移动机器人运动控制研究

摘要

移动机器人是目前机器人领域研究的重点之一，它有着巨大的应用潜力。运动控制是移动机器人系统的核心部分，对机器人的平稳运行起着至关重要的作用。目前，随着应用的深入和技术的发展，对运动控制系统提出了越来越高的要求，因此研究并设计出精确、实时、开放的运动控制系统具有重要的意义。

本文以三轮轮式机器人(后两驱动轮，前一万向轮)作为研究对象，建立运动学模型，然后按照控制系统的要求，选择高性能 ARM9 微处理器 S3C2410 作为处理器核心设计移动机器人系统平台，系统扩展 Flash、SDRAM 和键盘、LCD，Internet 等外围功能器件，选取 L298 作为电机驱动芯片，采用内电流环、外速度环的控制策略和积分分离的 PID 控制算法，文中详细介绍了各个硬件芯片型号的选取、硬件的性能参数、硬件引脚和寄存器参数，设计了各个硬件之间的接口电路。

系统的软件部分采用嵌入式 Linux 作为操作系统，移植了引导程序、裁剪后的 Linux 系统和文件系统，在此基础上设计了直流电机设备驱动程序、电流环控制子程序、速度环控制子程序和改进的积分分离 PID 控制算法等，实现机器人的精确、实时的运动控制。

实验结果表明，本系统能够较好的实现预期的功能，具有较好的稳定性和应用前景。

关键词：移动机器人，运动控制，ARM9，Linux，电机控制，设备驱动

RESEARCH ON MOTION CONTROL OF MOBILE ROBOT

ABSTRACT

Recently, mobile robot is one of important fields of robot research. It has a wide range of applications. The motion control is hard core of mobile robot system and has a vital influence on smooth running of robot. Now, it asks a higher and higher request for motion control system while developing of application and technology. So, it has a important sense to study and devise a precise、real-time and opening motion control system.

The paper choose 3-wheel robot (two deriving wheel on back and one universal wheel on front) as research object and study kinematics model. After that, we choose high-performance ARM9 processor S3C2410 as central processor to devise system platform based on demand. The Flash、SDRAM、keyboard、LCD、Internet and other peripheral function devices are added to the system. The motion control part choose L298 as deriving chip for DC motor and select strategy of double-loop (inside current loop and outside velocity loop) and PID algorithm (Integral separation). The paper also describe selecting of the hardware, the performance parameters、hardware pins、register parameters and design the interface between various hardware.

Embedded Linux is adopted as operating system and boot program、reduced Linux and file system is transplanted to the target board. DC motor

device driver program、current loop control program、velocity loop control program and PID algorithm (integral separation) is designed based on the target board. It achieve the precise、real-time control on motion.

Experiments proved that the system can achieve the desired functions and have good stability and prospect.

Keywords: Mobile robot, Motion control, ARM9, Linux, Motor control, Device driver

北京化工大学位论文原创性声明

本人郑重声明： 所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者签名：吴浩 日期：2009年6月4日

关于论文使用授权的说明

学位论文作者完全了解北京化工大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京化工大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。

保密论文注释：本学位论文属于保密范围，在2年解密后适用本授权书。非保密论文注释：本学位论文不属于保密范围，适用本授权书。

作者签名：吴浩 日期：2009年6月4日

导师签名：闫星文 日期：2009年6月4日

第一章 绪论

1.1 课题研究的背景

机器人技术是一门跨学科的综合性技术，机器人的研究、开发和应用涉及多刚体动力学、机构学、机械设计、传感技术电气液压驱动、控制工程、智能控制、计算机科学技术、人工智能和仿生学等学科。早在20世纪60年代初，美国就制造出了第一台工业机器人，目前已经发展到了第三代—智能机器人。由于机技术和机器人的重要作用，各工业化国家近几年都把它的研究和开发列为国策，制订一系列规划，采取多方面的措施，以加快发展。许多发展中国家也纷纷建立技术开发中心，应用和生产机器人，掀起一个世界范围的机器人研制热潮[1]。

以嵌入式计算机为技术核心的嵌入式系统是继网络技术之后，又一个 IT 领域新的技术发展方向。由于嵌入式系统具有体积小、性能强、功耗低、可靠性高以及面向行业具体应用等突出特征，目前已经广泛地应用于军事国防、消费电子、信息家电、网络通信、工业控制等各个领域。嵌入式的广泛应用可以说是无所不在。就我们周围的日常生活用品而言，各种电子手表、电话、手机、PDA、洗衣机、电视机、电饭锅、微波炉、空调器都有嵌入式系统的存在，如果说我们生活在一个充满嵌入式的世界，是毫不夸张的[2]。据统计，一般家用汽车的嵌入式计算机在24个以上，豪华汽车的在60个以上。据推测人们在2010年每天接触的嵌入式计算机有望达到800个以上。难怪美国汽车大亨福特公司的高级经理也曾宣称，“福特出售的‘计算能力’已超过了IBM”，由此可见嵌入式计算机工业的应用规模、应用深度和应用广度。

1.2 移动机器人发展概况

移动机器人是机器人学中的一个重要分支，是一种能够通过传感器感知环境和自身状态，实现在有障碍物的环境中，面向目标自主运动，从而完成一定功能的机器人系统。移动机器人研究始于 20 世纪五、六十年代，在这个阶段，人们首先以室内环境为背景，开始了移动机器人的探索性研究。例如:斯坦福研究院(SRI)的美国 Stanford 研究所的Nils Nilssen和Charles Rosen等人，在1966年至1972年研制出取名为 Shakey 的自主移动机器人[3]，如图 1-1 所示，此时并没有针对具体的任务或应用背景，而是面向室内结构化环境的基本技术预研。

进入20世纪八十年代后，在计算机技术、机器人技术和人工智能理论的推动下，国际上广泛开展了对移动机器人的研究。一方面，人们把目光集中于面向实际应用背景的室内移动机器人研制;另一方面，人们把室内移动机器人的研究拓宽到室外移动机器人的研究。对于室内移动机器人，比如：办公机器人、图书馆机器人、医院机器

人，家庭服务机器人等的研究都取得了很大的进展。图 1-2 所示为 TRC 公司生产的 Helpmate[3]医用轮式移动机器人[4]。该机器人由行走部分、行驶控制器及大量的传感器组成。可在医院中以 0.7 米/秒左右的速度自主运动，往返于病房与医务室之间，分发药品和换洗的衣物等。

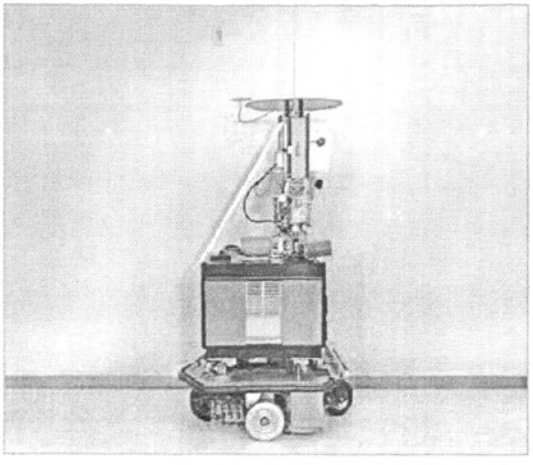

图 1-1 Shakey
Fig.1-1 Shakey

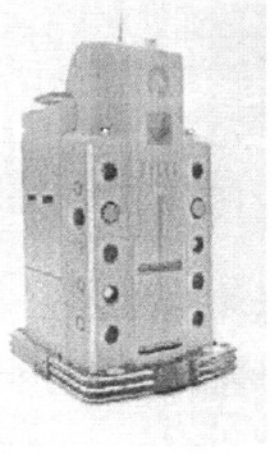

图 1-2 Helpmate
Fig. 1-2 Helpmate

进入 20 世纪 90 年代以研制高水平的环境信息传感器和信息处理技术、高适应性的移动机器人技术、真实环境下的规划技术为标志，开始了移动机器人更高层次的研究。空间机器人(Space Mobile robot)、远程操作机器人(Remote Operation Mobile Robot)、自主车辆(Autonomous Land Vehicle) 这些有着不同应用背景和功能的室外移动机器人纷纷亮相。最具代表性的当数美国火星探测器“勇气”号和“机遇”号，这对孪生兄弟先后经历了半年多，数以亿公里的星际旅行后，分别于 2004 年 1 月 4 日和 1 月 24 日成功着陆在目的地。“勇气”号和“机遇”号火星探测器是两个具有 6 个独立悬挂车轮的移动机器人，能够在崎岖的火星表面坚实地前进。图 1-3、1-4 分别是“勇气”号和“机遇”号的照片。

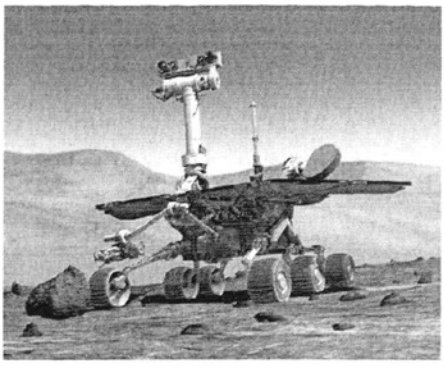

图 1-3“勇气”号探测车
Fig. 1-3 “Spirit” Mars Rover

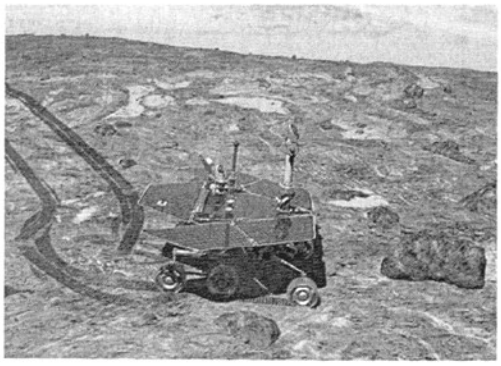

图 1-4 “机遇”号探测车
Fig. 1-4 “Opportunity” Mars Rover

我国移动机器人的研究相对而言起步较晚，但也取得了很大的进展。从 20 世纪 90 年代开始，许多科学院所和高等院校开展移动机器人的研究，并取得重要进展。如中国科学院自动化研究所研制的 CASIA-I；由国防科技大学、南京理工大学、浙江大学、清华大学、北京理工大学联合研制的国家五年计划 ATB-1 室外移动机器人系统[4]；清华大学研制的智能车 THMR-V，能够实现结构化环境下的车道线自动跟踪，以及复杂环境下的道理避障、道路停障，最高车速达到 150km/h；国防科技大学研制 CITAVT-IV，除此之外吉林大学、北京航空航天大学等院校也开展了移动机器人的研究。

我国火星探测器“萤火一号”与俄罗斯“火卫-土壤”探测器将于 2009 年 10 月发射，中俄联合火星探测计划命名为“萤火”计划。随着我国探月工程和火星探测计划的进展，必将推动我国移动机器人快速发展[5-6]。

1.3 嵌入式系统

1.3.1 嵌入式系统的发展

根据国际电气和电子工程师协会(IEEE)的定义，嵌入式系统是“控制、监视或者辅助设备、机器和车间运行的装置。广义上讲，嵌入式系统被定义为：以应用为中心，以计算机技术为基础，软硬件可裁减，适合应用系统对功能、可靠性、成本、体积和功耗要求的专用的计算机系统。它是计算机应用的最普遍的形式，是将先进的计算机技术、半导体技术、电子技术和各个行业的具体应用相结合后的产物。

目前基于嵌入式的系统大致可分为 3 类：基于单片机的微控制器(Micro Controller Unit 简称 MCU)系统；工业化的单板机；以 32 位嵌入式处理器(MPU 或者 DSP)为核心的高级专用计算机系统[7-10]。

嵌入式微处理器是由通用 CPU 演变而得，处理器在 32 位以上，可以运行操作系统，性能高，在工作温度、抗电磁干扰、可靠性等方面一般都做了各种增强。但是因为采用多线程，资源管理复杂。嵌入式微处理器目前主要有 Intel，AMD 及其他 X86 兼容厂商生产的 X86 嵌入式 CPU，Motorola 的 68000，IBM 和 Motorola 的 Power PC，日立公司的 SH，SGI 公司的 MIPS，以及在嵌入式精简指令体系(Reduce Instruction Set Computer 简称 RISC)处理器市场占绝对优势的 ARM 系列等。

此外，随着电子设计自动化(Electronic Design Automation 简称 EDA)的推广和超大规模集成电路(Very Large Scale Integration 简称 VLSL)设计的普及化及半导体工艺的迅速发展，在一个硅片上实现一个更为复杂系统的时代已经来临，即片上系统(System on Chip 简称 Soc)。除个别无法集成的器件以外，整个嵌入式系统大部分均可集成到一块或几块芯片中去，应用系统电路板将变得很简洁，对于减小体积和功耗、提高可靠性非常有利。

嵌入式系统提供了一种灵活、高效和高性价比的解决方案。随着数字技术的发展和新的体积更小的控制芯片和功能更强大的操作系统的出现，嵌入式技术已经被广泛地应用于科学研究、军事技术、工业控制以及消费电子产品等方面[11]。

1.3.2 嵌入式操作系统的发展

在嵌入式应用中，为了使嵌入式开发更方便、快捷，就需要具备相应的管理存储器分配，中断处理、任务间通信和定时器响应，以及提供多任务处理等功能的稳定的、安全的软件模块集合，即嵌入式操作系统(Embedded Operation System 缩写 EOS)。

嵌入式操作系统负责系统的全部软件、硬件资源的分配、调度、控制、协调；它必须体现其所在系统的特征，能够通过加载/卸载某些模块来达到系统所要求的功能。为了对复杂的嵌入式软件系统进行合理控制，嵌入式操作系统越来越重要。

1981 年，Ready System 发展了世界上第一个商用嵌入式实时内核 VRTX32。20 世纪 90 年代，现代操作系统的设计思想开始渗入实时操作系统(Real Time Operation System 缩写 RTOS)领域，国内的一些公司和科研院校也开始涉足嵌入式实时操作系统，并开发出一些优秀的实时操作系统。20 世纪 90 年代中期，互联网的流行推动 RTOS 的发展，这个时候的代表作品有 VxWorks，WinCE，Linux 等。1991 年，Linux 开发出 Linux 操作系统并通过互联网广泛发行，当时 Linux 并不是为嵌入式系统设计的。但是自从它首次公开发表以来，由于其源码开放性和设备独立性以及良好的移植性等特点，开发人员根据需要将 Linux 系统进行相应的裁减和改进，它的应用范围越来越广泛。现在工作站、服务器都有应用，已经嵌入式系统中也越来越多的得到应用。

目前，常用的嵌入式操作系统主要可分为商用系统和开放系统。商用系统功能强大，辅助工具齐全，可以应用于许多不同的领域。例如 WindRiver 公司的 VxWorks，

Microsoft 公司的 Windows CE，3COM 公司的 Palm OS，Symbian 公司的 EPOC，中科院凯思集团的 Hopen 等。开放系统主要有 Linux 和 μ C/OS[12]，它们的源代码可以免费获得，降低了开发成本，在实际中也有着很多的应用。

1.4 运动控制

1.4.1 运动控制系统概述

运动控制系统是一种通过电机驱动的执行机构对电机转矩、转速和转角进行控制以实现预定运动轨迹目标的电气系统。典型的运动控制系统有机器人、数控机床、运输机械等[1]。运动控制技术是一门融计算机、电子、机械、控制和传感器等多门学科知识于一体的交叉技术，广泛应用于工业、军事、航空航天、能源勘探和医疗等各个领域，它是自动化领域的一项关键技术。

1.4.2 运动控制系统的分类

运动控制系统的种类很多，根据控制原理与应用的不同，可按下述几种方法分。

1) 按运动控制轨迹分类

按照执行机构在空间中运动轨迹目标的不同，可将运动控制器分为点位控制、直线控制和连续控制三种。

(1) 点位控制：这类控制系统要求执行机构以最快的速度从一点移动到另一点并准确定位。对于点与点之间的移动轨迹(路径与方向)并无严格要求，各坐标轴之间的运动并不相关。

(2)直线控制：这类控制系统不但要求执行机构实现点与点的准确定位，而且要求两点之间的运动轨迹是一条直线，并且需要对运动的速度进行控制。

(3) 连续控制：连续控制也称为轮廓控制，其特点是能够对两个或两个以上轴的位移和速度同时进行相关控制，使末端执行机构按一定的空间曲线轨迹运动。

2) 按伺服控制方式分类

运动控制系统采用的伺服控制方式很多，主要分为开环控制、闭环控制和半闭环控制三种类型，此外还有开环补偿型和半闭环补偿型等混合控制方式。

(1) 开环运动控制：这类运动控制系统中没有位移检测反馈装置，运动控制器的控制指令直接通过驱动装置控制电机的运转，通过机械传动系统转化成执行机构的位移。由于这种控制系统没有反馈校正，因而多采用步进电机作为控制对象，其控制结构简单，性能稳定，价格低廉，广泛应用于经济型数控系统中。

(2) 闭环运动控制：此类运动控制系统的检测装置安装在执行机构上，用以直接

获得执行机构的实际运行位置，并将其与系统的指令位置相比较，用差值进行控制。这种控制方式可以消除整个传动过程中的传动累计误差，具有很高的定位精度。但由于它将丝杠、螺母等大惯量环节放在闭环之内，系统稳定性受到影响，调试困难而且结构复杂、价格昂贵。

(3)半闭环运动控制：这类运动控制系统的位置检测元件安装在伺服电机上，通过测量伺服电机的角位移间接计算出执行部件的实际位置(或位移)，然后进行反馈控制。由于将丝杠、螺母等大惯量环节排除在闭环控制系统之外，系统不能对它们的运动误差进行补偿，精度受到影响，但系统稳定性有所提高，调试比较方便，价格也较全闭环系统低。

1.4.3 运动控制技术的发展方向

早期的运动控制系统多采用专用硬件控制设备，必须设计专用的芯片及其他硬件电路以满足不同控制应用的需要，世界各国的系统生产商也都分别设计自己的专用硬件系统和软件系统。这种封闭式结构使运动控制系统的开发成本极高，开发周期很长，升级困难，可靠性、可扩展性、可维护性和易用性差，二次开发困难:。随着技术、市场、生产结构等的快速变化，人们对运动控制技术提出了更高的要求，希望能根据不同的应用需求、迅速、高效、经济地构筑面向客户的控制系统;大幅度降低系统维护费用，改变以往运动控制系统封闭性设计模式，使底层生产控制系统的集成更为简便和有效[3]。因此运动控制系统朝着开放的、实时的、可靠的方向发展[13-15]。

1.4.4 移动机器人的运动控制

移动机器人一切功能的实现离不开它的移动机构，移动机构是整个移动机器人系统的基础。传统的移动机器人移动机构主要有轮式、腿式、履带式等。轮式移动机构是移动机器人中应用最多的一种机器人，在相对平坦的地面上，用车轮移动的方式是相当优越的。车轮的形状或结构取决于地面的性质和地面的承载能力[16]。在轨道上运行的多采用实心的刚轮，室内路面行驶的多采用充气轮胎。轮式移动机器人根据车轮的多少可分为 1 轮、2 轮、3 轮、4 轮和多轮结构。1 轮移动机构在实现上的障碍主要是稳定性问题尚未解决，目前已经有人致力于直立控制的研究。2 轮移动机构，如摩托车和自行车等对象，因为直立稳定性问题差而未进入实用阶段。3 轮移动机构基本上具有直立稳定性，其主要的问题是移动方向的控制。代表性的车轮控制方法是一个前轮，两个后轮。前轮操舵，采用前轮或后轮驱动均可。也可以采用两后轮同时驱动，靠两轮的转速差进行操舵的控制方式，此时另外一个轮子只起到支撑作用，采用这种

轮式移动机构的时候，可以使两个驱动轮朝向不同方向转动，从而可以使车体实现灵活的小范围回转。4 轮移动机构是一种应用广泛的结构，这种移动结构适用于高速行走，稳定性好。如何使操舵性能进一步提高，是目前研究的重要课题之一。根据适用目的，还有适用 6 轮驱动车和车轮直径不同的轮胎车，也有人提出利用具有柔性机构车辆的方案。最典型的 6 轮机构是火星探测车，它的轮子可以根据地形上下调整高度，提高其稳定性，适合在外星表面行走[17]。

本文选用 3 轮机构为研究对象，移动机器人底盘的左右两侧分别安装直流伺服电机的驱动轮，独立驱动。这样，机器人的运动控制转化为对机器人运动模型的掌握以及对驱动电机的控制。

