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，独立进行研究所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的科研成果。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。本声明的法律责任由本人承担。

学位论文作者：王朋涛　　日期：2010 年 5 月 28 日

学位论文使用授权声明

本人在导师指导下完成的论文及相关的职务作品，知识产权归属郑州大学。根据郑州大学有关保留、使用学位论文的规定，同意学校保留或向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权郑州大学可以将本学位论文的全部或部分编入有关数据库进行检索，可以采用影印、缩印或者其他复制手段保存论文和汇编本学位论文。本人离校后发表、使用学位论文或与该学位论文直接相关的学术论文或成果时，第一署名单位仍然为郑州大学。保密论文在解密后应遵守此规定。

学位论文作者：王朋涛　　日期：2010 年 5 月 28 日

摘 要

粮食问题是关系国计民生的重要问题，实时准确地掌握粮食储藏数量及其质量对于制定政策和解决社会问题具有重要的意义。随着半导体技术的快速发展，越来越多的设备开始采用嵌入式处理器，该论文就是将嵌入式处理器 ARM9 和 Linux 操作系统相结合，设计了一个在线的网络化粮情监测系统。文章分别从硬件平台和软件设计两个方面详细的讨论了该粮情监测系统的设计和实现方法。

在硬件设计方面，本文以 ARM9 处理器为控制核心构建了硬件平台来代替传统的以单片机为处理器的粮情监测系统，增强了系统的监测功能，提高了监测系统的兼容性。

在软件设计方面，建立了嵌入式系统运行的软件环境，并进行了相应驱动程序和应用程序的设计。在设计的过程中具体实现了 bootloader 移植、linux 内核移植、文件系统的建立、温度和压力传感器驱动程序及其应用程序的开发。此外，在该系统上还构建了嵌入式网页服务器，设计了用于远程查询监测信息的 CGI 程序。在该系统中作者分析了现有粮情监测系统的优缺点，把嵌入式系统应用于粮情监测，利用 Linux 在通信和网络方面的优势，实现了监测系统的基本功能，具有一定的实用价值。

文章的最后给出了该监测系统的监测结果，分析了该系统在软件和硬件方面的不足之处，同时对下一步的研究方向作了展望。

关键词： ARM　LINUX　嵌入式系统 S3C2440　GPIO

Abstract

The food problem is an important issue to the people's livelihood. It is important to grasp the quantity and quality of grain storage accurately, because it can help us to formulate policies and solve social issues.

With the rapid development of semiconductor technology, more and more devices start using embedded processor, this paper also uses an embedded chip named ARM9. It presents a network-based multi-functional monitoring and measuring system of the grains. The measuring system is based on ARM9 embedded processor and the Linux operating system. The following paper gives a particular description of the hardware platform and software design.

On the hardware design, this paper uses ARM9 based hardware platform to replace the traditional single chip as the processor in Grain monitoring system. This design proposal enhanced the system's monitoring functions and improved its compatibility.

In software design, the author set up a software environment and made the appropriate drivers and its application. In the process of designing, some works have been done for example porting the boot loader and Linux kernel , the establishment of file system, writing device driver and its corresponding application of the temperature and pressure sensors .In addition, the embedded web server and some CGI program has been built on this system for remote query monitoring information. In this system, the author analyzed the situation of existing food surveillance system's advantages and disadvantages, and then introduced embedded systems to monitor the grain's situation. This design proposal takes advantage of the communications and network function of Linux to achieve the basic functions of the monitoring system, so it has some practical values.

Finally, the paper gives the results of the monitoring system, analyzes the shortcomings of this system in software and hardware, and points out directions for the future research.

Key words: ARM LINUX embedded system S3C2440 GPIO

目 录

1 绪论

1.1 粮情监测系统概述

粮食是人类生存和发展的第一需要，是关系国计民生的重要产品，储粮安全是国家粮食安全的重要组成部分，国家储粮数量及其质量数据是一个国家安全的重要信息之一。实时准确地掌握粮食储藏数量及其可靠性对国家制定粮食政策，解决社会保障问题、以及处理国际问题提供了重要的依据。国家储粮数量及其储粮的温度、水分、病虫害等情况的网络化实时监测是国家粮食部门一直迫切想要解决的问题。最近几年，国库内粮食被倒空卖空赚取粮食差价，然后又补不回倒卖的粮食的现象时有发生，给国家造成了重大的损失，更有甚者通过倒仓的方式来套取国家补贴。每年国家花费大量的人力、物力用于清仓查库，因此开发出简单方便、实时在线的网络化监测系统势在必行。该课题的研究有助于保证粮食储存水平，简化储粮管理，节省清仓查库的巨额资金，对于维护社会稳定和保证国家安全也有重要的意义。

1.2 粮情监测系统发展现状

现阶段粮情测控系统的主要功能还是局限于温度监测和简单的粮情分析，例如测定粮食的温度和湿度，而对粮食本身所含的水分、粮食所遭受的虫害情况以及粮库内的气体成分等粮情问题的监测还属于空白，另外市面上各家粮情监测软硬件平台多样化，常常不能相互通信，造成系统的软硬件兼容性差，这些问题给国家掌握储粮数量及其质量造成了诸多不便，同时也给储粮的安全带来了隐患。

目前，对传感网的研究成为了一个热点，将基于传感网的多传感器信息融合技术用于粮仓储存粮食的真实性及其质量的研究将会成为一个比较重要的方向。例如：在粮食储藏过程中，对粮食的数量有重要影响的是粮食的水分和湿度，而对储粮的质量有影响的是虫害情况。可以开发储粮专用传感器或者利用电磁波在不同介质中传播的特性不同，或者根据电磁波反射的参数，推导得出粮食数量及其水分等数据信息，然后对这些不同的传感器观测到的信息预处理，再依据信息融合算法进行信息融合，得出可信度最大的融合结果。

1.3 粮情监测系统结构框图

该系统包含客户机、交换机、ARM9 模块、通风控制模块。其结构图如图 1 所示：

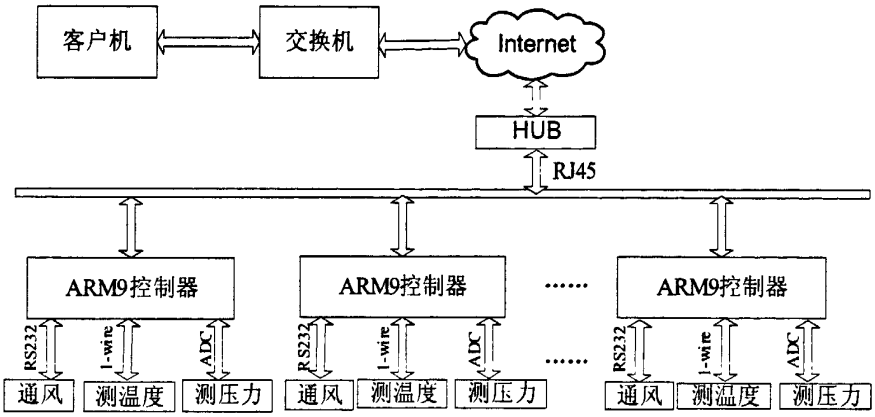


图 1-1 系统结构示意图

图 1-1 中客户机是指在局域网内任何一台带有浏览器的个人计算机，如果在外网，则需要给控制器提供一个固定的 IP 或者是固定的域名，然后使有权限的用户能够通过万维网访问本测控子系统。ARM9 控制器主要负责粮仓温度、压力等信息的采集和处理，控制器还负责与通风控制器的通信，发送命令到通风控制器进行手动控制，单片机本身也可以根据粮仓内的温度进行自动控制通风，此外该系统还利用 B/S 模式提供一个数据信息查询或控制的页面[1]。对于粮仓内粮食温度的测量则是用单总线温度传感器构成监测网络来实现；对于粮仓底部粮食的压力测量采用的是河南工业大学最新研制的压力传感器，然后根据粮仓底部的压力和粮仓的面积计算出整个粮仓内粮食的平均密度[2]。以上的所有模块中该嵌入式处理器是整个系统的核心，监测系统通过外扩的网卡电路使该系统可以接入网络，同时作为一个 web 服务器来响应客户机的请求，从而完成数据的传输和控制。

1.4 课题的研究背景及意义

该课题以研究安全储粮为背景，来自于国家十一五科技攻关项目“安全绿色储粮关键技术研究开发与示范”其中的关于网络化多功能粮情监控集成技术

和系统研究开发。该监测系统的开发是其中的一部分，对于建立多功能的粮情监测软硬件平台提供了一定的基础，同时也有助于实时的掌握国家粮食数量和粮仓内粮食的温度信息。

1.5 论文的主要研究内容

该课题以开发在线的网络化粮情监测和温度采集为目的，在 ARM9 处理器的核心平台上构建了 U-boot 和 Linux2.6 内核，实现了监测系统的测重、测温和通风控制等基本功能。在完成此课题的过程中解决了 u-boot 下驱动程序的移植、Linux 内核的裁剪和移植、yaffs2 文件系统的制作、Linux 下驱动程序和应用程序设计、CGI 程序设计、以及嵌入式 WEB 服务器的建立等几个关键问题。该论文的内容安排如下：

第 1 章介绍了粮情监测系统的发展现状，说明了本课题研究的背景和意义。

第 2 章在整体上介绍了监测系统的软硬件架构和平台，以及需要做的工作。

第 3 章详细的介绍了该监测系统硬件平台的各个模块及其工作原理。

第 4 章详细的介绍了系统软件平台的建立过程，其中主要包括：交叉开发环境的安装和配置、引导加载程序 u-boot 的移植、Linux 内核的裁剪和文件系统的建立。

第 5 章主要阐述了 Linux 设备驱动程序开发的过程和方法，以及开发中遇到的问题，并设计了单总线设备驱动、AD 转换驱动和键盘驱动等驱动程序。

第 6 章介绍了 linux 下应用程序的设计方法，并实现了温度监测、压力监测和串口控制等应用程序的设计。

最后在第 7 章对该课题进行了总结，分析了该粮情监测系统中存在的缺点，以及以后的设计中应该改进的方向。

2 系统总体设计

2.1 系统工作原理

该课题以 ARM9 S3C2440 为硬件平台搭建了一个模块化的粮情监测子系统，网络拓扑结构如图 1-1 所示。系统按模块划分为串行通信模块，GPIO 口通信模块和 AD 转换模块，串行通信模块用于粮库内的通风控制。GPIO 口通信模块用于连接温度传感器并通过该端口把温度值传给 ARM9 核心硬件平台，AD 转换模块用于连接压力传感器采集粮仓的重量信息。同时，在 ARM9 的软件平台上实现了一个嵌入式 WEB 服务器，并在其上实现了相应的 CGI 控制程序，可以利用其它计算机通过 Internet Explorer 浏览器远程访问 ARM9 嵌入式 WEB 服务器，点击网页内的控制按钮，就可以触发相应的 CGI 监控程序[3]。这样就实现了远程的客户机对服务器的在线数据的交互。

