持上位机与下位机通信，实时获取车辆位置坐标，实现了按照规定的路线自动驾驶的实验。

第五章对全文进行了总结和技术展望，分析了论文研究的结论及论文研究的不足之处，对今后的研究改进方向进行了展望。

第 2 章 车载通信网络设计与传输

2.1 车载 CAN 通讯网络

CAN 是一种串通信协议，相比其他的控制网络有着许多的通信优势。CAN 总线工作方式是多主进行的，不论节点的主从，随时都可以向其它的节点发送信息。CAN 通信时，不需要站点地址的节点信息，节点的最大数目由总线的驱动电路决定，如何增减节点不需要对整个系统进行修改，体现了系统扩展的灵活性。

车载通信网络一般都选用 CAN 总线做为车载数据传输的媒介。通过 CAN 总线实现车载 VCU 与众多车载传感器的数据传输以及车辆控制数据的传输。车载传感器的种类很多，包括压力、气体、温度、氧、位移、加速度、霍尔、光电、雷达等多种传感器，同时 VCU 要实现对车辆的控制发送控制指令，完成电池管理检测电池电量情况，显示器界面的数据等也要通过 CAN 总线进行传输，因此数据通信网络在车载传输过程中起到至关重要的作用。下图 2-1 为车载 CAN 通讯网路设计结构示意图。

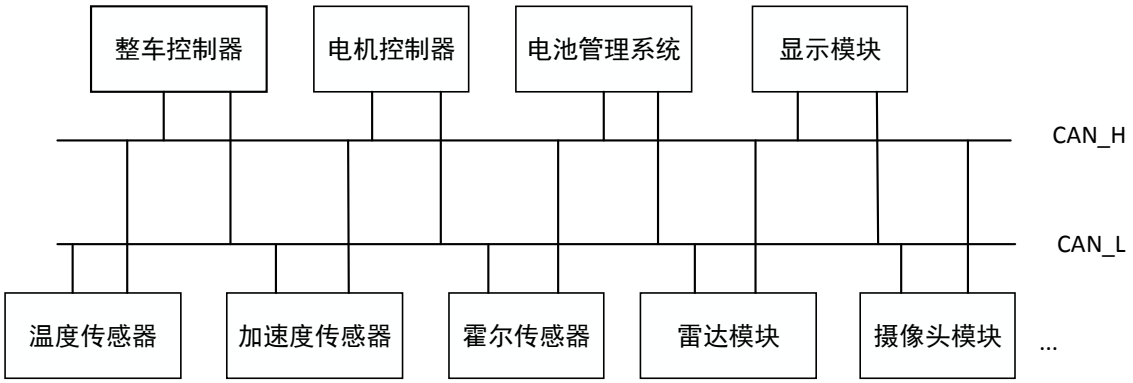

图 2-1 车载 CAN 通讯网络设计示意图

2.2 CAN 通信协议与报文帧格式

CAN 的基本通信协议是遵照规范并按照自己的需要制定的，以保证数据更严格的按照约束条件进行传输。本文选用的通信协议是 SAE J1939，它是 CAN 协议中应用最多并且是最为重要的协议，使用 CAN 总线协议作为核心对通信的不同层级要按照 OSI 模型制定相关的子标准，要明确通信传输的信息需在应用层子协议进行定义。SAE J1939 是高速通信网络可以支持闭环控制功能，这些电子控制单元在车辆的不同位置，只要总线处在空闲它们都可以向其发送信息，每个信息都是含有优先级、发送者和传输数据的标识符。传输时一般使用 CAN 总线仲裁进行标识符传输防止了总线的冲突，这样高优先级的数据在很快就可以通过网络。当同时有多个电子控制单元都要发送数据，即使它们的访问网络的资格是一样的，低优先级还是在冲裁时比不过高优先级的数据。

SAE J1939 使用 CAN 协议中定义的扩展帧 29 标识符组成一个完整的网络，并将 11 位标识符消息定义为特殊消息，这意味着允许在同一网络上使用具有 11 位标准帧标识符的设

备。尽管标准帧 11 位标识符不是 SAE J1939 的直接组成部分，但是将它们放在一起是为了确保用户可以与它们共存以避免冲突。SAE J1939 的数据帧格式和 CAN2.0 A/B 格式相同，表 2.1 表示了 CAN 报文标准格式、表 2.2 表示了 CAN 报文帧扩展格式。

表 2.1 CAN 报文帧标准格式

CAN 报文帧标准格式（帧最大长度=127 位）										
SOF	仲裁域		控制域			数据域	CRC 分隔符			
	标识符	RTR	IDE	R0	DLC		CRC	分隔符	ACK Field	EOF
1 位	11 位	1 位	1 位	1 位	1 位	0-64 位	15 位	1 位	2 位	7 位

表 2.2 CAN 报文帧扩展格式

CAN 报文帧扩展格式（帧最大长度=150 位）													
SOF	仲裁域					控制域			数据域	CRC 分隔符			
	标识符	SRR	IDE	扩展标识符	RTR	R1	R0	DLC		CRC	分隔符	ACK Field	EOF
1 位	11 位	1 位	1 位	18 位	1 位	1 位	1 位	4 位	0-64 位	15 位	1 位	2 位	7 位

其中仲裁段用于写明需要发送到目的 CAN 节点的地址、确定发送的帧类型（当前发送的是数据帧还是遥控帧），并确定发送的帧格式是标准帧还是扩展帧。仲裁段另一个重要的功能是通信优先级：当 CAN 总线空闲状态时，最先开始发送消息的单元获得发送权。多个单元同时开始发送时，各发送单元从仲裁段的第一位开始进行仲裁，连续输出显性电平最多的单元可继续发送。所以，帧 ID 值越小，优先级越高。

2.3 基于 CAN 总线传感器数据通信

2.3.1 车载传感器基本组成

当前汽车装载了众多种类的传感器，本文中选用了温度传感器 DS18B20、加速度传感器 MMA7361、转速传感器 A3144[43]作为应用实例，分析其使用方法和功能以及数据传输方法。应用 STM32 单片机对各个传感器数据进行采集并通过 CAN 总线的两根信号线 CAN_H 与 CAN_L 之间的差分电压将数据传输给 VCU 的 CAN 接口，可通过显示器显示数据值，可以实现多个传感器一起进行传输，并且 CAN 总线仲裁就根据传感器对应的优先级顺序进行有序传输，保证传输的实时性和有效性，通过显示器会显示数值，一旦出现了异常情况就会显示警告标识，此时驾驶员就要及时停下来进行车辆检测，保护其安全[44, 45]。

由于每个传感器都有安全的阈值，单片机将收到的信号经过 CAN 总线传输到 VCU 显示器上就是为了直观判断它们的数值是否安全。为了保证系统对发动机的温度、角度和转速等数值进行合理的判断，对其进行安全阈值的设定如表 2.3 所示。

表 2.3 传感器安全阈值

传感器	温度	加速度	转速
安全阈值	20-50 度	0-20 度	2000-6000 转

如果采集的数据在正常阈值之内，系统会直接将数据通过显示屏显示出来；如果采集数据在正常阈值之外，系统会将该项数据判定为异常数据，需要通过显示屏将异常符号和异常数据显示出来，达到通知驾驶员的目的。它的传输过程如图 2-2 所示。

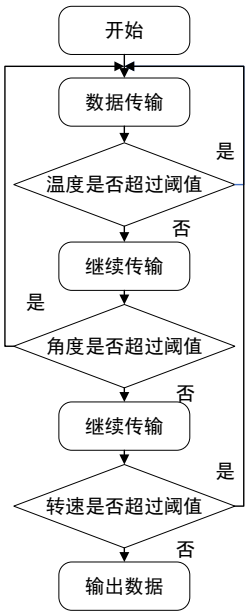

图 2-2 数据传输过程

将传输后的数据通过 CAN 总线按照优先级顺序传输到显示器上。根据 CAN 的 MAC 机制和总线仲裁，高优先级和低优先级的报文同时发送信息时，优先级高的会在冲裁中获胜，低优先级的只能先等待，但是还会一直尝试继续发送报文信息。只有优先级高的报文传输信息完成后，优先级低的报文才能获得传输权限，从而发送报文。所以在传输数据时要规定好优先级，这样系统才不会乱，能够有序准确的收到数据。

2.3.2 典型传感器工作原理与电路连接

温度传感器 DS18B20 是单总线结构，使用时可以提供 9 字节到 12 字节的温度值，它可以将采集到的数据直接串行发送到单片机上显示出对应的温度值。DS18B20 在上电之后处于低功耗的闲置状态，要想完成温度转换需要主设备向其发送温度转换指令 44h，转换完成之后，温度值会暂时保存在温度寄存器中，DS18B20 则会恢复原来的闲置状态，等待下一次的温度转换。如果 DS18B20 由外部电源进行工作，主设备可以在发送完温度指令 44h 后进行读取时序操作，如果温度正在进行转换则会响应“0”，转换完成之后会响应“1”。

加速度传感器 MMA7361 的工作原理就是通过采集 X、Y、Z 三个轴的变化，最终将其转化为对应的电压值。车辆本身静止的时候就会有对应的 X、Y、Z 三轴的值，其中 X 轴的值就是车辆前后移动产生的值，Y 轴就是车辆左右移动产生的值，Z 轴就是车辆上下移动所产生的值。加速度模块采集到的信号是模拟信号，并经过数模转换可以接到单片机的接收引脚。

霍尔传感器产生的霍尔电压随磁场强度变化而变化，它工作原理是当电机转动时，电机带动霍尔传感器运动，相应的就会产生频率脉冲信号，要完成脉冲的计数需要将各个周

期的脉冲信号在单片机脉冲寄存器下进行，同样可以完成转速的测量。霍尔传感器的数字信号就是判断车辆的的行驶或停止，模拟信号就是通过对磁感线不断地切割运动从而产生的电压变化达到对转速值的测量。它们与单片机的连接图如图 2-3 所示。

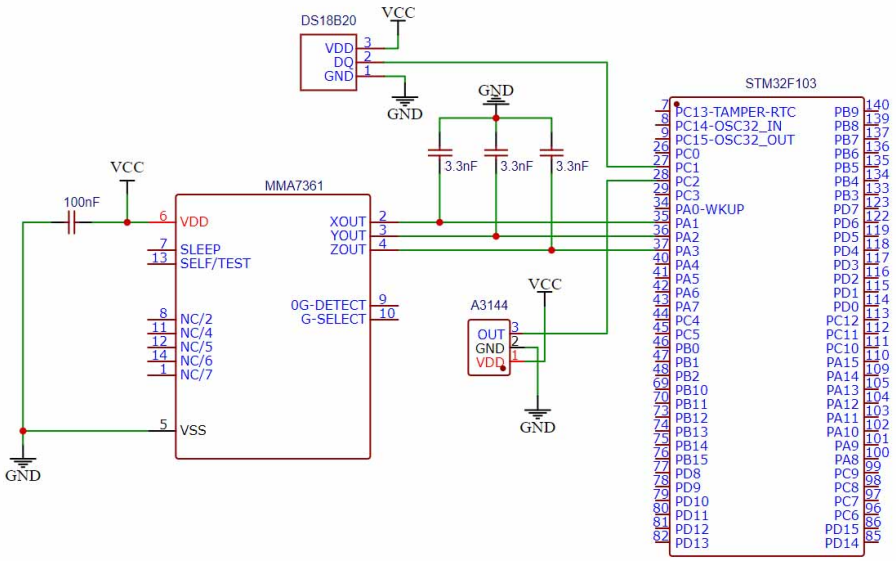

图 2-3 传感器与单片机连接图

从图中可以看出，温度传感器可以直接与 STM32 单片机进行连接，并且接到单片机的 PC1 接口，实现单片机直接对温度进行采集；将加速度传感器 X、Y、Z 引脚输出的信号分别接到单片机 PA1、PA2、PA3 接口；将霍尔传感器接到单片机的 PC2 接口。

2.3.3 传感器数据采集与传输

确定了不同传感器与单片机连接后，本节对其进行数据采集，采集到的数据是十进制的，数据接收端可以将数据转化成十六进制，采集时将传感器模块接到 5V 电源上。温度传感器采集的信号是电机实时温度数据，根据显示屏显示的温度情况及时关注当前温度，确保其安全。

加速度传感器采集的是将车辆进行前后、左右、上下运动之后变化的数据，规定向前、向左、向上为正，由于车辆是两驱的，所以它会有 L、R 的左、右的两个变化量。在静止状态时，车辆的角度变化为零。进行前后运动时，让车辆在平坦的路面上向前移动 30cm 的时候采集到的 X 轴的 L、R 变化量都为+500，当向后运动 30cm 时采集到的 X 轴的 L、R 变化量都为-494。在采集左右运动时，所用的方法就是向前向后保持同样的角度进行左右转弯，一定要保持在一向转弯的时候的角度不发生变化，一旦发生变化他的数据量就会出现误差。当进行向前左转弯的时候采集到 Y 轴的 L、R 变化量为+369、+630，进行向后左转弯的时候当保持角度一致时采集到的 L、R 变化量为-366、-625；当对向后转弯过程中对角度进行小幅改变后它的数据就变成-317、-541，进行大幅改变后数据变成-184、-315。当进行向前右转弯的时候采集到 Y 轴的 L、R 变化量为+585、+414，向后右转弯时采集到 Y 轴的 L、R 变化量为-580、-411；对向前运动时角度进行了改变，得到的数据为+527、