1.5 本文所做的工作以及意义

本文选取三轮式移动机器人作为研究对象，分析其运动学模型，研究电机控制方法，介绍机器人控制算法并作改进，结合嵌入式系统相关知识，选取 ARM9 芯片 S3C2410 作为主处理器，设计移动机器人系统硬件平台，采用 L298 作为电机驱动器，采用内电流环加外速度环的控制策略以及积分分离的 PID 控制算法，实现三轮移动机器人(后两驱动轮+前一万向轮)的直线、自旋转以及以及一定角度转弯的功能。扩展系统系统平台外围功能器件，主要有 Flash、SDRAM、网络、键盘和 LCD 等，选取 Linux 作为操作系统，进行软件开发环境的搭建和 Linux 系统以及文件系统的移植，进行系统软件的分析，设计直流电机驱动程序、电流环控制子程序和速度环控制子程序等，最后进行设备驱动程序的调试和系统试验。

本课题设计的移动机器人控制系统，具有很高的系统集成度和广泛的功能扩展空间，很好的兼顾了控制系统的通用性和实用性要求。可通过控制单元的扩充和升级，增加语音识别、人脸识别、视觉追踪等交互性更强的功能。同时，该控制系统的设计完成，对于降低上述各类型机器人的开发难度，缩短从客户提出需求到完成最终产品的开发周期，具有很强的指导意义。

另外，本课题设计的移动机器人控制系统。它可以在广泛的领域中得到应用，如商场导购服务、大型会展接待服务、观光景点导游、传染病房机器人、护士助手机器人、保安护卫等。除此之外，由于集成有通用 ARM 微控制器开发平台、电机驱动模块等多种功能单元，因此，可作为数字电子技术、自动控制技术、传感器技术、路径规划及人工智能等多学科多领域的通用实验平台。

第二章 相关技术理论

2.1 三轮轮式移动机器人建模

被控对象模型的建立是进行控制的第一环节，也是至关重要的一个环节。模型建立的好坏对控制性能有重要影响，如果模型能精确建立，则能大大减轻控制难度，改善控制效果，在建立模型之前，先分析一下移动机器人的运动结构。

2.1.1 运动结构

三轮移动结构是轮式移动机器人的基本移动结构，该结构有下述优点:能够可靠稳定地运动、能量利用率高、机构和控制相对简单，并且可以借鉴现有的技术和经验对其进行控制。

目前，三轮移动机构从结构上看有以下三种不同的情况。如图 2-1，在该结构中，前轮由操舵结构和驱动结构合并而成，由于操舵和驱动的驱动器都集中在前轮，所以该结构比较复杂。该结构旋转半径可以从 0 到无限大连续变化，但是由于轮子和地面之间存在滑动，绝对的 0 转弯半径很难实现。

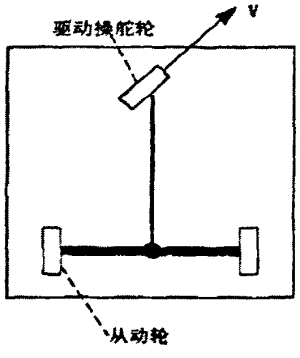

图 2-1 前驱动轮，后从动轮

Fig. 2-1 Front driver wheel, back slave wheel

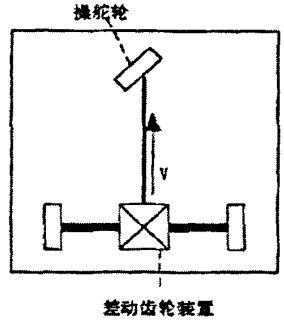

图 2-2 前方向轮，后驱动轮

Fig. 2-2 Front direction wheel, back driver wheel

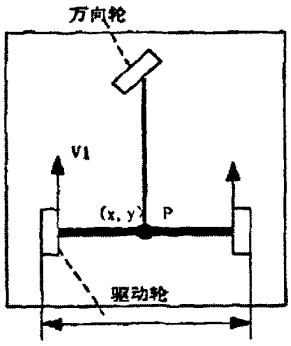

图 2-3 前万向轮，后驱动轮

Fig.2-3 Front universal wheel, back driver wheel

如图 2-2，在该结构中，前轮为操舵轮，后两轮由差动齿轮装置驱动，克服了图 2-1 中的缺点，但是该方法也不多用在移动机器人机构中。

如图 2-3，在该结构中，前轮为万向轮，仅起支撑作用，后两轮分别由两个电机独立驱动，结构简单，而且旋转半径可以从零到无限大任意设定。其旋转中心是在连接两驱动轴的直线上，所以旋转半径即使是 0，旋转中心也与车体的中心一致。鉴于该机构结构简单、便于控制的优点，

本课题所研究的移动机器人就采用这种三轮结构，前轮为一万向轮，后两轮由两台电动机分别独立驱动。

2.1.2 运动学模型

机器人的运动控制问题主要指出 Durrant-Whyte H.F 针对机器人导航提出的三个问题：I“我现在在何处?”；II“我要往何处去?”；III“我应该如何到达该目标点?”。其中，第一个问题可以通过外部传感器获取信息得到答案；第二个问题需要决策进行思考；第三个问题是：给定一个移动机器人的环境特征(环境变量可以通过视觉子系统得到并沿决策途径进行传递)，给定一个起点和一个期望终点，以及到达终点时需要具有的状态，寻求一条连接起点和终点的运动轨迹，并且给出控制该运动轨迹的控制步骤。这三个问题的提出定义了运动控制在整个流程中的位置和角色。本文的机器人为两轮独立驱动的三轮轮式移动机器人，对它的各种运动控制最终都要通过分别控制其左右轮速来实现，因此需分析机器人位姿与其左右轮速之间的约束关系，也就是建立机器人的运动学模型。机器人的后两轮为驱动轮，前轮为万向轮，万向轮为随动轮，它仅在运动失衡时起支撑作用，在运动学中可以忽略不计，该模型可以简化为两轮移动机器人的模型，建立模型时作如下假设：

(l)机器人所在路面为光滑平面；

(2)机器人运动过程中，在纵向作纯滚动，没有侧向滑移；

(3)机器人的有关参数，如左右轮半径 R，左右轮之间的距离 2L 在机器人负载与空载情况下相同。

机器人运动学模型可以用图 2-4 抽象表示:

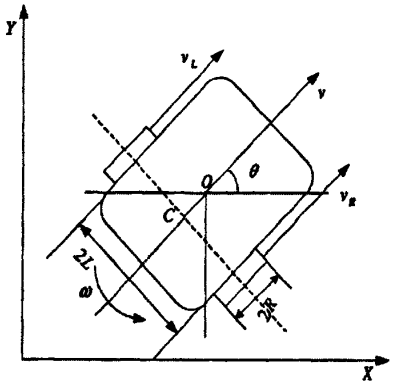

图 2-4 轮式移动机器人运动学模型

Fig.2-4 Kinematics model of wheeled mobile robot

在图 2-4 中，点 0 为移动机器人的几何中心，点 C 为两驱动轮的轮轴中心，车轮半径为 R，两个驱动轮轮心间的距离为 2L。在笛卡儿坐标系下，考虑 C 点为参考点，我们用坐标(x,y)表示 C 点坐标，θ 为机器人的姿态角，即机器人前进方向相对于 X 轴的方位角，v 是机器人的前进速度，ω 是机器人车体的转动角速度，v_L、v_R 分别为机器人左右轮的线速度，而 ω_L、ω_R 分别为机器人左右轮的转动速度。机器人的位姿可以表示为：

$$q=[x,y,\theta]^T \quad \text{..... (2.1)}$$

在无滑动、纯滚动的条件下，轮子在垂直于轮平面的速度分量为零，系统运动约束条件为：

$$x\sin\theta-y\cos\theta=0 \quad \text{..... (2.2)}$$

如果运动约束方程不可能积分为有限形式，则称为非完整约束，可以用反证法证明式(2.2)是一个非完整约束。

证：假设式(2.2)是完整约束，即可以把它积分成有限形式

$$f(x,y,\theta)=C \quad \text{..... (2.3)}$$

其中 C 为常量，对式(2.3)求导得：

$$f_x(x,y,\theta)=\sin\theta, \qquad f_y(x,y,\theta)=-\cos\theta, \qquad f_\theta(x,y,\theta)=0 \quad \text{....(2.4)}$$

由 $f_\theta(x,y,\theta)=0$ 可知，$f(x,y,\theta)$ 是一个与 θ 无关的函数，而这与 $f_x(x,y,\theta)=\sin\theta$，

$f_y(x,y,\theta)=-\cos\theta$ 相矛盾，因此式(2.4)不可积，即它是一个非完整约束。

机器人的质心运动方程为：

$$x=v\cos\theta\ ,\qquad y=v\sin\theta\ ,\qquad \theta=\omega \quad \text{.....(2.5)}$$

则机器人运动学方程为：

$$\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \quad \text{.....(2.6)}$$

根据刚体运动规律，可得下列运动方程：

$$v_L=\omega_L R\ ,\qquad v_R=\omega_R R \quad \text{.....(2.7)}$$

$$\omega=\frac{v_R-v_L}{2L}\ ,\qquad v=\frac{v_R+v_L}{2} \quad \text{.....(2.8)}$$

由式 2.8 可知，当 $v_L=v_R$ 时，质心的角速度为 0，则机器人沿直线运动；当 $v_L=v_R$ 时，质心的线速度为 0，则机器人可实现原地转身，此时机器人将以零半径转弯，其他情况下，机器人围绕零到无穷大的转弯半径的圆心做圆周运动，将(2.7)、(2.8)代入式(2.6)得：

$$\begin{bmatrix}x\\y\\\theta\end{bmatrix}=\begin{bmatrix}\frac{R}{2}\cos\theta & \frac{R}{2}\cos\theta\\ \frac{R}{2}\sin\theta & \frac{R}{2}\sin\theta\\ -\frac{R}{2L} & \frac{R}{2L}\end{bmatrix}\begin{bmatrix}\omega_L\\ \omega_R\end{bmatrix} \quad \text{.....(2.9)}$$

由式(2.9)可以看出，若知道 ω_L 和 ω_R 即可确定机器人的位姿，而驱动电机转速和电机角速度之间的关系为：$\omega=\frac{2\pi n}{60}$，则只要能测得左右轮电机的转速就可以求出 ω_L 和 ω_R，进而完成对机器人的直线、旋转和转弯等各种运动控制[18]。

2.2 电机控制

2.2.1 电机

电机是移动机器人的动力源泉，目前常用的控制电机有：电压控制感应电动机(制动电动机或两相伺服电动机)、电压控制直流电动机(DC 伺服电动机)、频率控制同步(SM)电动机(步进型伺服电动机)、频率控制感应(IM)电动机(感应型伺服电动机)、频率控制磁阻电动机(步进电动机)等，每种电机又衍生出不同的小类型，不同的电机控制方式、特点和应用场合也不相同。

目前移动机器人领域应用较多的是步进电机和直流电机两种。步进电动机是一种将电脉冲信号转换成机械角位移模拟量的控制电机，其输出的位移大小与输入脉冲个数成正比且时间上与脉冲同步，通过改变脉冲频率调节步进电机转速。步进电机一般分为反应式(VR)、永磁式(PM)和混合式(HM)三种。反应式结构简单、工作可靠、运行频率高但转子阻尼大、噪声大、步距角一般在“1.5° ～15° ”之间；永磁式功率小、效率高、价格低，步距角一般在“7.5° ～18° ”之间；混合式介于二者之间，具有步距角小、频率高、功率小等优点，但结构相对比较复杂，步距角一般在“0.36° ～3.6° ”之间。

直流电机是最早出现的电动机，也是最早能实现调速的电动机。长期以来，直流电动机一直占据着调速控制的统治地位。由于它具有良好的线性调速特性，简单的控制性能，高的效率，优异的动态特性，尽管近年来不断受到其它电动机(如交流变频电动机、步进电动机等)的挑战，但目前为止，它仍然是大多数调速控制电动机的优先选择。其调速方法有多种，如果采用 PWM 控制，只需要通过软件改变 PWM 波的占空比就可实现调速，这对提高移动机器人运动中的灵活性非常有用。另外，随着 PWM 输出在机器人控制器中的广泛应用，采用直流电机作为驱动电机的应用也越来越多。

对比二者的特点，尤其考虑使用嵌入式 ARM 作为控制器时设计可靠性，本课题的驱动电机就选用直流电机。

2.2.2 直流电机控制

直流电机具有良好的运行和控制特性，长期以来，一直占据着垄断地位，就其控制方法，也有多种，在讨论之前先看一下他的供电原理图，如图 2-5 所示

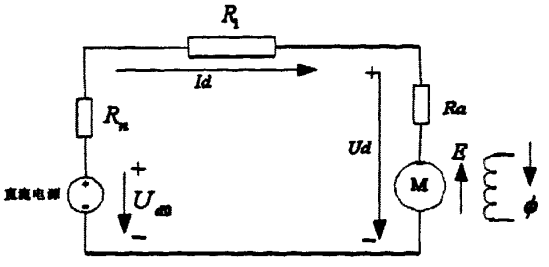

图 2-5 直流电机供电原理图

Fig.2-5 Schematic diagram of power for DC motor

直流电机的转速方程为：$n=\frac{E}{C_e}=\frac{U_{do}-I_dR}{C_e}=\frac{U_d-I_dR_a}{C_e}$，式中$R=R_a+R_1+R_n$ 由此可知，直流电机有三种调节转速的方法：

(1) 调节电枢电压调节；

(2) 减弱励磁磁通调速；

(3) 改变电枢回路电阻调速；

一般工程上常用的主要是调节电压调速。实现调压调速，首先要有一个平滑可调的直流电源。常用的可调直流电源有以下三种：

(1)旋转变流机组：用交流电动机和直流发电机组成的机组，以获得可调的直流电压；

(2)静止可控整流器：用静止的可控整流器，如晶闸管可控整流器，以获得可调直流电压；

(3)直流斩波器或脉宽调制变换器：用恒定直流电源或不可控整流电源供电，利用直流斩波器或脉宽调制变换器产生可变的直流平均电压。

近年来，随着电力电子技术的发展，已普遍采用由晶闸管整流器供电的直流电机调速系统，以取代以前广泛应用的交流电动机－直流发电机供电系统。但由于相控整流中电网输入电流的功率因数直接与移相触发角相关，在电机低速运行整流器输出电压较低时，电网输入电流功率因数底，谐波分量比较大，对电网特别是对公共直流电网有不利影响。而采用直流脉宽调制的调速系统，在许多方面有较大的优点：①主电路线路简单，需要的功率元件少；②开关频率高，电流容易连续，谐波少，电机损耗和发热都较小；③低速性能好，稳速精度高，因而调速范围宽；④系统快速响应性能好，动态抗干扰能力强；⑤主电路元件工作在开关状态，导通损耗小，装置效率较高；⑥直流电源采用不可控三相整流时，电网功率因数高。

直流脉冲宽度调制(Pulse Width Modulation 简称 PWM)是指输出一定周期的信号，通过调整基本周期内的工作周期的大小来控制输出电压，从而调整直流电机的转速。图 2-2- 所示是脉宽调制型调速系统原理图。

图中的开关 S 表示脉宽调制器，调速系统的外加电源电压U_s为固定的直流电压，当开关 S 闭合时，直流电流经过 S 给电动机 M 供电；开关 S 断开时，直流电源供给 M 的电流被切断，M 的储能经二极管 VD 续流，电枢两端电压接近为零。如果开关 S 按照某一定频率开闭而改变周期内的接通时间时，控制脉冲宽度相应改变，从而改变了电动机两端平均电压，达到调速的目的。

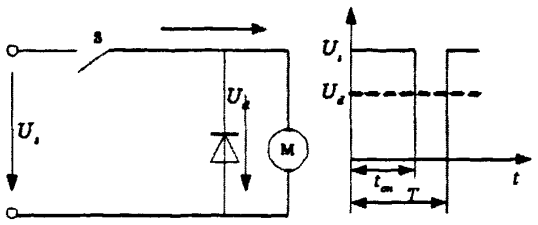

图 2-6 脉宽调速系统原理图

Fig.2-6 Schematic diagram of PWM timing

由图 2-6 的脉冲波形，其平均电压为 $U_d = \frac{1}{T}\int U_s dt = \frac{t_{on}}{T} U_s = \rho U_s$，式中，T 为脉冲周期，$t_{on}$ 为接通时间，ρ 为占空比。可见，在电源 U_s 与 PWM 波的周期 T 固定的条件下，U_d 可随 ρ 的改变而平滑调节，从而实现电动机的平滑调速[19-21]。

本文中就是采用 PWM 脉宽调速[22-23]。直流电机 PWM 调速系统有可逆和不可逆之分。其中可逆 PWM 系统可以控制电机正反转，在工程上有更大的应用价值。可逆 PWM 控制主要有三种工作模式：双极模式、单极模式和受限单极模式。工程上常用双极模式，双极模式控制系统有 T 型和 H 型两种，T 型电路结构简单，便于实现电能反馈，但要求双电源供电，且晶体管承受的反压较高，因此只适用于小功率低压伺服系统。H 型 PWM 功率转换电路只需单一电源供电，晶体管耐压相对较低，应用较为广泛，如图 2-7 所示。

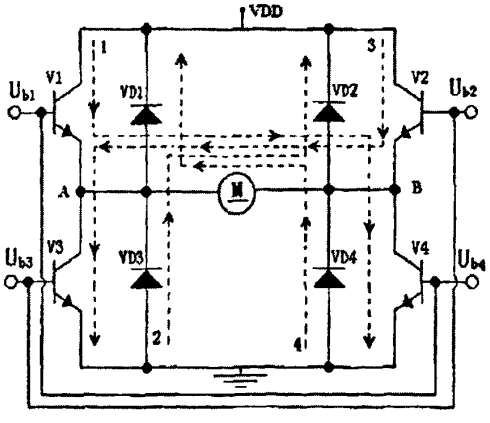

图 2-7 H 型双极模式 PWM 功率转换电路

Fig.2-7 PWM power conversion circuit of H type bipolar mode

H 桥结构又称为全桥结构[24]，如图 上所示，他由两个半桥驱动器、四个功率开关和四个续流二极管组成。利用 H 桥驱动电路和 PWM 控制可实现对直流电动机正反两个方向的调速和伺服控制。双极方式的特点是四桥臂对角线两组开关分别控制，V1 和 V4 为一组同时导通或关断，V2 和 V3 为一组，也是同时导通和关断，在任一时刻最多只允许有一组是导通的。

本文就是选用这种 H 桥式控制结构作为 PWM 控制方案。

2.3 机器人运动控制算法

机器人的运动控制，从伺服控制角度来看，主要有位置反馈控制和速度反馈控制。对于本文研究的双轮驱动+一个方向轮结构的机器人，对电机控制的好坏，直接影响到机器人的运动性能。从理论上讲，传统的 PID 控制就能满足控制的需要[1]。

PID 控制由于其算法简单、鲁棒性好和可靠性高，被广泛用于工业控制。随着计算机技术的发展，在控制过程中，用计算机 PID 控制算法来实现数字 PID 控制器，可以灵活的改变 PID 参数，同时可以改变控制策略来达到控制目的。这是传统的模拟 PID 控制器所无法实现的。

数字 PID 控制算法，由于计算机控制的特点，需要对模拟 PID 算法离散化，假设采样周期为 T，在采样时刻 $t=kT$，可得数字 PID 控制算法如下：

$$u(kT)=K_p\left(e(kT)+\frac{T}{T_i}\sum_{i=0}^{k}e(iT)+\frac{T_d}{T}(e(kT)-e(kT-T))\right)\text{.....(2.10)}$$

为了书写方便，将 e(kT) 简化成 e(k)，u(kT) 简化成 u(k) 省去 T，式(2.10)可简化为：

$$u(k)=K_p\left(e(k)+\frac{T}{T_i}\sum_{i=0}^{k}e(i)+\frac{T_d}{T}(e(k)-e(k-1))\right)\text{.....(2.11)}$$

或

$$u(k)=K_pe(k)+K_i\sum_{i=0}^{k}e(i)+K_d(e(k)-e(k-1))\text{.....(2.12)}$$

K_i 为积分系数，$K_i=K_pT/T_i$；K_d 为微分系数，$K_d=K_pT_d/T$。只要采样周期 T 足够小，就能保证有足够的精度。(2.11)以及(2.12)称式为位置式 PID 算法[25]。