2.2 硬件功能分析

该课题采用的嵌入式处理器模块主要有：基于 arm920t 内核的 32 位精简指令集处理器、存储器模块、串行接口、以太网模块、通用输入输出接口、电源管理和复位电路模块、扫描调试电路、USB 接口模块、SD 卡控制存储模块和 AD 转换模块等各个子模块组成。其结构示意图如图 2-1 所示。

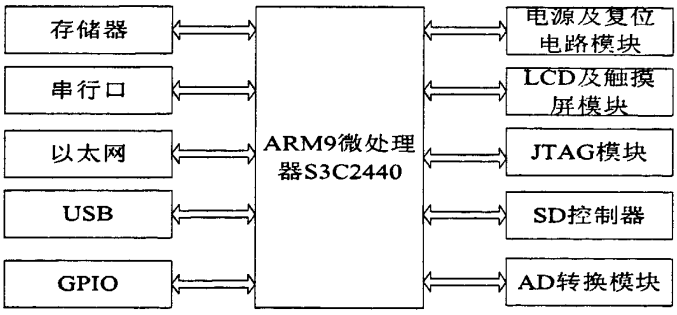


图 2-1 ARM9 控制器模块示意图

微处理器采用三星公司的 32 位 ARM9 处理器 S3C2440A;存储器模块由三星的 64MB 的 NAND FLASH K9F128UOM 构成闪存型存储系统，用于存放该监测系统

的程序和文件系统。另外，由 SDRAM HY57V561620 构成程序运行的片外 RAM。以太网模块选用DAVICOM公司的低功耗以太网控制器DM9000A搭配HS9016以太网脉冲变压器构成[4]。保留 1 个 RS232 电平的串行接口来实现与其它粮情测控系统的数据交换，以提高改善该监测系统的兼容性。USB 模块电路可以作为无线通信的扩展模块，用以扩展无线网卡使该系统支持 Wi-Fi 网络。GPIO 接温度传感器，AD 转换接压力传感器，SD 卡模块作为大容量存储设备用来存储数据采集结果和保存嵌入式监测系统的运行日志。

2.3 软件框架及开发流程

由于 S3C2440A 的内核已经具有全性能的 MMU（内存管理单元），因此本系统采用 Linux 作为操作系统，因为 Linux 操作系统有很多的优点：如内核源代码免费，可以节省开发费用；Linux 移植性好，适用于多种硬件平台；Linux 微内核内嵌网络协议栈；Linux 上进行软件开发可以从互联网中获取大量的帮助。

软件框架如下图 2-2 所示，最底层的是硬件平台，在硬件平台之上的是操作系统，其中设备驱动程序、板级支持包 BSP 和 TCP/IP 协议栈是操作系统和硬件平台之间的桥梁，再上层是应用程序。该平台在工作时一般是应用程序首先向操作系统发出控制硬件的请求，由操作系统调用相应的驱动程序，然后驱动程序根据具体的请求去执行相应的控制。

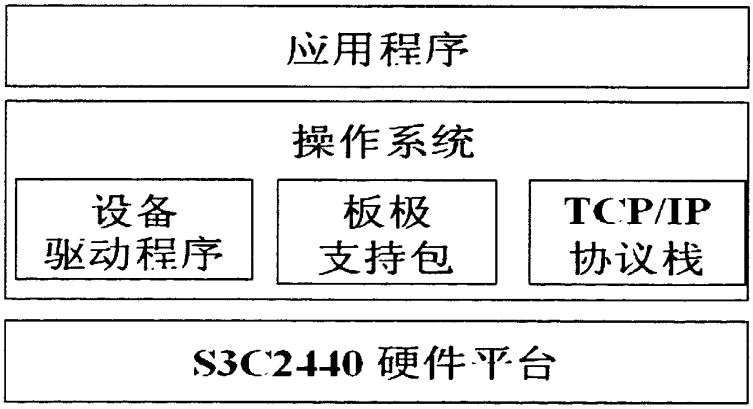


图 2-2 软件框架示意图

嵌入式开发往往不仅需要从总体上把握软件、硬件、安装、调试等各个方面，而且也要关注嵌入式系统的鲁棒性、精度和实时性。嵌入式开发首先需要

进行需求分析，并根据需求划分出各个子功能模块，依据各个功能选择合适的硬件进行系统设计，这时再次确认需求分析，依据子功能模块画出软件的流程图，然后由流程图实现相关代码，最后再联合调试以便实现系统的所有功能。

本课题所要做的软件工作如下：

(1) 交叉编译环境的搭建

(2) 移植 u-boot

(3) Linux 内核的裁剪移植

(4) 文件系统的创建和打包

(5) 温度传感器驱动程序设计

(6) 压力传感器驱动程序设计

(7) 应用程序设计

其中应用程序的设计又包括以下几个部分：

(1) 温度、压力传感器的应用程序设计

(2) 串口应用程序设计

(3) web 应用程序设计

(4) CGI 程序设计

3 系统硬件设计

3.1 系统硬件平台的选择

本系统根据成本、功耗和性能等原则采用了性价比较高的 Samsung 公司的 S3C2440A 处理器。该处理器内部集成了 ARM 公司 ARM 920 T 内核的 32 位微控制器，主频最高可达 400MHz，该芯片集成度高，片上资源丰富，非常适合于嵌入式网络方面的应用。

3.2 核心板组成

核心板一般采用多层板工艺，以提高电气性能和抗干扰性能，核心板资源包括：ARM9 处理器芯片、SDRAM 和 FLASH 存储电路。

3.2.1 核心处理器

S3C2440A 是 Samsung 公司推出的一款 32 位精简指令集处理器，其内部结构如图 3-1 所示：

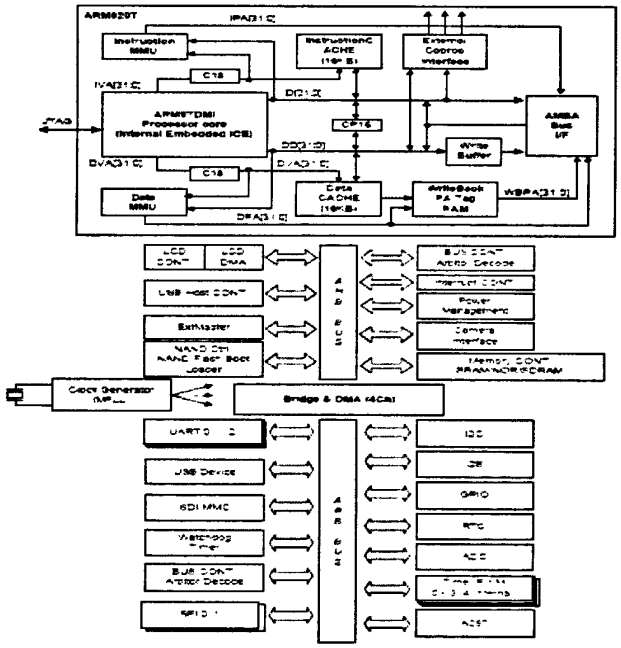


图 3-1 S3C2440A 内部结构框图

S3C2440A 基于 ARM 公司的 ARM920T 版本 IP 核，实现了 MMU、USB 等资源，此外片上还集成了 LCD 控制器，3 通道 UART、ADC 和触摸屏等丰富的接口，详细的介绍可以参考相关的芯片手册[5]。

3.2.2 SDRAM 存储电路

S3C2440A 支持内存控制器，其 BANK6、BANK7 支持外接 SDRAM 的扩展，可以支持 8 位、16 位和 32 位数据总线的 SDRAM，由于每个 BANK 最大可以外扩 128MB 的存储器，所以可以外扩的 SDRAM 最大容量是 256MB。另外 S3C2440A 的内存控制器还支持 SRAM，然而由于 SRAM 往往价格较高，所以在大多的嵌入式系统都是选用同步动态 RAM 来作为外扩内存的。SRAM 是一种静态 RAM，不需要刷新电路，存取速度快，一般用于 CPU 内部 cache 和数字信号处理芯片的外接内存，但是其缺点就是成本较高，功耗也较大[6]。DRAM 是一种动态的 RAM，相对与 SRAM，它需要不断的刷新，但是其较低的功耗和成本，及其较高的集成度，使得它在嵌入式系统中得到了广泛的应用。动态内存的发展从早期的 DRAM，到 FPM DRAM、EDO DRAM，再到 SDRAM 和近几年的 DDR、DDRⅡ、DDRⅢ,其内核主频也从早期的 133MHz 提高到了今天的 1333MHz。SDRAM 是一种同步内存，即它需要一个同步时钟，来依次执行存取动作。由于 S3C2440A 的外部数据总线是 32 位的，而市场上很少有单片 32 位宽度的 SDRAM 芯片，所以一般都是采用 2 片 16 位的 SDRAM 拼接成 32 位宽度的来用。该系统采用 2 片 hynix 公司的 HY57V561620BT-H 芯片构成 32 位的外接内存，HY57V561620BT-H 单片容量是 32MB，时钟频率为 133MHz，其连接原理图如图 3-2 所示：

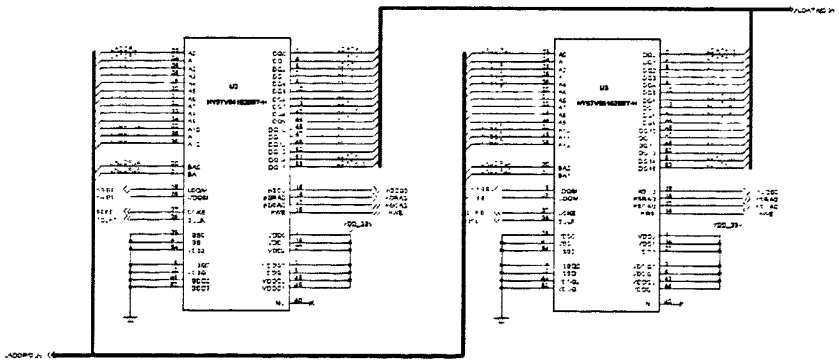


图 3-2 SDRAM 电路连接原理图

从上图中可以看出芯片的 A0 引脚，并没有接在 S3C2440A 的 ADDR0 地址线上，而是接在了 A2 上，这是因为当 SDRAM 为 32 位数据宽度的时候，其每一个存储单元将会占用 4 个字节，当芯片的地址线 A1A0=01 时，对应的处理器上的空间应就是 ADDR3ADDR2=01。同理，当外扩存储器的数据总线为 16 位宽度的时候芯片的 A0 引脚应当接到 S3C2440A 的 ADDR1 引脚；当外扩存储器的数据总线为 8 位宽度的时候芯片的 A0 引脚应当接到 S3C2440A 的 ADDR0 引脚。由于 SDRAM 具有较高的主频，所以在布线的时候需要考虑其走线的长度和走线之间的干扰，布线的原则一般是所有的数据线走线长度要尽量做到等长，控制信号和地址信号线的长度也应尽量相当，必要的时候也可以考虑采用串联电阻的方式来做阻抗匹配[7]。