+335，+398、+154。这足以说明说明角度改变的越大，数值变化也就越大。当进行上下运动时，模拟就是把车辆抬起放下，抬起的时候 L、R 变化量都为+497，放下的时候 L、R 变化量为-495。然后根据 A/D 转化之后就可以把数据变成对应的电压值，在根据角度与电压值的对应公式可以得到相应角度值，电压值和角度两者关系如下：

$$V_{out} = V_{offest} + (\frac{\Delta V}{\Delta g} \times \lg \times \sin\alpha) \tag{2.1}$$

其中，V_{out} 为输出电压（V）；V_{offest} 为重力加速度为 0 时候的偏转电压（V）；$\frac{\Delta V}{\Delta g}$ 为传感器的灵敏度；$\lg$ 为重力加速度；α 为偏转角度；

通过上式可求得角度为:

$$\alpha = \arcsin(\frac{V_{out} - V_{offest}}{\Delta V / \Delta g}) \tag{2.2}$$

已知 V_{offest}=1650Mv，$\Delta V / \Delta g$=800mv/g，可求得角度为:

$$\alpha = \arcsin(\frac{V_{out} - 1650}{800mv / g}) \tag{2.3}$$

霍尔传感器采集的是对其进行切割磁感线，根据霍尔效应原理，输出霍尔效应传感器测量产生周期性电压变化的形式，通过记录变化的脉冲数量的脉冲，根据每单位时间的脉冲数量转换成电机转速测量的实际值。转速测量的方法是利用脉冲计数来测量转速。脉冲计数的主要方法是找出给定时间内所有的旋转脉冲。目前常用的电机转速测量方法有 M 法、T 法和 M/T 法。由于本文使用的车辆速度没有那么快，因此采用 T 法测量转速。T 法的应用原理是通过测量两个相邻旋转脉冲之间的时间来测量旋转速度[46]。VCU 用于捕获相邻的旋转脉冲，记录捕获电平变化的时间，并再次记录电平变化的时间值，在计算脉冲时，由于起始点的选择不同，难免会产生相对误差。测量转速公式为：

$$N = \frac{60f}{nM_2} \tag{2.4}$$

公式中 N 表示被测对象的转速，并且把电机转轴在每旋转一圈时所产生的脉冲数记作 n，f 为高频的脉冲频率，M_2 是电机轴旋转的脉冲周期中所有包含高频脉冲的次数。

在进行实验时，单片机接的是 5V 电源，霍尔传感器的初始值即为 1024，然对其进行切割磁感线运动，就可以看出明显的数据变化，当一直进行切割磁感线运动时它的数值变为 9，最后通过转速测量公式得到相应的转速，从而对转速的数值进行判断，通过采集到的数据可以说明，通信可以正常进行，实现了传输。

2.3.4 传感器数据传输协议的制定

本文中单片机负责处理传感器模块的信息，由于传感器采集的模拟量但是其本身可以对数模量进行转换，转换后对数字量进行传输，而树莓派则负责将处理后的信息进行执行

操作，它们之间通过 CAN 总线进行数据传输。下表 2.4 对通信协议的数据格式进行定义。

表 2.4 通信协议数据格式

报头	地址码	功能码	数据	报尾
0XFFFF	0X01	0X01	0X00000000	0X0d0a
2 字节	1 字节	1 字节	4 字节	2 字节

设置报头报尾的功能就是为了保证数据的完整性，地址码就是每台设备的地址唯一性，功能码就是设备完成不同功能的指令。下表 2.5、2.6 就是对地址码和功能码进行定义。

表 2.5 不同地址码的定义

序号	地址码	相关介绍
1	0X10	加速度传感器 X 轴分配的地址码
	0X11	加速度传感器 Y 轴分配的地址码
	0X12	加速度传感器 Z 轴分配的地址码
2	0X20	霍尔传感器转速分配的地址码
3	0X30	温度传感器分配的地址码
4	0X40	雷达传感器分配的地址码
5	0X50	摄像头分配的地址码

表 2.6 不同功能码的定义

序号	功能码	功能介绍
1	0XA0	读取加速度传感器采集的 X 轴指令
	0XA1	读取加速度传感器采集的 Y 轴指令
	0XA2	读取加速度传感器采集的 Z 轴指令
2	0XB0	读取霍尔传感器采集的转速指令
3	0XC0	读取温度传感器采集的温度指令
4	0XD0	读取雷达传感器采集的温度指令
5	0XE0	读取摄像头采集的温度指令

对地址码和功能码进行定义之后，使用 CAN 总线进行单片机与 VCU 的通信。通信时在单片机的 CAN 接口外接一个 CAN 收发器来进行数据传输。CAN 收发器选用 TJA1050，将 CAN 控制器传输的逻辑电平信号转换为 CAN 总线的差分电平信号。并且其支持 CAN2.0A 和 CAN2.0B 协议，所以只需要在外接一个 CAN 收发器 TJA1050 就能进行数据通信，CAN 发送引脚是 PA12(CAN_TX)，接收引脚是 PA11(CAN_RX)，与 CAN 收发器对应的串行输出 TXD 和输入 RXD 连接。收发器的 GND 和 RS 引脚接地，VCC 接 5V 电源，CAN_H 和 CAN_L 接到 CAN 总线上，由于 TJA1050 在通信时的波特率在 60Kb/s 以上，所以在 CAN 总线传输数据出现错误时能立刻阻止信号继续传输，保证了其正常的通讯。图 2-4 是设计的硬件电路。

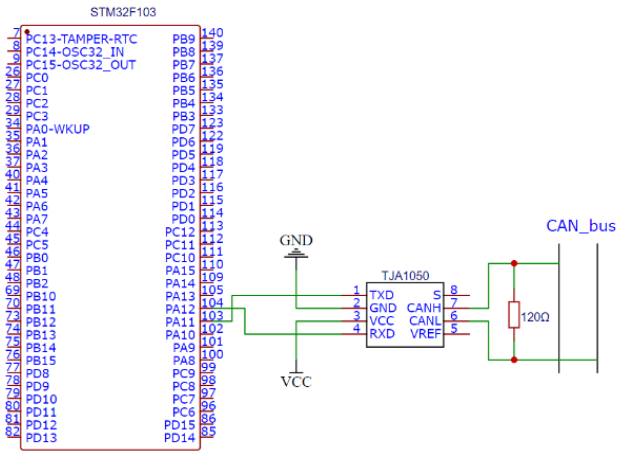

图 2-4 单片机与 CAN 收发器连接图

根据功能码的定义对传感器通信地址进行设置，加速度传感器 MMA7361 的 X 轴的地址为 0XA0、Y 轴的地址为 0XA1、Z 轴的地址为 0XA2，转速传感器 A3144 地址为 0XB0，温度传感器 DS18B20 地址为 0XC0。当 CAN 总线同时接收到传感器信息时，根据它们的地址编码进行优先级的排序，在传输时要先传输加速度传感器信号，它对车辆平稳起到了关键的作用，所以一旦车辆不平稳可及时进行调整，它的 X、Y、Z 轴在传输时前后顺序对车辆没什么影响，所以就按 X、Y、Z 轴的顺序传输，然后传输霍尔传感器，由于温度传感器的信号对车辆的影响最小所以它可以最后传输。规定 CAN 总线优先识别加速度传感器的编码 0XA0，0XA1，0XA2 然后识别转速传感器的编码 0XB0，最后识别温度传感器的编码 0XC0。

根据传感器地址码设定优先级传输顺序。它们的优先级顺序为：0X10＞0X101＞0X12＞0X20＞0X30。传输报文时首先让加速度传感器发送 0X10 的报文，这样就可以利用高优先级的报文避免总线空闲并可以一直处在工作状态。由于霍尔传感器与温度传感器的报文优先级不同，而且霍尔传感器的优先级更高，所以它们要一直实时监听总线的传输结果，只要有优先级高的报文还在发送，它们就得在旁边继续等待。当加速度传感器发送完报文后，霍尔传感器与温度传感器就要开始竞争下一个谁发送，就出现了竞争总线的情况。由于霍尔传感器优先级高，所以它将发送报文信息，温度传感器的报文将继续等待，等到它完全发送完报文后，温度传感器接收到信号，开始发送报文信息。

在发送过程中，将严格遵循 CAN 总线优先级机制。当低优先级消息发送消息时，如果它检测到高优先级消息也需要传输，那么低优先级报文需要停止发送并允许高优先级报文发送，低优先级报文在高优先级没发送完之前不能再次发送。所以在实际发送时要注意，尽量不要出现这种情况，要严格按照优先级顺序传输，这样容易出现接收到的数据不可靠，使数据没有说服性，所以在发送报文时，必须要等到高优先级的报文发送结束后低优先级再进行发送，如果在传输过程中出现了数据异常要停止发送报文并对数据进行处理，处理后在继续发送。

2.4 本章小结

本章根据车载通信网络设计与数据传输针对车载传感器数据的特点和控制信息的要求应用 CAN 总线进行数据传输。在分析了 CAN 总线的通信协议的基础上，分析了温度传感器、加速度传感器、霍尔传感器等车载传感器数据信息以及数据与车辆的行驶状态的关系，根据 CAN 的报文帧格式和 SAE J1939 协议制定了多传感器和控制节点的数据通信传输协议，实现了车载信息的可靠传输。

第 3 章 车辆能量管理控制策略设计与仿真

3.1 能量管理控制的设计要求

能量管理控制也是 VCU 设计的主要内容之一，由于纯电动汽车使用的电池的数量比较有限，汽车能够承载的能量也有限，续航里程达不到要求，一般采用两种方法进行提高[47]。第一种就是对电池的效率进行进一步开发提高，为了使电池在使用均匀性更好、充电速度更快，储能效率更高，但是这种方法目前来讲实现还有些困难。第二种就是结合电池研究一个更合理的能量管理策略，能够对电池组的电量进行合理的利用和分配，不仅可以提高使用率还可以保持电池的持续使用，从而达到提高车辆的续航里程的目的[48, 49]。

为研究能量管理控制策略，需要首先构建车辆行驶工况，本文采用当前常用的 NEDC 和 WLTC 行驶工况标准，然后按照不同速度范围对本文车辆行驶片段进行提取，经过聚类算法对行驶片段优化得到最优的行驶工况曲线，最后选择动态规划算法进行能量管理优化，在行驶相同距离相同起始 SOC 值的情况下，对比在使用算法前后剩余 SOC 的值，验证在能量控制策略下能够使车辆可以剩余更多的电量，从而完成能量管理优化策略，能量控制管理框图如图 3-1 所示。

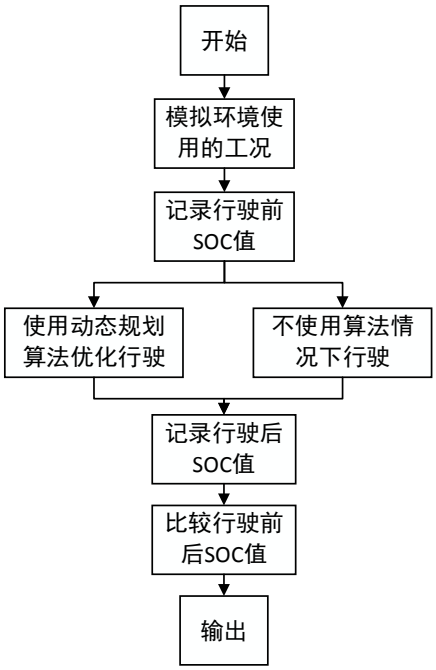

图 3-1 能量控制管理框图

3.2 行驶工况的构建

车辆的行驶工况主要是为了描述不同的车辆在不同的交通环境情况下的运动学特征，这种特性的描述一般使用速度-时间曲线来表示。曲线的时间区间可以根据此时的环境自行设定，现在公开常用的行驶曲线 NEDC 时间长度是 1180 秒，WLTC 是 1800 秒。汽车行驶工况大多应用在汽车排放、油量变化、动力情况，为汽车研制过程中的规则制定和性能优

化奠定了基础。在研究一个地方因交通情况导致的汽车运动变化时，为了设计出更合理的交通网络、汽车排放量、燃油利用率，对行驶工况的研究就变得尤为重要，另外建立车辆能效、评估各项指标和环保性能，以及开发和测试新车，也是汽车行业不可或缺的技术。

3.2.1 NEDC 行驶工况

NEDC（新欧洲驾驶循环工况）是欧洲的续航测试工况标准，最新的是 1997 年版本，包含 4 个市区循环和 1 个郊区循环。其中市区工况共 780 秒，最高车速 50km/h；郊区工况 400 秒，最高车速 120km/h。下表 3.1 表示的是 NEDC 循环工况数据特征以及图 3-2 表示的是 NEDC 循环工况示意图。

表 3.1 NEDC 循环工况数据特征

特征参数	数值
平均速度（km/h）	33.6
运行速度（km/h）	43.5
平均加速度（m/s^2）	0.53
平均减速度（m/s^2）	-0.75
加速比例%	23.2
减速比例%	16.6
匀速比例%	37.5
怠速比例%	22.6