2.4 嵌入式系统

2.4.1 嵌入式系统概述

所谓的“嵌入式系统”是“嵌入式计算机系统”的简称，电气工程师协会的一个定义是用来控制或监视机器、装备或工厂等大规模系统的设备。嵌入式系统主要由嵌入式处理器、相关支撑硬件、嵌入式操作系统及应用软件系统等组成。嵌入式系统的设计可以分为三个阶段：分析、设计和实现。分析阶段是确定要解决的问题及需要完成的目标；设计阶段主要是解决如何在给定的约束条件下完成用户的要求；实现阶段主要是解决如何在所选择的硬件和软件的基础上进行整个软、硬件系统的协调工作。通常，硬件和软件的选择包括处理器、硬件部件、操作系统、编程语言、软件开发工具、硬件调试工具，软件组件等。

嵌入式系统开发一般包括硬件部分设计和软件部分设计，由于嵌入式系统自身的特殊性，注定了它自身的资源和内存空间都是十分有限，不可能像开发 PC 软件那样在其上运行所有的开发工具，而且很多嵌入式系统都没有像显示器那样的输出设备，这些都决定了嵌入式系统软件开发应当采用一种特殊的模式：主机—目标机模式，使用交叉开发的方式进行开发。

开发环境就运行在主机上，用户所有的开发工作都在主机开发环境中进行，包括编辑源代码、编译、连接、下载和调试等。主机的操作系统是通用的 Windows 或 Linux 系统，目标机就是嵌入式应用系统，它运行嵌入式实时操作系统。在主机上通过交叉编译生成目标平台上可运行的二进制文件，通过串行口或 Internet 网口下载到目标机。在目标机执行时，可以把执行结果回显到主机上，主机还可以通过开发环境提供的调试工具对代码进行调试。

2.4.2 ARM 微处理器

嵌入式处理器是嵌入式系统的核心，是控制、辅助系统运行的硬件单元。目前流行的体系结构包括 MCU、MPU 等 10 多个系列。嵌入式处理器课分为：嵌入式微控制器(EMCU，Embedded Micro Controller Unit)、嵌入式微处理器(EMPU，Embedded Micro Processor Unit)、通信领域的嵌入式 DSP 处理器(EDSP，Embedded Digital Signal Processor)、高度集成的嵌入式片上系统(SOC，System On Chip)。微控制器俗称单片机，随着技术的发展，8/16 位单片机的速度和内存的限制较难满足复杂的应用要求；而 32 位嵌入式微处理器的价格不断下降，并且其高速度、低功耗等诸多优异的性能，受到了广大用户的青睐，而成为各类产品中选用较多的处理器。嵌入式微处理器是由通用的计算机 CPU 演变而来，目前市场上主流的微处理器有 Power PC，386EX，Motorola 68000，MIPS，ARM 等[26]，其中 ARM 系列的芯片占据了市场上约 60%的份额，是目前最为流行的微处理器。

ARM 是一家设计 32 位嵌入式 RISC 芯片内核的公司。ARM 处理器采用 RISC 的架构技术，具有小体积、低功耗、低成本、高性能等特色，支持 Thumb(16 位)和 ARM(32 位)双指令集，能很好的兼容 8 位/16 位器件。目前，ARM 处理器已成为最流行的嵌入式处理器。ARM 处理器分为 ARM7、ARM9、ARM9E、ARM10、ARM11 和 SecurCore 系列。其中常用的 ARM7 内核有 ARM7TDMI、ARM7TDMI-S、ARM720T、ARM7EJ-S 等。其中，T：支持 16 位 Thumb 指令。D：支持在线 Debug。M：内嵌乘法器 Multiplier。I：内嵌入式 ICE，支持在线断点和调试。E：DSP 指令，表示支持 DSP 的特定指令，主要是 16 位的 Thumb 指令。S：可以综合，提供 VHDL 或者 Verilog 语言设计，可以实现自己定义的硬件。J：支持新的 Java 功能。ARM9 内核有 ARM920T，ARM922T 和 ARM940T。ARM9E 内核有 ARM926EJ-S、ARM946E-S、ARM966E-S。ARM10E 的内核有 AMR1020E、AMR1022E 及 ARM1026EJ-S[44]。

1 、ARM 微处理器选型

ARM 微处理器有几十种架构，几十个芯片生产厂家以及各种各样的内部功能配置，因此开发时需要对芯片做一些对比分析，选择合适的芯片。对比选择的因素主要考虑如下几点：

1.ARM 微处理器内核的选择

不同的内核，适用于不同的应用领域。如 ARM7 内核没有 MMU，而 ARM9 内核有 MMU。由于 uCLinux 等不需要 MMU 单位，因而可以在 ARM7 上运行，相反，嵌入式 Linux 具有 MMU，因而可以在 ARM9 上运行。

2.系统的工作频率

系统的工作频率很大程度上决定了系统处理任务的能力。但是系统的工作频率越高，其功耗也较高。因此在实际应用中，需要根据需要来选择工作频率。

3.芯片内存储器的容量

多数的 ARM 微处理器片内存储器的容量不大，因而需要用户在设计系统时进行外部扩展，但是也有芯片内部有较大的片内存储空间。因而，用户可以根据需要选择合适的方案。

4.片内外围电路的支持

几乎所有的芯片都有各自不同的适用领域，扩展了相应的外围模块功能，并集成在芯片内部，称之为片内外围电路。开发人员根据系统设计的需要，选择合适的 ARM 外围电路，可以大大地降低开发成本，节约开发时间。

2、S3C2410 芯片资源

ARM 公司于 1997 年推出了能提供五级流水线和哈佛结构的微处理器 ARM9TDMI 系列，新一代的 ARM9 处理器通过增加时钟频率和减少指令执行周期可以使其处理能力达到 ARM7 处理器的两倍以上。主要包括：ARM920T、ARM922T 和 ARM940T 三种类型。其主要特点有：

1. 采用 ARMv4T 结构，支持 32 位元 ARM 指令集和 16 位元 Thumb 扩展指令集；
2. 与 ARM7 代码高度兼容，仅仅简化了数据中断处理方式；
3. 采用 5 级流水线和哈佛体系结构，拥有 32 位的线性地址空间，分别存储代码和数据；
4. 提供协处理器接口，支持片内协处理器，用以支持浮点运算，数字信号处理，图形加速；
5. 扩展片内仿真调试方式，支持硬件单步调试和异常中断；ARM9 系列处理器都采用了 ARM9TDMI 处理器核，兼有 16 位元 Thumb 指令集，使得代码密度提高了 35%。

S3C2410 处理器是 Samsung 公司基于 ARM 公司的 16/32 位 ARM920T RISC 微处理器。该处理器同时增加了丰富的外围资源，为手持设备和通用嵌入式应用提供了片上集成系统解决方案，对系统开发非常有利。

该处理器的 CPU 内核采用 0.18um CMOS 标准单元结构，实现了存储管理单元(MMU)和高速缓存体系结构。其中，MMU 可以管理虚拟内存，高速缓存由独立的 16KB 地址和 16KB 数据高速 Cache 组成。

该芯片集成了一系列完整的外围设备及接口，包括 LCD 控制器、NAND FLASH 控制器、SDRAM 控制器、3 个通道的 UART，8 路 10 位 ADC，1 个 USB 设备等控制器和丰富的外部接口；在时钟方面也有突出的特点，该芯片集成了一个具有日历功能的实时控制(Real-time control 简称 RTC)和具有锁相环(Phase-Locked Loop 简称 PLL)(MPLL 和 UPLL)的芯片时钟发生器。MPLL 产生主时钟，能够使处理器工作频率最高达到 203MHz。该工作频率能够使处理器轻松运行 WinCE，Linux 等操作系统以及进行较为复杂的数据处理。该芯片提供了一套完整的通用系统的外围设备，并且使得整个系统消耗最小[27]。

因此，本系统采用以 ARM920T 为内核的具有以上优点的 S3C2410 作为系统处理器。这使得后续的嵌入式操作系统的选择以及功能的扩展有了更大的灵活性。而且，该芯片在国内应用较为广泛、成熟，可以参考的资料也较多，这对缩短开发周期也会有很大的好处。

2.4.3 嵌入式操作系统

嵌入式操作系统是支持嵌入式系统工作的操作系统。它在知识体系和技术本质上

与通用操作系统没有太大的区别，一般用于比较复杂的嵌入式系统软件开发中。由于嵌入式操作系统应用的硬件资源远没有通常的操作系统丰富，因此一般嵌入式系统占用系统资源较小，且一般可以根据需要进一步裁减，使其体积进一步减小。正因为其体积的可裁减性，因而应用非常广泛。尽管不同的场合对嵌入式操作系统的要求会有所不同，但是这些操作系统的主要功能仍然是提供一些系统服务供应用程序调用，包括多任务管理、存储管理、周边资源管理、中断管理等。

嵌入式操作系统按照是否免费可分为两种：商用型和免费型。目前商用型的操作系统主要有 VxWorks、Windows CE、Psos、Palm OS、OS-9、QNX、LYNX 等。它们的优点是功能稳定、可靠，有完善的技术支持和售后服务，而且提供了图形用户界面和网络支持等高端嵌入式系统需要的许多高级功能；缺点是价格昂贵且代码封闭。目前免费型的操作系统主要有 Linux 和 μC/OS-II，在价格方面有很大的优势。

WINCE 系统特点：支持针对小内存体积；“硬”实时内核：支持 256 个优先级别和嵌套中断；强健的内存管理：虚拟地址空间从 32MB 扩展到当前的 64MB。另外还支持内存映射文件。这些基于 Win32 的文件映射 API 除了允许数据文件映射到内存当中并建立类似于内存指针的引用之外，还可以用来分配能够在多个进程之间进行共享的内存；开放的通信平台：TCP/IP，IPv6 等；远程和系统管理；对标准的支持；广泛的存储和文件系统；缺点是系统实时性不足。

VxWorks 操作系统：VxWorks 操作系统是美国 WindRiver 公司设计开发的一种嵌入式实时操作系统。良好的持续发展能力、高性能的内核以及友好的用户开发环境，在嵌入式实时操作系统领域占据一席之地。它以其良好的可靠性和卓越的实时性被广泛地应用在通信、军事、航空、航天等高精尖技术及实时性要求极高的领域中。但由于是商业系统，所以价格昂贵。

μC/OS-II 操作系统：μC/OS-II 结构小巧、具有可剥夺实时内核的实时操作系统。其内核提供任务调度与管理、时间管理、任务间同步与通信、内存管理和中断服务等功能。适合小型控制系统，具有执行效率高、占用空间小、实时性优良和扩展性强等特点，最小内核可编译至 2KB。抢占式内核。

Linux 操作系统：继承标准 Linux 的优良特性，针对嵌入式处理器的特点设计的一种操作系统，具有内嵌网络协议、支持多种文件系统，开发者可利用标准 Linux 先验知识等优势。其编译后的目标文件可控制在几百 KB。

在所有的操作系统中，Linux 是发展最快、应用最为广泛的。由于 Linux 的源码开放和设备独立性以及良好的移植性等特点，开发人员将 Linux 系统进行相应的裁减和改进就可以应用到嵌入式处理器上。但是由于各类嵌入式处理器的资源不同，实时性要求也不同，因而对 Linux 的裁减和相关的技术性改造也就不一样。

在没有内存管理单元(Memory Manage Unit 简称 MMU)的微处理器中，最初只能运行一些很简单的单任务操作系统，或者简单的控制程序，甚至没有操作系统系统而

直接运行应用程序。这样要求设计人员对硬件相当熟悉，因此也阻碍了嵌入式系统的应用和发展。

由于标准的 Linux 内核采用了虚拟内存管理技术，这样可以提高系统运行的效率。但是因此需要处理器上有内嵌的 MMU 的支持。因此，在许多没有 MMU 的处理器中应用标准的 Linux 显得不太恰当，甚至因为虚拟管理内存技术的存在给应用造成了一定的困难。因此，为了解决上述问题而开发出一种新的嵌入式操作系统——μCLinux。

该系统是为微控制领域而量身定做的 Linux 版本，源代码完全开放，完全符合 GNU/GPL。其设计的基本思想是针对现有的无 MMU 的处理器而对标准内核进行裁减，去除虚拟内存管理部分代码，并且对内存分配进行优化，从而提高了系统运行的效率，减小系统的体积的同时也保留了 Linux 系统稳定、良好移植性和优秀的网络功能以及对各种文件系统支持的功能。

然而因为没有 MMU 的支持，该系统能对内存单元进行直接的读写，因此当有非法的内存地址的时候，系统可能运行不平稳甚至崩溃，这也要求对系统中的应用程序进行良好的设计来保证系统运行的健壮性和安全性。

在有 MMU 的微处理器中使用 Linux 系统同样也需要进行裁减，这样才能满足外围硬件的限制，针对不同的开发设计的要求，进行裁减和改进的方式也不一样。

众多优良的特性使得 Linux 系统内核可以裁剪得非常小巧，很适合于嵌入式系统的需要。近年来，由于嵌入式 Linux 操作系统特有的优点，嵌入式 Linux 操作系统已经受到了业界的极大关注。

2.4.4 嵌入式操作系统选型

概括地讲，在进行系统选型工作时，要考虑到系统成本、市场进入时间以及技术支持、可移植性、可利用资源、系统定制能力等问题。考虑到系统的价格问题，决定选取免费的嵌入式操作系统。由于 μC/OS 相比较 Linux 来说网络的支持功能没有后者强大，因此对本系统决定采用 Linux 操作系统。

嵌入式 Linux 经 Linux 裁减而来的，属于免费资源，无需考虑版权费用的问题：Linux 在国内市场用的很广，发展非常迅速，尽管没有商业组织为其提供技术支持，但是由于开源，很容易找到相应的技术资料，甚至可以找到相应的设备的驱动程序，所以可以大大缩短产品开发时间，节省产品成本；Linux 经过裁减、改进，可以移植到目前市场上绝大部分嵌入式处理器上，具有良好的移植性；Linux 系统由于其自身的优点，可以根据设计系统的要求定制所需要的内核模块，很容易将系统精简到最小。Linux 系统是由许多组件组成的，一般 Linux 系统的架构及其包含的组件如图 2-8 所示。

图 2-8 Linux 系统的一般架构

Fig.2-8 General configuration of Linux OS

μClinux 是一种 Linux 的变型版本，它通过对标准 Linux 内核裁减，去除内存管理部分的代码，并对内存分配进行优化，达到提高系统运行效率的目的。但是 μ Clinux 的应用程序开发要求用户自己正确的处理内存管理，一旦不慎错误地修改了其他进程的内存，将可能造成系统死机。使用 MMU 的内存管理，对进程进行保护，可以提高嵌入式系统中多进程的保护能力，使用户应用程序的可靠性得以提高，降低了用户的开发难度。

针对本系统的开发，本系统的微处理器选择的是 S3C2410 芯片，带有 MMU 单元。考虑到系统开发的费用和开发的周期，以及以后系统的完善、升级和维护，选择经裁减的 Linux 作为本系统的嵌入式操作系统[28-30]。

2.5 小结

本章介绍本系统设计所涉及的相关理论，首先，介绍三轮移动式机器人，按运动结构的分类选取了后两驱动轮和一前万向轮的三轮式移动机器人作为研究对象，分析其运动学模型。接着介绍电机控制，主要介绍了本系统所要采用的直流电机和直流电机控制方法以及机器人运动控制算法，最后介绍嵌入系统相关技术，选取了 ARM9 芯片 S3C2410 作为主处理芯片，选用 Linux 作为操作系统。

第三章 系统硬件设计

3.1 移动机器人运动控制系统功能要求

本移动机器人运动控制系统，按照运动控制的：开放的、实时的、可靠的主要发展方向，采用嵌入式系统设计，完成移动机器人最底层的硬件系统平台和伺服系统设计。实现键盘交互、LCD 显示功能、设计 Internet 网络通讯功能，设计高性能的伺服控制器实现三轮移动机器人(后两驱动轮+前一万向轮)的直线、自旋转以及以及一定角度转弯的功能，此外比较重要的是系统应具有很高的系统集成度和广泛的功能扩展空间，兼顾控制系统的通用性和实用性要求。另外，可通过控制单元的扩充和升级，增加语音识别、人脸识别、视觉追踪等交互性更强的功能[31]

3.2 系统原理框图

根据设计要求，本课题采用嵌入式 ARM 处理器作为核心，采用 ARM9 系列。ARM9 系列处理器是 ARM 公司设计的主流嵌入式处理器，主要包括 ARM9TDMI 和 ARM9-S 等系列。它以更加强大的性能正在逐渐代替 ARM7 处理器。新一代的 ARM9 处理器通过增加时钟频率和减少指令执行周期可以使其处理能力达到 ARM7 处理器的两倍以上。

使用微处理器嵌入式系统的优点在于嵌入式微处理器是由通用 CPU 演变而得，处理器在 32 位以上，可以运行操作系统，能够按照应用的要求对硬件实现完全剪裁，有嵌入式操作系统支持的强大软件平台，具有最低冗余度和最大专用性，性能高，在工作温度、抗电磁干扰、可靠性等方面都做了各种增强。不足之处是微处理器采用了多线程，资源管理比较复杂。

根据课题任务要求，考虑到成本、抗干扰性、系统可靠性与稳定性等因素，选用 Samsung 公司生产的 S3C2410 处理器作为控制芯片。该处理器基于 ARM 公司的 ARM920T 处理器核，集成了手持设备和通用嵌入式应用所需的多个功能单元，其主频最高可运行在 206MHz[32]。

图 3-1 为系统硬件原理框图。系统由电机伺服控制器、Internet 通讯接口模块、存储器扩展模块、键盘控制接口、LCD 液晶显示模块、ARM 处理器模块组成。

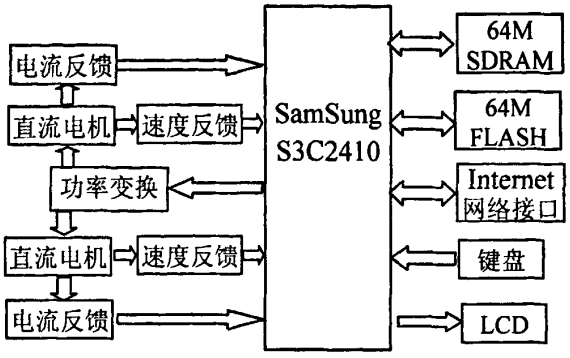

图3-1 系统硬件原理框图
Fig.3-1 System hardware principle diagram

3.3 电机控制器设计

3.3.1 电机控制总体结构

直流电机的理想启动过程，即在整个启动过程中，使启动电流一直保持最大允许值，此时电机以最大转矩启动，转速迅速以直线规律上升，以缩短启动时间；启动结束之后，电流从最大值迅速下降为负载电流值且保持不变，转速维持在给定转速不变。

为了达到这个目标，需要采用一定的控制策略。由于直流电机的开环控制技术不能消除转速差率以及不能满足实时性的要求，实际中常采用闭环控制技术来调节转速。电机的闭环控制系统可以是单闭环系统(速度闭环)，也可以是双闭环(速度环和电流环)。转速负反馈单闭环控制系统可以保证系统稳定的条件下实现转速无静差，但是如果对系统的动态性能要求较高的话，如快速启制动，单闭环系统就很难满足要求，这是因为单闭环系统不能完全按照需求来控制动态过程的电流和转矩。为了改善动态性，就要在速度反馈单闭环控制的基础上再引入电流反馈来控制系统动态过程中的电流和转矩，系统采用双闭环控制系统(外速度环和内电流环)，设计速度调节器和电流调节器，总体控制结构如下图 3-2：

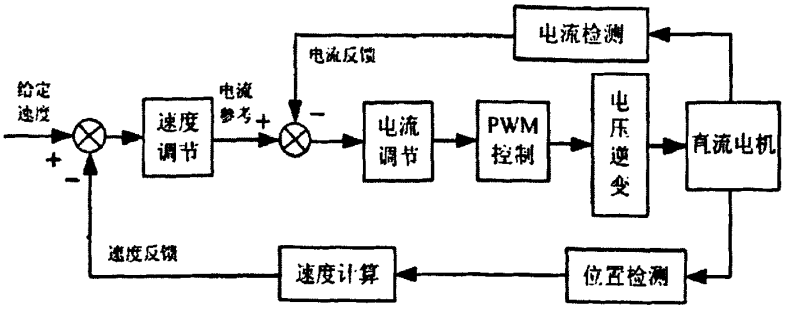

图 3-2 系统控制结构

Fig.3-2 Control framework of the system

速度调节器的作用是对给定速度与反馈速度之差按一定规律进行运算，并通过运算结果对电动机进行调速控制。由于电动机轴的转动惯量和负载的存在，使速度时间常速较大，系统的响应较慢。电流调节器的引入一方面在启动和加减速时起到电流调节和限幅作用，这种情况下，速度调节器的输出作为极限给定值加在电流调节器上，电流调节器的作用使绕组电流迅速达到并稳定在其最大值上，从而实现快速加减速和电流限流作用。另一方面的作用是使系统的抗电源干扰和负载扰动的能力增加。如果没有电流环，扰动会使绕组电流随之波动，使电动机的速度受影响。虽然速度环可以最终使速度稳定，但需要的时间较长。如果有电流环，由于电的时间常数较小，电流调节器会使扰动的电流很快稳定下来，速度的波动不会很大，系统的快速性和稳定性得到改善[33]。