3.2.3 FLASH 存储电路

该系统所用的处理器已经自带了 NAND FLASH 控制器，可以从 NAND FLASH 自举启动，因此 NAND FLASH 可以和 S3C2440A 直接相连。常见的 FLASH 一般可以简单的分为二种：NAND FLASH 和 NOR FLASH。NOR 是一种 linear 技术，它可以实现单字节读和写（即编程），并且执行速度快，一般用来存放程序。NAND FLASH 则必须以块为单位来进行读写，速度比 NOR 慢一点，但它较大的容量、较低的成本和较好的耐用性等特点，使其成为了嵌入式系统中最主要的存储设备，一般用来存储数据。市场上几乎全部的存储卡和 U 盘都是以 NAND FLASH 为基础的，如 CF(Compact Flash)卡、SM(Smart Media)卡、XD(eXtreme Digital)卡、SD(广泛用于手机和数码相机)卡、记忆棒(Memory Stick)等都是用的 NAND FLASH。该系统采用的 NAND FLASH 型号为三星公司的 K9F1208U0M，容量为 64MB，其连接图如图 3-3 所示：

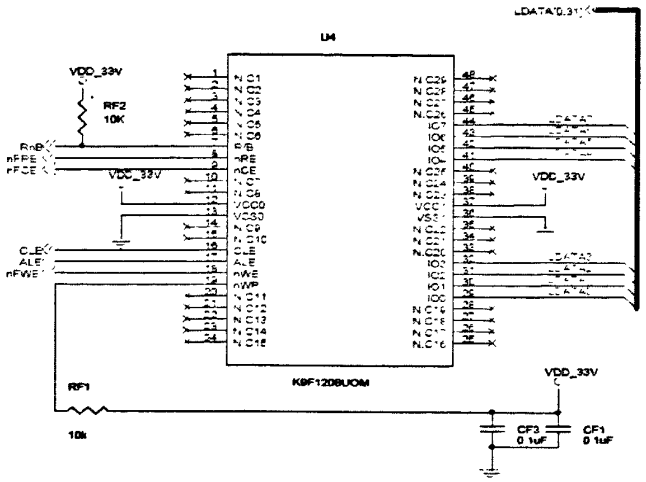


图 3-3 FLASH 电路连接原理图

3.3 底板电路设计

底板为了降低成本，通常采用双层板工艺，底板上的硬件资源包括：串行接口、以太网模块接口、USB 接口、GPIO 扩展接口、电源管理模块和复位电路、JTAG 扫描调试电路、SD 卡控制器接口和数模转换接口等部分。

3.3.1 串行口硬件电路

串行通信是最简单的一种数据通信方式，该系统采用的是异步通信方式，传输一次的数据以字节为单位，分为 1bit 起始位，8bit 数据位、1bit 停止位。起始位固定是 0，停止位是 1[8]。S3C2440A 内部带有 3 通道的 UART 串口，其中一个通道是 5 线制串口，另两个通道是 3 线制串口，所以对串口的硬件设计一般就是对电平转换电路设计，通常采用 MAX232 电平转换芯片来实现。串口 0 转换成 RS232 电平接口用于嵌入式系统的调试控制台，通过 PC 机的 COM 口进行人机交互和打印调试过程中输出的信息。串口 1 用于和辅助控制板相连接，实现辅助控制板和 ARM9 核心的通信，并根据得到的命令来实时的控制通风。如图 3-4 为串口电平转换电路图：

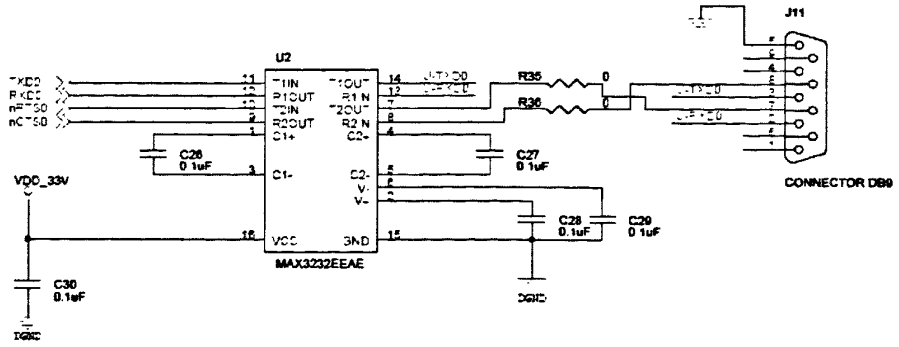


图 3-4 串口电平转换电路原理图

3.3.2 以太网模块接口

网络作为数据交换的一种介质，使得其不论在 PC 领域还是在嵌入式领域都成为了数据通信的一种重要方式，同时它也是嵌入式系统中调试应用程序和驱动程序的重要手段。在嵌入式系统中常常在系统启动的时候挂载 NFS 服务器的共享目录，在主机上的共享目录下编译应用程序和驱动程序，然后再到目标板上去执行已经编译好的应用程序或驱动，这样就不用来回拷贝编译好的驱动或应用程序，提高了效率[9]。该系统采用了低功耗以太网接口芯片 DM9000A，该芯片内部集成了物理层和数据链路层控制协议，同时支持 8 位、16 位、32 位访问模式，该系统采用 16 位访问模式，把以太网接口芯片接在 BANK3 上，BANK3 的地址范围是 0x18000000-0x20000000，由于 DM9000A 默认的 IO 口地址为 300H，所以它的实际物理地址应该为 0x1800003000-0x180003fff，其连接原理图如图 3-5：

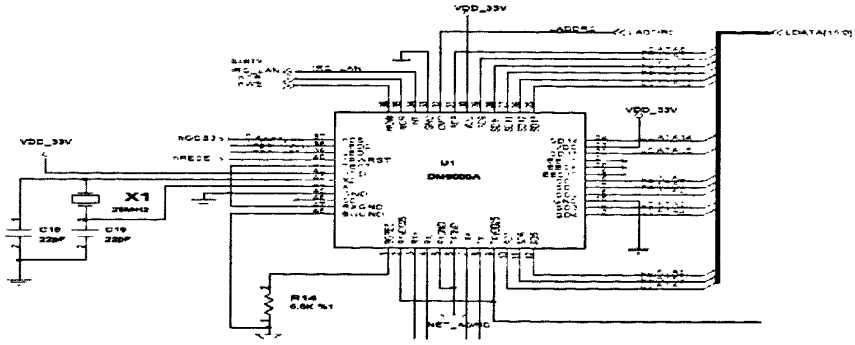


图 3-5 以太网接口电路原理图

3.3.3 AD 转换电路

AD 转换的主要过程就是采样、量化和编码，它可以分为逐次逼近型、双积分型、VF 型、二进制斜坡式、并行比较式和量化反馈式等，用的最广泛的还是逐次逼近型 AD 转换。ARM S3C2440A 芯片自带一个 8 通道 10 位 A/D 转换器，其最大转换率为 500K[10]。ADC 转换硬件电路主要涉及了 8 通道的 ADC,其中通道 0-3 为正常通道，而 AIN4-AIN7 被触摸屏所占用，要用做 ADC 时需要禁用触摸屏的功能。硬件连接时只需要把其直接引出来，如图 3-6 所示：

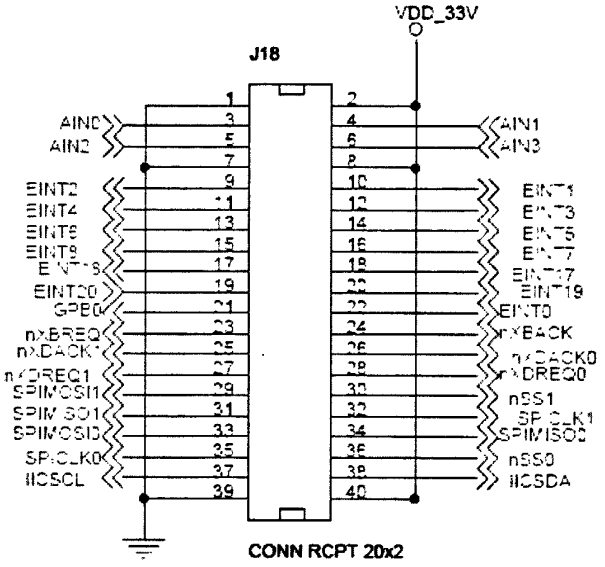


图 3-6 ADC 接口电路原理图

3.3.4 GPIO 扩展接口

该系统测温采用 DALLAS 的数字式温度传感器 DS18B20,它可以在单条线上挂接多个温度探测点。测温范围为-55 度—— +125 度，测量分辨率为 0.0625 度，内部含 64 位只读存储器 ROM，用户可分别设定各路温度的上、下限[11]。单总线设备与 S3C2440 的连接是通过其 GPIO 管脚来实现的。利用 ARM9 的 GPIO 接口接 18B20 进行单总线传输，接 GPB0 即板子上的 J18 第 21 脚，VCC 接 2 脚，GND 接 39 脚。S3C2440A 可以用 GPB0 作为单总线的连接端口来实现数据通信。另外，多余的 GPIO 口还可以用来扩展键盘或显示接口，例如通过 GPIO 扩展的数码管，或者扩展 LCD 接口和 LED 发光管显示电路。

3.3.5 USB 接口

S3C2440A 内部自带 1 通道 USB 主机接口，接口类型为 USB1.1,可以外接 USB HUB 进行扩展。USB 接口在本系统中主要用来作为备用的接口，以便使系统可以通过外接 USB 无线网卡实现无线通信，其接口电路如下图 3-7 所示：

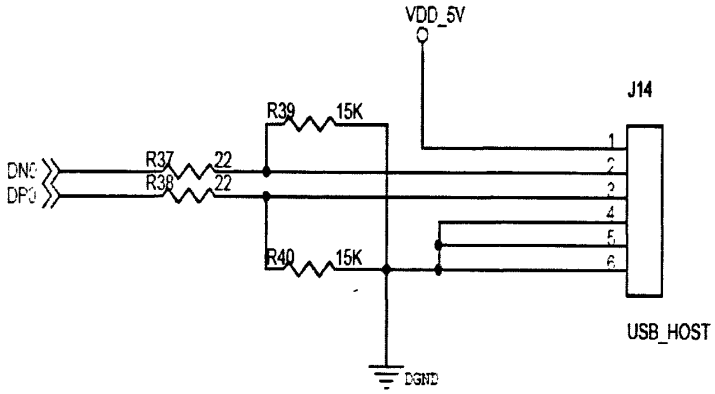


图 3-7 USB 接口电路原理图

3.3.6 SD 卡控制器接口

S3C2440A 内部自带 SD 主机接口，可以支持 SD 卡，TF 卡等[12]。该系统主要用 SD 卡来存储采集到的粮堆温度数据信息，对温度的采集时间及周期一般设定为每 2 小时定时采集一次各监测点的数据信息，存放到 SD 卡上，监测采集周期定为 2 年。其连接原理图如下：