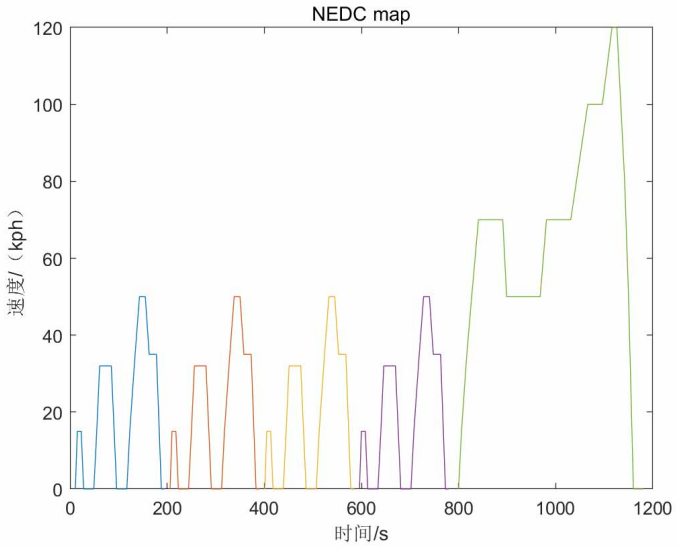

图 3-2 NEDC 循环工况示意图

一个完整 NEDC 测试循环共计 1180 秒，由四个市区工况循环（1 部）和一个郊区工况（2 部）组成。市区工况循环测试的时候最高车速为 50km/h，平均速为 18.77km/h，每个循环时间 195 秒，行驶 1.013km，最大加速度 1.042m/s²，平均加速度 0.599m/s²；郊区工况占用 380 秒，测试时最高车速为 120km/h，平均车速为 62.6km/h，有效行驶时间 400 秒，共行驶 6.955km 路程；最大加速度 0.833m/s²，平均加速度 0.354m/s²[50]。

3.2.2 WLTC 行驶工况

WLTC（世界轻型汽车测试循环工况）2017 年 9 月 1 日开始进行使用。相比 NEDC 工

况，WLTC 测试工况更加严格。WLTC 工况中没有周期性的加速、减速，更能体现在不同拥堵程度的路面以及车速时快时慢的情况。而且，由于工况变化没有周期性，车企在标定发动机时很难对其动手脚。另外，相比于 NEDC 工况测试体系，WLTC 工况的测试周期从 1180s 延长到 1800s，测试平均速度也从 34km/h 增至 46km/h，总里程为 23.25km，如图 3-3 所示

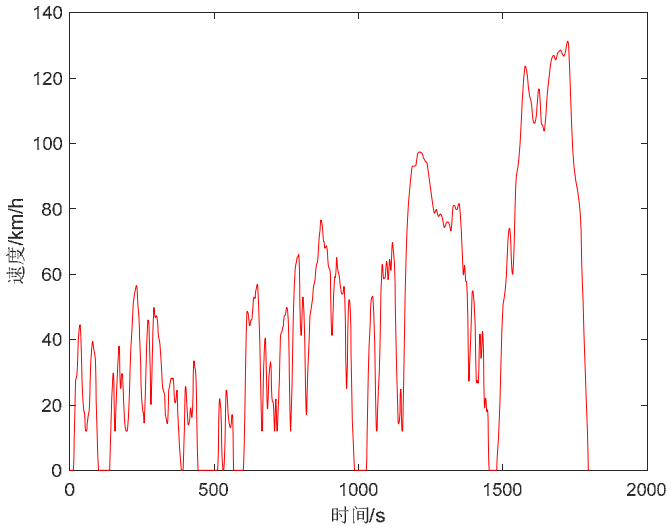

图 3-3 WLTC 循环工况示意图

WLTC 循环标准首先按照车辆功率和质量的比值分为三个等级，分别为≤22、22～34 和≥34。在每一个级别的驾驶测试中，WLTC 测试循环分为低速、中速、高速与超高速四部分，模拟了城市、环路、乡村和高速路段 4 种不同工况。低速、中速、高速与超高速四部分的持续时间分别为 589 秒、433 秒、455 秒和 323 秒，最高速度达到了 131km/h。加速、减速、匀速、怠速的占比约为 30%、27%、28%、12%[51]。

3.2.3 车载行驶工况构建

为分析能量管理控制策略算法的有效性，本文根据城市乡村等道路环境，以及标准工况实现的要求，构建了城市道路环境的行驶工况数据，车辆在模拟环境下行驶，并把获取行驶片段拼在一起得到一个完整的片段。有些采集到的数据无法直接使用，需要对数据进行筛查，删除不满足的，留下符合要求的片段，从而得到更优的片段。

原始工况采集数据包含时间、GPS 车速、加速度、经纬度、发动机转速、扭矩、油耗，这些数据通过行驶数据采集设备来记录。在采集的过程中或传输时原始数据经常会因为数据丢失、数据异常导致采集的信号质量不过关，倘若还继续使用这些数据进行处理运用则得到的结果也会不准确。经过查阅资料和文献[52]可知，将那些采集的行驶片段中有明显数据错误的进行删除，把加减速出现异常的行驶片段进行数据缺失处理，然后对时间上不连续的行驶片段挨个进行数据填补，最后对一直处在堵车、停车和怠速不正常的行驶片段再进行集中统一处理。处理结束后就需要对数据进行分析，由于行驶过程中会受到天气、交通情况、启停等多种因素影响，所以要将这些数据进行划分，划分为低速、中速、高速三

个类别，该工况全程是 1800s，模拟了市区、市郊、乡村三种不同工况，最高速度为 154km/h，总里程 25.5km。速度时间曲线如图 3-4 所示。

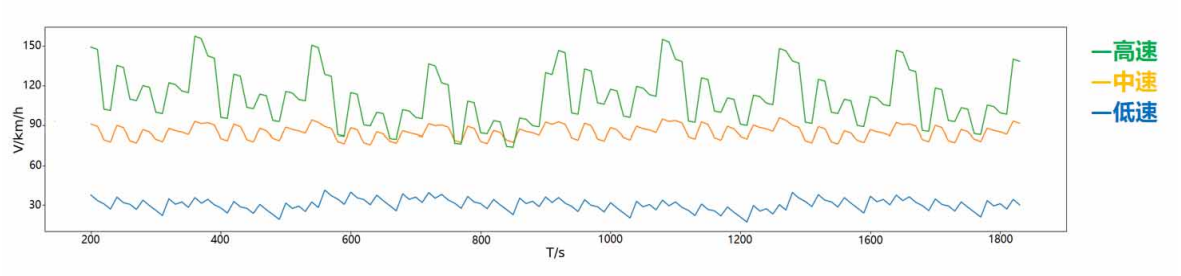

图 3-4 三种速度范围的行驶工况曲线图

行驶工况的构建是基于对原始数据处以及行驶工况片段提取之后完成的，通过使用数学模型和方法对数据进行分析处理，再将处理完的数据用聚类算法对不同行驶特征的行驶片段进行提取，这样就能组成行驶工况，最终将这些数据进行拟合，构造出能够体现采集数据的车辆行驶工况曲线。

聚类分析方法在进行行驶片段处理时扮演了非常重要的角色，目前聚类方法有很多种，其中应用较为普遍的是 K-means 聚类。K-means 算法中的 K 代表类簇个数，means 代表类簇内数据对象的均值（这种均值是一种对类簇中心的描述），因此，K-means 算法又可以说是 K-均值算法。K-means 算法是对数据进行划分的一种算法，通过衡量数据对象间的距离来判断它们的相似度，它们之间是呈现正比的关系，相似度越高它们在同一类簇的概率就越大[53, 54]。在计算数据间的距离时有很多不同方法，本文选用的 K-means 算法通常使用欧氏距离的方法对数据间的距离进行计算。

通过查阅文献[55]可得 K-means 聚类的具体分析过程如下：

（1）对 n 个样本依据特定的规则选取 k 个样本为初始类簇中心$(Z_1, Z_2, \cdots, Z_k)$；

（2）采用欧式距离将剩余的任意样本 x_i 的距离进行计算，欧式距离是指两个样本内所有的 n 个变量值差的平方和的平方根，源自欧氏空间中两点间的距离公式：

$$d_{xy} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.1}$$

式中，x_i 是样本 x 中第 i 个变量的值,y_i 是样本 y 中第 i 个变量的值。

（3）重新计算每个簇中的平均值，把得到的新平均值作为新聚类中心；

（4）对上述步骤重复进行，等到类簇中心不再变化时停止。

对行驶工况片段进行聚类分析时，设置聚类数为 6，迭代次数为 50 次。经过 16 次迭代，聚类中心收敛，对行驶片段的主成分进行聚类分析，选取主成分中的前三个，进行标准化得到聚类结果如下图 3-5 所示。

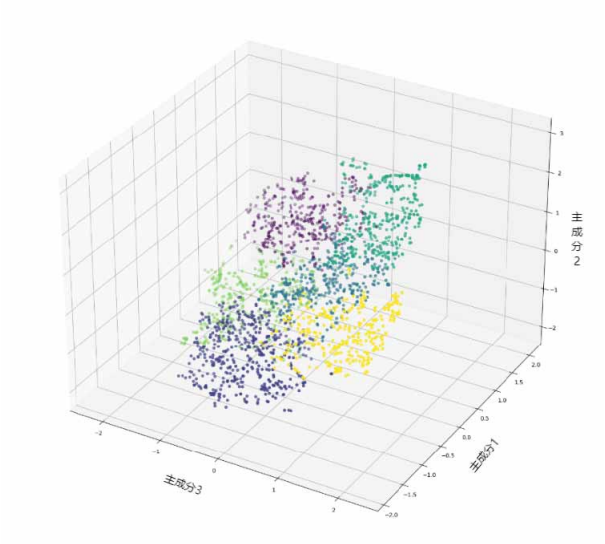

图 3-5 K-means 聚类结果

最后对各类运动片段进行选取，计算各类运动学片段到所属簇中心的距离，并分别在类内按距离从小到大的顺序排序，选取靠近簇中心的运动学片段构成典型工况，选取运动学片段的具体步骤如下：

（1）在所有类当中抽取若干运动学片段，使抽取的时间片段总和接近于该类占行驶工况的时间；

（2）对三类运动学片段进行判断，判断其所构成的片段是否满足行驶工况的时间，如果满足，这个片段可以作为备选行驶工况。

（3）将那些运动学片段时间总和小于占行驶工况时间的类，适当增加所选运动学片段的数目来获得更长的运动学片段，再转到第二步进行判断；

（4）将那些运动学片段时间总和大于占行驶工况时间的类，适当减少所选运动学片段的数目来获得更长的运动学片段，再转到第二步进行判断。

完成上诉步骤行驶工况的构建完成，结果如图 3-6 所示。

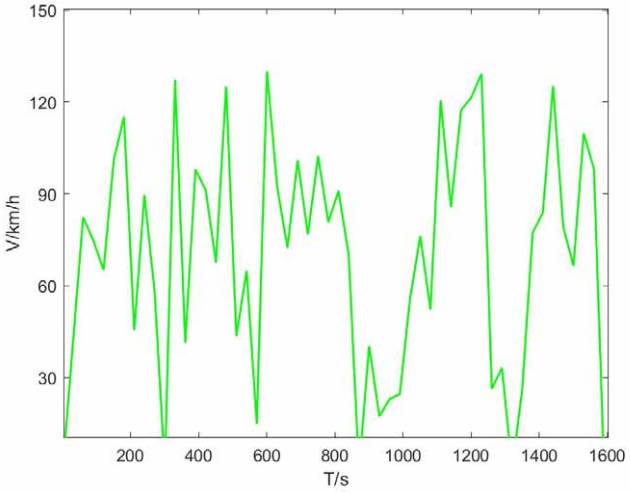

图 3-6 车辆行驶工况曲线图

3.3 传统能量管理策略

3.3.1 恒温器型控制策略

恒温器策略具计算性强的特点，是混合动力系统中广泛使用的一种模式，它的控制策略需要设置 SOC 的上下限值。当 SOC 低于 SOC 下限值时，发动机启动（发动机功率一部分驱动电机，一部分给电池组充电）；当 SOC 高于 SOC 上限值时，发动机停止进行工作仅由电池组单独进行驱动。策略中规则（例如 SOC 上下限）是根据工程师经验决定，因此此策略本身存在着局限性。因为恒温器策略实用性低、经济性差，常见的实用型能量控制规则都会将恒温器策略与其他策略相结合，先后进行后循环和前循环确定策略合理性。

恒温控制策略的优点是，它可以确保发动机始终在其最佳工作点上运行，而不是过于频繁地开启和关闭，从而在整个驾驶周期内满足里程需求，但是却没有达到最佳经济性。为了防止电池的 SOC 进一步下降，发动机必须以高功率运行，如果电池过度充电，将不利于电池的寿命[56]。

3.3.2 功率跟随型控制策略

在基于功率的控制策略中，发动机的状态根据电池组充电功率的 SOC 值而变化。如果电池 SOC 值较高，而车辆功率需求较低，发动机就会停止运行；如果电池 SOC 值较低，而车辆功率需求较高，发动机就会正常运行。当发动机正常运行时，功率输出在一定范围内上下波动。这种控制策略确保了发动机不会频繁启动和停止，功率始终是可变的，因此不会导致持续的大电流充电，同时通过缩短电池周期来保护电池寿命，不但对发动机不利，而且不能达到成本效益和排放的目标，也不能确保在周期结束时提供全部电池电量[57]。

传统的能量优化算法虽然操作过程简单，有各自的优点的同时都可以达到能量优化的效果，但是整体来看还是不满足要求，对电池的损坏同样存在，现今传统能量算法已经不在是首选，有很多更佳的能量优化算法。