控制算法上采用增量式 PID 控制算法。但是由于在增量式 PID 调节控制系统中，虽然积分环节可以消除静差、提高精度，但加入积分校正后，会造成积分积累，产生过大的超调量，在电机的运行过程中，这是不理想的。所以，为了减少电机在运行过程中积分校正对控制系统动态性能的影响，需要在电机的启停阶段或大幅度加减速时，采用积分分离 PID 控制算法，即只加比例、微分运算，取消积分校正。而当电机的实际速度与给定速度偏差小于一定值时，则恢复积分校正作用。

电流调节器采用 PI 调节，就是将速度调节得到的参考电流与实际检测到的电机电流进行比较，他们的偏差值经过 PI 调节后得到控制量用于改变 PWM 的占空比。

3.3.2 PWM 控制

本文以 MAXON 高电压、低电流的电机为研究对象。电机型号：EB256105，齿轮箱：GP110309。该型号电机的额定电压为 12V，标称功率为 1.5W，转矩输出为 1.59mNm，空载转速为 12500rpm，齿轮减速比 16：1，经减速后的输出转速为 785rpm。

由控制单元给出的PWM信号并不能直接驱动电机，需要经过功率放大才能驱动电机，比较常用的电机驱动电路分为两种：

(1) 由分立的功率元件(如MOS管、功率三极管等)和必要的保护电路、逻辑电路构成，这种驱动电路成本低、占用空间大、设计难度大、可靠性难以保证、测试开发周期长。

(2) 采用专业公司开发的集成电机驱动模块(芯片)来设计，其成本根据集成芯片的功能而不同，占用的PCB空间小、结构紧凑、可靠性高、外围电路设计简单，一般都有成熟的应用方案，涉及开发时间短。常用的直流电机驱动芯片有：L298，L293，L9110，L6203，BA6209，BA6287，LMD18200等。

本文选用L298芯片作为驱动芯片。L298是SGS公司生产的电机双桥驱动器，可以用于直流电机和步进电机的驱动。它的每一个集成电路内部都包括四路半桥式功率驱动器，这些驱动器可以单独使用，也可以分成两组构成两个全桥电路使用，每一路驱动器都有一个内部的TTL逻辑电路输入端，这个输入端可以控制一组桥路。该芯片内部的逻辑控制电路有单独的电源输入端，并且由内部稳压，从而使逻辑电路可以在较低的电源电压下工作，大大减少了逻辑控制电路的损耗。L298是15脚封装，在工作电压达到46V时，驱动管的工作电流可达到2A，其内部带有热保护装置，具有很强的抗干扰能力。有时为了使驱动器得到更大的输出电流，单片集成电路内的两组桥式电路可以并联使用。L298的输出端并联使用时，可使输出电流达到3.5A。

其中的一个H桥式(A桥)结构如下图3-3所示：

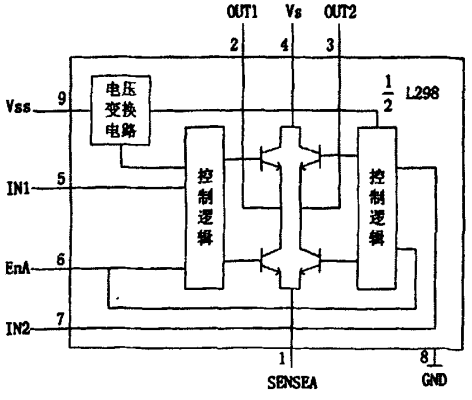

图3-3 L298的H桥式结构

Fig.3-3 H bridge of L298

管脚1为A桥的电流检测接口；管脚2和3为A桥的输出端口；管脚4为驱动电压接口，管脚5和7为A桥的输入信号接口，管脚6为A桥使能端。采用L298可

以很方便地实现直流电机的双极式、单极式驱动。双极性驱动是指在一个 PWM 周期里，电机电枢的电压极性呈正负变化。双极性可逆系统虽然有低速运行平稳的优点，但是由于在一个周期里面电流始终存在，因而功率损耗较大，这在使用电池供电的移动机器人系统设计中最应该注意的，所以本系统使用单极性的驱动方式，在一个 PWM 周期内，电动机电枢只承受单极性的电压。设计原理图如下图 3-4：

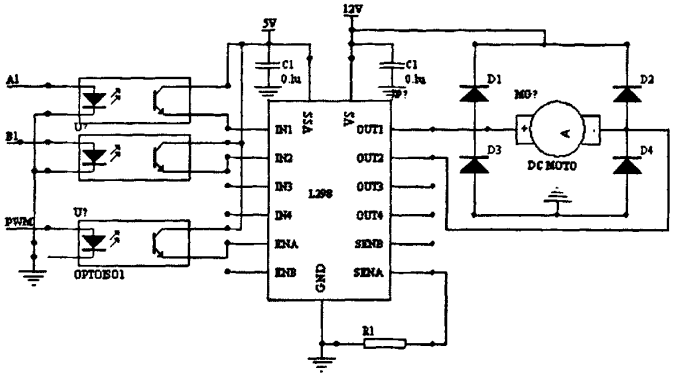

图 3-4 L298 和电机连接图

Fig.3-4 Interface of L298 and DC motor

L298 的电机工作控制端真值表，以 A 桥为例，具体关系如下表 3-1 所示：

表 3-1 L298 电机工作控制端真值表

Table 3-1 Logic table of L298 pins for DC motor running

控制端电平			电机工作情况
ENA	IN1	IN2	
H	H	L	正转
H	L	H	反转
H	H	H	刹车
H	L	L	
L	X	X	停止

由表可知，当且仅当 IN1 与 IN2 反相的情况下，电机才工作，亦即，不论电机正转还是反转，IN1 与 IN2 电平始终绝对相反，而两者同相时所达到的控制目的--电机停转，完全可以通过 PWM 占空比为零来实现。

在本设计中，在处理 IO 口和 L298 控制端之间添加反向器来达到单电机单线控制

转向的目的，简化了微控制器的程序设计和硬件端口资源分配。光耦隔离部分采用6N173进行光耦隔离，原理图如下图3-5所示[34]：

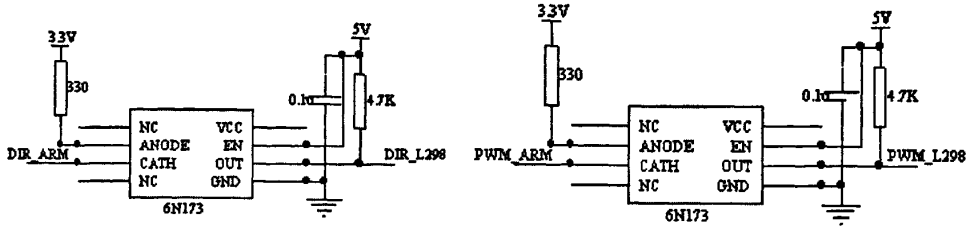

图3-5 光耦隔离电路

Fig.3-5 Circuit of optical isolation

3.3.3 控制策略

直流电机的理想启动过程，即在整个启动过程中，使启动电流一直保持最大允许值，此时电机以最大转矩启动，转速迅速以直线规律上升，以缩短启动时间；启动结束之后，电流从最大值迅速下降为负载电流值且保持不变，转速维持在给定转速不变。

为了达到这个目标，需要采用一定的控制策略。由于直流电机的开环控制技术不能消除转速差率以及不能满足实时性的要求，实际中常采用闭环控制技术来调节转速。电机的闭环控制系统可以是单闭环系统(速度闭环)，也可以是双闭环(速度环和电流环)。转速负反馈单闭环控制系统可以保证系统稳定的条件下实现转速无静差，但是如果对系统的动态性能要求较高的话，如快速启制动，单闭环系统就很难满足要求，这是因为单闭环系统不能完全按照需求来控制动态过程的电流和转矩。为了改善动态性，就要在速度反馈单闭环控制的基础上再引入电流反馈来控制系统动态过程中的电流和转矩，系统采用双闭环控制系统(外速度环和内电流环)，设计速度调节器和电流调节器。

速度调节器的作用是对给定速度与反馈速度之差按一定规律进行运算，并通过运算结果对电动机进行调速控制。由于电动机轴的转动惯量和负载的存在，使速度时间常数较大，系统的响应较慢。电流调节器的引入一方面在启动和加减速时起到电流调节和限幅作用，这种情况下，速度调节器的输出作为极限给定值加在电流调节器上，电流调节器的作用使绕组电流迅速达到并稳定在其最大值上，从而实现快速加减速和电流限流作用。另一方面的作用是使系统的抗电源干扰和负载扰动的能力增加。如果没有电流环，扰动会使绕组电流随之波动，使电动机的速度受影响。虽然速度环可以最终使速度稳定，但需要的时间较长。如果有电流环，由于电的时间常数较小，电流调节器会使扰动的电流很快稳定下来，速度的波动不会很大，系统的快速性和稳定性

得到改善[35-37]。

3.3.4 电流检测

电流检测通常有两种方法：

(1) 使用霍尔电流传感器

(2) 在逆变桥的下桥臂和地之间串接电阻，将电阻上的电压信号作为采样信号。

霍尔电流传感器是利用霍尔效应制成的电流传感器，它是一种很好的隔离式电流检测装置，能有效地实现主电路和控制回路的电气隔离，但是霍尔传感器较贵，增加了系统成本。第二种方法是直接将电机电流信号转化为电压信号送给控制电路，简单、方便，输出电压正比于主电路流过的电流。

本系统采用串接电阻法测电流。在电机回路中中串联四个并联的 1 欧电阻，相当于串联一个 0.25 欧的采样电阻，测量其上的电压推断出电流的大小。采用 4 个 1 欧的电阻并联是为了降低电压分压和获得更大的功率承受能力。根据实验知道，正常运行时电机的电枢电流在 0-400mA 之间，则采样电阻上的电压变化范围为 0-100mV。需要通过放大再送给 AD 采集部分。

但是只检测出电流还是不够的，电枢电流不是一个恒定量，而是周期和 PWM 调速信号相同的脉动信号(标准 PWM 方波电压信号经过滤波后加在电机感性负载后的结果)，显然不能反映电枢电流当前有效值，加之处理采样时间非常快，如处理器直接将该脉动信号采样回来做电流闭环反馈，就必然会造成 PWM 输出震荡。

下图 3-6 是电机电流检测信号的处理电路：

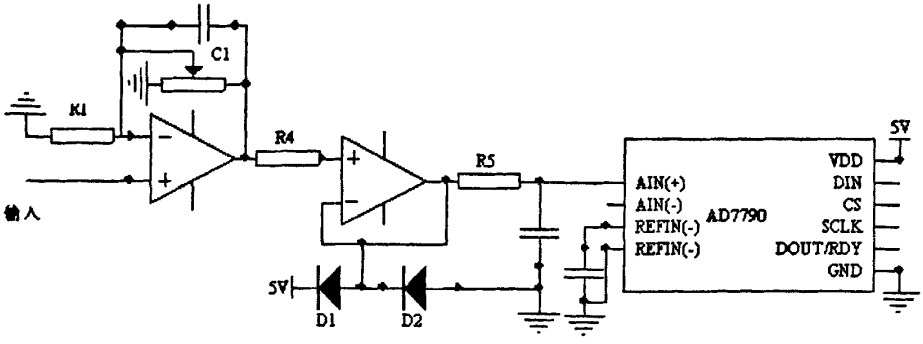

图 3-6 电流信号检测处理电路

Fig.3-6 Measuring and processing circuit for current signal

放大电路有积分电容，对输入信号进行滤波，时间常数的选以 PWM 信号频率为基准；放大电路之后为跟随电路，增加驱动能力，后续电路为一级阻容滤波电路，起

进一步平滑检测信号的作用，该时间常数亦和 PWM 信号频率一致，二极管 D1 和 D2 起保护作用，将检测信号限制在 0-5V 之间。

关于 AD 部分采用 AD7790 芯片完成，AD7790 是一个 16 位的 A/D 转换芯片，通过 SPI 接口和处理器相连，可以编程确定范围、转换速率和操作模式等。

3.3.5 速度检测

本系统在选用直流电机的同时选择了与电机配套使用的速度编码器(MR250899)，该编码器的工作电压为 5V，兼容 TTL 电平，两通道输出，每旋转一周计数 256 个脉冲。电机运行时编码器输出两路相位相差 90 度的方波信号 A、B，当车轮顺时针旋转时，A 脉冲超前 B 脉冲 1/4 个周期，而当车轮反转时，A 脉冲落后 B 脉冲 1/4 个周期，如图 3-7。控制器基于 A、B 脉冲的相位关系的不同来判断车轮的转向，基于其中任一路的脉冲来计数计算机器人的速度。

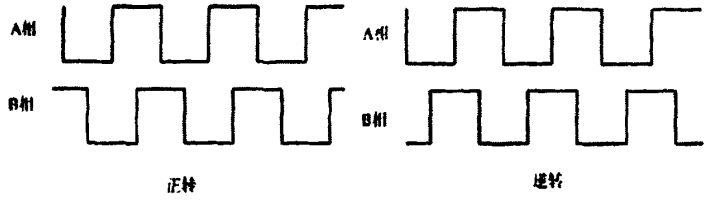

图 3-7 速度编码器输出信号图

Fig.3-7 Output signal of speed encoder

因 ARM9 S3C2410 内部没有正交编码器，本设计选用正交编码计数器 HCTL2020 来辅助检测，HCTL2020 内部集成有数字滤波器、正交编码器、16 位位置计数器和寄存器，其计数值为输入的 4 倍频，对外有 8 位 3 态总线，可以通过处理控制其相应的管脚，来实现分两次并行输出 8 位，一共 16 位的位置计数器。系统中电机编码器输出的信号是 5V 的方波信号，而 ARM 输出口线的驱动电压为 3.3V，二者信号电压不一致，必须进行转换，在设计中采用 74LVC245 作为转换芯片[38-40]，原理图 如下图 3-8

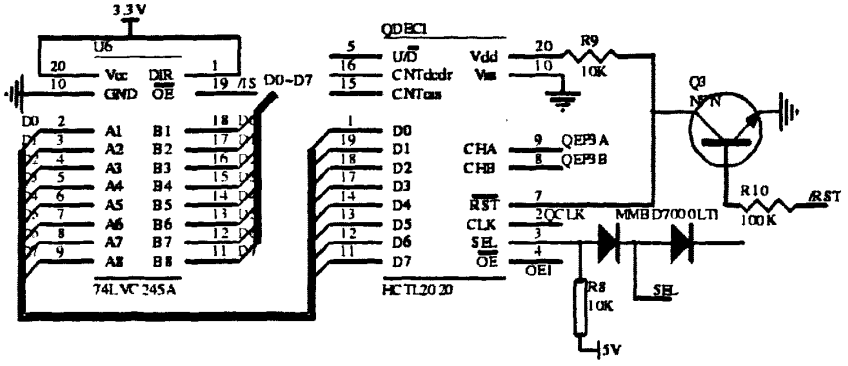

图 3-8 速度检测电路设计

Fig.3-8 Circuit design for velocity measuring

3.4 外围功能器件硬件设计

3.4.1 Flash 存储器接口电路设计

S3C2410 将系统的存储空间分成 8 个 Bank，每个 Bank 的大小是 128M 字节，共 1G 字节。Bank0 到 Bank5 的开始地址是固定的，用于 ROM 或 SRAM。Bank6 和 Bank7 用于 ROM，SRAM 或 SDRAM，这两个 Bank 可编程，且大小相同在。Bank7 的开始地址是 Bank6 的结束地址，灵活可变。所有内存块的访问周期都可编程，外部 Wait 管脚扩展了访问周期。S3C2410 采用 nGCS[7: 0] 8 个通用片选线选择 8 个 Bank 区。

S3C2410 内部集成了一个Nand Flash 控制器，CPU 可以直接由Nand Flash 启动。与NorFlash 相比，Nand Flash 拥有读写速度快，易集成大容量等特点，使得它在嵌入式系统中得到了广泛应用。Nand Flash 采用块和扇区的方式读写，而Nor Flash 采用线性地址读写，这是两种Flash 的最大区别。

S3C2410 具有三种 boot 方式，由 OM[1: 0]管脚选择:00 时处理器从 NAND 闪存 boot；O1 时从 16 位宽的 ROM boot；10 时从 32 位宽 ROM boot。将 bootloader 代码和操作系统镜像放在外部的 NAND 闪存，采用 NAND 闪存 boot。处理器上电复位时，通过内置的 NAND 闪存访问控制接口将 bootload 代码自动加载到内部的 4KB SRAM(此时该 SRAM 定位于起始地址空间 0x00000000)并且运行，在 boot SRAM 运行的 bootloader 程序将操作系统的镜像加载到 SDRAM 之后操作系统就能够在 SDRAM 运行。启动完毕后，4KB boot SRAM 就可以用于其他用途。如果从其他方式 boot, boot ROM 就要定位于内存的起始地址空间 0x00000000，处理器直接在 ROM 上运行 boot 程序，此时 4KB boot SRAM 被定位于内存地址 0x40000000 处。

系统的 Nand Flash 选用的是三星片上（SOP）K9F1208，工作电压为 2.7V—3.6V，

采用 48 脚 TSOP 封装或 48 脚 FBGA 封装. 该 Nand Flash 容量为 64MB，它主要被用来存放启动代码(bootloader)、Linux 内核、根文件系统以及用户程序等等。将 S3C2410 的 CLE,ALE,RnB,nFRE, nFWE ,nFCE 分别接至 K9F1208 的 CLE,ALE,R/B, /RE, /WE ,/CE 端。K9F1208 的/WP(写保护)上拉至 VCC；8 位数据总线[I/O7—I/O0]与 S3C2410 的低 8 位数据总线[DATA07—DATAO]相连. 电路原理图如图 3-9 所示。

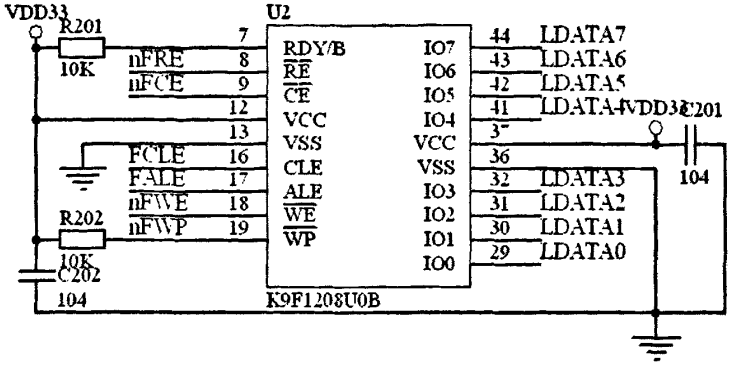

图 3-9 K9F1208 与 S3C2410 连接电路图

Fig.3-9 The connection circuit diagram of K9F1208 and S3C2410

3.4.2 SDRAM 接口电路设计

SDRAM 不具有掉电保持数据的特性，但是其操作都是由时钟作为同步，存取速度大大高于 Flash 存储器，数据吞吐量更大，且具有读/写的属性。SDRAM 在系统中主要用作程序的运行空间。当系统启动时，微处理器首先从复位地址 0x0 处读取启动代码，在完成系统的初始化后，程序代码调入 SDRAM 中运行，以提高系统的运行速度。同时，系统及用户堆栈、运行数据也都放在 SDRAM 中。

本系统扩展 64MB 的 SDRAM，满足嵌入式操作系统及各种相对复杂的算法的运行要求。SDRAM 选用两片 HY57V561620（4M*16bit*4Bank），工作电压为 3.3V，兼容 LVTTL 接口，支持自动刷新(Auto-Refresh)和自刷新(Self-Refresh)，16 位数据宽度。其中一片为高 16 位，另一片为低 16 位，两片拼成 32 位数据宽度的 SDRAM 存储系统，并映射到 S3C2410 的 SROM/SDRAM 的 BANK6，地址范围是 0x30000000--0x33FFFFFF。将其相应的管脚连接于 BANK6、BANK7 上相应的 SDRAM 控制器的管脚，连接图如图 3-10：