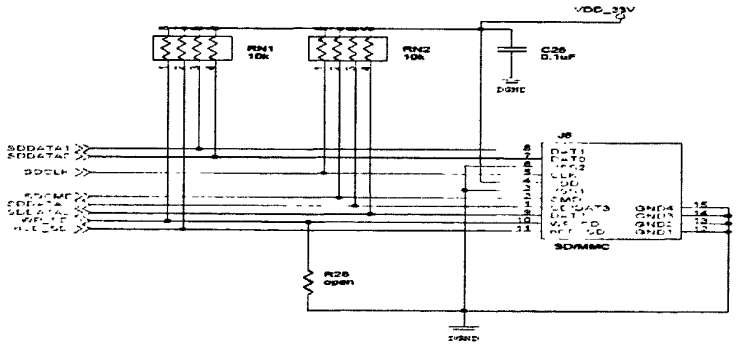


图 3-8 SD 卡接口电路原理图

3.4 辅助控制板

辅助控制板主要是由单片机自动控制板构成，在单片机板上实现了串口通信和温度的自动控制。当粮仓内温度大于设定温度的时候，自动开启通风设备进行通风，当粮仓内温度小于设定温度的时候，则关闭通风设备。也可以通过串口发送命令进行强制通风或者是强制关闭通风。ARM9 目标板与辅助控制板的连接：计算机的 RS232 串行 COM 口应该与 ARM9 的 UART0 串口相连，ARM9 的 COM1 和单片机的串口相连，ARM9 通过网线连接到与 PC 机相连接的网络。其硬件电路图如下图 3-9：

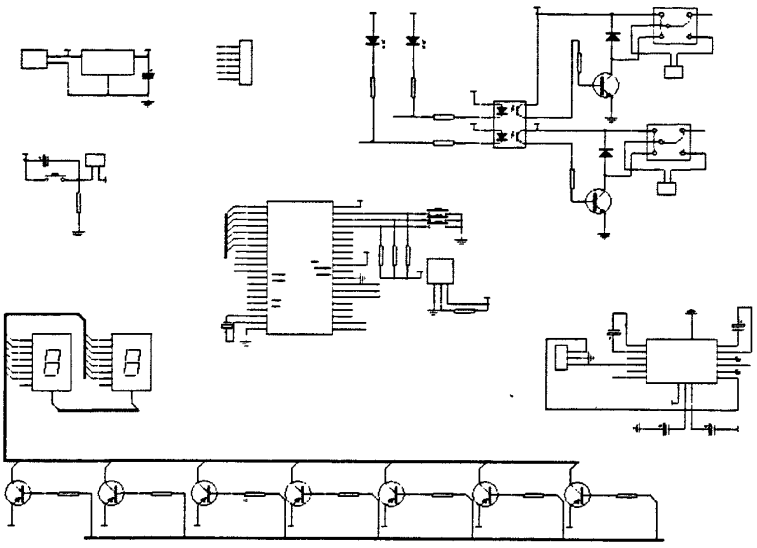


图 3-9 辅助控制板电路图

4 系统软件平台构建

4.1 搭建嵌入式系统开发环境

在进行嵌入式系统开发的时候，往往都需要搭建一些开发环境，如开发 51 系列单片机的时候需要安装 Keil uVision 编程环境，开发 AVR 系列的单片机时需要安装支持 AVR 的 gcc 编译器，用 ARM 作为处理器时，也需要安装支持 ARM9 的 gcc 编译器[13]。另外，因为选用 linux 作为操作系统，所以还需要在 linux 下安装相应的软件、配置一些服务，如配置 linux 下的 tftp、nfs、samba 等服务器。由于嵌入式系统所用的处理器处理速度与 PC 机的 CPU 处理速度相差甚远，所以它需要采用一种交叉编译的开发方式，即在开发嵌入式系统的驱动和应用程序时，不能直接在目标板上编程和调试，而是首先在 PC 机上进行编写、编译和调试，然后下载到目标板子上去执行。一般所需要的工作有安装发行版 linux、安装交叉编译器、配制宿主机开发环境、移植 u-boot，下载 linux 内核源代码、裁剪和移植 linux 内核、制作文件系统[14]。

4.1.1 安装发行版 linux

如今从网络上可以看到有各种不同的操作系统如 Windows、DOS、unix、netware、Mac OS X、Solaris、MontaVista、VxWorks、SUSE linux 和 RedHat linux 等等，但是从早期的操作系统的发展来看仍然可以分为 DOS、Windows（包括 win9X、XP、2K、Vista）、Unix（SCO Unix）、类 Unix（Mac OS X）、Linux（linux 也是一种类 Unix）等。常见的嵌入式操作系统有 WINCE，linux、uClinux、VxWorks、MontaVista、Uc/OS2 等，这其中有大部分都是商业化的产品，每套授权的价格相当昂贵，而且源码相对比较封闭，从而增加了开发的难度，提高了开发的成本。Linux 是一种源码开放的操作系统，它是免费的，支持多种硬件架构如 X86、ARM、PPC、MIPS、Sparc、NEC 等硬件平台，其内核直接支持网络协议栈，支持多任务、多进程，该系统选用 linux 作为嵌入式操作系统[15]。

安装 linux 到 PC 机上一般有 2 种方法：一种是直接安装进硬盘，另一种是通过虚拟机软件在 XP 下虚拟出一台 PC 机，再在虚拟出来的 PC 机上安装操作系统。该系统采用后一种方法，操作系统采用 RedHat 9.0 Linux 桌面安装版，计算机安装双网卡：一个用于虚拟机和 XP 共享文件，另一个用于 XP 上网用。

虚拟机用的网卡在 XP 下设定其 IP 地址为：192.168.1.1 子网掩码为：255.255.255.0，在 Linux 下设定为：192.168.1.200 子网掩码为：255.255.255.0 网关为：192.168.1.1。虚拟机内存设定为 256MB、硬盘容量设置为 15GB，安装的时候首先要分区，根分区分 8GB，交换分区分 512MB，其余的设定为用户分区，选择自定义安装、没有防火墙，选择软件安装包的时候需要选择 Everything 即完全安装所有 RPM 包。安装完成后去掉了 root 的登陆密码，先用 root 登陆系统，然后修改根目录下 etc 文件夹下的 shadow 文件的第一行内容 root 后 2 个冒号之间的内容删除，保存重启[16]。

4.1.2 搭建交叉编译环境

绝大多数的嵌入式软件开发都是以交叉编译的方式进行的，主要的原因是嵌入式系统本身没有足够的资源在本机来运行开发调试工具，交叉编译工具安装在宿主机上即安装好发行版 Linux 的主机，对应的 ARM 板称为目标板[17]。把 PC 机的 COM 口与目标板的串口 0 相连接，PC 机上虚拟机占用的那个网卡通过交叉网线与目标板的 RJ45 接口相连接。模型如下图 4-1 所示：

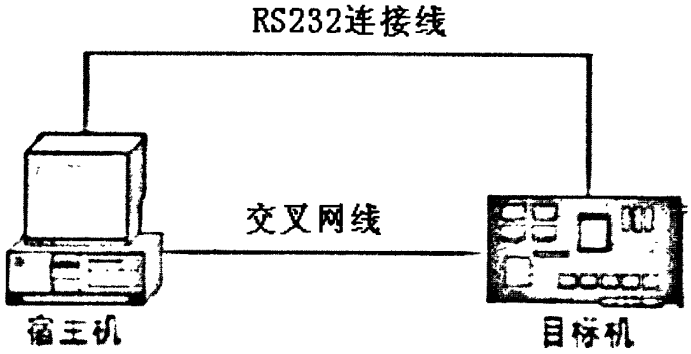


图 4-1 交叉编译模型

网络上已经有编译好的交叉编译器，编译 u-boot 时采用较低的版本即可，编译内核和应用程序时采用较高版本。具体的安装过程是下载 arm-linux-gcc-3.4.1.tar.bz2 和 arm-linux-gcc-2.95.3.tar.bz2 到 root 目录，然后用“tar -xjvf arm-linux-gcc-2.95.3.tar.bz2 -C /”命令解压安装到根目录下，同理对 3.4.1 版本的安装类似。安装完后用 vi 命令修改 root

文件夹下的隐含文件.bash_profie,给“PATH”变量添加绝对的路径PATH=$PATH:/usr/local/arm/2.95.3/bin，保存退出，然后执行：source /root/.bash_profile,使设置的环境变量生效。如果要用 3.4.1 版本的交叉编译器则需要修改为PATH=$PATH:$HOME/bin: /usr/local/arm/3.4.1/bin。

4.1.3 配置宿主机

对于编译好的内核或者是uboot引导文件如果要测试能否在该平台运行，可以通过该系统的网卡下载到目标板，而对于应用程序则是通过 nfs 方式挂载到目标板的一个目录，然后在此目录中直接运行即可。对于XP与linux之间的文件交换则采用共享文件夹的方式，在 linux 下共享文件要用到 samba 服务器[18]。

tftp 是一种小文件传输协议，它基于 UDP 协议实现。tftp 服务器的建立是用 windows 下的一个应用程序 tftpd32 来提供。网络文件系统 NFS 则是可以把局域网上的共享文件做为 local 的文件来操作。NFS 服务器需要在 linux 下配置，首先建立一个用户名为 hank,密码为 111111 的用户，建立完后在 home 目录下就会有一个 hank 的文件夹，用 chmod 777 hank 修改其属性，然后再用 vi 编辑 etc 文件夹下的 exports 文件，输入内容“/home/hank 192.168.1.200/24(rw,sync,no_root_squash)”保存退出，在 shell 里输入 service portmap restart & service nfs restart 命令，等待NFS服务启动。nfs 服务启动成功后可以用“mount -t nfs 192.168.1.200:/home/hank /mnt”来测试服务是否配置成功，如果挂载成功则说明配置好了，反之，需要重新配置。Samba是可以把linux上的文件共享出来的一个软件，它能把linux下的硬碟和打印机共享出来，相当于 windows 下的网络邻居，在 shell 下输入 redhat-config-samba 后,会弹出 samba 服务器的配置界面,首先需要添加 samba 用户 hank,密码设为 111111，其次是添加共享目录，目录设置为/home/hank,访问属性设置为读写，保存退出，再用 service smb restart 来启动 samba 服务，samba 服务启动成功后，可以在 XP 下的运行里输入：\\192.168.1.200\hank 来测试是否连接上 Linux 下共享出来的文件夹[19]。

4.2 U-Boot 的移植

在嵌入式系统中用来引导系统初始化的部分被称为 bootloader，它相当于 PC 机中的 BIOS。嵌入式系统中常见的 bootloader 有 ppcboot、redboot、u-boot 和 vivi 等。该系统中采用 u-boot 作为引导加载程序即 bootloader，它支持多种不同的硬件平台[20]。