3.4 基于动态规划的能量管理策略

本节开展对行驶工况的能量管理优化进行研究。研究其能量管理的目的就是尽可能的减少行驶工况过程中电池的电量消耗，在同样的时间内电池 SOC 越多表明能量消耗越少从而达到了能量优化的目的，本文选用动态规划算法对行驶工况进行优化求解，比较使用算法之前与使用之后电池剩余电量的情况，并通过对 NEDC 和 WLTC 行驶工况进行比较，以防结果的偶然性。

3.4.1 动态规划算法原理

采用动态规划算法求解多阶段重复决策问题，该算法的核心思想是将多个重复决策阶段转化为一个个独立的子阶段逐个求解，其中任何一个子阶段和状态都可以看作是初始状

态和初始阶段，其余的决策必须由该初始状态构成最优策略[58, 59]；算法中各阶段的状态变化基于递推关系，可以存储前一阶段的结果，避免重复计算，从而提高求解优化问题的效率。根据求解问题的出发点和寻求最优决策的顺序进行计算[60]。

本文研究的车辆是纯电动的，它的电池 SOC 剩余的电量值根据行驶工况的变化而不断改变，存在很多未知性，但是初始的 SOC 的值是确定的而且容易得到它的数值，所以本文中选用前向动态规划方法会更加方便，即以 SOC 的初始值为研究起点，按照正向规划算法规律进行求解，对最后的 SOC 的值进行比较，证明能量管理优化的可行性。

3.4.2 动态规划算法的优化方法

由于车辆的行驶工况明确，其总需求转矩能保证的前提下，能量管理策略需要决策出前后电机的最优转矩分配规则，合理分配电池的电量，使车辆能够最大程度的达到节能效果。下面就是其主要的求解过程。

（1）能量优化问题阶段化

通过选用时间来将已知的行驶工况离散化，设置时间间隔 1s，这样就可以用非线性的离散时间系统来代替车辆的能量优化问题。

（2）设置系统的状态变量和控制变量

状态变量就是用来对系统变化过程进行描述。在车辆中，电池的 SOC 就是为了可以直观的感受到在行驶过程中的能量变化，即把电池 SOC 作为本次系统的状态量如式（3.2）所示。则控制变量显而易见的设置成前电机转矩 T_{m_f} 和后电机转矩 T_{m_r} 两者构成的控制向量如式（3.3）所示。

$$x(t) = SOC(t) \tag{3.2}$$

$$u(t) = [T_{m\_f}(t), T_{m\_r}t)] \tag{3.3}$$

（3）确定状态转移方程的递推公式

确定状态转换方程时要对用到的不同变量进行分析，其中电流就是通过需求功率和电压的输入得到的，而需求功率则是取决于电机功率和所属附件功率，表达关系式如式（3.4）所示。将电压模块获得的电压采用基尔霍夫电压定律得到电压值，它们之间具体的公式如式（3.5）所示。要确定初始的 SOC 的值，选用积分估算法估算其 SOC 初始值，把电流在时域上进行积分，积分之后的结果就是 SOC 的值，计算公式如式（3.6）所示；电池组的内阻、开路电压取决于功率流动方向、目前温度情况、电池当前 SOC 值。开路电压的计算公式如式（3.7）所示，内阻的计算公式如式（3.8）所示。

$$P = \frac{P_m}{\eta_m^b} = P_{acc} \tag{3.4}$$

$$I = \frac{P}{U} \tag{3.5}$$

$$SOC=\frac{\left(Q-\int\frac{I}{3600}dt)\right)}{Q} \tag{3.6}$$

$$V_{oc}=f\left(T_{temp},SOC\right) \tag{3.7}$$

$$R_{batt}=f\left(I,T_{temp},SOC\right) \tag{3.8}$$

其中，P_m 为电机的输出功率；η_m^b 为当前状态下电机的效率值，b 为效率常量，b 只能取正负 1，取 1 代表放电；取-1 代表充电；P_{acc} 为车载附件功率；I 为此刻的电流 Q 为电池的容量，T_{temp} 为当前电池的使用温度。由此可以得到相邻两个子阶段的 SOC 的关系式如式（3.9）所示。

$$SOC(t+1)=\frac{\left(V_{oc}-\sqrt{V_{oc}^{\ 2}-4R_{batt}P}\right)\Delta t}{2R_{batt}Q}+SOC(t) \tag{3.9}$$

（4）建立系统的代价函数

在 t 时刻内阶段代价函数要对状态量施加一个控制量，然后就会产生代价。本文中将动力系统的功率最小消耗量作为目标，在动态规划算法下即可得到阶段代价函数如式（3.10）、（3.11）所示。

$$L_t=\frac{P_{m\_f}}{\left(\eta_{m\_f}\right)^{k\_f}}+\frac{P_{m\_r}}{\left(\eta_{m\_f}\right)^{k\_r}} \tag{3.10}$$

$$k\_i=\begin{cases}1,P_{m\_i}>0\\-1,P_{m\_i}<0\end{cases}i=\{f,r\} \tag{3.11}$$

要想将动态规划算法所决策出来的最优控制量能够按照要求在车辆上完成，在最开始求解时要对系统中不同的变量加以限制，具体的限制条件如式（3.12）所示。

$$s.t.=\begin{cases}SOC_{\min}\le SOC(t)\le SOC_{\max}\\T_{\min\_f}\le T_{m\_f}(t)\le T_{\max\_f}\\T_{\min\_r}\le T_{m\_r}(t)\le T_{\max\_r}\\n_{\min\_f}\le n_{m\_f}(t)\le n_{\max\_f}\\n_{\min\_r}\le n_{m\_r}(t)\le n_{\max\_r}\end{cases} \tag{3.12}$$

其中，$SOC_{\min}$、$SOC_{\max}$ 分别为状态量 SOC 的上下边界值；$T_{\min\_f}$、$T_{\max\_f}$ 分别为前电机输出的最低转矩及最高转矩；$n_{\min\_f}$、$n_{\max\_f}$ 为前电机输出的最低转速及最高转速；$T_{\min\_r}$、$T_{\max\_r}$ 分别为后电机输出的最低转矩及最高转矩；$n_{\min\_r}$、$n_{\max\_r}$ 分别为后电机输出的最低转速及最高转速。

对系统方程建立之后，车辆在选取行驶工况中最小功率消耗问题就可以转化为在各个阶段求最优控制变量的问题，其具体步骤如下所示。

（1）把确定的车辆运行工况所需的出行时间离散为 T 个子阶段（初始子阶段不在离散范围），根据车辆纵向动力学方程对不同子阶段的车载需求扭矩 $T_{req}(t)$ 进行计算。

（2）将每个子阶段的状态变量用 ΔSOC 离散化，不同子阶段分别对应不同状态变量集合，则在 t 阶段 SOC 的集合可用式（3.13）表示。

$$x(t)=\{SOC_{\min}(t),\Delta SOC+SOC_{\min}(t),\ldots SOC_{\max}(t)\} \tag{3.13}$$

（3）前向求解各子阶段的控制量及对应阶段到初始阶段的累积代价函数。离散化处理电机扭矩变量，离散间隔为 ΔT，则 t 阶段的 j 状态对应的控制集合如式（3.14）所示。

$$u_{t\_j}=\{T_{\min\_i}:\Delta T:T_{\max\_i}\} \tag{3.14}$$

其中，$T_{\min\_i}$、$T_{\max\_i}(i=f,r)$ 为约束范围内各电机对应的最小和最大转矩。

采用前向规划算法的数学表示为：系统处于初始阶段时，状态量 $x(0)$ 为 SOC 的初始值 SOC_{ini}，对应的代价函数如式（3.15）所示；系统处于阶段时，最优代价函数为（3.16）所示。

$$J_0(x_0)=L(x_0) \tag{3.15}$$

$$J_t=\min[J_{t-1}+L_t] \tag{3.16}$$

其中，L_t 为 t 阶段采取控制量所产生的直接代价；J_{t-1} 为前一阶段累积的代价。重复上述过程，直到系统的末阶段结束。

（4）逆向寻找最优控制量。通过比较系统最后一阶段的累积成本，得到从前一阶段到最后一阶段的最优控制量，并根据相似的思路逐级搜索各阶段的最优控制量，直到初始阶段，最后确定最优控制序列，得到各阶段对应的 SOC 状态点的全局最优轨迹。

3.4.3 能量管理优化策略仿真分析

将得到的 NEDC、WLTC 行驶工况和构建的车辆行驶工况在动态规划算法下进行对比，设置初始的 SOC 值为 0.9，整个仿真时间定为 1600s，下图 3-7、3-8、3-9 为 NEDC 工况、WLTC 工况、车辆行驶工况的仿真结果。

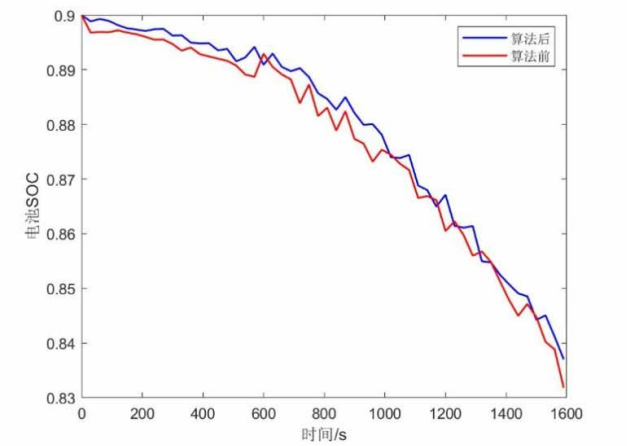

图 3-7 NEDC 工况算法前后仿真图

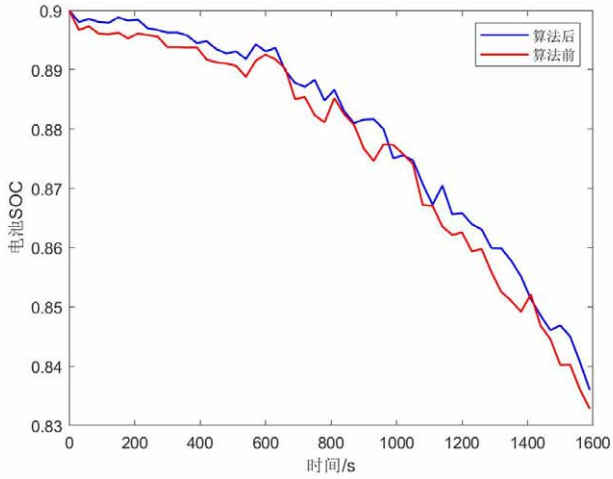

图 3-8 WLTC 工况算法前后仿真图

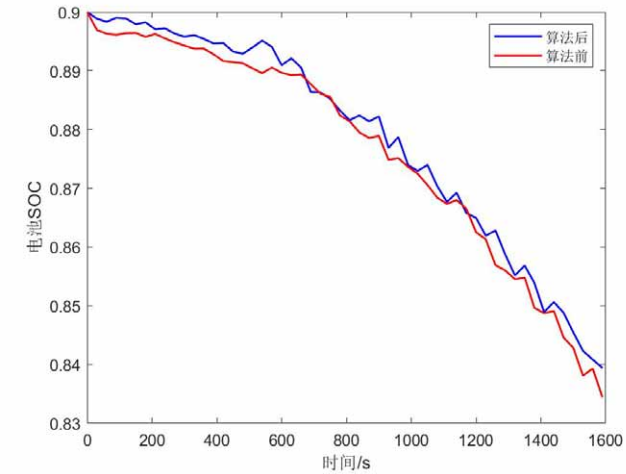

图 3-9 车辆行驶工况算法前后仿真图

表 3.2 为未使用算法和使用算法 SOC 对比值。

表 3.2 使用算法前后 SOC 对比表

	NEDC 工况	WLTC 工况	文中构建的工况
没有使用算法 SOC 值	0.8217	0.8376	0.8462
使用算法后 SOC 值	0.8725	0.8624	0.8986

从图和表中可以看出在初始 SOC 值都是 0.9 时，相同时间和相同行走路径的情况下，各个工况不使用算法时剩余的电池量都比使用算法时剩余的电池量会低，而且电池 SOC 之间有一定的差距值，说明动态规划算法起到了作用，可以使车辆的剩余电池量的值增多，从而达到能量优化的目的。

3.5 本章小结

本章主要研究了车辆的能量管理策略，首先明确了能量管理的要求，对车辆要实现的能量管理进行设计，其次对新欧洲驾驶循环工况 NEDC 和世界轻型汽车测试循环工况 WLTC 进行研究分析，为后续做对比提供基础，然后构建车辆的行驶工况，构建时对不同速度的片段进行分类组合，构成一个初始的行驶片段，再通过聚类算法对行驶工况进行聚类分析

筛选得到优化的行驶工况片段，最后通过动态规划算法实现对整车能量管理，将车辆的电机功率变化转换成 SOC 的值，通过剩余 SOC 的值判断出车辆实现对能量管理优化策略算法有效性的分析和对比。