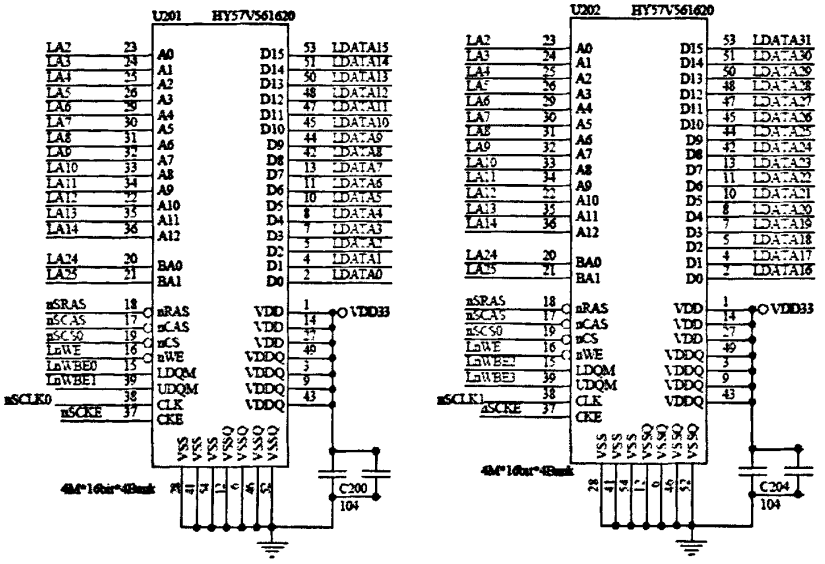

图 3-10 SDRAM 和 ARM9 连接图

Fig.3-10 Interface of HY57V561620 and S3C2410

3.4.3 键盘接口设计

系统需要键盘来实现交互。系统采用 17 键键盘实现对话的功能。键盘由芯片 ZLG7290 驱动，通过 IIC 总线和 CPU 接口。接口连接如图 3-11 所示。

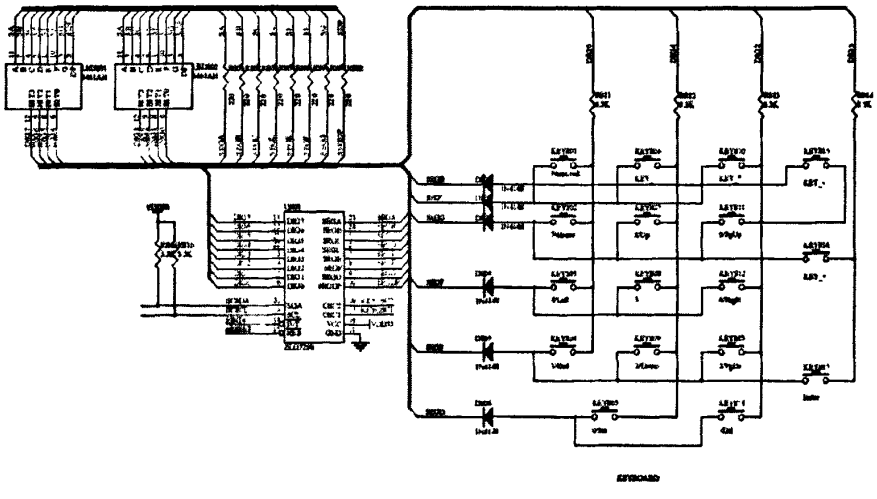

图 3-11 键盘与 S3C2410 连接图

Fig.3-11 The connection circuit diagram of keyboard and S3C2410

键盘由 ZLG7289 完成动态扫描，按键去抖，CPU 通过 IIC 总线进行读键操作，

电路中对 IICSDA 和 IICSCL 两个信号节接有 3.3K 上拉电阻。如果有按键按下会产生中断，ZLG7290 占用 EINT4。

3.4.4 LCD 液晶显示模块

带有汉字显示功能的 LCD 液晶模块，由微处理器的 I/O 口组成模拟串口方式控制模块的显示和数据交换，实时显示图像。

S3C2410 中具有内置的 LCD 控制器，它具有将显示缓存(在系统存储器中)中的 LCD 图像数据传输到外部 LCD 驱动电路的逻辑功能。

S3C2410 中内置的 LCD 控制器可支持灰度 LCD 和彩色 LCD。在灰度 LCD 上，使用基于时间的抖动算法(time-based dithering algorithm)和 FRC(Frame Rate Control)方法，可以支持单色、4 级灰度和 16 级灰度模式的灰度 LCD。在彩色 LCD 上，可以支持 256 级彩色，使用 STN LCD 可以支持 4096 级彩色。对于不同尺寸的 LCD，具有不同数量的垂直和水平象素、数据接口的数据宽度、接口时间及刷新率，而 LCD 控制器可以进行编程控制相应的寄存器值，以适应不同的 LCD 显示板。

内置的 LCD 控制器提供了下列外部接口信号：

VFRAME/VSYNC/STV：帧同步信号(STN)/垂直同步信号(TFT)/SEC TFT 信号；

VLINE/HSYNC/CPV：行同步脉冲信号(STN)/水平同步信号(TFT)/SEC TFT 信号；

VCLK/LCD_HCLK：象素时钟信号(STN/TFT)/SEC TFT 信号；

VD[23:0]：LCD 像素数据输出端口(STN/TFT/SEC TFT)；

VM/VDEN/TP：LCD 驱动交流偏置信号(STN)/数据使能信号(TFT)/SEC TFT 信号；

LEND/STH：行结束信号(TFT)/SEC TFT 信号；

LCD_PWREN：LCD 面板电源使能控制信号；

LCDVF0：SEC TFT OE 信号；

LCDVF1：SEC TFT REV 信号；

LCDVF2：SEC TFT REVB 信号。

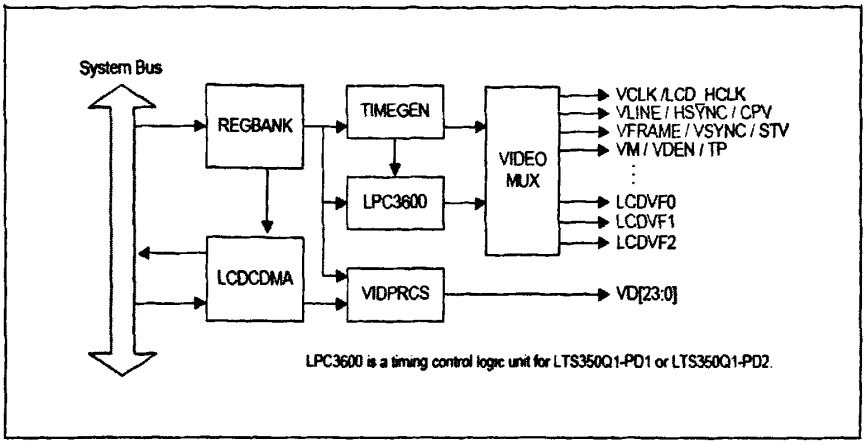

图 3-12 LCD 控制器逻辑框图

Fig.3-12 LCD controller logic frame chart

图 3-12 为 S3C2410 中内置的 LCD 控制器的逻辑框图，它用于传输显示图像数据并产生必要的控制信号，如 VFRAME，VLINE，VCLK，和 VM 等信号。除了控制信号，还有显示数据的数据端口 VD[23:0]。LCD 控制器包含 REGBANK，LCDCDMA，VIDPRCS，TIMEGEN 和 LPC3600。REGBANK 具有 17 个可编程寄存器和 256×16 颜料存储器，用于配置 LCD 控制器。LCDCDMA 为专用 DMA，它可以自动地将显示数据从帧内存中传送到 LCD 驱动器中。通过使用这一专用的 DMA，可以实现在不需要 CPU 介入的情况下显示数据。VIDPRCS 从 LCDCDMA 接收数据，变换为合适的数据格式(比如 4 / 8 位单一扫描和 4 位双扫描显示模式)后通过 VD[23:0]发送到 LCD 驱动器。TIMEGEN 包含可编程的逻辑，以支持常见 LCD 驱动器所需要的不同接口时序、速率要求。TIMEGEN 部分产生 VFRAME，VLINE，VCLK，VM 等信号。

本系统将上述的 24 根数据线和若干根控制线引出，经过 74HC245 隔离后，留作待扩展用。其中对于 256 色 LED 只需要其中低 8 位数据线即可。这些信号线经过 74HC245 隔离后接到 LCD 模块。部分的扩展原理图如图 3-13 所示：

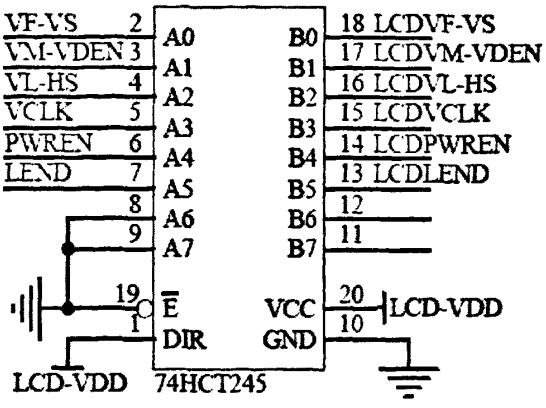

图 3-13 LCD 扩展原理图

Fig.3-13 Expansion diagram for LCD

3.4.5 以太网接口设计

以太网接口控制器主要包括 MAC 和 PHY 两部分。S3C2410 内部没有集成 MAC 控制器，需要外部通过总线扩展以太网接口，通用的方法就是通过 MAC+PHY 的以太网芯片进行扩展。本系统扩展了以太网控制器 RTL8019AS 芯片，为用户提供了一个 10M 的以太网接口，可以完成网络通讯功能、以及实现 FTP、PING 之类的网络服务。

RTL8019AS 通过总线和 S3C2410X 相连，如图 3-14 所示。中断通过 S3C2410X 的外部中断接管。

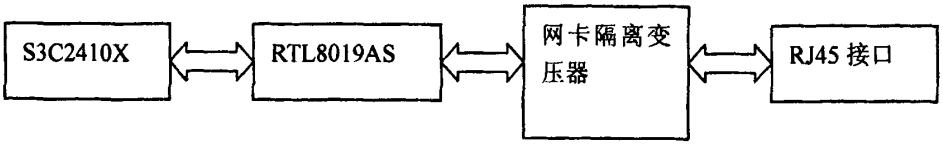

图 3-14 以太网接口连接结构

Fig.3-14 Connection structure of Ethernet interface

RTL8019AS 是一种高度集成的以太网芯片，能简单的实现 Plug and Play。由于它拥有三种等级的掉电模式，所以它是作为绿色电脑的网络设备的理性选择。在全双工模式下，如果是连接到一个同样是全双工的交换机或集成器，就可以实现同时的接收和发送。这个特性虽然不能把传输速率从 10Mbps 提高到 20M bps，但是在执行以太网协议 CSMA/CD 协议时，可以避免更多的冲突的发生。而 Microsoft's Plug and Play 功能就可以减轻用户对资源配置时的烦恼(如 IRQ， I/O Memery Address 等)。又或者在一些特殊的场合，为了与不支持 Microsoft's Plug and Play 功能的器件兼容，RTL8019AS 还可以选择跳线模式和非跳线模式。

为了支持完整的 PnP，RTL8019AS 提供的自动监测接口类型的功能，能自动识别接口是 10BaseT 集成收发器(RJ45)还是 AUI BNC 接口。此外 10BaseT 可以自动修正极性，8 路中断请求和 16 位 I/O 地址都可以根据资源情况机动配置。

RTL8019AS 支持 16K、32K、64K 字节的 BROM，另外还支持 FLASH MEMERY 和页访问方式，最大支持 4M 字节(16K*256)的外存地址。此外还支持在运行完 BROM 后自动释放内存，以供系统其他程序的运行。

3.4.6 串行接口设计

异步串口是 PC 和嵌入式处理器上最常用的资源。几乎所有的微控制器、PC 都提供串行接口，使用电子工业协会(EIA)推荐的 RS-232-C 标准。RS-232-C 标准采用的接口是 9 芯或 25 芯的 D 型插头，以常用的 9 芯 D 型插头为例，各引脚定义如表 3-2 所示。

表 3-2 RS-232-C 9 芯接口的定义

Table 3-2 9-pins interface definition

引脚	名称	功能描述
1	DCD	数据载波检测
2	RXD	数据接收
3	TXD	数据发送
4	DTR	数据终端准备好
5	GND	地
6	DSR	数据设备准备好
7	RTS	请求发送
8	CTS	清除发送
9	RI	振铃指示

本系统扩有二个 UART 端口，分别为 UART0 和 UART1，为标准 9 针 D 型 RS-232C 接口，UART0 作为调试端口，UART1 则作为与其他串口设备的通讯端口。由于 RS-232-C 标准所定义的高、低电平信号与 S3C2410 系统的 LVTTL 电路所定义的高、低电平信号完全不同，LVTTL 的标准逻辑"1" 对应 2V-3.3V 电平，标准逻辑"0"对应 OV-0.4V 电平，而 RS-232-C 标准采用负逻辑方式，标准逻辑"1"对应-5V 至-15V 电平，标准逻辑"0"对应+5V 至+15 V 电平，显然，两者间要进行通信必须经过信号电平的转换，系统使用常见的电平转换芯片 MAX3232，UART0、UART1 的应用电路见图 3-15 所示。

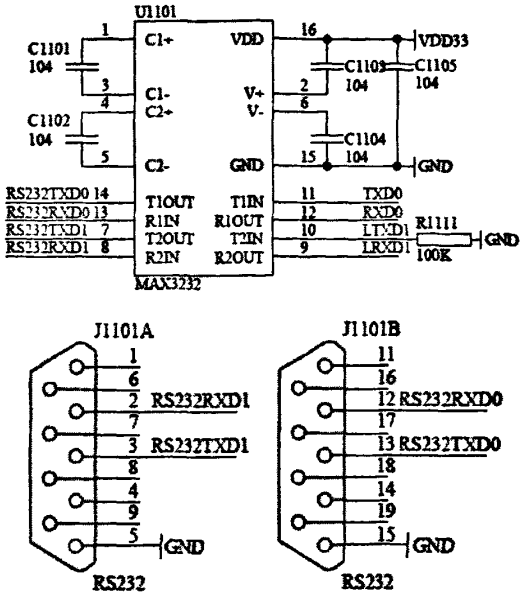

图 3-15 串口接口电路

Fig.3-15 Interface circuit of serial ports

3.4.7 电源和 JTAG

1.电源电路

在本系统中，需要使用 12V、5V、3.3V 和 1.8V 的直流稳压电源，其中 S3C2410 系统内核采用 1.8V 供电，外围采用 3.3V 供电，部分外围器件需 5V 电源，直流电机部分用 12V。为简化系统电源电路的设计，整个系统的输入电压为 12V 的直流稳压电源。电源芯片 LM1085-3.3V 为系统外围电路提供 3.3V 电源，AS1117-1.8V 为内核提供 1.8V 电源，电源电路如图 3-16 所示。扩展板选用 MAX748 DC-DC 转换器完成 5V 到 3.3V 的转换。

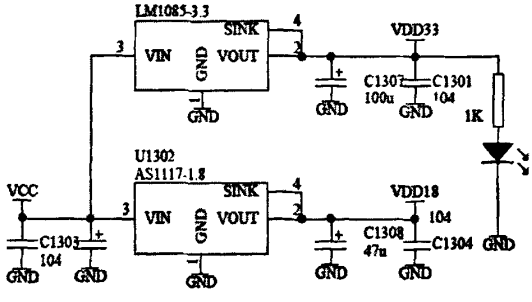

图 3-16 电源转换电路

Fig.3-16 Power conversion circuit

2. JTAG 电路

JTAG 是 Joint Test Action Group(联合测试行动小组)的简称，由于 IEEE 1149.1 标准是由 JTAG 这个组织最初提出的，最终由 IEEE 批准并且标准化。所以 IEEE 1149.1 这个标准一般也俗称为 JTAG 调试标准。

JTAG 标准主要用于芯片内部测试及对系统进行仿真、调试。JTAG 技术是一种嵌入式调试技术，它在芯片内部封装了专门的测试电路 TAP(测试访问口)，通过专门的 JTAG 测试工具对内部节点进行测试。标准的 JTAG 接口包括以下 4 线：TMS(测试模式选择)、TCK（测试时钟)、TDI(测试数据输入)、TDO(测试数据输出)。基于 JTAG 接口的边界扫描测试技术，是在芯片引脚和芯片内部逻辑之间(即芯片边界位置)增加串行连接的边界扫描测试单元，实现对芯片引脚状态的设定和读取，使芯片引脚状态具有可控性和可观测性。JTAG 与 S3C2410 的连接电路如图 3-17 所示[41-45]。

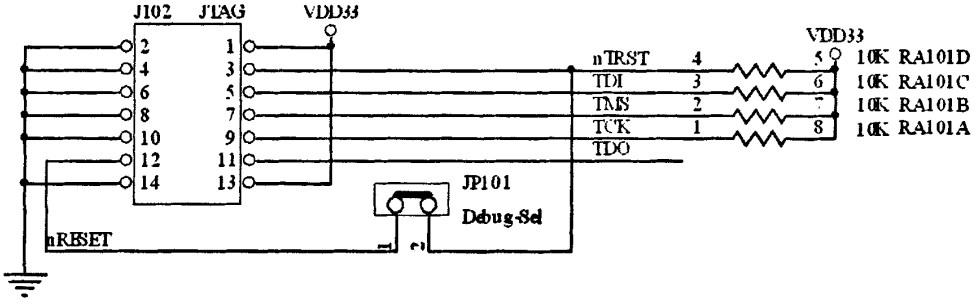

图 3-17 S3C2410X 与 JTAG 连接电路图

Fig. 3-17 The connection circuit diagram of S3C2410X and JTAG

3.5 小结

本章首先介绍了本机器人运动控制系统的功能要求，然后给出整体的硬件设计框图，接着详细介绍电机控制器的设计，包括：功率驱动、控制策略、电流检测和速度检测，和系统硬件设计，包括：存储器扩展、键盘接口、LCD 接口、Internet 网络接口、电源、串行口以及 JTAG。

本章在硬件上对机器人运动控制平台的建立给予了详细的介绍，并为以后系统功能的扩展打下了基础。

第四章 嵌入式系统软件开发平台建立

一个嵌入式linux的开发，根据应用需求的不同有不同的配置开发方法，但是一般都要经过如下的过程：

1. 建立开发环境，PC机上安装LINUX操作系统，安装交叉编译器。
2. 建立引导装载程序BOOTLOADER，根据具体系统进行移植修改。
3. 下载移植好的LINUX操作系统，根据目标平台对源码进行必要的修改(主要是体系结构相关的部分)，添加需要的硬件驱动程序，打造成一款适合目标平台的新的操作系统。。
4. 建立根文件系统，使用BUSYBOX软件进行功能裁减，产生一个最基本的根文件系统。

4.1 建立交叉编译环境

首先，在宿主机(一台2.8GHz、512MB内存的Intel处理器PC)上安装标准Linux 操作系统（Redhat 9.0版本），确保计算机的网卡驱动、网络通讯配置正常。第二，宿主机上要安装交叉编译器。交叉编译就是在一个平台上生成可以在另一个平台上执行的代码。由于在目标板上无法安装编译器，因此要借助于宿主机，在宿主机上对即将运行在目标机上的应用程序进行编译，生成可在目标机上运行的代码格式。本系统使用了博创公司提供的工具包，安装了所需的各种工具软件，其中arm Linux编译器为arm4l-unknown-linux-gcc，它是gcc的arm改版，开发库包括arm4l C库和arm4l C库头文件。

本系统采用NFS方式对嵌入式Linux的应用程序进行开发，需要对宿主机 (NFS 服务器)和目标板（S3C2410系统）进行设置，步骤如下：

（1）编写etc/exports文件，格式如下：

/arm2410 192.168.0.*(rw,synch)

含义是允许ip 地址范围在192.168.0.1-192.168.0.254的计算机，可以以读写的权限来访问/ arm2410目录。也称为服务器输出共享目录。

（2）接着执行如下命令，启动端口映射：

/etc/rc.d/init.d/portmap start

（3）执行如下命令启动NFS 服务，此时NFS 会激活守护进程，然后就开始监听 Client 端的请求：

/etc/rc.d/init.d/nfs start

对目标板进行设置，步骤如下：

（1）内核需要支持NFS，进入内核源码目录，make menuconfig

Enable:

File Systems -->

Network File Systems -->

NFS file system support

Provide NFSv3 client support

配置完后，重新编译内核

（2）在嵌入式目标系统的Linux Shell 下，执行如下命令来进行NFS 共享目录挂载：

mkdir /mnt/nfs //建立Linux服务器输出共享目录的挂载点；

#mount –t nfs 192.168.0.20:/arm2410 /mnt/nfs

此时，嵌入式目标系统端所显示的内容即为Linux 服务器的输出目录的内容，即Linux 服务器的输出目录/arm2410 通过NFS 映射到了嵌入式目标系统的/mnt/nfs 目录[47]。