4.2.1 Bootloader 介绍

U-boot 的全称是 universal boot loader，它是由德国 denx 的软件工程师 wolfgang denk 开发的。u-boot 是一种开放源码的 bootloader，它不仅支持多种嵌入式系统的引导如 linux、netbsd、vxworks、qnx 等，而且支持许多不同的硬件架构，u-boot 的代码编写风格比较相似于 linux[21]。U-boot 在嵌入式系统中主要用来加载和引导 linux 内核、初始化硬件设备、传递内核引导参数、加载文件系统、通过网络协议或串口通信协议将二进制文件下载到内存的任意位置，另外它还可以把内存中的数据烧写到 flash 中。该系统移植的 u-boot 版本是 1.2.0。

4.2.2 u-boot 启动过程

U-boot 启动过程分为 2 个阶段：阶段一和阶段二。第一阶段主要是与体系结构密切相关的汇编代码，第二阶段是与 CPU 体系结构不太相关的系统功能扩展。第一阶段主要是定义程序入口、初始化 CPU 时钟和工作模式、初始化内存、初始化堆栈、启动代码重定向[22]。第二阶段进入 C 语言代码，主要是初始化 flash，根据硬件的配置信息初始化网卡、扩展 u-boot 的命令和初始化控制台，初始化完毕后进入主循环等待用户输入 u-boot 命令。

4.2.3 u-boot 的移植

移植 u-boot 就是根据目标板的硬件配置信息，修改相关的文件和配置选项，然后编译下载调试，直到完成所有需要的功能。一般 u-boot 应可以通过网络和串口下载文件，可以加载可执行文件，可以烧写程序或数据等功能。根据上面的功能可以知道需要做的工作主要是移植 NAND FLASH 驱动和网卡驱动程序[23]。U-boot 并没有直接提供对 S3C2440 平台的支持，因此需要找到最相近的板子 SMDK2410 来移植，该系统用的 u-boot 中已经支持了标准的 2410 开发板。

移植过程如下：

(1) 添加目标板

拷贝 board/smdk2410 文件夹及其子文件到 board 文件夹下并命名为 eck2440。修改顶层 Makefile 添加对 S3C2440 板子的支持、修改交叉编译器使其指向 arm-linux-gcc,重命名 smdk2410.c 为 eck2440.c，同时需要修改顶层目录下 Makefile[24]。

(2) 第一阶段需要修改的文件

U-boot 第一阶段需要修改的文件有 2 个：cpu/arm920t/Start.S 和 board/utu2440/lowlevel_init.S。Start.S 是程序的入口，这个文件的主要功能是设置中断向量，初始化 CPU 时钟，设置堆栈的起始地址和大小。在这里需要修改的地方有:添加 S3C2440A 相关寄存器的定义和内存设置值代码，根据芯片说明设置中断屏蔽寄存器的值，初始化 UART 以便打印调试信息，屏蔽从 NOR FLASH 启动，添加从 NAND FLASH 启动代码,设置 SDRAM 所在 BANK 的寄存器以初始化内存[25]。S3C2440A 支持从 NAND FLASH 启动，系统启动时 CPU 会自动把 NAND 设备的第 0 块的前 4KB 数据读入片内 SRAM，因此在 u-boot 的最前面的一段程序中只要把 u-boot 的运行代码搬移到 SDRAM，然后再跳转到 SDRAM 中去执行即可实现自举启动。lowlevel_init.S 文件功能主要是：设置 SDRAM 的刷新率，设置每个 BANK 的工作模式和匹配每个 BANK 的时序。这里需要根据每个 BANK 上实际所连接的设备，参考其操作的时序图来进行设置。

(3) 第二阶段需要修改的文件

该阶段需要实现存储设备的读写和擦除，网络的下载功能。即存储设备 NAND FLASH 支持需要添加 NAND FLASH 的读写擦除等驱动函数，添加对本系统所用的 K9F1208U0M flash 的支持。针对网络下载功能则需要修改网卡驱动和网卡所在的物理地址[26]。在 include/linux/mtd/nand_ids.h 文件中增加 K9F1208U0M flash 的 ID,以便系统在初始化的时候能够识别该 FLASH，在 include/configs/utu2440.h 文件中增加存储器 NAND 控制器的寄存器地址，设置 DM9000A 网卡的实际基地址为 0x18000300，设置 DM9000A 网卡工作在 16 位模式[27]。在 board/utu2440 文件夹下加入 NAND Flash 读写函数；在 board/utu2440/utu2440.c 文件中增加 NAND FLASH 底层初始化、读写、校验、片选等函数；修改 drivers/dm9000x.c,在检查数据包是否准备好的地方增加对 DM9000_MRRH 和 DM9000_MRRL 寄存器的读操作，主要是让接收缓冲区的内存地址

复位到0xc00。

(4) 需要注意的地方

(a) 在没有移植网卡的时候应该在目标板的头文件里注释掉网卡的配置信息，否则编译出来的u-boot下载到目标板执行时会死机。

(b) 为了使每次接收数据不需要等待，在 u-boot 启动的时候应该直接初始化DM9000A驱动，而不是等到运行tftp或ping命令时再初始化网卡驱动，所以需要在 board.c 文件中增加网卡初始化函数的调用eth_init()。

(c) 如果网卡移植完成后用 tftp 命令下载调试时出现“checksum bad”错误，这时需要修改网卡所在BANK的时序即修改lowlevel_init.S文件，把#define B3_BWSCON (DW16 + WAIT + UBLB)修改为：#define B3_BWSCON (DW16)。

(d) 由于交叉编译器用的是硬件来做浮点运算，而u-boot 用的则是 软件来实现该运算，所以编译的时候会出错。需要将 cpu/arm920t/config.mk文件中-msoft-float这个参数注释掉。

(e) 由于交叉编译器不支持 apcs-gnu 标准，故编译过程中会出现错误，此时需要将cpu/arm920t/config.mk文件中的“-mabi=apcs-gnu”参数去掉，这是因为函数会检查编译器是否支持给定的apcs选项，apcs指的是处理器过程调用标准，在 APCS 标准中有两个版本 apcs-32 和apcs-gnu[28]。cc-option首先检查是否支持apcs-32标准，如果不支持的话就使用 apcs-gnu 标准，要是给定的两个都不支持的话就不需要指定了。

(f) 对于板子mach type的定义U-boot中应该和linux内核中的定义相一致，否则u-boot无法正确的引导linux内核。U-boot中对机器号的定义在 mach-types.h 文件中定义，linux 中对 mach type 的定义在mach-types.h文件中定义，都是在include/asm/目录中定义的[29]。

4.2.4 制作补丁文件

diff 命令是制作补丁文件的，而 patch 命令是安装补丁的。diff 相当于windows下的fc命令，它们都是文件比较命令，即比较2个文件的不同之处然后记录下来，不同的是diff可以依次比较两个不同文件夹下具有相同文件名

的所有文件，而 fc 则只能比较同一个文件夹下的 2 个文件。diff 命令有 3 个比较重要的参数：

-r 代表递归的意思，即是说要比较 2 个不同文件夹下的所有文件，包括子文件夹；

-N 表示补丁文件对已经建立或者删除的文件能正确的处理；

-U 表示按一定的格式生成补丁文件。

打补丁需要用到 patch 命令，这个命令是根据 diff 命令制作出来的补丁文件来修改源文件或目标文件的，即这个命令可以根据补丁文件把源文件打补丁变成新文件，也可以把打过补丁的新文件变成老的源文件[30]。它常用的参数是-p0 或-p1 表示是从第一级目录还是从第二级目录来查找源文件。该系统所移植的 u-boot 版本是 1.2.0，制作补丁文件和打补丁的过程如下：

（1）制作补丁

先把移植好的 u-boot 命名为 myok-u-boot-1.2.0，然后解压 uboot 源码到同级目录，再用 diff 命令制作出补丁文件 uboot-1.2.0.patch，具体操作命令如下：

```
mv   u-boot-1.2.0 ok-u-boot-1.2.0

tar –xjvf u-boot-1.2.0.tar.bz2

diff –Nura   u-boot-1.2.0    ok-u-boot-1.2.0>uboot-1.2.0.patch
```

（2）打补丁

该系统打补丁的方法是进入到解压好的 u-boot 源码目录下然后，用 patch 命令对补丁文件输入重定向，操作命令如下：

```
cd u-boot-1.2.0

patch –p0 < uboot-1.2.0.patch
```

（3）去除补丁文件

去除补丁文件需要用到 patch 命令的 R 和 E 参数：

-R 表示把打过补丁的文件恢复成源码文件；

-E 表示如果遇到空文件就删除空文件；具体的命令如下：

```
patch    -RE –p0 < uboot-1.2.0.patch
```

（4）将移植好的源码打包

一般移植好的 u-boot 源码应该先打包做备份，然后再制作补丁文件，打包相当

于 windows 下用压缩软件压缩文件，linux 下打包用 tar 命令，它常用的参数有：
c 表示创建压缩包的意思；
j 和 z 表示 2 种不同的压缩格式，j 表示 bz2 格式的压缩包，z 表示普通格式的压缩包；
-f 表示创建压缩包的具体文件名，具体的命令如下：
可以用 bz2 格式压缩
tar –cjf ok-uboot.tar.bz2 ok-uboot/
或者也可以用 zip 格式压缩
tar –czf ok-uboot.tar.gz ok-uboot/

4.2.5 烧写 u-boot 到目标板

如果板子是裸板，则第一次烧写 U-boot 到目标板需要借助 JTAG 烧写工具和软件。JTAG 是 JOINT TEST ACTION GROUP 的简称，JTAG 调试的基本内容是 TAP(Test Access Port)和边界扫描[31]。TAP 是一个有 TAP 控制器控制的一个通用的端口，通过这个端口可以访问芯片内部所有的数据寄存器和指令寄存器。TAP 共有 5 个信号：TCK, TMS, TDO, TDI, TRST。一般采用并口简易 JTAG 下载电缆，借助于三星公司提供的下载工具可以把编译好的 u-boot.bin 文件烧写到 NAND FLASH 中去[32]。

如果不是第一次烧写，则可以用 u-boot 来更新自己，既可以通过串口下载 u-boot 到内存中更新，也可以通过网口下载 u-boot 到内存中然后再更新。用串口下载时需要在 u-boot 下首先输入命令 loadb，然后用超级终端发送到目标板，最后用 nand flash 烧写命令把下载的文件从 SDRAM 写入到 FLASH 中。用网口下载时首先需要在主机上运行 tftp32.exe 以开启 tftp 服务器并设置好包含 u-boot.bin 的下载目录，然后在目标板的 u-boot 下首先输入命令 tftp 32000000 u-boot.bin，最后再用 nand write 命令把 u-boot 写入 FLASH。