第 4 章 车载控制器软硬件设计与实现

4.1 车载控制器方案设计

车载控制器(VCU)平台是实现 VCU 控制算法和控制对象的载体，通过采集传感器信息数据和构建管理策略实现对车辆的控制，同时具有多种可控制策略的软件编程接口和信息采集与传输接口，具有信息通信传输协议。VCU 的主要原理就是采集传感器采集数字或模拟信号，并通过 CAN 总线以及 CAN 总线协议与其他的子系统的电控单元进行连接，接收各个子系统电控单元发送过来的数据信息。VCU 将接收到的信息进行处理，当某个环节出现故障时，可以对故障进行分析诊断，可以保证数据的正确传输，维持车载通信系统的正常，如果出现严重故，车辆就不会继续行驶，等待下一步检修和处理。车辆 VCU 实现的功能包括：

（1）车载网络管理。在整个车辆网络管理系统中，CAN 是主要通信方式，VCU 是网络控制的中心，主要就是负责报文的组织与传输、CAN 网络的监控、故障诊断以及处理。

（2）车载能量优化管理。为了使车辆在相同条件下能够更省电行驶更远的距离，VCU 对车载的电机控制系统、电池管理系统及其它的车用附件进行能量优化控制策略，目的就是为了提高行驶时间增加续航里程。

(3）道路规划与避障。实现对车辆运行道路环境下的最优路线规划和障碍规避等功能。

根据车辆实现的功能设计车载控制器的组成，下图 4-1 为车载控制器系统组成。

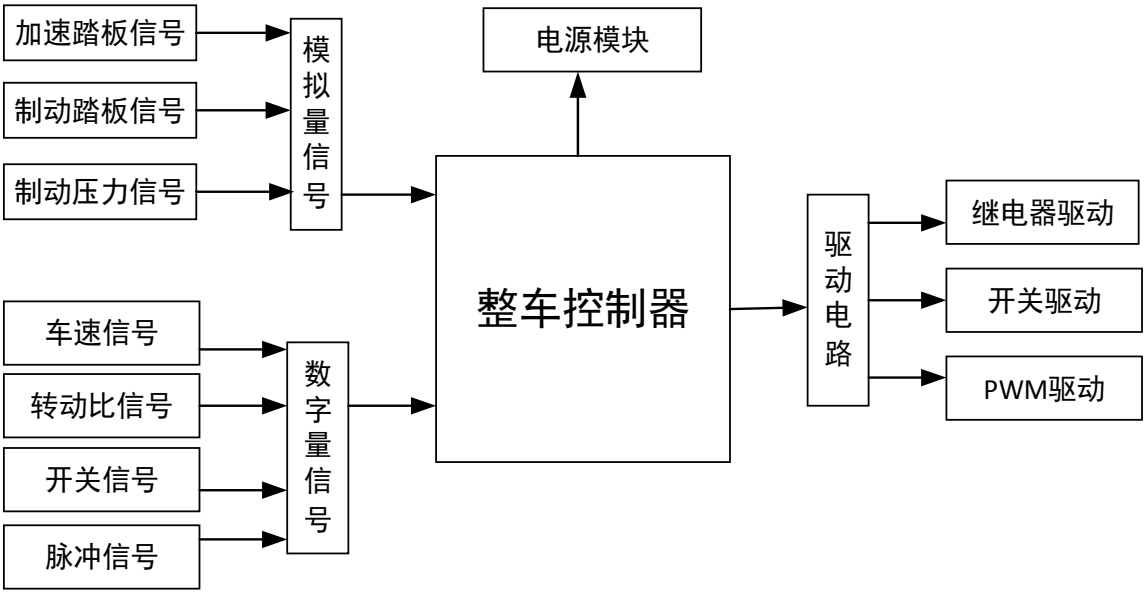

图 4-1 车载控制器的组成

本文采用 ros 小车开展试验研究，由于目前树莓派在工业方面使用广泛，而且它是具有微型电脑控制的嵌入式的设备，运行的是 Linux 系统，与此同时还集成通信、控制、感知、存储等各种不同的资源，还可以完成车辆按照预定轨迹预测和行走，在行走过程中遇

到障碍物还会有主动识别和避障的任务。在价格方面又低于 NVIDIA 系列嵌入式产品，而且性能明显高于日常的计算机，特点也比较显著，在自动驾驶方面有很大的空间，所以本文选用树莓派来进行设计，与此同时结合 STM32 搭建一个总体的控制器平台，在平台上操作控制车辆完成自动驾驶设计。

4.2 车载控制器的硬件设计

4.2.1 主控芯片选择

本文选用的主控芯片是 32 单片机和树莓派 4B 型结合起来来实现 VCU 各种复杂的控制功能。其中 STM32 单片机是由 ST 厂商推出的 STM32 系列单片机，在与其它系列产品比较时性价比更有优势，功能非常强大[61, 62]。

树莓派 4B 型是新一代树莓派计算机中的第一款，支持更多的 RAM，并显著增强了 CPU、GPU 和 I/O 性能，有很多方便实用的特点可以满足 VCU 的功能需求，下面就是树莓派的一些特点。

（1）硬件特点：可以运行在 1.5GHz 的四核 64 位 ARM-Cortex A72；1,2,4 GB LPDDR4 RAM 可供选择使用。

（2）接口特点：有 2 个微型 HDMI 端口支持双显示器，最高可达 4Kp60 分辨率；1 个千兆以太网端口(支持 PoE，并添加 PoE HAT)；有一个 SD 卡还有 28 个 GPIO 口可供使用，其中包括 6 个串口、6 个 I2C 接口、5 个 SPI 接口、1 个 SDIO 接口、1 个 DPI 接口可并行 RGB 显示、1 个 PCM 接口、2 个 PWM 通道、3 个 GPCLK 输出接口。虽然其不具备 CAN 总线接口但是它本身具备 CAN 功能，只不过使用 CAN 总线时要在 SPI 接口连接一个 CAN 控制器，还可以搭配一个 CAN 收发器，这样就可以用 STM32 单片机采集数据，处理后通过 CAN 总线传输到 VCU 树莓派上，VCU 接收到信号后在进行执行操作，这样分布进行更快捷，在出现故障时方便检测。

（3）软件特点：ARMv8 指令集；有着成熟的 Linux 软件栈；最新的 Linux 内核支持，使用标准应用程序接口提供图形处理单元功能，并且有很多稳定的用户支持和使用，使用范围广泛。

树莓派 4B 和传统计算机不同，传统计算机的内存和硬盘是分开的，树莓派 4B 则采用的 SD 卡作为统一的内存和外存，可以使用 HDMI 或 VGA 接口连接屏幕显示，USB 外接摄像头、鼠标、键盘等一些其它外设，构成了一个微型的计算机[63, 64]。

4.2.2 自动驾驶系统基本技术架构

自动驾驶由三个部分构成，分别是感知层、认知与决策层、执行层。其中感知层主要任务是通过车载传感器以识别周围环境、自身定位、采集数据信息，包括车辆运动信息、

环境感知情况。过程中比较重要的传感器是车辆运动传感器它的原理就是通过角度和速度将传感器采集到的横向和纵向信息反馈到车辆线控系统。在上文已经对角度传感器进行了分析，通过采集的数据，根据对应的公式转换成相应的角度信息，然后对其进行分析判断，判断得到的角度数据是否在安全的阈值内，可以确保车辆安全平稳的工作。惯性导航系统主要作用就是提供车辆定位信息，它是由陀螺仪和加速度计组成计算车辆导航参数的系统，其中根据陀螺仪输出的数据对车辆的坐标系进行构建，根据加速计输出的数据对车辆在坐标系内的速度和位置信息进行计算，它们之间相互配合构成了完整导航系统。其对环境没有依赖性不受其它信息和辐射的影响，而且它的工作环境也比较广，不但包括空气、地面，还包括水下。惯性导航的基本原理来自于牛顿力学定律，牛顿力学定律是通过测量载体在整个系统的加速对时间进行整合，将整合后的数据转化为坐标系，根据坐标系中的数据就可以得到载体速度、位置和偏航角等信息[65]。

环境感知传感器是与人类视觉和听觉非常相像的用来感知环境的传感器，采集时根据雷达和摄像机数据融合送到计算单元并运用算法对数据进行处理，如果没有环境传感器就不能实时采集环境信息，没法提供给车辆环境情况即自动驾驶就无法完成。摄像机利用其图像传感器将采集到的光信号转换为电信号，最后经过 AD 转换变成数字图像信号，最后发送到数字信号处理芯片进行处理。

由于雷达的类型有很多种，本文选用的是 RPLIDAR A1 360 度激光扫描测距雷达，测量时的基本原理为：每次测距过程中，RPLIDAR A1 将发射通过调制的红外激光信号，该激光信号在照射到目标物体后 RPLIDAR A1 的视觉采集系统接收产生的反光。将反光通过 RPLIDAR A1 内部的 DSP 处理器进行实时解算，被照射到的目标物体、RPLIDAR A1 的距离值以及当前的夹角信息将从通讯接口中输出[66]。

认知与决策层首先是认知理解，感知层通过采集信息对车辆的位置进行精确定位；其次是决策规划，由于不确定接下来会发生什么所以要对其提前进行预测，从而可以对接下来的行动做出正确的判断和合理的规划。下图 4-2 就是认知与决策层过程。

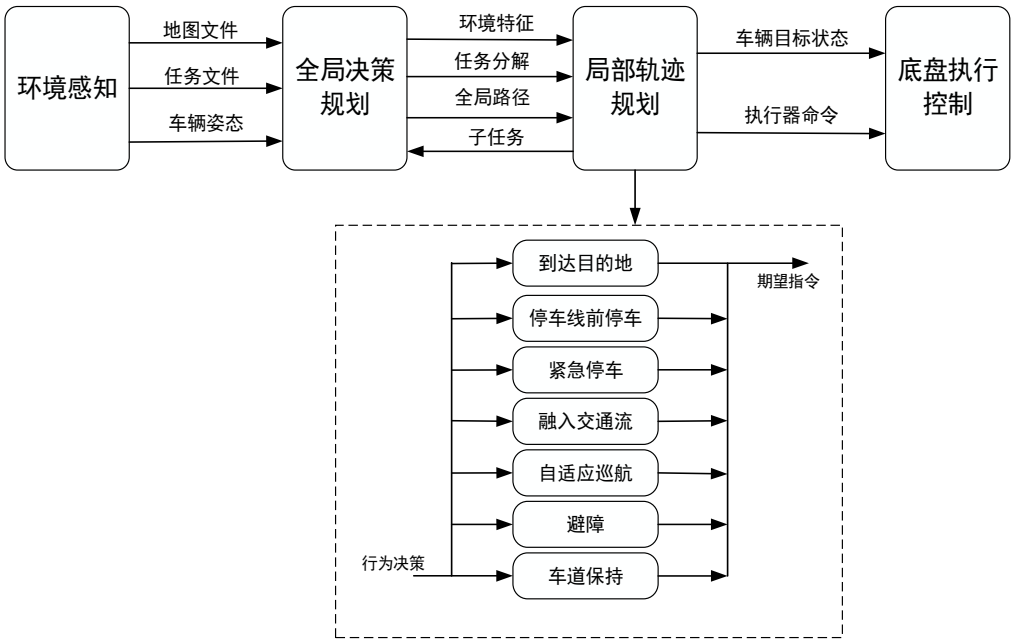

图 4-2 认知与决策层过程

其中全局决策规划是指自动驾驶汽车根据地图上的信息，进行自身任务的确定，生成从起点到终点最优路径；局部运动规划就是知道了车辆怎么走后，需要车辆判断眼前是否有障碍物、是否进行避障、以及运动的规划等信息。

执行层则是根据认知与决策层运动规划的期望目标，控制车辆达到预期结果，例如车速，位置等信息[67]。

4.2.3 电源电路设计

单片机最小系统包括电源电路、时钟电路、复位电路和下载电路。其中电源电路采用 5V 工作电压；时钟电路则是整个芯片的核心，芯片只要开始工作就得在基准时钟频率下进行。时钟电路的组成部分包括晶振芯片、电容、晶体振荡器，晶振芯片有很多规格，按照要求选合适的晶振；复位电路同样也特别重要，它是用来将芯片重置在最开始的状态，在使用或者测试过程中，遇到的大多数情况都可以通过复位来解决，方便了 VCU 的调试也可以防止在跑程序的时候程序跑飞；下载电路则就是对 VCU 进行程序的烧录和硬件的电路调试。电源电路可以给车辆供电，是完成自动驾驶的前提条件，本文中电源电源电路都使用的是 5V 工作电压，其中包括树莓派电源电路如图 4-3，USB 电源电路如图 4-4，整个 VCU 电源电路如图 4-5 所示。