4.2 BootLoader 程序

BootLoader 主要是对系统进行初始化、系统引导、Flash操作等功能。本系统选用了支持Nand Flash启动的BootLoader—VIVI。主要是对系统的硬件进行初始化，包括时钟和PLL、定时器、调试串口（Debug UART）等等。

进入VIVI界面后，可以通过VIVI的命令，对VIVI 的启动参数进行设置。如启动延时时间，更改linux 的console，还有就是对Nand Flash进行分区。在运行linux时，需要挂载根文件系统cramfs以及用户的yaffs文件系统。因此在烧写linux内核文件zImage和根文件系统cramfs前，需要对nand flash进行分区。表4-1为本系统的nandflash 分区表。块地址和线性地址对应关系为：将块地址乘以nand flash每块的容量16KB 即N*16KB，就可以得到线性地址。

Nand Flash分区命令为：bon part 0 192K 1245184K 3270246K:m

表4-1 Nand Flash 分区表

Table.4-1 Subarea of Nand Flash

序号	起始块	对应线性地址	容量大小	存放内容
0	0	0	192KB	VIVI
1	12	192k	1M	zImage
2	76	1245184	30M	Cramfs
3		32702464	32M	Yaffs

4.3 Linux 移植

所谓 Linux 移植就是把 Linux 操作系统针对具体的目标平台做必要改写之后，安装到该目标平台使其正确的运行起来。其基本内容是：获取某一版本的 Linux 内核源码，根据具体目标平台对这源码进行必要的改写（主要是修改体系结构相关部分），然后添加一些外设的驱动，打造一款适合于目标平台的新操作系统，对该系统进行针对我们目标平台的交叉编译，生成一个内核映象文件，最后通过一些手段把该映象文件烧写（安装）到我们目标平台中。内核的裁减和移植工作需要在宿主机上完成。

4.3.1 Linux 内核文件的修改

(1) 设置目标平台和指定交叉编译器

在根目录的 makefile 文件中，作如下改动：

ARCH:= arm

CROSS_COMPILE= /opt/host/armv41/bin/armv41-unknown-linux-gcc

(2) 修改 arch/arm/boot 中的 init.S 文件

由汇编写的文件，是引导 Linux 内核在 ARM 平台上启动的初始化代码。里面定义了一个全局符号_start，它定义了默认的起始地址。同时也是整体内核二进制镜像的起始标志。主要完成的功能如下

1) 定义数据段、代码段、bbs 起始地址变量并对 bss 段进行初始化
2) 设置寄存器以初始化系统硬件
3) 关闭中断
4) 初始化 LCD 显示
5) 将数据段数据复制到内存
6) 跳转到内核起始函数 start_kernel 继续执行
7) 对主寄存器的修改

(3) 修改 arch/arm 目录下的 makefile 文件

系统的起始代码是通过这个 makefile 文件产生。在 Linux-2.4.18 内核中添加：

Ifeq($(CONFIG_ARCH_S3C2410) ,y)

TEXTADDR= 0xC0008000

MACHINE= s3c2410

Endif

TEXTADDR 确定内核开始运行的虚拟地址。

(4) 修改 arch/arm 目录下 config.in 文件

配置文件 config.in 能够配置运行“make menuconfig”命令时的菜单选项，由于 Linux-2.4.18 内核中没有 S3C2410 的相关信息，所以要在该文件中进行有效的配置。完成该步，在 Linux 内核配置时就可以选择刚刚加入的内核平台了。

(5) 修改 arch/arm/boot 目录下的 makefile 文件

编译出来的内核存放在，这里指定内核解压到实际硬件系统的物理地址

```
ifeq($ (CONFIG_ARCH_S3C2410),y)
ZTEXTADDR= 0x30008000
ZRELADDR= 0x30008000
endif
```

要根据实际的硬件系统，修改解压后内核开始运行的实际物理地址

(6) 修改 arch/arm/boot/compressed 目录下的 makefile 文件

该文件的功能是从 vmlinux 中创建一个压缩的 vmlinux 镜像文件。该文件中用到的 SYSTEM、ZTEXTADDER、ZBSSADDR 和 ZRELADDR 是从 arch/arm/boot/Makefile 文件中得到的。添加如下代码：

```
ifeq ($(CONFIG_ARCH_S3C2410).y)
OBJS+= head_-s3c2410.o
endif
```

(7) 在 arch/arm/def-configs 目录下添加 head-s3c2410.s 文件

该文件主要用来初始化处理

(8) 在 arch/arm/def-configs 目下添加配置好的 S3C2410 的配置文件

(9) 修改 arch/arm/kernel 目录下的 makefile 文件

该文件主要用来确定文件类型的依赖关系。修改如下：

```
No-irq-arch:= $ (CONFIG_ARCH_INTEGRATOR) $ (CONFIG)ARCH_CLPS711X) \
$ (CONFIG_FOOTBRIDGE) $ (CONFIG_ARCH_EBSA110) \
$ (CONFIG)ARCH_SA1100) $ (CONFIG_ARCH_CAMELOT) \
$ (CONFIG_ARCH_S3C2400) $ (CONFIG_ARCH_S3C2410) \
$ (CONFIG_ARCH_MXIADS) $ (CONFIG_ARCH_PXA)
```

(10) 修改 arch/arm/kernel 目录下的 debug-armv.s 文件

在该文件中添加如下代码，目的是关闭外围设备的时钟，以保证系统正常运行。此文件修改如下：

```
#elif defined(CONFIG_ARCH_S3C2410)
  .macro addruart,rx
  mrc    p15,0,\rx.c1,c0
  tst    \rx,#1              @MMU enabled ?
  moveq \rx,#0x50000000    @ physical base address
```

```
.endm
.macro  senduart ,rd,rx
str     \rd ,[\rx，#0x20]   @UTXh
.endm
.macro    busyuart,rd,rx
1001:  ldr\rd, [\rx, #0x10]        @ read UTRSTAT
       tst      \rd, #1<<2         @ TX_EMPTY?
       beq   1001b
       .endm
```

(11) 修改 arch/arm/kernel 目录下的 entry-armv.s

该文件是 CPU 初始化时处理中断的汇编代码，需要做如下修改：

```
#elif define (CONFIG_ARCH_S3C2410)
#include < asm/hardware.h>
        .macro disable_fiq
        .endm
       .macro get_irqnr_and_base, irqnr, irqstat, base, tmp
       mov r4, #INTBASE     @ virtual address of IRQ registers
       ldr   \irqnr, [r4, #0x8]   @ read INTMSK
       ldr   \irqstat, [r4,#0x10] @ read INTPND
       bics \irqstat, \irqstat, \irnr
       bics \irqstat, \irqstat, \irqnr
       beq 1002f
       mov \irqnr, #0
1001:   tst\irqstat, #1
       bne 1002f@ found IR
       add \irqnr, \irnr' #1
       mov \irqstat, \irqstat, lsr #1
       cmp \irqnr, #32
       bcc 1001b
1002:
       .endm
       .macro ir_prio_table
       .endm
```

(12) 修改 arch/arm/mm 目录下的相关文件

此目录下包含移植好的有关 ARM 内存管理的代码。在 mm-armv.c 中，使

“init_maps->bufferable=1;”即可。Init_maps 是一个 map_desc 型的数据结构，nao_desc 的定义在/include/asm-arm/mach/map.h 文件中。

(13) 修改 arch/arm/mach-s3c2410 目录下的相关文件。

此目录下包含的代码是专门针对 S3C2410 处理器的

4.3.2 内核的编译

第一步首先通过 make menuconfig 命令进入配置界面，进行配置和功能的裁减。然后执行 make dep(仅在第一次编译时需要，以后就不需要了，为的是在编译时知道文件之间的依赖关系，在进行了多次编译后，make 会根据这个依赖关系确定哪些文件需要重新编译，那些文件可以跳过)。接着执行 make clean：清除以前编译内核时生成的所有目标文件、模块文件和临时文件。最后执行 make zImage：编译内核。编译通过后，在目录 arch/arm/boot 下生成 zImage 内核文件。

4.4 根文件系统的制作

文件系统就是文件和目录按一定的组织结构组织起来的实体。嵌入式linux 中需要需要在flash(本系统中在nand flash)上创建文件系统，才能被linux 识别和使用。Linux 支持绝大多数文件系统，本系统在nand flash 中创建cramfs 文件系统，该文件系统是只读的，有利于保护内核等重要内容不被意外修改。本系统采用linux作为操作系统，对 cramfs 有很好的支持。在配置内核时，选择 File systems→Miscellaneous filesystems→Compressed ROM file system support (cramfs)，系统内核就可得到cramfs 文件系统的支持。

本系统通过 BusyBox-1.00 工具对已有的根文件系统进行修改，根据系统的功能要求对 BusyBox 进行配置和裁剪。选择完成所需的全部命令后，保存配置文件退出，并执行如下命令进行编译：

#make

#make install

当命令执行完毕以后，会在当前目录中会生成“_install”目录，进入该目录，可以看到其中有四个子目录：bin、linuxrc、sbin和usr。将文件系统源码的压缩包root-2410.tar.gz和mkcramfs工具拷贝到宿主机的某个目录，然后执行如下命令：

chmod 777 mkcramfs //改变该文件属性；

tar root-2410.tar.gz //解压文件包；

将_isntall目录中的文件拷贝到root-2410目录，覆盖原有文件，然后执行如下命令：

./mkcramfs ./root-2410 ./root-2410.cramfs

该命令执行以后，就会在当前目录下生成新的根文件系统root-2410.cramfs。

4.5 内核和根文件系统的烧写

在超级终端的vivi命令行下输入“load flash kernel x”，将通过xmodem协议发送kernel文件：zImage。

在超级终端的vivi命令行下输入“load flash root x”，将通过xmodem协议发送root文件：root-2410.cramfs[48-51]。

4.6 小结

本章基于系统平台，介绍了嵌入式 Linux 开发交叉编译环境的建立、引导程序bootloader、内核的移植(内核文件修改、编译、烧写)和根文件系统的制作、编译和烧写，为应用开发打下了基础。

第五章 机器人运动控制系统软件设计

5.1 Linux 下直流电机的设备驱动程序

5.1.1 Linux 设备驱动概述

Linux 下的设备驱动程序是内核的一部分，运行在内核模式，操作系统通过各种驱动程序来控制硬件设备，设备驱动程序为内核提供了一个 I/O 接口，用户使用这个接口实现对设备的操作。图 4-1 为 Linux 输入/输出系统层次结构和功能。

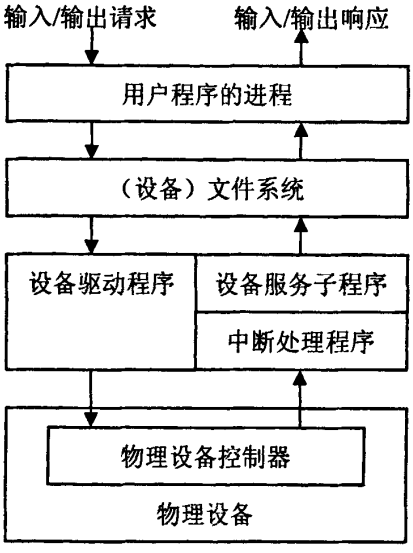

图 4-1 Linux 输入/输出系统层次结构和功能
Fig.4-1 Hiberarchy of input and output system for Linux

Linux 定义的主要设备类型：字符设备、块设备及网络设备。

(1) 字符设备是指能像字节流（如文件）一样被访问的设备。

(2) 块设备则仅以数据块为单位进行读写，典型的块大小为 512 或 1024 字节，块设备的存取一般是通过缓冲方式来进行的。

(3) 网络设备，是指系统的网卡，包括实际设备和虚拟设备。

Linux 为每种不同类型的设备驱动程序维护相应的数据结构，以便定义统一的接口并实现驱动程序的可装载性和动态性。

5.1.2 电机的驱动程序分析

由第三章的直流电机驱动硬件设计可知，本系统是通过改变输出信号占空比的方法来调节电机转速，改变占空比有两种调制方法。一种是开关周期恒定，通过改变导通脉冲宽度来改变占空比的方式，即脉冲宽度调制(Pulse Width Modulation 缩写为PWM)；另一种方式为导通脉冲宽度恒定，通过改变开关频率($f=\frac{1}{T}$)来改变占空比，亦即脉冲频率调制(Pulse Frequency Modulation，缩写为 PFM)。由于 PFM 控制是依靠脉冲频率来改变占空比德，当遇到某个特殊的频率下的机械谐振时，常导致系统震动和出现音频啸叫声，这一严重的缺点导致 PFM 控制在伺服系统中不使用。本系统采用第一种即 PWM 方式，使用 PWM timer0 和 PWM timer2 来产生两路 PWM 信号，来控制两侧的电机。下面以其中的一路 PWM timer0 来介绍 Linux 下电机设备驱动程序的编写。

电机的设备驱动实际为操作定时器产生相应的 PWM 信号，属于字符设备，字符设备驱动程序是 Linux 系统最基本、最常用的驱动程序结构。通常字符设备提供给应用程序的是一个流控制接口，主要包括 open、close(或 release)、read、write、ioctl、poll 和 mmap 等。在系统中添加一个字符设备驱动程序实际上就是给上述操作添加对应的代码。Linux 内核把对字符设备和块设备的这些操作进行了统一的抽象，把它们定义在结构体 file_operations 中。对于 Linux2.4.19 内核，结构体 file_operations 的定义(include/linux/fs.h)如下：

```
struct file_operations {
        struct module *owner;
        loff_t (*llseek) (struct file *, loff_t, int);
        ssize_t (*read) (struct file *, char *, size_t, loff_t *);
        ssize_t (*write) (struct file *, const char *, size_t, loff_t *);
        int (*readdir) (struct file *, void *, filldir_t);
        unsigned int (*poll) (struct file *, struct poll_table_struct *);
       int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long);
        int (*mmap) (struct file *, struct vm_area_struct *);
        int (*open) (struct inode *, struct file *);
        int (*flush) (struct file *);
        int (*release) (struct inode *, struct file *);
        int (*fsync) (struct file *, struct dentry *, int datasync);
        int (*fasync) (int, struct file *, int);
        int (*lock) (struct file *, int, struct file_lock *);
        ssize_t (*readv) (struct file *, const struct iovec *, unsigned long, loff_t *);
```

```
        ssize_t (*writev) (struct file *, const struct iovec *, unsigned long, loff_t *);
        ssize_t (*sendpage) (struct file *, struct page *, int, size_t, loff_t *, int);
        unsigned long (*get_unmapped_area)(struct file *, unsigned long, unsigned
long, unsigned long, unsigned long);
};
```

5.1.3 电机的设备驱动程序设计

上述 struct file_operations 中的成员很多，对设备驱动而言并不是每个都要实现，尤其对于嵌入式设计，在资源及其珍贵的情况下，这点更加重要。对本系统的电机驱动程序分以下几部分来完成

(1) 初始化部分

```
int __init s3c2410_dcm_init(void)
{
    int ret;
   static devfs_handle_t devfs_dcm_dir, devfs_dcm0;
    ret = register_chrdev(0, DEVICE_NAME, &s3c2410_dcm_fops);
    if (ret < 0) {
        DPRINTK(DEVICE_NAME " can't get major number\n");
        return ret;
    }
    dcmMajor=ret;
#ifdef CONFIG_DEVFS_FS
    devfs_dcm_dir = devfs_mk_dir(NULL, "dcm", NULL);
    devfs_dcm0 = devfs_register(devfs_dcm_dir, "0raw", DEVFS_FL_DEFAULT,
                dcmMajor, DCMRAW_MINOR, S_IFCHR | S_IRUSR | S_IWUSR,
&s3c2410_dcm_fops, NULL);
#endif
    DPRINTK (DEVICE_NAME"\tdevice initialized\n");
    return 0;
}
```

首先使用 register_chrdev()函数向内核注册一个 Linux 字符驱动程序，它在 fs/device.c 中被定义，在此，该函数返回一个系统自动分配的未使用的主设备号给 dcmMajor，CONFIG_DEVFS_FS 来判断系统在配置内核时是否使用了设备文件系统(devfs)，也就是第四章中说明的在编写设备驱动程序时要考虑对设备文件系统支持。devfs_mk_dir() 函数在设备文件系统中创建一个名为 dcm 的目录，devfs_register() 函数创建名为 0 的设备文件节点，并赋予相应的权限。主设备号是 register_chrdev()自动分配的，次设备号被 DCMRAW_MINOR 定义，这样，系统在挂载(mount)设备文件系统到/dev 目录后，就会看到有/dev/dcm/0 这个字符设备存在。

```
void __exit s3c2410_dcm_exit(void)
{
#ifdef CONFIG_DEVFS_FS
    devfs_unregister(devfs_dcm0);
    devfs_unregister(devfs_dcm_dir);
#endif
    unregister_chrdev(dcmMajor, DEVICE_NAME);
}
```

此函数在模块卸载时候被调用，用于注销设备、释放资源和主设备号。

```
module_init(s3c2410_dcm_init);
module_exit(s3c2410_dcm_exit);
```

对于编写好的初始化和退出函数一定要用宏 module_init()、module_exit()对函数名称进行记录，对于模块加载的驱动程序，他们指定了加载(或卸载)模块时调用的函数；对于内核中的驱动程序，他们让编译器把驱动程序的入口函数放到一个特殊的段(section)中，在内核启动过程中，专门有代码负责加载这些驱动程序。

module_init()、module_exit() 定义在内核源码 include/linux/init.h 中。在 Linux2.4 和 2.6 内核中，几乎所有 Linux 驱动程序都是依靠 module_init 宏形式被加载的。

(2) file_operations 中函数的实现

设备驱动程序中需实现 file_operations 数据结构中几个最为关键的函数，表识如下，在 GUN C 中，允许使用冒号(：)为特定的结构体赋值，其他成员值为零。

```
static struct file_operations s3c2410_dcm_fops = {
    owner:  THIS_MODULE,
    open:   s3c2410_dcm_open,
    ioctl:  s3c2410_dcm_ioctl,
    release: s3c2410_dcm_release,
};
```

owner-所属的设备模块，open-设备被打开时执行的操作(应用程序执行 open 系统调用)；release-设备被关闭时执行的操作(应用程序执行 close 系统调用)；ioctl-执行对设备控制的操作(应用程序执行 ioctl 系统调用)。下面逐个分析：

```
static int s3c2410_dcm_open(struct inode *inode, struct file *filp)
{
    MOD_INC_USE_COUNT;
    DPRINTK( "S3c2410 DC Motor device open!\n");
    tout0_enable();
    dcm_start_timer();
    return 0;
```

```
}
```

该函数在设备被打开时调用，MOD_INC_USE_COUNT 宏在/include/linux/module.h 中定义。负责记录设备模块使用情况，打开则增加，防止当有应用程序在使用驱动程序时，此模块被意外的卸载。

tout0_enable() 该函数完成处理器 IO 口的初始化，初始化为定时器输出端口，用于输出 PWM 波，dcm_start_timer(); 初始化定时器，装在计数计时器和比较寄存器，并启动定时器，在本系统，主频 FCLK=202M，FCLK：HCLK：PCLK=1：2：4，其中 PCLK 为送到 PWM timer 的时钟为 50M，经过 PWM timer 配置寄存器分两次频后，时钟在 8M 左右，设置的 TCNTB0=4000，按照设置，PWM 波的周期在 5ms 左右。设置 TCMPB0 在 4000-0 之间，会产生占空比在 100%-0%之间的 PWM 信号。

```
static int s3c2410_dcm_ioctl (struct inode *inode, struct file *filp, unsigned int cmd,
unsigned long arg)
{
    switch(cmd){
    case   DCM_IOCTRL_SETPWM:
        return dcm_setpwm((int)arg);
   default:
            return -EINVAL;
    }
    return 0;
}
```

此函数在应用程序调用 ioctl 时被调用，用于传递参数控制设备。dcm_setpwm() 函数实时的改变 TCMPB0 比较寄存器的值，来调节 PWM 占空比，达到调节电机转速的目的。

```
static int s3c2410_dcm_release(struct inode *inode, struct file *filp)
{
    MOD_DEC_USE_COUNT;
    DPRINTK( "S3c2410 DC Motor device release!\n");
    tout01_disable();
    dcm_stop_timer();
    return 0;
}
```

该函数在关闭设备时被调用，同样 MOD_DEC_USE_COUNT 宏用来记录设备的使用情况，后两个函数用于完成撤销 Open 系统调用完成的操作[52-56]。