4.3 Linux 内核移植

移植内核就是把Linux 内核根据具体的目标硬件做必要选择配置和改动，编译后安装到目标板上使其能正常稳定的运行起来。该系统采用稳定版内核2.6.14为基础，在其上进行裁剪和移植，最终产生二进制的内核映像文件。要移植linux首先需要对linux的内核结构比较了解，其次要对硬件的连接组成和工作原理有深入的认识，同时还要对ARM体系结构的汇编语言有所了解[33]。移植的一般过程是：下载linux内核源代码和体系结构相关的补丁，并打补丁到内核源码上，添加和修改相关硬件驱动和文件系统驱动，裁剪内核，交叉编译，下载调试，固化内核。

4.3.1 Linux 内核结构

Linux 内核由 5 个具有独立功能的子系统组成，它们是进程调度、内存管理、虚拟文件系统、网络接口和进程间通信[34]。Linux 中的进程（process）可以看成是一个独立的程序，在内存中有其完备的数据空间和代码空间。进程调度就是协调和控制各个进程所占用的 CPU 资源，用以保证所有的进程都能有机会得到 CPU 的资源。内存管理就是管理系统的内存使得进程间可以共享系统的内存，虚拟文件系统是将外部的存储设备如硬盘、U 盘、FLASH 等抽象成一个统一的接口，它是上层应用软件和底层存储系统间的桥梁。进程通信是两个进程之间或者是多个进程之间数据的传递和交换。网络接口是指网络设备驱动及大量的网络协议。

linux 源码大部分采用 C 语言实现，也包含一部分的汇编代码。Linux 内核源码下有很多不同的目录，认识内核的目录结构对于快速便捷的移植内核和驱动程序有很重要的意义[35]。内核目录下的文件夹以及 arch 子目录和 net 子目录如下图 4-2 所示：

```
[root@localhost linux-2.6.14]# ls
10-9.config       COPYING                  include     Makefile        security
2009-3-18.config  CREDITS                  init        mm              sound
3-17.config.bak   crypto                   ipc         Module.symvers  System.map
6-16.config       Documentation            Kbuild      net             usr
arch              drivers                  kernel      README          vmlinux
bak.config        flashok_2009-3-18.config lib         REPORTING-BUGS
config-6-15       fs                       MAINTAINERS scripts
[root@localhost linux-2.6.14]# ls ./arch/
alpha  arm26  frv    i386  m32r  m68knommu  parisc  ppc64  sh    sparc   um    x86_64
arm    cris   h8300  ia64  m68k  mips       ppc     s390   sh64  sparc64 v850  xtensa
[root@localhost linux-2.6.14]#
```

图 4-2 Linux-2.6.14 目录结构

arch 目录下主要包含着与处理器硬件体系结构相关的代码，每种架构的硬件处理器对应一个文件夹，例如该粮情监测系统用的是 ARM 架构的处理器，与硬件平台相关的代码如寄存器设置、内存映射、内存管理、引导程序都放在 arm 子文件夹下。Documentation 存放的是内核相关的文档说明。drivers 目录是用来存放内核中包含的设备驱动程序，像字符设备、网络设备、各种总线、USB 设备驱动等，例如该粮情监测系统所需要修改的有 DM9000A 网卡驱动、USB 驱动、NAND FLASH 驱动和串行口驱动就是放在此目录下。fs 目录主要包含各种文件系统的驱动代码和操作实现，每个子目录对应着一种文件系统。include 目录存放的是编译内核时需要依赖的头文件[36]。init 目录是 linux 内核 main 函数所在的目录，内核的启动过程就是从 main 函数的各个调用开始的。ipc 目录是进程间通信所需要依赖的函数实现。kernel 存放的是内核最核心的代码，内核的最基本功能就是从这里开始的。lib 目录存放的是内核的核心代码库。net 目录存放的是网络协议的实现代码，mm 存放的是与硬件平台无关的内存管理的代码。

4.3.2 下载内核并打补丁

Linux 内核源码可以很容易的从 linux 内核官方网站 http://www.kernel.org 上下载到，而与 ARM 体系结构相关的硬件移植代码可以从英国的网站 http://www.arm.linux.org.uk 上下载[37]。补丁文件一般由补丁头、块、块的缩进、块的第一列组成。打补丁同 u-boot 打补丁的方法类似，一般用 patch 命令加参数-p0，表示从当前的目录中查找目标文件或文件夹。

4.3.3 添加相关驱动

Linux 内核源码分为与硬件平台相关的部分和与硬件平台无关的部分，一般需要修改的就是与硬件平台相关的部分的代码[38]。例如添加一些外设驱动，根据硬件的连接修改相关代码，重新生成一款适合该监测系统的操作系统，在该系统中主要需要添加的有 NAND FLASH 驱动、DM9000A 网卡驱动、USB 驱动、串行口驱动和 yaffs2 文件系统驱动，此外还需要设置 NAND FLASH 的分区信息。对于 NAND FLASH 驱动的移植和 DM9000A 网卡驱动移植，其原理同 u-boot 下的一样，不再做详细的阐述。对于 yaffs 文件系统，linux 内核中没有自带 yaffs2 文件系统的源代码,所以需要从 yaffs 的官方网站上下载 yaffs2 对于 linux 2.6 版本内核的补丁，然后在配置文件和编译文件里加入对 yaffs 的支持，重新编译就可以让内核支持该文件系统了，打补丁 yaffs2.tar.gz 的具体操作步骤如下：

tar xvjf yaffs2.tar.gz

cd yaffs2

./patch-ker.sh c ../linux-2.6.14

注：此处 c 是指 copy，可以改为 l，l 表示是 link 的意思。

对于NAND FLASH的分区信息的设置是在arch/arm/machs3c2410/devs.c文件中设置的，可以按自己的需要在此文件里设置详细的分区信息；对于内核在启动的过程中需要初始化的硬件设备的设置需要添加和修改arch/arm/mach-s3c2410目录中的utu2440.c文件[39]。

4.3.4 配置内核

配置内核其实就是裁剪，添加与目标板相关的选项，去除一些不必要的和无用的模块。在内核目录下通过 make menuconfig 进行设置。配置的时候首先需要配置内核启动时的参数,在Boot options选项下Default kernel command string设定为：

noinitrd root=/dev/mtdblock2 init=/linuxrc console=ttySAC0,115200。

其余的选项按自己的需要来选择。

4.3.5 编译调试

内核配置完成后，就可以在终端下运行 make zImage 或者 make uImage 命令，等待交叉编译器编译出二进制的内核映像文件，生成二进制的内核映像文件 zImage 在 arch/arm/boot/目录下。编译内核时需要使用 3.4.1 版本的交叉编译器。在移植的过程中需要不断的排除编译过程中的错误，反复的修改、编译、下载、调试，最终生成稳定的内核映像文件。

（1） zImage 与 uImage 的区别

这 2 个文件的区别是：zImage 可以用 u-boot 的 go 命令直接执行， zImage 头部带有解压代码，所以执行 go 命令后，内核会直接解压启动，但是用这种方法启动的内核不能直接加载文件系统，因为 go 命令不会传递启动参数给内核。如果想要传递启动参数给内核映像，则需要用到 u-boot 的 bootm 命令，而 bootm 命令只能处理 uImage 格式的内核映像[40]。uImage 是通过 u-boot 自带的一个工具 mkimage 生成的，它在 zImage 的基础上创建 u-boot 首部，加在 Linux 内核映像之上构成了 uImage。

（2） 把 zImage 转化为 uImage

把zImage转化为uImage可以通过创建一个脚本文件来完成，脚本文件如图4-3下：

./mkimage –A arm –O linux –T kernel –C none –a 0x30008000 –e 0x30008000 –n “linux-2.6.14” –d ./zImage ./uImage

此处的 mkimage 是由 u-boot 的源代码目录下 tools 目录里编译生成的。参数 a 表示 u-boot 的加载地址即让 u-boot 把内核加载到内存的位置，参数 A 表示处理器的体系结构，参数 e 表示映像文件的入口地址，参数 n 可以作为用户设置内核版本信息的标记。

（3） 烧写内核映像

在内核还没有最终调试好的时候，一般利用 u-boot 的网络下载功能把内核镜像文件下载到 RAM 中直接运行，观察调试信息是否正确。操作过程如下：

```
tftp 30008000 uImage; bootm
```

内核调试成功后，就可以通过网络下载到 SDRAM 中，然后利用 u-boot 的烧写功能把内核映像从 SDRAM 烧写到 NAND FLASH 中固化，可以用以下命令：

```
tftp 38000000 uImage
nand erase 40000  200000
nand write 38000000  40000  200000
```

4.4 文件系统构建

嵌入式系统中用的最多的存储介质还是闪存，由于 NOR FLASH 存储器容量较小，所以嵌入式系统的文件系统一般都是存放在 NAND FLASH 上。文件系统需要驱动程序的支持，基于 linux 的文件系统通常采用内存驱动技术的支持。该技术的英文全称是 Memory Technology Device，它主要目的是为了实现操作系统与硬件存储设备的无关性，对上层的应用和操作提供一系列通用的函数接口，对下层硬件则是提供具体的各种类型的存储器的驱动函数。常用的嵌入式文件系统有：CRAMFS 文件系统、YAFFS 文件系统、JFFS 文件系统、RAM 文件系统、FAT 文件系统等[41]。CRAMFS 文件系统是一种基于 RAM 的且经过压缩的只读的文件系统，它在运行的时候不是一次性的把文件系统解压到内存中，而是临时的把需要用的到数据解压到 SDARM 中，由于是只读文件系统所以不适合做日志的写入和保存。YAFFS 文件系统是一种日志型的文件系统，它是专门为 NAND FLASH 而设计的一款文件系统，它是在充分的考虑了 NAND 型 FLASH 的优点和缺陷后而设计的，它有效的提高了文件系统的挂载速度，优化了损耗平衡使得 FLASH 的寿命得到了提高[42]。JFFS 是由 redhat 公司开发的一款免费的嵌入式文件系统，主要是针对 RedHat 公司的嵌入式操作系统 eCos 而设计，它主要用于 NOR 型的 FLASH，最新的 eCos 操作系统也有用 exFAT 做为文件系统的。 基于 RAM 的文件系统常见的就是 ramdisk，它是把内存的一部分空间分割出来作为文件系统的载体，相当于 windows 下的虚拟内存盘或者是 DOS 下的 RAM DRIVE，用 RAM 做文件系统读写速度毋庸置疑，肯定是最快的，但是 RAM 同时也具有掉电不保存的缺点，所以它不是日志型的文件系统。经过多方面的考虑，该粮情监测系统中采用 yaffs 文件系统。

基于 linux 内核的嵌入式文件系统的结构与发行版的 linux 没有太大的差别，前一个几乎就是后一个的精简版本，只不过嵌入式系统中的这些目录下的文件大多都是通过交叉编译得到的。文件系统目录结构如下图：