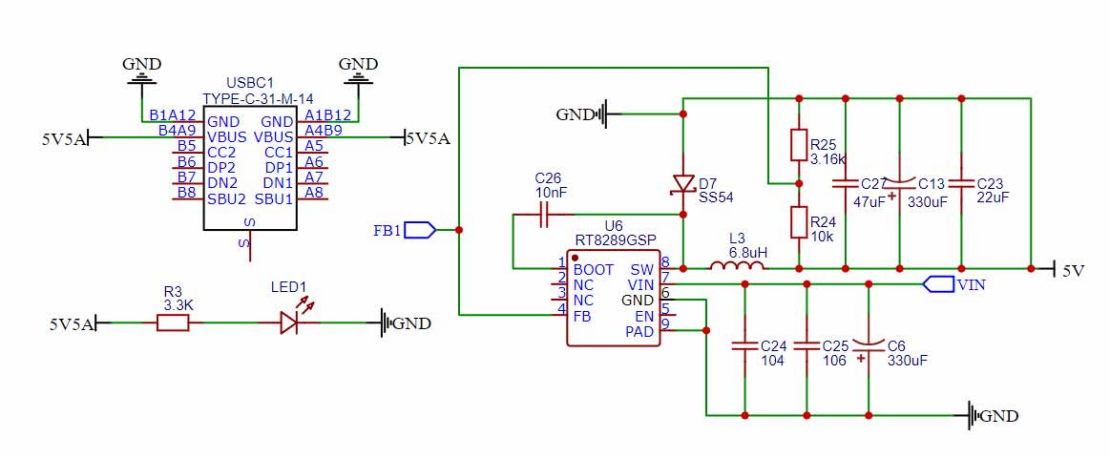

图4-3树莓派5V电路

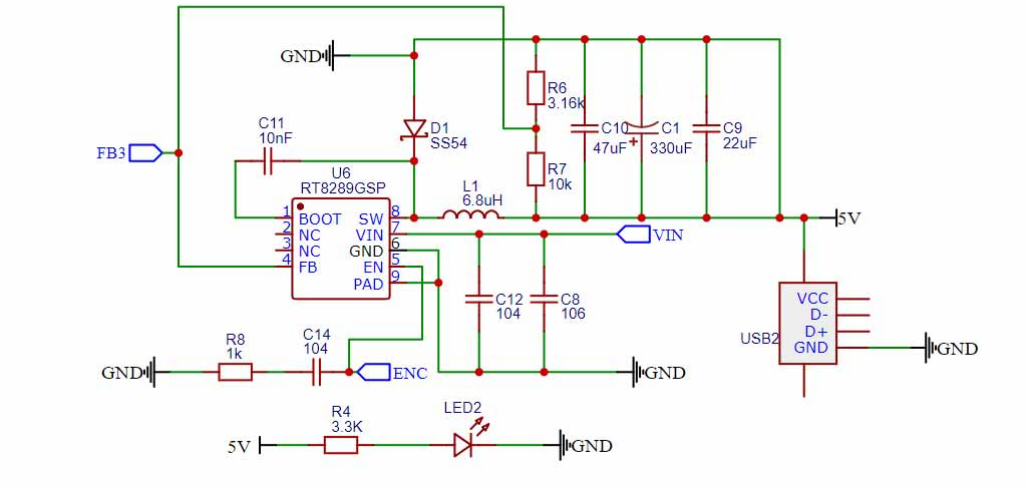

图 4-4 USB 5V 电路

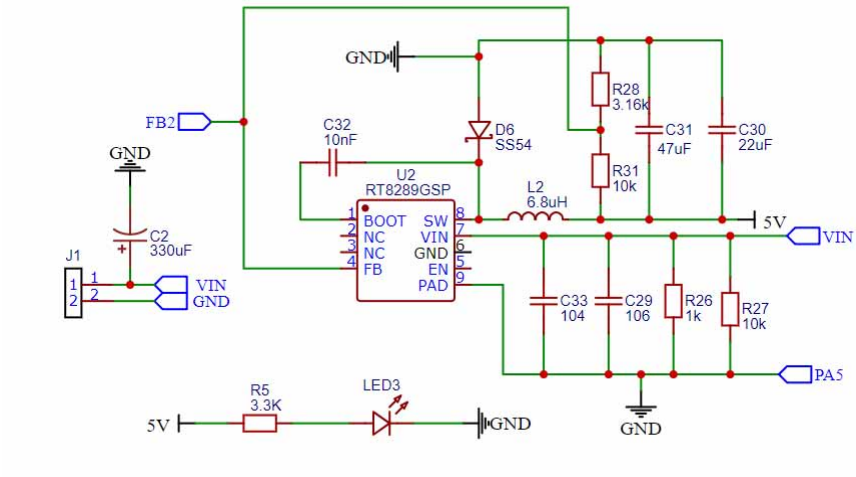

图 4-5 VCU 5V 电路

4.2.4 树莓派外围电路设计

要想实现自动驾驶需要对树莓派外围电路进行分析，由于激光雷达扫描的是周围一圈的环境，是一个平面，摄像头扫描的是前方的环境情况，再通过 STM32 使用 CAN 总线与

树莓派进行数据传输，不断的把采集到的数据传输过去，其中 MMA7361 采集到的数据可以与上面采集的数据结合，这样就可以扫描出一个 3D 环境，最后通过电机控制驱动车辆完成自动驾驶。树莓派外围整体硬件结构如图 4-6 所示。

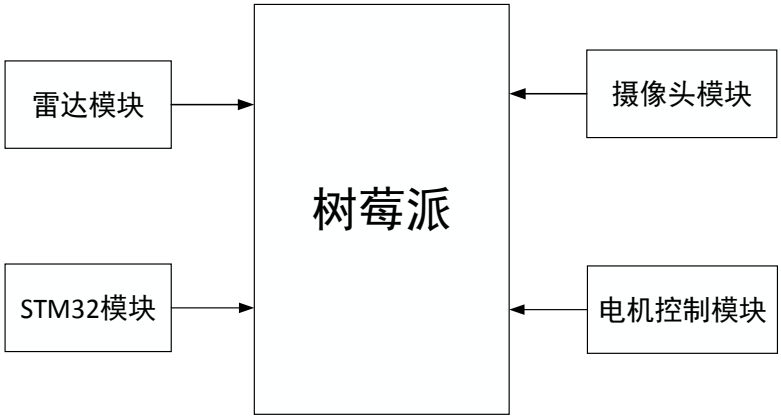

图 4-6 树莓派整体硬件结构图

由于 STM32 对普遍的传感器进行 CAN 总线的接收和发送比较方便，在传输时也不容易出错，对于雷达和摄像头模块达不到那么高的精度和要求，树莓派则可以，所以雷达模块和摄像头模块通过 USB 口接到树莓派上面；STM32 模块和电机控制模块则是通过一个 USB 转 CAN 的模块，树莓派端接 USB，CAN 接口接到 STM32 模块和电机控制模块，这样树莓派和 STM32 就能分布式进行数据采集和传输。

使用雷达模块和摄像头模块进行工作时要构建一个封闭环境，模拟路况并设置障碍物让激光雷达和摄像头分别对环境进行扫描，雷达扫描的结果如图 4-7 所示，摄像头扫描的结果如图 4-8 所示。

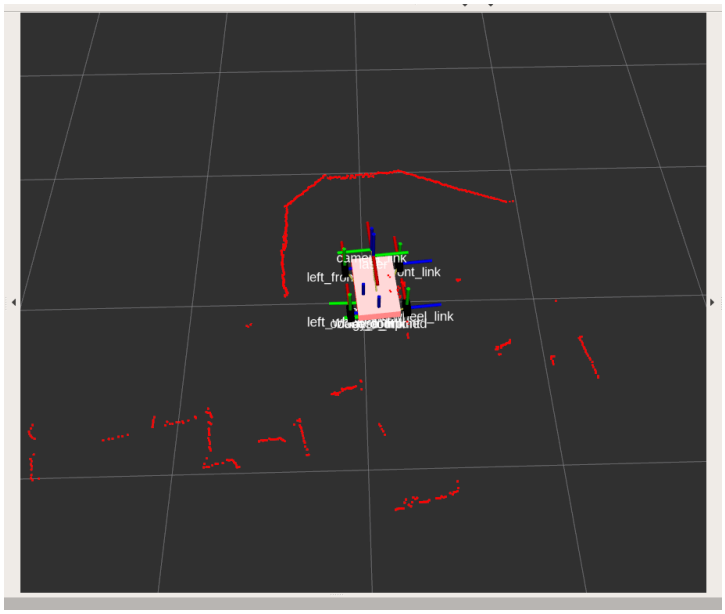

图 4-7 雷达扫描示意图

图 4-8 摄像头扫描示意图

图 4-7 中由于激光雷达是 360 度扫描的，车辆前方红色部分就是搭建一个密闭环境的障碍物，后方则是没有搭环境，在后方可以看出红色部分就是一些零碎的障碍物，可以得出激光雷达扫描的全面性，但是它扫描不到前方的小型障碍物这也是它的局限所在；而图 4-8 则是摄像头扫描的结果，虽然它扫描不了那么多方位，只能扫描前方的物体，但是小型的障碍物就明显的扫描了出来，然后与激光雷达进行结合，就能扫描出 3D 效果的环境，为自动驾驶提供了先决条件。

4.3 车载控制器的软件设计

4.3.1 软件架构总体设计

在设计时考虑到设计和开发的难度，所以采用模块化的思想将各个模块的功能要求进行模块化设计。为了减少软件过程中的负责性，提高软件的可靠性，采用模块化时要先明确 VCU 的软件要实现和怎么实现什么功能，这样才能准确设计出主程序和各个子程序的框架，之后根据模块化思想设计将其分为不同的子模块，对其分别进行设计调试，最后对整体进行调试和改进。按照其软件功能划为如图 4-9 的软件架构。

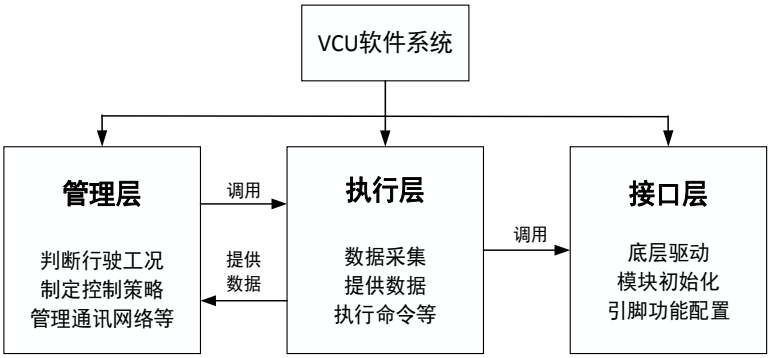

图 4-9 车载控制器软件架构图

管理层的主要功能是：分析和计算执行层采集到的车辆自动驾驶过程中的运行信息，根据计算结果判断车辆的行驶工况并制定合理的控制策略，最后调用执行层去执行对应的指令；在信息传输过程中要管理车载网络通讯各个通讯节点之间的报文传输，保证车辆可

以平稳安全的运行。

执行层的主要功能是：对数据进行采集，采集之后给管理层去进行处理；执行管理层下达的指令；调用接口层中的模块。

接口层的主要功能是：负责底层的驱动；实现 CAN 总线的通讯、初始化 A/D 等模块；配置不同 I/O 端口的引脚等。

4.3.2 建立控制车载的软件系统

车载操作系统使用 ROS 操作系统，它可以根据需要建立属于自己的模型，在此模型下可以对车辆进行路径规划、避障和导航等。使用之前要把虚拟机软件 VMware 在电脑端进行安装，然后还需要在虚拟机上再安装一个 Ubuntu 找适合于树莓派的 Ubuntu 镜像文件并进行下载，下载完要把镜像文件烧到 SD 卡中，这样就可以实现树莓派车辆的 ROS 与虚拟机 ROS 进行通信，因为它们之间的通信是在 ROS 工作环境下的，所以要在虚拟机上构建一个工作空间。

构建工作空间时首先要构建工作空间的文件夹，工作空间的文件夹名字可以自己定义，本文定义工作空间名字为 catkin_ws，文件夹的路径也可以自己进行定义，本文选择的是根目录下创建新的文件夹 catkin_ws，创建的时候最好使用命令行去创建；然后在 catkin_ws 文件夹下再创建一个 src 文件（这个文件一定要命名为 src）；下一步进入 src 文件夹执行如图 4-10 的指令生成 CMakerLists.txt 文件。

```
passoni@passoni:~/catkin_ws/src$ catkin_init_workspace
Creating symlink "/home/passoni/catkin_ws/src/CMakeLists.txt" pointing to "/opt/
ros/melodic/share/catkin/cmake/toplevel.cmake"
```

图4-10 生成CMakerLists.txt的指令图

构建之后要对工作空间进行编译。返回到 catkin_ws 文件夹，执行如图 4-11 所示的指令去编译工作空间，编译完成之后可以通过如图 4-12 所示的指令看到工作空间的文件夹里面多了 build 和 devel 文件夹，此时编译完成。

```
passoni@passoni:~/catkin_ws/src$ cd ..
passoni@passoni:~/catkin_ws$ catkin_make
```

图 4-11 编译空间指令图

```
passoni@passoni:~/catkin_ws$ ls
build  devel  src
```

图4-12查看工作空间文件夹指令图

最后设置其环境变量。通过图 4-13 所示的指令去设置环境变量，在设置好的环境变量后可以通过图 4-14 所示的指令去查看环境变量。

```
passoni@passoni:~/catkin_ws$ source devel/setup.bash
```

图4-13设置环境变量指令图

```
passoni@passoni:~/catkin_ws$ echo $ROS_PACKAGE_PATH
/home/passoni/catkin_ws/src:/opt/ros/melodic/share
```

图4-14查看环境变量指令图

设置完环境变量之后需要重启终端窗口才能生效，此时 ROS 的工作空间构建完成，建好之后就可以运行虚拟机然后对车辆进行操作和通信。对车辆进行初始化控制，启动车辆，在车辆行驶过程中会对环境进行判断，随后会自行做出下一步行驶的决定，直到完成行驶，详细代码见附录。下图 4-15 为车辆控制流程图。