5.2 PID控制算法及其改进

PID控制算法由于其算法简单、鲁棒性好和可靠性高，被广泛用于工业控制，尤其适用于能建立精确数学模型的确定性系统。在控制理论和技术迅速发展的今天，PID控制仍然具有极强的生命力，也是伺服系统首选的控制策略之一。

根据前面第二章的介绍有，数字式PID控制算法公式如下

$$u(k)=K_p e(k)+K_i\sum_{i=0}^{k}e(i)+K_d(e(k)-e(k-1)) \quad (2.12)$$

此为位置式PID控制算法。由(2-12)可推导出下式：

$$u(k-1)=K_p e(k-1)+K_i\sum_{i=0}^{k-1}e(i)+K_d(e(k-1)-e(k-2)) \quad (2.13)$$

令$\Delta u(k)=u(k)-u(k-1)$则有：

$$\Delta u(k)=K_p(e(k)-e(k-1))+K_i e(k)+K_d(e(k)-2e(k-1)+e(k-2)) \quad (2.14)$$

上式称为增量式PID算法，可将此进一步改写为：

$$\Delta u(k)=Ae(k)+Be(k-1)+Ce(k-2) \quad (2.15)$$

式中$A=K_p\left(1+\frac{T}{T_i}+\frac{T_d}{T}\right)$，$B=-K_p\left(1+\frac{2T_d}{T}\right)$，$C=K_p\frac{T_d}{T}$

由(2.14)以及(2.15)可得：

$$u(k)=Ae(k)+Be(k-1)+Ce(k-2)+u(k-1) \quad (2.16)$$

从上面分析可知，增量式PID算法由于控制算法中不需要累加，控制增量$\Delta u(k)$仅与最近几次的采样有关，所以误差动作时影响小，而且较容易通过加权处理获得比较好的控制效果。本系统就选用增量式PID作为控制算法。

另外，在速度控制中为了减少电机在运行过程中积分校正对系统动态性能的影响，在电机的启停阶段或大幅度加减速时，采用积分分离PID控制算法，即只加比例、微分运算，取消积分校正。而当电机的实际速度与给定速度偏差小于一定值时，则恢复积分校正作用。积分分离的算法具体实现如下：

(1) 根据要求，人为设定最大偏差 e

(2) 当[e(k)]>e，即偏差[e(k)]比较大时，采用PID控制，不进行积分运算，只进行比例预算，即把积分项分离出去。这样可避免过大的超调，又使系统有较快的响应

(3) [e(k)]<<e，即偏差[e(k)]比较小时，采用PID控制，可以保证系统的控制精度。

积分分离 PID 控制算法程序流程图，如下图 4-2 所示，分析与实验表明，积分分离 PID 控制算法显著降低了被控制量的超调量和过渡时间，改善了调节性能。

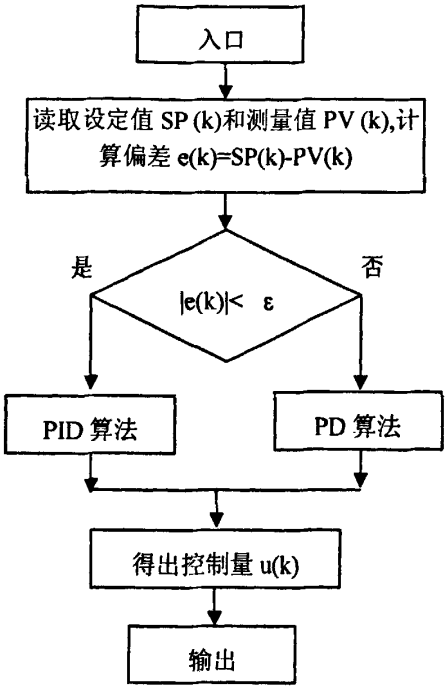

图 4-2 积分分离 PID 算法流程图

Fig.4-2 Flow chart of PID algorithm(integral separation)

5.3 控制方式切换程序设计

电机的运行是靠 PWM 脉冲波来驱动，本系统中设定的 PWM 脉冲周期为 5ms，在每个 PWM 周期末，输出的是高电平 ，此时的电流是电机绕组中最大电流，采集此时的电流，再按照已知的 PWM 周期即可算出绕组的平均电流。电流环采样的周期在本系统中设定为两个 PWM 周期，即为 10ms，速度的采样周期设为 40ms，用两个全局变量 Count_Current 和 Count_Speed 来实现，Count_Current 用来维护电流环控制周期，Count_Speed 用来维护速度环控制周期，定时器中断服务程序每运行一次，Count_Current 和 Count_Speed 就自加一次，如图 4-3 所示。

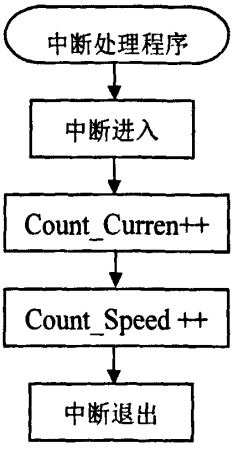

图 4-3 定时器中断子程序
Fig.4-3 Interrupt program of timer

当 Count_Current=2 时，调用电流环控制子程序并将 Count_Current 清零，当 Count_Speed=8 时调用速度环控制子程序，速度环子程序调用之后将 Count_Speed 清零。这样就实现了 10ms 电流环采样控制和 40ms 的速度环采样控制。流程图 4-4 如下。

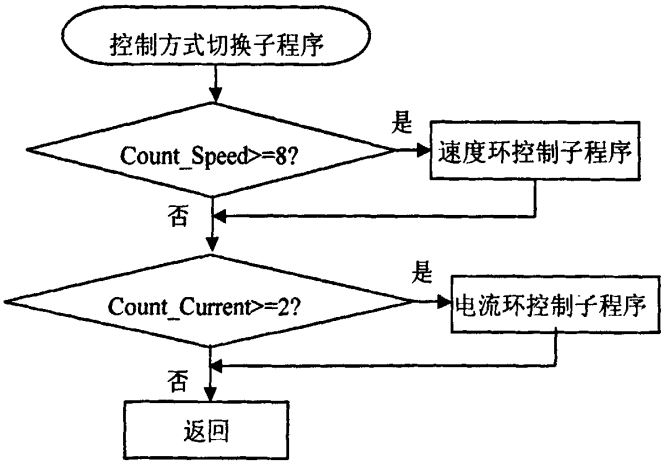

图 4-4 控制方式切换流程图
Fig.4-4 Flow chart of control mode change

通过改变此处的设置量，可以灵活的改变电流控制周期和速度控制周期。

5.4 电流环控制子程序设计

电流环控制子程序要实现的功能为：读 A/D 转换结果得到电机绕组电流测量值，根据速度调节得到的电流参考值和电流测量值进行比较，对得到的偏差值再经过电流 PI 控制，最后得到的控制变量改变 PWM 的占空比。电流环子程序流程如图 4-5 所示：

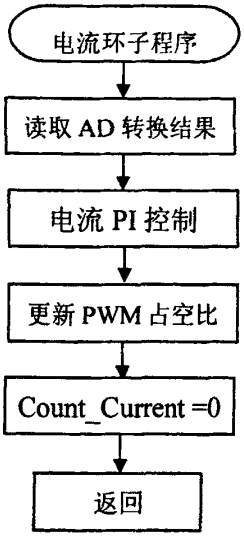

图 4-5 电流环控制子程序

Fig.4-5 Subroutine of current loop

AD 部分采用慢速 AD7790 来实现，提前启动转换，当两个 PWM 周期之后，读取转换结果，此时的电流值是前两个 PWM 周期内的峰值，这种方式使得 AD 转换始终在功率开关器件导通周期最后进行电流采样，保证了同步性，避免功率开关器件过电流，保证了系统安全。

AD7790 是一款 16 位精度的 AD 转换芯片，通过 SPI 接口和处理器相连，其操作流程图如图 4-6

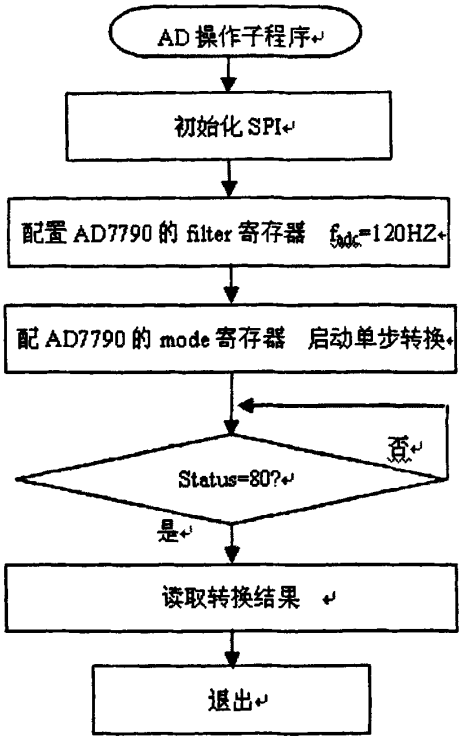

图 4-6 AD779 操作流程图

Fig. 4-6 Flow chart of AD7790 operation

5.5 速度环控制子程序设计

速度环采用积分分离 PID 控制。积分分离的具体内容如上述所述。在实现上本系统在积分项上乘以一个系数 β

$$\beta=\begin{cases}1 & |e(k)|\le\varepsilon\\ 0 & |e(k)|>\varepsilon\end{cases}$$

控制器中设置 ε =20，也就是说当当电机实际速度与给定速度的偏差绝对值 err>20 时，采用 PD 控制，消除积分校正对控制系统动态性能的影响，使系统有较快的反应。当 err<20 时，又采用 PID 控制，以保证系统的控制精度，最后输出的参考电流 Is 经过限幅处理后送入电流调节环。速度调节流程图如图 4-7 所示：

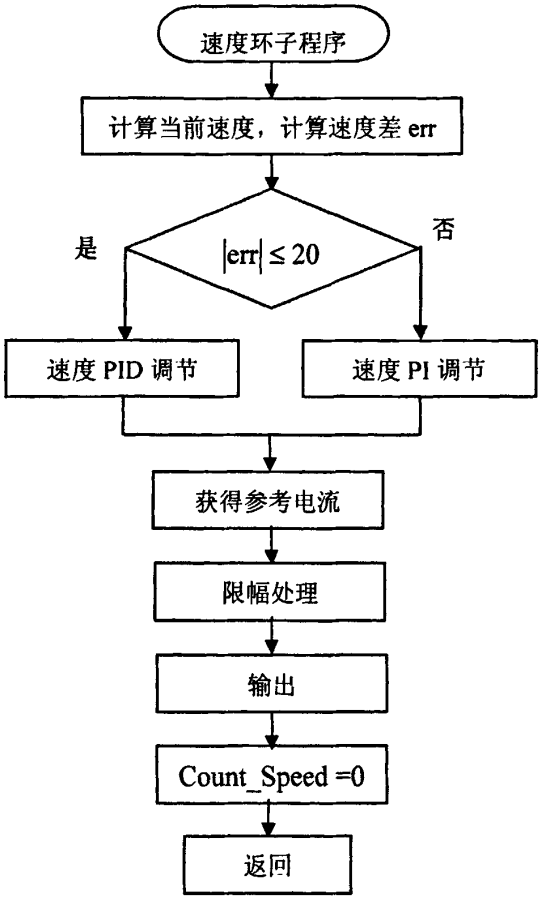

图 4-7 速度环控制子程序

Fig. 4-7 Control subroutine of velocity loop

在检测速度上，本设计采用速度码盘。在电机闭环控制系统中，根据脉冲计数来测量转数的方法主要有 M 法测速、T 法测速和 M/T 法测速。

(1) M 法测速：在规定的时间间隔内，测量所产生的脉冲数 m，来获得被测速度值，适合于高速测量场合。

(2) T 法测速：测量相邻两个脉冲的时间间隔来确定被测速度的方法叫做 T 法测速，适合于低速时测量。

(3) M/T 法测速：M/T 法是同时测量检测时间和在此检测时间内脉冲发生器发送的脉冲数来确定被测转速，兼有 M 法和 T 法的优点。

综合对比三种方法的优缺点，以及本系统电流控制周期和速度控制周期的特点，本设计采取 M 法测速[1]。初始化之后，在每个速度控制周期时调用读取位置计数器的数值程序模块，读完之后，复位 HCTL_2020 开始下一个周期的计数，T=40ms..在周期 T 之后读取可逆位置计数器中数值，通过读取 U/D 管脚的状态(CHA 超前 CHB 时

为 1，此时正转)，得知其是增计数还是减计数。

其操作流程图如图 4-8、4-9 所示：

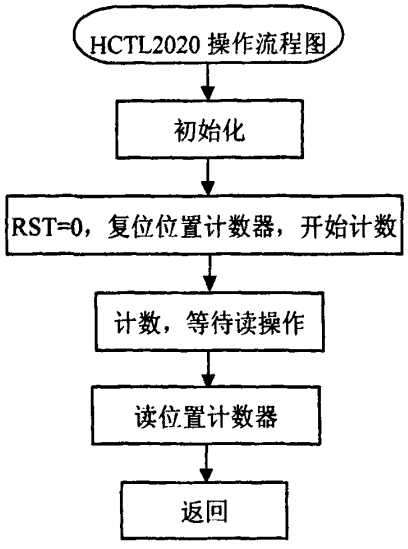

图 4-8 HCTL2020 操作流程图

Fig.4-8 Flow chart of HCTL2020 operation

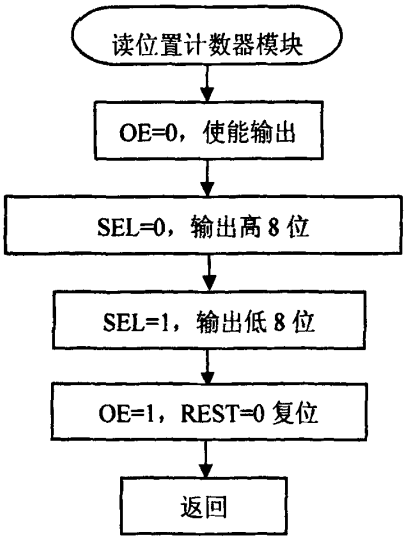

图 4-9 读位置计数器操作流程图

Fig.4-9 Flow chart of reading position counter

5.6 系统调试与实验

系统的调试分两部分，第一部分为 Linux 系统下，设备驱动程序的调试，也就是本章第一节介绍的直流电机驱动程序。第二部分为系统性能的调试，也即所设计的机器人运动控制器的性能试验。

5.6.1 驱动程序的调试

驱动程序的调试一般都是在模块加载的方式下进行，可在不重新启动系统的前提下，反复地调试和修改模块，直至整个驱动可以满足要求，再把它编译到内核中。具体操作步骤如下：

1）修改相应的通过修改相应的 makefile 文件，建立起所写的电机设备驱动程序的 makefile 文件，将设备驱动程序编译成模块 s3c2410-dc-motor.o

2)ipconfig 设置目标板的 ip 192.168.0.20

3）挂载目标板和主机 mount -t nfs 192.168.0.20:/arm2410 /host

4) insmod s3c2410-dc-motor.o 向内核插入驱动模块。 使用 lsmod 列出所有插入的模块，察看 s3c2410-dc-motor.o 的插入情况

5）编写 test.c 测试程序。本系统所编写的 test.c 测试程序完成的功能是：打开电机驱动模块，延时 10ms，一步步地递增增加 PWM 的占空比，从 0%-100%

6）编译 test.c 程序，挂载运行

运行 test.c 可以看出电机的速度逐步稳定的提升，在 PWM 占空比为 100%，电机表现的有些不稳定。从试验现象看，电机的驱动程序安全的进入内核，和应用程序实现了交互，基本上完成功能，为以后复杂的功能设计打下了基础。

最后把驱动程序连接到内核

在 drivers/char/Config.in 文件中添加：

```
dep_tristate 'S3C2410 DC support' CONFIG_S3C2410_DC
$CONFIG_ARCH_S3C2410
```

其含义是：只要定义了 CONFIG)ARCH_S3C2410 为 y 或 m，在配制内核时(make menuconfig)，Character devices 分类下，就会出现 S3C2410 DC support 选项，它对应了 CONFIG_S3C2410_DC 的定义。把驱动程序编译为模块时，CONFIG_S3C2410_DC 定义为 m，把驱动程序连接到内核中，CONFIG_S3C2410_DC 定义为 y。在此，我们选择把驱动程序连接到内核中，此时 CONFIG_S3C2410_DC 定义为 y。

在 derivers/char/Makefile 文件中添加：

```
obj-$ (CONFIG_S3C2410_DC) += s3c2410-dc-motor.o
```

Makefile 会根据 obj-m 和 obj-y 编译并连接对应的源码。由上面所叙，此时

CONFIG_S3C2410_DC 定义为 y。obj-y 会在 make 或 make zImage 时被编译。这样，直流电机驱动程序就会在启动时直接被内核加载。

实验表明，所设计的驱动程序，实现了预期功能，满足要求。

5.6.2 系统实验

在以上理论研究的基础上，我们开始机器人实际运动的实验，将移动机器人的运动轨迹分为直线、旋转和圆弧三种，分别进行如下实验和误差分析。

1、直线运动

当差动轮式移动机人左右驱动轮的伺服控制电路参数和直流电机型号参数相同时，若给定电机的控制量大小相等且方向相同时，移动机器人的运动轨迹应为直线。对于本系统而言，左轮轮径为 160.8mm，右轮轮径为 161.1mm，而且由于移动机器人两驱动轮与驱动轴直径的同轴精度不高、负载分布不均等问题,移动机器人的直线运动控制会产生偏差。

在程序中我们设定周期计数寄存器 TCNTB0=40000，通过改变 TCMPB0 的值来控制占空比，从而改变两驱动轮的速度。下表 4-1 为在给定相同控制量的情况下测得的左右轮速度。

表 4-1 机器人直线运动试验数据表

Table.4-1 Experimental data table of robot linear movement

TCMPB	占空比	左轮(r/min)	右轮(r/m)	左轮(cm/s)	右轮(cm/s)
36000	90%	46.42	44.3	43.26	39.68
32000	80%	40.84	38.21	36.52	34.2
28000	70%	34.87	32.24	31.18	28.89
24000	60%	28.54	26.57	25.52	23.8
20000	50%	22.84	21.32	20.42	19.1
16000	40%	16.23	15.43	14.51	13.82
12000	30%	11.43	10.84	10.22	9.71
8000	20%	7.42	7.15	6.63	6.4
4000	10%	3.46	3.45	3.09	3.35

根据以上测量数据，我可以绘出如图 4-10 所示的左右轮速与占空比的直接关系，如下所示：

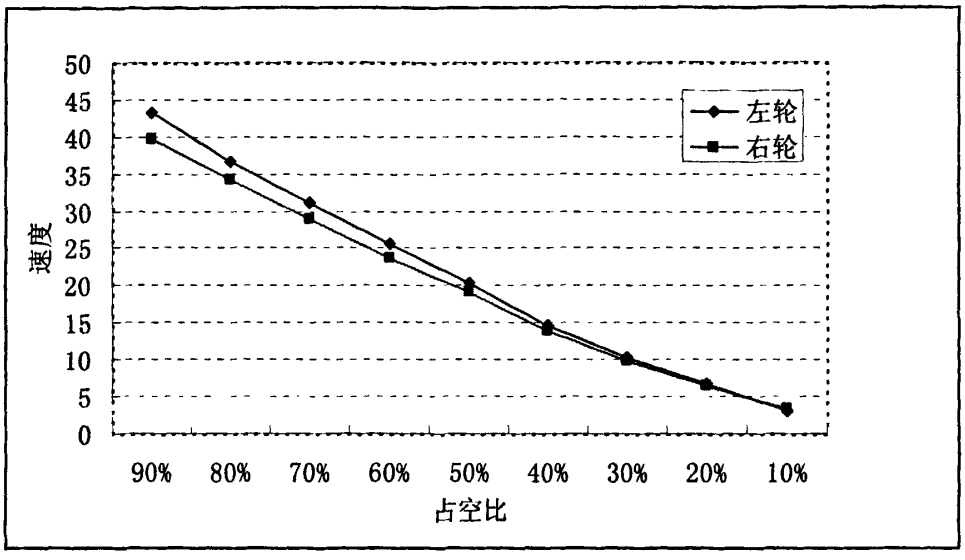

图 4-10 速度和占空比的关系图

Fig.4-10 Relation between velocity and Duty cycle

从图中可以看出，当给定相同的占空比时，两轮的速度有偏差，而且随着速度的不同，偏差也不同；并且左轮的速度一般情况下大于右轮速度。

2、旋转运动

当差动轮式机器人左右两轮的速度大小相等而方向相反时，机器人在原地自旋转。实现移动机器人的旋转运动控制只要保证左右两轮的速度大小相等且方向相反即可。

实验中我们设定机器人中心的起始坐标为{0，0}，在规定的运动圈数和运动速度下，使机器人按照正向和反向两种方向旋转，最后测得机器人的终点坐标。如表 4-2 所示：

表 4-2 机器人旋转运动实验数据表

Table.4-2 Experimental data table of robot rotary movement

	顺时针旋转		逆时针旋转	
圈数	v=10cm/s	v=20cm/s	v=10cm/s	v=10cm/s
10	{3.2, 2.5}	{6.5, 4.4}	{2.9, 2.5}	{5.1, 4.2}
20	{6.4, 5.3}	{10.3, 8.5}	{5.8, 5.2}	{8.2, 9.6}
30	{9.8, 9.2}	{14.2, 11.7}	{8.9, 7.4}	{12.7, 12.4}