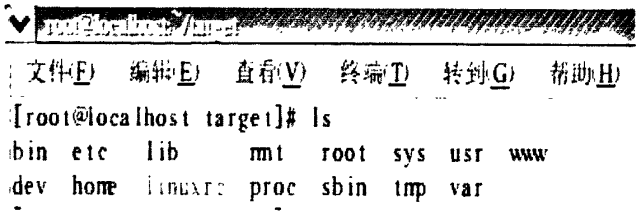


图 4-3 嵌入式系统根文件系统的目录

bin 文件夹下存放的是 linux 常用的基本命令，大部分都是 busybox 的链接；
sbin 下存放的大多是 linux 的扩展命令，例如网卡配置命令 ifconfig；
dev 文件夹下存放的是设备文件，例如温度传感器设备文件 s3c2440-ds；
etc 里存放的都是一些配置文件和启动时用的脚本文件，如果要让自己编写的应用程序在 linux 开机后自启动，就需要修改该目录下的 rcS 文件；
lib 中存放的是运行 linux 时所需要的静态链接库。

制作文件系统的一般步骤是：先用 BusyBox 的源码交叉编译出 linux 下的常用命令和工具拷贝到 bin 和 sbin 文件夹下，其次根据具体的硬件资源创建设备文件到 dev 目录，然后再创建 OS 的启动脚本和系统配置文件并拷贝到根目录和 etc 文件夹下，接着拷贝共享链接库到 lib 文件夹下，最后再把这些创建好的文件和文件夹用 yaffs 文件系统里的一个工具软件 mkyaffsimage 来制作出 yaffs 文件系统镜像文件。制作文件系统的另外一种方法是直接拷贝其他目标板上未打包的文件系统，然后修改相应的启动脚本和配置文件，再用文件系统制作工具直接打包成文件系统映像文件。BusyBox 可以称为“linux 的瑞士军刀”，它实际上是一个 unix 的工具集合，它利用最少的代码实现尽可能多的 GNU 常用命令。交叉编译后的 BusyBox 虽然只有 500KB 左右，但是却包含了很多的命令。编译后的所有命令几乎都是可执行二进制文件 BusyBox 的链接[43]。

在移植过程中如果碰到 NAND FLASH 上提示有较多的坏块，则原因可能是由于错误的移植文件系统使得 NAND FLASH 上原本是好的块被标记成了坏块，这需要在 U-BOOT 下用 nand scrub 命令重新擦除一下 NAND FLASH，擦除的时候会重新标记坏块。

5 嵌入式 Linux 设备驱动设计

驱动程序是 linux 内核很重要的一部分，linux 内核源码中驱动程序占用了几乎一半的代码。在 linux 操作系统中执行的程序可以分为内核态和用户态，驱动程序一般是运行在内核态的，它对上通过 VFS 给操作系统中的应用程序提供标准的文件读写控制接口函数，对下提供具体的硬件操作函数接口[44]。内核态和用户态分别对应于处理器不同的处理级别，内核态的级别高于用户态的级别。

5.1 设备驱动程序概述

Linux 是把硬件作为文件来处理的，linux 对硬件外设的管理称为设备管理，它是操作系统的重要组成部分。Linux 把几乎所有的硬件都当作文件来处理，这种文件是一种特殊的文件即设备文件，linux 系统通过处理文件的接口即虚拟文件系统来管理和使用各种硬件设备的资源和功能[45]。与 PC 机和外设的数据传输方式一样，linux 内核与设备之间的数据传输也分为三种方式：软件查询、中断处理、DMA 直接存储器存取。例如字符型设备驱动的工作过程为：当用户需要使用某个硬件设备时，应用程序首先调用标准的文件操作函数提交数据请求，根据设备文件的名字去决定操作何种硬件设备，设备驱动程序再根据内核中该硬件的声明和注册的设备来管理各种操作，设备驱动程序和应用程序是通过一个结构体 file_operations 来衔接在一起的。file_operations 结构体里又链接了对硬件设备的具体的打开、读写、关闭、释放等操作。

5.1.1 虚拟地址

对于 X86 架构的处理器，有三种不同的地址：逻辑地址、线性地址和物理地址，而 linux 操作系统却只是用了线性地址和物理地址。几乎所有的处理在处理地址总线的时候用的都是线性地址，只有当 CPU 要访问外设或内存时才会通过内存管理单元做地址转换，所以任何访问外设的请求都需要先做页查找，页表中就存放了地址的转换[46]。对于 Linux 系统，它在访问设备的时候用的是虚拟地址，需要经过 MMU 来进行地址映射。没有 MMU 内存管理单元的处理器，

其访问外部存储器的时候使用的是实际的物理地址来直接访问的，在其上运行的应用程序需要考虑程序越界问题。MMU 解决了大程序在小内存平台上的运行问题，让操作系统在不同的时刻调入程序的不同部分，这样就形成了虚拟地址的概念。例如一台 PC 机采用 32 位的处理器，那么它理论上可以访问的空间是 4G，但是它的内存可能只有 256M 或者更小。在 linux 驱动程序中访问实际的物理地址是不能直接访问的，而是要用 ioremap 函数把实际的物理地址映射到 linux 内核驱动可以访问的虚拟地址。

5.1.2 设备驱动程序类型

Linux 内核本身包含了许多的驱动程序，这些驱动程序大体上可以分为 3 种不同的类型：字符设备驱动程序、块设备驱动程序和网络设备驱动程序。

字符型设备驱动程序就是操作和控制字符的设备驱动程序，这些字符可以是单个字符，也可以是长短相同或者长短不同的字符序列。这种设备驱动在 linux 内核里占了绝大多数，例如：键盘设备驱动、A/D 转换设备驱动、LCD 驱动等。应用程序可以直接通过标准的文件操作函数 open、close、read、write、ioctl 等来操作和控制硬件设备[47]。

块设备就是按一定的数据块来处理数据的设备驱动程序，应用程序一般不会直接调用块设备驱动程序，而是通过文件系统接口来管理和操作块设备，例如存储卡设备驱动程序和 IDE 硬盘设备驱动程序。

网络设备驱动程序就是与网络相关的一些驱动程序，具体包含了与硬件操作有关的驱动和网络层相关的程序，字符型设备驱动和块设备驱动这 2 种设备驱动程序都可以通过访问设备文件来操作具体的硬件，网络设备驱动程序则不能，它是通过 linux 内核内部的网络协议栈来操作和控制网络硬件设备。

5.1.3 中断处理

根据 linux 下三种不同的 I/O 控制方式：查询、中断和 DMA，linux 下的设备驱动可以分为查询型驱动程序和中断型驱动程序[48]。有些驱动程序采用程序查询的方式来监测输入输出端口，例如 A/D 转换驱动程序，有些则采用中断方式，例如键盘的驱动程序往往采用中断处理的方式。

(1) 中断的申请：

中断型的驱动程序首先需要申请中断，中断的申请一般是在初始化程序里用

request_irq()函数来实现的，以按键的中断初始化程序为例：

```
Eg：static int s3c2440_key_init(void)
{
err=request_irq(key_info_tab[i].irq_number,key_irq,SA_INTERRUPT,KEY_DEV,N
 ULL);
}
```

这里有 5 个参数：

第一个参数表示需要申请的中断号，与硬件密切相关；

第二个表示中断服务子程序，自己定义的，通过 key_irq 来定义了中断服务子程序，其中中断号和中断服务子程序是一一对应的；

第三个 SA_INTERRUPT 表示不允许其他中断，可选的有 SA_SHIRQ 共享相同的中断号，SA_SAMPLE_RANDOM 影响 RANDOM 处理；

第四个 device 表示中断的所有者；

第五个是共享中断时的中断区别 ID。

(2) 中断类型

中断型的设备驱动常用 set_irq_type 函数来设置中断的类型。在 set_irq_type（irq，type）中的 type 是表示中断类型的，一般中断可以设置为上升沿、下升沿或双边沿有效、低电平有效和高电平有效。具体设置如下：

```
#define IRQT_RISING (__IRQT_RISEDGE)//表示把中断设置为上升沿有效
#define IRQT_FALLING (__IRQT_FALEDGE)//表示把中断设置为下升沿有效
#define IRQT_BOTHEDGE (__IRQT_RISEDGE|__IRQT_FALEDGE)//表示把中
断设置为双边沿有效
#define IRQT_LOW (__IRQT_LOWLVL)//表示把中断设置为低电平有效
#define IRQT_HIGH (__IRQT_HIGHLVL)//表示把中断设置为高电平有效
```

(3) 中断驱动程序与 file_operations 结构体

在 linux 中，驱动程序的接口是由 file_operations 结构体向内核说明的，对设备的操作都是通过 file_operations 这个结构体作为入口点来进行的，而中断申请函数与 file_operations 结构体的关系可以通过两种方式来关联：

第一种是在用 register_chrdev()函数注册时，通过该函数传递的第三个参数和 file_operations 相关联。如下：

```
ret=register_chrdev(KEY_MAJOR,KEY_DEV,&key_fops);
```

第二种是在用 cdev_add()函数注册时是通过其初始化函数 cdev_init()的第二个参数来关联。如下：

```
cdev_init(&keyboard_dev,&key_fops);
```

(4) 中断服务程序和阻塞型输入输出

中断设备驱动程序为了提高工作效率，需要对设备驱动进行阻塞处理。一般的读取程序要比中断程序执行的慢的多，所以读取数据的速度就比较慢，读数据读的大部分是中断处理完后所得到的结果，而读函数read需要等待中断处理完成，所以在read函数里需要加入使进程睡眠的等待队列。如果外设的处理速度比较慢，当应用程序向设备驱动程序发出操作请求时，在外设还没有准备好数据的时候使进程进入睡眠的状态，当外设准备好数据后，再唤醒相应的进程来读取数据，从而可以提高linux的运行效率[49]。

以键盘为例，在键盘的读函数中需要设置阻塞，如下：

```
if(!ready)
   {
   if(flip->f_flags&O_NONBLOCK)return -EAGAIN;
   else
   {
// wait_event_interruptible(key_wait,ready);
   interruptible_sleep_on(&key_wait); }
}
```

其中ready是一个标志位，主要是作为判断中断是否结束了，数据是不是可读的标记。如果ready为零，那么进入if判断，首先读取flip->f_flags标志，判断是否需要等待，然后看O_NONBLOCK标志判断设备文件是否以非阻塞模式打开，只有这2个条件同时不满足，才可以进入等待。如果设备文件是否以非阻塞模式打开的，那么就不能使用interruptible_sleep_on()函数使进程进入睡眠。而另一种方法就是采用wait_event_interruptible()函数来使进程进入睡眠状态，ready是一个条件表达式，表示直到ready为“真”时才被唤醒。执行的过程是在中断申请函数中首先需要屏蔽其他中断，进行一段时间的延时以消除按键的抖动，然后再把GPIO管脚设置为普通的输入模式读取该管脚的电平，恢复屏蔽的中断，根据读到的GPIO管脚的高低电平，来判是否有键按下。在中断子程序处理完的时候，应该用wake_up_interruptible(&key_wait)函数来唤醒等