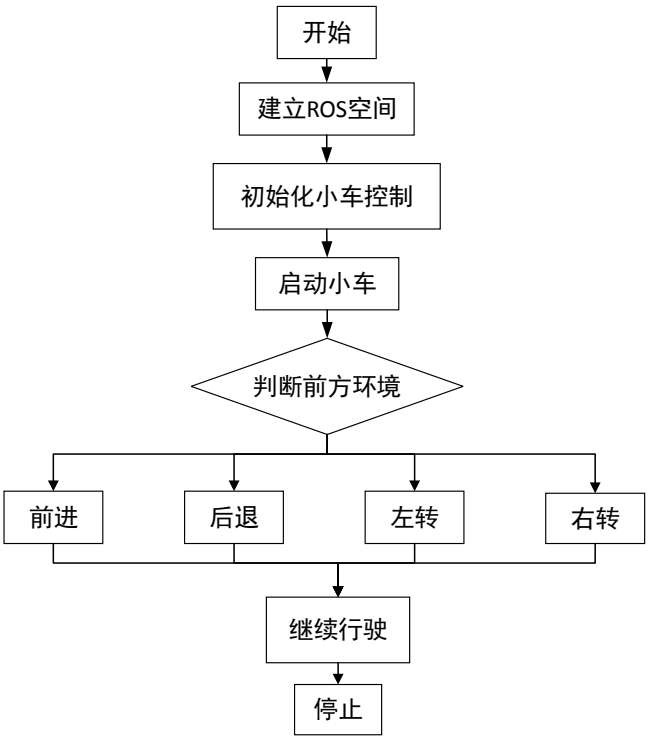

图4-15 车辆控制流程图

4.4 车载自动驾驶的设计与实现

4.4.1 自动驾驶车辆数据采集

在对软硬件进行设计之后，就可以进行自动驾驶车辆的数据采集，其中软件系统主要就是负责接收实时位置信息、航向信息、角度速度等信息，把接收到的信息进行算法控制并输出，与此同时还负责和上位机进行通信，把采集到的数据信息发送到上位机中并进行显示，上位机处理后会发出指令，然后对发出的指令进行执行。

自动驾驶车辆的数据包括了很多种，数据采集过程中主要对数据内容、通信协议进行采集和设定。数据内容包括车载的电池、电机、电控等信息，并可以通过这些信息监测车辆的运行状态；还包括雷达、摄像头的数据，它们通过扫描周围的 3D 环境对车辆的行驶路径进行解析，遇到障碍物时也可以进行自主避障。通信协议则还是按照上文提及的 CAN 总线的通信协议，把激光雷达、摄像头扫描的环境信息以及车辆实时运行情况的各个电控单元信息通过 CAN 总线传输到 VCU 上，VCU 在经过解析处理控制车辆运行。

4.4.2 自动驾驶车辆定位解析

车辆的实时位置信息在自动驾驶中非常重要，它可以实时了解到车辆的行驶轨是否按

照设定的轨迹运行，要保证运行过程中车辆的安全以及对道路正确判断，所以实时得到车辆的位置信息可以在第一时间将车辆调整到正确的路径上面。

在车辆行驶之前要对其行驶路径进行设置，给系统上电后，开始对车辆参数、状态、起始点进行设置，主控系统会接收到设置的数据，并把接收的设置参数信息存储，把车辆启动的点设置为起始点并命名为 A 点，通过解算得到 A 点的直角坐标，算出 A 的直角坐标，随后把 A 的信息进行存储，行驶一段距离后设置为 B 点，通过计算得到 B 点的直角坐标，也要进行存储。然后开启自动导航，实时对车辆的位置信息进行采集，将接收到的 RTK 定位数据进行分析，通过公式计算出当时的经纬度，由于经纬度是地理坐标，需要经过公式换算变成直角坐标，把坐标信息发送到上位机进行显示，显示出实时车辆的坐标；随后再根据采集到的角度传感器信息对车轮的转角进行计算，通过对车辆位置的实时追踪在根据采集到的航向角和车轮转角计算出车辆的航向偏差值，对航向偏差值进行 PID 控制算法解算，把结果送到驱动液压控制阀，控制车辆按预定的路径进行航向修正，保证车辆根据预设的路线直线行驶。

4.4.3 自动驾驶车辆行走路径分析

开启车辆的电源开关，让车辆在规定路线上行驶，行驶过程中定位模块会接收发来的实时位置信息。定位模块分为基站和移动站，基站是固定的。车辆的定位数据通过天线接收并将原始的数据解析之后通过电台发送给移动站，移动站固定在车辆上，随车辆移动而移动。同时也接收来自基站的信息，基站要与移动站保持一定的距离，它们之间通过天线进行数据接收和发送，移动站最终要通过解算 RTK 数据然后将其经过串口发送到上位机进行显示。

在行驶过程中难免会出现偏离路线的情况，车辆的航姿模块就是对车辆的行走方向以平面直角坐标系为基准纠正车轮需要转动的角度偏差。航姿模块数据通过串口采集并在电脑端显示，将波特率设置为 115200，输出的一帧数据包括 70 个字节，其中一个字节表示 8 位数据；数据以及其组成为：帧头 FA，FF（2 字节）+数据类型 MTData2（0X36）+数据区的 byte 数 LEN（1 字节）+数据 DATA（65 个字节）+校验 CHECKSUM（1 字节）。其中 4-8 字节数为计数器包（37 20 30 04 3E），9-23 字节数包括横滚角、俯仰角、航向角（37 22 34 11 42 2D 5A 1C 39 C4 A1 5F 3E EB B9），24-38 字节数为加速度数据（D8 3C 22 0D C0 31 87 10 41 E6 A0 B4 42 1F DD），39-53 字节数为角速度数据（2F 81 20 OF 3C C6 2D 0E 38 D1 1B ED 3A 10 78），54-68 字节数为磁场强度数据（3D C2 20 0C 41 D7 73 AC 40 21 E2 D3 BF 25 AD），经过对这些数据的处理计算，得到车辆的航姿数据。

角度传感器主要采集车辆的转角信息，根据采集到的转角信息进行修正，从而提高车辆的行驶准确度。编码器作为媒介把转角信息发送到上位机当中，编码器模式包括分为主动和被动模式，应用时使用主动模式，其数据长度为 16 位，采用的则是 ASCII 码字符编

码形式，格式为 YAB>+-DATA，Y 为前导字母，>为分隔符，+-代表转角的方向，DATA 则为数据位，以十位 ASCII 格式表示，范围为-9,999,999,999~+9,999,999,999，最后一位用 0D 表示回车符。在被动模式时，上位机和编码器之前是应答模式，过程中上位机会向编码器发送询问指令，指令包括 4 位数据，用 16 位 ASCII 码表示，格式为 D+AB，AB 表示的是编码器的地址范围（0~99）。

将采集到的航姿和角度等数据，由上位机接收指令并对其进行分析处理，处理之后需要转向控制系统驱动车辆的车轮转向，它是自动驾驶系统的核心，主要负责接收来自控制器的信号并输出控制信息，将 AD 模块输出的模拟信息的功率进行放大，最后通过控制液压阀来控制车轮的转向。根据采集的数据使用 PID 算法进行解算并输出控制值，最后由 AD 模块输出，由于 AD 转换器参考电压是 3.3V，而输出电压最高是 8V，所以在进行 AD 转换之前要对输出电压进行三倍缩小，在经过 AD 转换，转换之后在对电压进行三倍放大，这样就能保持之前的电压控制值，最后再把功率进行放大，输出到液压控制阀来驱动车辆的车轮转向。

4.5 本章小结

本章主要分析研究车辆软硬件的设计和实现，首先对车辆要完成的功能进行了描述，根据车辆要完成的功能进行设计。然后对 VCU 的主控芯片进行了选型，选择了 STM32 和树莓派共同作为硬件控制平台，分析芯片的特点和对应引脚设置，对外围电路进行了设计还对自动驾驶进行研究，分析其对应的传感器、雷达、惯导。硬件设计完成后对软件架构也进行了设计，使用 ROS 系统实现树莓派和车辆之间的通信。对自动驾驶车辆进行数据采集，将采集的信息发送到上位机，上位机处理之后再给下位机发送指令去执行，行驶过程中计算车辆实时的坐标情况并对车辆进行实时的定位监测，一旦车辆偏离轨道上位机接收到信息，就开始对车辆的航向进行修正，使车辆能够一直按照设定的路线进行自动驾驶。

第 5 章 总结与展望

5.1 总结

本文主要研究了车载控制器平台的实现方法，并在搭建的平台上实现自动驾驶的功能和对车载进行能量优化，主要的具体工作有：

（1）在设计实验的前期，查阅了大量的文献资料，对本文所研究的背景意义、自动驾驶的国内外研究现状和 VCU 控制平台进行了调查研究，根据纯电动汽车的特征确定主控单元的设计框架，明确了研究方向和研究内容，为下文研究自动驾驶平台控制器奠定了基础。

（2）根据主控单元的设计和方案，对传感器和 VCU 的之间通信方式进行选择，最终选用了 CAN 总线进行通讯。研究传感器与 CAN 总线进行通讯时，选择温度传感器、加速度传感器、霍尔传感器，分析其工作特性和工作原理，并对其安全的阈值进行设定，使车辆可以在规定的安全阈值内进行行驶，一旦超过阈值就会显示异常，这样就可以确保车辆的安全，避免发生事故。在进行通讯的时候在 SAEJ1939 协议的基础上建立与 VCU 通讯的通讯协议，设定通讯协议的数据格式和功能码，在进行传输时严格按照制定的协议有序的进行传输，保证了数据的实时性和可靠性。

（3）根据车辆的耗电情况，为了节省电量和增加续航里程对车辆进行能量管理优化策略。由于单一工况研究时容易出现偶然情况，不能保证结果的信服力，所以选择了常用于进行比较的 NEDC 和 WLTC 工况进行对比，根据它们行驶工况的数据对其行驶工况进行构建。车辆的行驶工况的构建需要在模拟的环境下对行驶片段进行提取，对不同速度范围的运动片段进行分类，在 K-means 聚类算法下对片段整合优化，得到最后的行驶工况曲线，选择了动态优化算法对电池剩余电量进行能量管理优化，设定车辆初始电量值，把车辆的电机功率情况转换成 SOC 值以此来判断 SOC 值的变化量，通过对不使用算法和使用算法 SOC 值进行比较，比较 NEDC 和 WLTC 剩余电量值的情况，证明了车辆能量管理优化策略可以成功实现。

（4）根据自动驾驶的要求对车辆软硬件进行设计，明确了车辆使用的主控芯片是树莓派和 STM32 联合构成 VCU 平台，对树莓派和 STM32 的软硬件特性进行分析，各个部分的电源电路都在 5V 电源下工作，明确了自动驾驶的技术架构，使用雷达、摄像头和角度传感器对周围 3D 环境构建。考虑到软件设计的难度采用模块化，不同模块实现 VCU 不同软件功能，构建的自动驾驶控制平台要在虚拟机系统下进行执行，并在 ROS 工作环境下使树莓派车辆的 ROS 与虚拟机 ROS 进行通信，对自动驾驶车辆的数据进行采集并基于 CAN 总线保持上位机与下位机的通信，实时获取车辆的位置坐标，观察在复杂环境下行驶的轨迹，一旦出现了轨迹偏差，及时对航向值进行修正，确保其可以在正确的路线上自动

驾驶。

5.2 展望

虽然本文完成了对自动驾驶平台控制器的设计但是还有很多不足之处需要研究和后续改进：

（1）本文中只完成了部分的传感器通讯实验，还有很多传感器被使用，虽然对通信协议进行了设计，也达到了传输的要求，还应该对更多的传感器的通信协议进行编写，全面性还尚未达到要求。

（2）本文在进行行驶工况构建的时候，对工况行驶片段的提取还不是非常完善，使得到的工况行驶片段具有一些偶然性，今后要对行驶片段的分类和提取进行完善；通过构建的工况行驶片段进行能量优化时，只使用了一个算法进行比较，在下面的工作中，要学习多种算法，并把其应用到能量优化的实际问题中，在今后还需要多学习，多尝试。

（3）本文中对车载软硬件进行了设计，目的是完成车辆的自动驾驶，其中在用雷达及其它设备进行周围环境构造时，构造的环境范围不够大，在后面的实验中要尝试让环境扩大，验证其扫描的结果和对周围环境构造的情况；在进行自动驾驶过程中，坐标设置时车辆行驶的距离较短，在后面的使用中要让车辆多行驶一段距离；再判断其在设置路线情况下自动驾驶和进行避障的情况。

参考文献

[1] 陈秀娟. 全国汽车保有量达 2.81 亿辆 [J]. 汽车观察, 2021, 1(7).

[2] 杨威. 自动驾驶车辆的动作规划算法研究 [D]. 北京 :北方工业大学, 2021.

[3] 贺宜, 杨鑫炜, 吴兵等. 中美交通事故数据统计方法比较研究 [J]. 交通信息与安全, 2018, 36(1): 1-9.

[4] 王科俊, 赵彦东, 邢向磊. 深度学习在无人驾驶汽车领域应用的研究进展 [J]. 智能系统学报, 2018, 13(1): 55-69.

[5] GUO J, LUO Y, LI K. Adaptive coordinated collision avoidance control of autonomous ground vehicles [J]. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 2018, 232(9): 1120-33.

[6] HUANG Y, WANG H, KHAJEPOUR A, et al. A novel local motion planning framework for autonomous vehicles based on resistance network and model predictive control [J]. IEEE Transactions on Vehicular Technology, 2019, 69(1): 55-66.