3、圆弧运动

当差动轮式机器人左右两轮的运动方向相同、速度大小保持不变且速度差固定不变时机器人的运动轨迹为圆弧，并且当移动机器人的起点在坐标系的原点时，

圆弧的圆心坐标为：$\left\{0,\frac{(2v_L+\Delta v)L}{2\Delta v}\right\}$，圆弧的半径为：$\frac{(2v_L+\Delta v)L}{2\Delta v}$

按照移动机器人逆运动学原理，假设要使机器人按半径为 r 的圆弧运动，则根据 r 可以计算出机器人左右轮所需的速度和速度差。为分析方便，取 r=50cm 和 r=100cm 进行分析。

(1) $\frac{(2v_L+\Delta v)L}{2\Delta v}=50cm$ 可以看出，机器人所需要的速度和速度差可以有多个解。

下面考虑 $v_L=10cm/s$ 和 $v_L=20cm/s$ 两种情形

A:当 $v_L=10cm/s$ 时，两驱动轮距离为 $L=40cm$，则可得 $\Delta v=13.33cm/s$ 因而右轮 $v_R=23.33cm/s$；

B:当 $v_L=20cm/s$ 时，则 $\Delta v=26.66cm/s$，$v_R=46.66cm/s$

(2) $\frac{(2v_l+\Delta v)L}{2\Delta v}=100cm$ 时，一样分 $v_L=10cm/s$ 和 $v_L=20cm/s$ 两种情形

A：当 $v_L=10cm/s$ 时，则 $\Delta v=5cm/s$，$v_R=15cm/s$

B：当 $v_L=20cm/s$ 时，则 $\Delta v=10cm/s$，$v_R=30cm/s$

对相应的情况做了实验，所得的试验数据如下表 4-3：

表 4-3 圆弧运动实验数据表

Table. 4-3 Experimental data table of robot arc movement

运动方向	轮速	运动半径
前进	$v_L=10cm/s$ $v_R=23.33cm/s$	R = 58.4cm
	$v_L=20cm/s$ $v_R=30cm/s$	R = 112.6cm
后退	$v_L=10cm/s$ $v_R=23.33cm/s$	R = 56.7cm
	$v_L=20cm/s$ $v_R=30cm/s$	R = 110.3cm

机器人作圆弧运动实验数据表

4、结果分析

由以上的试验数据可以看出，可以实现预期的功能，虽然伺服控制器还是存在一定的误差，而且误差随着时间和速度的增大而增大，但是这样的误差是在允许之内的。究其原因，主要有以下几个方面：两驱动轮的直径不完全相等；两驱动轴和地面接触点之间距离不确定性；两驱动轮的轴心不在同一直线上；前后导向轮在运动过程中的阻碍作用；机器人的左右负载分布不同等。另外光电编码有限的分辨率以及整体的运动控制算法也还有待改进[57-61]。

5.7 小结

本章在以上第三章硬件平台搭建和第四章 Linux 系统移植、软件开发平台搭建的基础上，介绍了直流电机 Linux 下驱动程序设计、增量式 PID 算法以及改进、电流环控制子程序设计、速度环控制子程序设计和两种控制方式协调的设计，最后进行系统的调试与实验。

第六章 结论与展望

6.1 结论

随着机器人技术和嵌入式技术的发展，二者的结合成为不可阻挡的趋势。本文首先分析了国内外移动机器人发展现状，介绍了运动控制概述、分类以及发展方向结合嵌入式系统的概述，提出采用高性能的 ARM9 处理器 S3C2410 作为处理器核心，结合嵌入式 Linux 操作系统设计机器人系统平台。

系统选取三轮移动机器人(后两驱动轮，前一万向轮)作为对象，分析了运动学模型。介绍了直流电机控制方法以及相应的算法，采用 L298 作为电机的驱动芯片，扩展 HCTL2020 正交编码芯片检测转速，采用内电流环加外速度环的控制策略，采用改进积分分离 PID 控制算法。软件上，进行了 bootloader、Linux 系统和文件系统的移植，设计了直流电机的设备驱动程序、电流环控制子程序和速度环控制子程序。

文章最后进行了设备驱动程序的调试，和控制系统性能验证的实验，实验表明系统存在有一定的误差并且误差随着速度和时间的增加而增加，但这样的误差是在容许的范围之内的，整体的设计达到了预期要求。影响的原因，文中也进行了分析，虽然控制系统以及算法还有待做更进一步的改进，但是机械等原因也是造成误差的一个重要而不可忽视的因素。

6.2 展望

虽然课题的研究取得了一定的成就，但是系统还是有很多待改进的地方，需要在进一步的工作中进行研究。后续的研究开发可以从以下几个方面进行。

1、 电机的控制只使用了积分分离的 PID 控制算法，而 ARM9 的处理能力和处理速度都是非常高的，可以完成各种复杂的控制算法，为改善控制性能，适应各种不同的复杂环境，可以引入智能控制算法，比如模糊控制、专家 PID、自适应 PID 等

2、系统平台上可以扩展超声波、红外等测距传感器建立机器人蔽障系统，更进一步可以建立基于 CCD 的视觉导航系统提高移动机器人蔽障成功的概率。

3、增加数字信号处理器以添加语音和图像识别功能，同时增加摄像头和高速无线通信模块，增强移动机器人的交互和沟通能力，更扩展了其应用的范围

4、开发界面友好的上位机配置环境，以便非专业人员可以自行设计移动机器

人的运行路线

参考文献

[1] 于平. 移动机器人运动控制系统研制开发[D]. 哈尔滨: 哈尔滨理工大学, 2005

[2] 田泽. 嵌入式系统开发与应用[M]. 北京: 北京航空航天大学出版社, 2005

[3] NilssonNJ. An APPlication of Artificial Intelligence Technique Proe. of IJCAI[J]. Autonomous Mobile Robots: Control, Planning and Architecture, 1969, 2(1): 233 一 239

[4] A.M.Flynn. Survey of mobile robot. AD-A183 179, 1985

[5] httP://eees.vanderbilt.edu/CIS/IRL/html/helpmate_mobile_robot.html

[6] J.Borenstein, H.R.Everett, L.Feng. Where am I? Sensors and Methods for Mobile Robot Positioning, The University of Miehigan, 1996

[7] 桑楠. 嵌入式系统原理及应用开发技术[M]. 北京: 航空航天大学出版社, 2002

[8] 潘巨龙, 黄宁, 姚伏天, 陈科杰, 道克刚. ARM9 嵌入式 Linux 系统构建与应用[M]. 北京: 北京航空航天大学出版社, 2006

[9] 徐英慧, 马忠梅, 王磊等. ARM9 嵌入式系统设计-基于 S3C2410 与 Linux[M]. 北京: 北京航空航天大学出版社, 2007

[10] USER'S MANUAL S3C2410X 32-Bit RISC Microprocessor Revision 1.2. Samsung Electronics Co., Ltd[OL]. http://www.samsungsemi.com

[11] 潘宁, 邓燕. 妮基于 S3C44B0X 的嵌入式机器视觉系统设计[J]. 机械与电子, 2006(10) 52-54

[12] Jean J. Labrosse 著, 邵贝贝等译. 嵌入式实时操作系统 μC/OS-II [M]. 北京: 北京航空航天大学出版社, 2003

[13] 尔桂花, 窦口轩. 运动控制技术[M]. 北京:清华大学出版社, 2002

[14] 丁仕燕, 韩江, 朱方洲. 开放式数控系统软件平台的研究[J]. 机械与电子, 2002(2): 35-38

[15] RAndre WRussell. Survey of Robot Applications for odor-Sensing Technology[J]. The International Journal of Robot Research, 2001, 20(2): 10-162

[16] 柳宏义, 宋伟刚. 机器人技术基础[M]. 北京: 冶金工业出版社, 2002

[17] 大熊繁. 机器人控制入门[M]. 北京: 科学出版社, 2000.

[18] 龚建伟, 陆际联, 黄文宇. 轮式机器人的航向预估算法[J]. 机器人, 2001, 23(3): 193-197

[19] 陈伯时. 电力拖动自动控制系统[M]. 机械工业出版社, 2004

[20] 李发海, 朱东起. 电机学[M]. 北京: 科学出版社, 2001.

[21] 秦继荣等. 现代直流伺服控制技术及其系统设计[M]. 北京: 机械工业出版社, 1993 143-149

[22] 程福. 双轮驱动足球机器人的动态性能及踢球研究[D]. 天津: 天津大学, 2004

[23] 秦继荣, 沈安俊. 现代直流伺服控制计数及其系统设计[M]. 北京: 机械工业出版社, 1993

[24] 李铁才, 杜坤梅. 电机控制技术[M]. 哈尔滨: 哈尔滨工业大学出版社, 2000

[25] 胡寿松. 自动控制原理[M]. 北京: 科学出版社, 2001

[26] 周立功. ARM 微控制器基础与实战[M]. 北京: 北京航空航天大学出版社, 2003

[27] 李驹光. ARM 应用系统设计开发详解[M]. 北京: 清华大学出版社, 2003

[28] 黄力,覃纪武. 嵌入式 Linux 的现状与前景研究[J]. 电脑知识与技术, 2005(10): 69-71

[29] 卞正岗. 论嵌入式系统的发展[J]. 仪器仪表与分析监测, 2004, (1): 1-3
[30] 吕京建, 肖海桥. 面向二十一世纪的嵌入式系统综述[J]. 电子质量, 2001, (8): 10-13
[31] 范永, 谭民. 机器人控制的现状及展望[J]. 机器人, 2001, 3
[32] S3C2410A 中文数据手册. 杭州立宇泰电子有限公司编著
[33] 许大中, 贺益康. 电机控制(第二版)[M]. 杭州: 浙江大学出版社, 2002
[34] 齐书聪, 蔡胜乐. 模拟电子技术基础[M]. 沈阳: 东北大学出版社,1997
[35] 薛方正, 徐心和. 集控式足球机器人决策与控制系统设计[J]. 机器人, 2005, 27(6): 431-434
[36] 曹卫华, 吴敏, 陈鑫. 基于 DSP 控制的足球机器人小车的设计与实现[J]. 机器人技术与应用, 2002, 3: 19-21
[37] 宋铁群, 杜华生, 王德新. 一种小型移动机器人的控制系统研究[J]. 机械研究与应用, 2004, 17(6): 37-8
[38] 何存富, 周龙, 宋国荣. 基于 DSP 的直流电机驱动控制电路设计[J]. 测控技术, 2007, 26(01): 64-67
[39] 谢元平, 罗晖. 正交解码与可逆计数专用芯片 HCTL-2020 及其应用[J]. 仪表技术 2000, (3): 37-39
[40] 李克东. 移动机器人控制系统设计及规划研究[D]. 江南大学, 2008
[41] 李驹光等. ARM 应用系统设计开发详解[M]. 北京: 清华大学出版社, 2003
[42] 杜春雷. ARM 体系结构与编程[M]. 北京: 清华大学出版社, 2003
[43] S3C2410 用户手册.
[44] RTL8019AS.pdf. REALTEK SEMI-CONDUCTOR CO., LTD. HEAD OFFICE
[45] www. busybox. net
[46] 2410-S 实验指导书. 北京博创兴业技术有限公司
[47] http://kegel.emo/erosstool/
[48] 徐英慧, 忠梅. ARM9 嵌入式系统设计[M]. 北京: 北京航空航天大学出版社, 2007
[49] 李善平, 刘文峰, 王焕龙. Linux 与嵌入式系统[M]. 北京: 清华大学出版社, 2003.
[50] 毛德操等. Linux 内核源代码情景分析(上、下)[M]. 浙江: 浙江大学出版社, 2004 : 15-300
[51] 陶列骏等. Linux 可移植性研究[J]. 小型计算机系统, 2002, 23(1): 51-53
[52] Alessandro Bubini 等. Linux Device Drivers[M]. 北京: 中国电力出版社, 2000.: 57-86
[53] 李玉波等. Linux C 编程[M]. 北京: 清华大学出版社, 2005: 129-148
[54] 刘淼. 嵌入式系统接口设计与 Linux 驱动程序开发[M]. 北京: 北京航空航天大学出版社,2006: 18-59
[55] http://~'arm.eom/pdfs/AMRg_family_flyer_34_5.pdf
[56] Alessandro Rubini, Jonathan Corbet, Linux Device Drivers 2[M]. 北京: 机械出版社, 2002.
[57] 刘丽, 王祥. 基于 MCS-51 单片机的转速测量系统[J]. 测量测试技术, 2007, 3.
[58] 王晓明. 电动机的单片机控制[M]. 北京: 北京航空航天大学出版社, 2002
[59] 谷海涛, 彦湘武, 曲伟. 正交解码电路和捕获单元在转角和转速测量中的应用[J]. 电

器应用, 2005, 24(5): 112-115.

[60] 朱瑞. 自主移动机器人运动控制研究[D]. 北京：北京交通大学, 2006

[61] 刘复. 8098 单片机及其应用(十)—直流电机脉宽调速系统[J]. 电子与自动化, 1994, (l): 3-5

附录

Linux 下直流电机的设备驱动程序(部分实验程序)

```
#include <linux/config.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/sched.h>
#include <linux/delay.h>
#include <linux/mm.h>
#include <asm/uaccess.h>
#include <asm/arch/S3C2410.h>
#ifdef CONFIG_DEVFS_FS
#include <linux/devfs_fs_kernel.h>
#endif
#define DEBUG
#ifdef DEBUG
#define DPRINTK(x...) {printk(__FUNCTION__"(%d): ",__LINE__);printk(##x);}
#else
#define DPRINTK(x...) (void)(0)
#endif
#define DEVICE_NAME              "s3c2410-dc-motor"
#define DCMRAW_MINOR  1          //direct current motor
#define DCM_IOCTRL_SETPWM              (0x10)
#define DCM_TCNTB0                     (16384)
#define DCM_TCFG0                      (2)
static int dcmMajor = 0;
#define tout01_enable()   //enable 定时器宏定义
    ({  GPBCON &=~ 0xf;
        GPBCON |= 0xa;       })

#define tout01_disable()    //disable  定时器宏定义
    ({  GPBCON &=~ 0xf;
        GPBCON |= 0x5;
        GPBUP &=~0x3;     })
```

```
#define dcm_stop_timer()({ TCON &= ~0x1; })  //停止 PWM 定时器
#define dcm_start_timer()      //启动 PWM 定时器
    ({ TCFG0 &= ~(0x00ff0000);
        TCFG0 |= (DCM_TCFG0);
        TCFG1 &= ~(0xf);
        TCNTB0 = DCM_TCNTB0;  //PWM 波频率 调节寄存器
        TCMPB0 = DCM_TCNTB0/2;   // 占空比调节寄存器
        TCON &=~(0xf);
        TCON |= (0x2);  //装载计数寄存器和比较寄存器
        TCON &=~(0xf);
        TCON |= (0x9); })  //配置自动装载，并启动

static int s3c2410_dcm_open(struct inode *inode, struct file *filp)    //open 函数定义
{
    MOD_INC_USE_COUNT;
    DPRINTK( "S3c2410 DC Motor device open!\n");
    tout01_enable();
    dcm_start_timer();
    return 0;
}
static int s3c2410_dcm_release(struct inode *inode, struct file *filp)  //release 函数定义
{
    MOD_DEC_USE_COUNT;
    DPRINTK( "S3c2410 DC Motor device release!\n");
    tout01_disable();
    dcm_stop_timer();
    return 0;
}
static int dcm_setpwm(int v)  //调节占空比函数
{
    return (TCMPB0 =DCM_TCNTB0/2 + v);
}
static int s3c2410_dcm_ioctl (struct inode *inode, struct file *filp, unsigned int cmd,
unsigned long arg)  //ioctl 函数定义
{
```

```
    switch(cmd){
    case DCM_IOCTRL_SETPWM:
        return dcm_setpwm((int)arg);
    }
    return 0;
}

static struct file_operations s3c2410_dcm_fops = {
    owner:  THIS_MODULE,
    open:   s3c2410_dcm_open,
    ioctl:  s3c2410_dcm_ioctl,
    release: s3c2410_dcm_release,
};   // s32410_dcm_fops 结构体成员初始化

#ifdef CONFIG_DEVFS_FS
static devfs_handle_t devfs_dcm_dir, devfs_dcm0;
#endif
int __init s3c2410_dcm_init(void) //初始化函数
{
    int ret;
    ret = register_chrdev(0, DEVICE_NAME, &s3c2410_dcm_fops); //向内核注册一个
字符设备
    if (ret < 0) {
        DPRINTK(DEVICE_NAME " can't get major number\n");
        return ret;
    }
    dcmMajor=ret; //返回主设备号

#ifdef CONFIG_DEVFS_FS //判断是否使用了设备文件系统
    devfs_dcm_dir = devfs_mk_dir(NULL, "dcm", NULL); //设备文件系统中创建一个
名为 dcm 的目录
    devfs_dcm0 = devfs_register(devfs_dcm_dir, "0raw", DEVFS_FL_DEFAULT,
                dcmMajor, DCMRAW_MINOR, S_IFCHR | S_IRUSR | S_IWUSR,
&s3c2410_dcm_fops, NULL); //创建设备文件节点
#endif
```

```
    DPRINTK (DEVICE_NAME"\tdevice initialized\n");
    return 0;
}

module_init(s3c2410_dcm_init); //对初始化函数名称进行记录

#ifdef MODULE
void __exit s3c2410_dcm_exit(void) //退出函数
{
#ifdef CONFIG_DEVFS_FS
    devfs_unregister(devfs_dcm0); //撤销目录
    devfs_unregister(devfs_dcm_dir); //撤销节点
#endif
    unregister_chrdev(dcmMajor, DEVICE_NAME); //撤销向系统注册的字符设备
}

module_exit(s3c2410_dcm_exit); //对退出函数名称进行记录
#endif
MODULE_LICENSE("GPL");   //GPL 申明
```

致谢

首先要感谢我的指导老师田景文教授。本文是在田老师的亲切关怀和悉心指导下完成的。从选题、研究方案制订、理论分析、实验方案制订到最后最终定稿，田老师无不付出了心血和汗水。

三年来，田老师无论在学习上、思想上、还是在生活上都给了我莫大的帮助。进入课题研究阶段以来，田老师要求我深入思考和研究课题的各个细节与技术问题，使我在学术上获益匪浅。同时，田老师以严谨的治学态度、敏锐的洞察力、渊博的知识给我留下了深刻的印象也成为我学习的榜样，这些都将是我终身的宝贵财富。在此向导师田景文教授致以诚挚的感谢和崇高的敬意。

师母高美娟老师在这三年里给予了我许多生活上和学习上的关心与厚爱，对此表示诚挚的谢意

感谢实验室的师兄师弟，李凯、周浩、徐进、张帆等在课题上给予的帮助，对论文提出的意见和建议。

最要感谢的是我的父母，是他们对我的多年来的支持，让我健康成长！最后向所有关心、指导、支持、鼓励我的师长、同学和朋友们表示衷心的感谢！

研究成果及发表的学术论文

已发表的论文

[1] 田景文，吴浩，高美娟. 基于支持向量机的火车滚轴故障诊断[J]. 机床与液压，2007，7(35): 248-250

[2] 田景文，吴浩，高美娟. 基于开关磁阻电机的抽油机节能控制系统[J]. 机床与液压，2008，7(36): 314-316

作者和导师简介

作者简介：

吴浩，男，1984 年 2 月出生，汉族，安徽省桐城市人，北京化工大学，硕士研究生。

专业：检测技术与自动化装置

研究方向：智能检测与信息处理

联系方式：206001255@grad.buct.edu.cn

导师简介：

田景文，男，1962 年 6 月，黑龙江人，汉族，北京联合大学，教授，博士。

研究方向：智能测控技术，机器人技术

发表学术论文 50 多篇，其中国际三大检索收录 20 多篇，获得专利 12 项。

联系方式：ttjjww999@163.com