[7] FUNKE J, BROWN M, ERLIEN S M, et al. Collision avoidance and stabilization for autonomous vehicles in emergency scenarios [J]. IEEE Transactions on Control Systems Technology, 2016, 25(4): 1204-16.

[8] 于钊, 韩衍东, 刘元治等. 新型整车控制器关键技术分析 [J]. 汽车文摘, 2021, 2(5).

[9] TSUGAWA S, YATABE T, HIROSE T, et al. An automobile with artificial intelligence[C]. proceedings of the Proceedings of the 6th international joint conference on Artificial intelligence-Volume 2, 1979: 893-895.

[10] LI S G, SHARKH S M, WALSH F C, et al. Energy and battery management of a plug-in series hybrid electric vehicle using fuzzy logic [J]. IEEE Transactions on Vehicular Technology, 2011, 60(8): 3571-85.

[11] VELENIS E, KATZOURAKIS D, FRAZZOLI E, et al. Steady-state drifting stabilization of RWD vehicles [J]. Control Engineering Practice, 2011, 19(11): 1363-76.

[12] THRUN S, MONTEMERLO M, DAHLKAMP H, et al. Stanley: The robot that won the DARPA Grand Challenge [J]. Journal of field Robotics, 2006, 23(9): 661-92.

[13] 熊璐, 杨兴, 卓桂荣等. 无人驾驶车辆的运动控制发展现状综述 [J]. 机械工程学报, 2020, 56(10): 127-43.

[14] 章江. 开启无人驾驶商业化时代 [J]. 汽车与配件, 2019, (11): 44-6.

[15] 连志鹏. 自动驾驶农机转向控制研究 [D]. 上海: 上海工程技术大学, 2016.

[16] 明鑫朗. 基于 CAN 总线的拖拉机控制系统平台研究 [D]. 武汉: 湖北工业大学, 2015.

[17] 王嘉明. 纯电动汽车整车控制器与控制策略研究 [D]. 哈尔滨: 哈尔滨理工大学, 2020.

[18] 罗润. 纯电动汽车整车控制策略及控制器研究 [D]. 太原: 太原理工大学, 2019.

[19] RAZI M, MURGOVSKI N, MCKELVEY T, et al. Design and comparative analyses of optimal feedback controllers for hybrid electric vehicles [J]. IEEE Transactions on Vehicular Technology, 2021, 70(4): 2979-93.

[20] CHEN Y, FENG G, WU S, et al. A new hybrid model predictive controller design for adaptive cruise of autonomous electric vehicles [J]. Journal of Advanced Transportation, 2021, 2021: 1-25.

[21] 刘永山. 纯电动汽车整车控制器开发及控制策略研究 [D]. 武汉: 武汉理工大学, 2014.

[22] RUGGIERI R, RUGGERI M, VINCI G, et al. Electric mobility in a smart city: European overview [J].

Energies, 2021, 14(2): 315.

[23] WANG H, SUN T, ZHOU X, et al. Research on the Electric Vehicle Control System [J]. International Journal of u-and e-Service, Science and Technology, 2015, 8(8): 103-10.

[24] MECKLING J, NAHM J. The politics of technology bans: Industrial policy competition and green goals for the auto industry [J]. Energy Policy, 2019, 126: 470-9.

[25] HONG J, WANG Z, LIU P, et al. Research on parameter matching and control strategy of load isolation pure electric driving vehicles [J]. Energy Procedia, 2017, 105: 4040-5.

[26] ZHAO Y, HOU J, WANG C, et al. Design of vehicle control research and development platform for a pure electric vehicle [J]. Advances in Mechanical Engineering, 2019, 11(2): 1687814019826427.

[27] HONGYU W, YONG F, YUNQIAN Q, et al. Study on High Voltage power up down control strategy of pure electric vehicles[C]. proceedings of the 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2015: 624-628.

[28] 张家旭, 张振兆, 赵健等. 采用极点配置的自动驾驶汽车换道路径规划与跟踪控制 [J]. 西安交通大学学报, 2020, 54(10): 160-7.

[29] 陈慧, 徐建波. 智能汽车技术发展趋势 [J]. 中国集成电路, 2014, 23(11): 64-70.

[30] 刘海锋. 无人驾驶汽车技术发展现状和前景展望 [J]. 山东工业技术, 2018, (14): 71.

[31] 杜卓洋. 无人驾驶车辆轨迹规划算法研究 [D]. 杭州: 浙江大学, 2019.

[32] 张亚娇. 农机自动驾驶监控终端关键技术研究与系统开发 [D]. 广州: 华南农业大学, 2016.

[33] 张雁. 水田环境下的无人农机自动驾驶与作业系统研究 [D]. 上海: 上海交通大学, 2019.

[34] 田军辉. 纯电动客车整车控制器硬件在环测试系统开发及驱动控制策略研究 [D]. 长春: 吉林大学, 2013.

[35] 朱晓琪. 纯电动汽车整车控制器开发 [D]. 长春: 吉林大学, 2015.

[36] 张志鹏. 插电式单轴并联混合动力汽车 DCT 换挡控制技术研究 [D]. 北京: 北京理工大学, 2017.

[37] 许同盟. 纯电动汽车整车控制器设计与控制策略研究 [D]. 济南: 山东大学, 2019.

[38] 王炜, 杨新苗, 陈学武. 城市公共交通系统规划方法与管理技术 [M]. 科学出版社, 2002, 1-3.

[39] 闫龙涛. 基于 XC2268N 的纯电动汽车整车控制器设计 [D]. 天津: 天津大学, 2012.

[40] 宋晓妍, 王群淞, 刘晶郁等. 汽车驾驶模拟器控制系统的设计与实现 [J]. 计算机测量与控制, 2020, 28(2): 82-7.

[41] 周美兰, 高肇明, 吴晓刚等. 五种 PWM 方式对直流无刷电机系统换相转矩脉动的影响 [J]. 电机与控制学报, 2013, 17(7): 15-21.

[42] 张庆, 熊会元, 于丽敏. 纯电动汽车转矩控制策略及仿真 [J]. 计算机仿真, 2014, 31(12): 142-6.

[43] 费为伟. 纯电动客车整车控制策略开发与硬件在环测试 [D]. 西安: 长安大学, 2021.

[44]董小辉, 胡锦波, 黄浩等. 基于 SAE J1939 的商用车智能车载终端系统设计 [J]. 计算机测量与控制, 2019, 27(7): 209-13.

[45] 王荣. 基于 CAN 总线的智能车辆数据采集与处理 [D]. 重庆: 重庆交通大学, 2016.

[46] 陈诗桓. 车辆控制系统设计与开发中 CAN 总线通信实施方法 [J]. 现代制造技术与装备, 2022, 58(01): 49-51.

[47] 常智. 基于 CAN 总线在车辆通讯协议中的研究和应用 [D]. 太原: 中北大学, 2019.

[48] 张玲娜. 基于霍尔传感器的电机转速测量与调速系统设计和研究 [D]. 西安: 西安石油大学, 2016.

[49] 王星琦. 考虑电池寿命的插电式混合动力物流车能量管理优化研究 [D]. 长春: 吉林大学, 2020.

[50] 洪日. 增程式电动物流车能量管理策略研究与性能优化 [D]. 长春: 吉林大学, 2021.

[51] 臧怀泉, 张琦, 强鹏辉等. 插电式混合动力汽车能量优化管理策略研究 [J]. 燕山大学学报, 2020, 44(4): 388-96.

[52] 李日业. 基于 NEDC 循环工况纯电动车能量回收的应用研究 [J]. 汽车实用技术, 2021, 46(24): 1-4.

[53] 王永广, 付江涛, 李鹏飞等. WLTC 和 CLTC 循环测试工况对比分析 [J]. 中国汽车, 2020(11): 14-21.

[54] PELKMANS L, DEBAL P. Comparison of on-road emissions with emissions measured on chassis dynamometer test cycles [J]. Transportation Research Part D: Transport and Environment, 2006, 11(4): 233-41.

[55] 段宇帅. 基于主成分分析与 K-means 聚类的汽车行驶工况构建 [J]. 软件导刊, 2022,21(05):175-180.

[56] 周溪召, 刘启超. 基于 K-means 聚类分析的汽车行驶工况构建 [J]. 物流科技, 2020, 43(11): 93-96.

[57] 蔡锷, 李阳阳, 李春明等. 基于 K-均值聚类算法的西安市汽车行驶工况合成技术研究 [J]. 汽车技术, 2015, (8): 33-6.

[58]郭宁远. 基于工况预测的插电式混合动力汽车能量管理策略研究[D]. 昆明: 昆明理工大学, 2018.

[59] ZHU H, LI L, JIN M, et al. Real-time yaw rate prediction based on a non-linear model and feedback compensation for vehicle dynamics control [J]. Proceedings of the Institution of Mechanical Engineers, Part D: Journal of automobile engineering, 2013, 227(10): 1431-45.

[60] 于田雨. 双电机四驱汽车动力系统匹配设计和能量管理策略研究 [D]. 长春: 吉林大学, 2021.

[61] WANG C, GUAN Z, WANG W. Design of Remote Operated Vehicle Based on STM32[C] proceedings of the Proceedings of the 11th International Conference on Computer Engineering and Networks, 2022: 1208-1215.

[62] 耿兴华, 王克欣, 钱希兴等. 基于 STM32 的电机控制实验平台设计与实现[J]. 实验室科学, 2022, 25(3): 66-69, 74.

[63] 闫悦, 蒋祝鹏. 基于树莓派 4B 的智能物流小车 [J]. 智能制造, 2021, 29(6): 84.

[64] 刘天君, 常昊, 马准等. 基于树莓派的智能探测小车设计 [J]. 电子测试, 2021(01): 5-7+11.

[65] 刘增文. 浅谈汽车自动驾驶技术的原理及应用 [J]. 中国设备工程, 2022, (12): 209-11.

[66] 周厚波. 基于 ROS 的移动机器人同-异质传感器融合与路径规划研究 [D]. 重庆: 重庆大学, 2021.

[67] LI C. Technical Description of Self-driving Cars [J]. Academic Journal of Engineering and Technology Science, 2022, 5.0(6.0).

附录

```
初始化小车控制
        GPIO.setup(ENA, GPIO.OUT, initial=GPIO.HIGH)
        GPIO.setup(IN1, GPIO.OUT, initial=GPIO.LOW)
        GPIO.setup(IN2, GPIO.OUT, initial=GPIO.LOW)
        GPIO.setup(ENB, GPIO.OUT, initial=GPIO.HIGH)
        GPIO.setup(IN3, GPIO.OUT, initial=GPIO.LOW)
        GPIO.setup(IN4, GPIO.OUT, initial=GPIO.LOW)
        GPIO.setup(ServoUpDownPin, GPIO.OUT)
        GPIO.setup(ServoLeftRightPin, GPIO.OUT)
        GPIO.setup(buzzer,GPIO.OUT,initial=GPIO.HIGH)
        GPIO.setup(lightPin, GPIO.LOW,initial=GPIO.LOW)
        # 设置 pwm 引脚和频率为 2000hz
        pwm_ENA = GPIO.PWM(ENA, 2000)
        pwm_ENB = GPIO.PWM(ENB, 2000)
        pwm_UpDownServo = GPIO.PWM(ServoUpDownPin, 50)
        pwm_LeftRightServo = GPIO.PWM(ServoLeftRightPin, 50)
        pwm_ENA.start(0)
        pwm_ENB.start(0)
        pwm_UpDownServo.start(0)
        pwm_LeftRightServo.start(0)
        初始化小车控制结束
    def getCarStatus(self):
        return ("speend:%d,light:%s"%(CarSpeedControl,self.lightOpen))
# 小车前进
    def run(self):
        GPIO.output(IN1, GPIO.HIGH)
        GPIO.output(IN2, GPIO.LOW)
        GPIO.output(IN3, GPIO.HIGH)
        GPIO.output(IN4, GPIO.LOW)
        pwm_ENA.ChangeDutyCyclCarSpeedControle()
        pwm_ENB.ChangeDutyCycle(CarSpeedControl)
        time.sleep(1)
    # 小车后退
```

```
def back(self):
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.HIGH)
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.HIGH)
    pwm_ENA.ChangeDutyCycle(CarSpeedControl)
    pwm_ENB.ChangeDutyCycle(CarSpeedControl)
    time.sleep(1)
# 小车左转
def left(self):
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)
    leftSpeed=5
    if CarSpeedControl-10>0:
        leftSpeed=CarSpeedControl-10
    pwm_ENA.ChangeDutyCycle(leftSpeed)
    pwm_ENB.ChangeDutyCycle(CarSpeedControl)
    time.sleep(1)
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.LOW)
# 小车右转
def right(self):
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)
    rightSpeed = 5
    if CarSpeedControl - 10 > 0:
        rightSpeed = CarSpeedControl - 10
    pwm_ENA.ChangeDutyCycle(CarSpeedControl)
    pwm_ENB.ChangeDutyCycle(rightSpeed)
```

```
            time.sleep(2)
            GPIO.output(IN1, GPIO.LOW)
            GPIO.output(IN2, GPIO.LOW)
            GPIO.output(IN3, GPIO.LOW)
            GPIO.output(IN4, GPIO.LOW)
# 小车停止
    def brake(self):
            GPIO.output(IN1, GPIO.LOW)
            GPIO.output(IN2, GPIO.LOW)
            GPIO.output(IN3, GPIO.LOW)
            GPIO.output(IN4, GPIO.LOW)
```