待队列。

(5) 中断的屏蔽和恢复

中断的屏蔽一般要用local_irq_save函数来保存当前中断的状态，再用local_irq_disable函数禁止中断，然后处理自己的操作，最后再用local_irq_restore函数恢复中断标志。如下：

```
local_irq_save(flags);//保存中断标志
local_irq_disable();//关中断
key=key_value;//读键值
local_irq_restore(flags);//恢复中断标志
```

5.1.4 设备注册和注销

在该监测系统中注册设备驱动程序主要是指字符型设备驱动程序的注册和注销，对于块设备的注册和注销不在研究范围之内。字符型设备驱动的注册既可以用 register_chrdev()和 register_chrdev_region()函数来实现，也可以用 cdev_add()函数来注册实现。2.4 版本的 linux 内核常常用 register_chrdev()函数来注册设备，用 unregister_chrdev()来注销设备，对于 2.6 版本的内核两种注册和注销方式都可以，推荐用 register_chrdev_region()函数来注册设备，用 unregister_chrdev_region()函数来注销设备[50]。

对于设备文件的创建，既可以在设备驱动程序加载后用 mknod 命令在 dev 目录下手工的创建，也可以在驱动程序中加入自动生成设备文件的代码来创建。例如手工创建 LED 设备文件的命令是：**mknod /dev/s3c2440_led c 97 0**

其中 c 表示字符设备，97 是代表主设备号，0 代表次设备号，字符型设备的主次设备号，可以通过查看/proc/devices 文件得到的，devices 文件的内容如图 5-1 所示：

```
[root@utu-linux /]# cat /proc/devices
Character devices:
  1 mem
  2 pty
  3 ttyp
  4 /dev/vc/0
  4 tty
  5 /dev/tty
  5 /dev/console
  5 /dev/ptmx
  7 vcs
 10 misc
 13 input
 14 sound
 29 fb
 81 video4linux
 89 i2c
 90 mtd
128 ptm
136 pts
180 usb
204 s3c2410_serial
254 utu2440-buttons
```

图 5-1 设备文件的主次设备号

要在程序中自动生成设备文件的方法是在驱动中加入一个结构体 struct class，然后用 class_create()函数创建这个结构体，最后再用 class_device_create()函数来创建该设备文件。具体的代码如下图 5-5 所示：

```
static struct class *irq_class;
.....
dev_t devt=0;//设备号
ret=register_chrdev(KEY_MAJOR,KEY_DEV,&key_fops);//注册设备
irq_class=class_create(THIS_MODULE,"s3c2440_key");//关联类
devt=MKDEV(KEY_MAJOR,0);
class_device_create(irq_class,devt,NULL,"s3c2440_key");
```

5.1.5 驱动程序头文件

设备驱动程序常用的头文件如下：

(1) #include <linux/module.h>

与模块有关的头文件，需要依赖此头文件的函数主要是：模块声明、模块的版本和作者声明等函数，例如 MODULE_AUTHO()和 MODULE_LICENSE("Dual BSD/GPL")函数、THIS_MODULE 常量和模块的计数 try_module_get()函数等。

(2) #include <linux/kernel.h>

主要是与内核有关的以及打印内核信息的函数，例如 printk，vsfprint 和 min()，max()等函数。

(3) #include <linux/fs.h> /*文件系统有关的*/

主要是对驱动中用到的 file_operations 结构体的定义。

(4) #include <linux/init.h>

模块的初始化函数 module_init()和移除函数 module_exit()需依赖此头文件。

(5) #include <linux/sched.h>

系统调用要用到的函数，同时也包含了各种信息处理函数的结构定义。

(6) #include <linux/errno.h>

错误信息和错误代码依赖文件，例如用到 ENODEV 常量时需要包含此头文件。

(7) #include <linux/types.h>

在此头文件中定义了__u16,__32,ssize_t 等数据类型。

(8) #include <linux/fcntl.h>

与文件的属性相关的常量的定义需要依赖此头文件。

(9) #include <linux/device.h>

自动创建设备文件名的函数 class_create(),class_device_create ()的定义。

(10) #include <linux/delay.h> /*delay*/

与延时有关的如毫秒延时 mdelay()和纳秒延时 ndelay()函数的声明。

(11) #include <linux/poll.h> /*poll*/

等待队列 poll_wait()函数需要依赖的头文件，该函数会监测进程队列 button_waitq 里的进程。

(12) #include <asm/irq.h>

中断触发方式的定义：IRQT_NOEDGE,IRQT_RISING 上升沿有效,RQT_FALLING 下升沿有效,IRQT_BOTHEDGE 双边沿有效，IRQT_LOW 低电平有效，IRQT_HIGH 高电平有效等常量需要依赖此头文件。

(13) #include <linux/interrupt.h> /*linux 中断*/

中断申请、中断释放、中断禁止、中断使能等函数需要依赖此文件例如：request_irq()，free_irq(),enable_irq()等。

(14) #include <asm/uaccess.h>

用户空间和内核空间数据传递用的函数 copy_to_user(),copy_from_user()等函数需要依赖的头文件。

(15) #include <asm/arch-xxx/regs-gpio.h> /*寄存器设置*/

具体的硬件平台中断寄存器定义,例如： asm/arch-s3c2410/regs-gpio.h。

(16) #include <asm/arch-xxx/irqs.h>

具体的硬件平台的中断号的定义，例如： asm/arch-s3c2410/irqs.h

(17) #include <asm/hardware.h> /*hardware*/

一般是在<asm/arch/hardware.h>或<asm/arch-s3c2410/hardware.h>,定义管脚配置置位函数，如 s3c2410_gpio_cfgpin()和 s3c2410_gpio_setpin()等。

(18) #include <asm/io.h>

32 位寄存器读写用的函数，如：__raw_writel 和__raw_readl。

(19) #include <linux/wait.h>

阻塞输入输出函数 wait_event_interruptible()函数需要依赖此文件。

(20) #include <asm/system.h>

中断的屏蔽和恢复函数的定义需要依赖的头文件，如 local_irq_save()和 local_irq_disable()。

(21) #include <linux/cdev.h>

字符设备的初始化函数和注册函数 cdev_init(),cdev_add()的定义。

5.2 设备驱动程序设计方法

Linux 内核的源码和驱动程序都是按模块来分类存放的。在 linux 下设计驱动，主要就是设计结构体 file operations 中定义的函数，然后再向虚拟文件系统注册一下设备就可以使用。当驱动程序的代码编写完成后，应该把驱动的源文件拷贝到 linux 内核的相应目录下，同时需要修改该驱动所在文件夹下的 Makefile 文件和 Kconfig 文件，修改 Makefile 文件的目的是添加编译规则到 Makefile 文件。使得在执行 make 命令的时候可以编译新加入的驱动程序，而修改 Kconfig 文件是为了在内核中加入配置选项，使得通过执行 make menuconfig 后可以对该驱动程序进行配置和选择，在 Kconfig 里可以配置驱动程序在编译的时候是加入到内核里，还是以模块的方式来编译。需要注意的是编译内核要用 arm-linux-gcc-3.4.1 版本的编译器，因此要修改环境变量使其指向 3.4.1 版本的交叉编译器。

5.3 温度传感器驱动程序设计

该系统采用 DALLAS 的数字式温度传感器 DS18B20,对它的操作都是通过一个管脚来实现的，这种传感器称为单总线设备，它广泛应用于粮仓中的温度检测，它的应用已经是一种非常成熟的技术。由于单总线温度传感器，只有一条信号线，所以它对时序要求比较严格，设计的时候需要首先根据其时序图来设计出它的基本操作：读一个字节、写一个字节、读 ROM、查找 ROM、匹配 ROM 等。该设备的操作比较简单，所有的基本操作都是按时序来置位和复位 GPB0 管脚。对于多个传感器的查找和搜索，则是一个二叉树的遍历问题，只要把单总线上挂接的所有传感器用二叉树的遍历算法访问一遍，就可以实现对所有单总线上温度传感器数据的采集。下面主要给出驱动的初始化和注册部分的代码如下，初始化 18B20 温度传感器的关键代码如下所示：

```
static int __init ds18b20_init(void)
{
    printk("set s3c2410_gpio_cfgpin==========\n");
    s3c2410_gpio_cfgpin(GPB0,CFG_OUT_PIN);
    printk("set s3c2410_gpio_setpin==========\n");
```

```
    s3c2410_gpio_setpin(GPB0,0);
    pr_info("debug==========%s\n",UTU_DS18B20_DRIVER);
    misc_register(&ds18b20_dev);
    return 0;
}
```

在这里注册设备的时候是通过 ds18b20_dev 这个结构体来指向对此设备的具体的操作的，而这个结构体的定义如下所示：

```
#define UTU_DS18B20_DRIVER "utu2440 ds18b20 driver v1.00"
static struct miscdevice ds18b20_dev =
{
    MISC_DYNAMIC_MINOR,
    "s3c2410-ds",
    &ds18b20_fileops
};
```

在 ds18b20_dev 这个结构体中，定义了设备文件的次设备号和设备文件的名称，另外又通过一个指针指向了 ds18b20_fileops 这个结构体，而 ds18b20_fileops 这个结构体就是 file_operations 类型的结构体，驱动程序就是把这个结构体作为入口点来进行具体的硬件操作，如下所示：

```
static struct file_operations ds18b20_fileops = {
  .owner = THIS_MODULE,
  .write = ds18b20_write,
  .read = ds18b20_read,
  .release = ds18b20_close,
  .open = ds18b20_open,
}
```

当需要注销设备的时候需要调用misc_deregister()函数来注销,如下

```
void __exit ds18b20_exit(void)
{
    misc_deregister(&ds18b20_dev);
}
```

在这个驱动程序里使用了 miscdevice 结构体， miscdevice 表示杂项设备，所有的杂项设备共用一个主设备号 10，不同的杂项设备是根据设备的次设备号来区别的，所以在用 misc_register()来注册设备的时候，实际是以主设备号 10 来调用 register_chrdev()函数实现设备的注册的。驱动程序编制完成后，还需要在内核源码目录下的 driver/char 目录下修改 2 个文件：

（1）Makefile文件中加入

```
obj-$(CONFIG_UTU2440_DS18B20) += utu2440-ds18b20.o
```

（2）Kconfig 文件中加入

```
config UTU2440_DS18B20
        bool "UTU2440 DS18B20 Driver"
    depends on ARCH_S3C2410
    help
        This option enables support for the DS18B20 control.
```

5.4 压力传感器驱动程序设计

压力传感器检测到信号后，输出的是0-20毫安的电流信号，但是ARM9的A/D转换接口接受的是电压信号，所以需要把电流信号转换成电压信号再送给ARM9的AD转换接口。一般把电流信号转换为电压信号的方法是在电路中并接一个0.1%精度的精密电阻，然后根据欧姆定律就可以得出电流与电压的对应关系。ARM S3C2440 芯片自带 8 通道10 位精度的A/D 转换器，因此对于AD转换驱动程序就比较简单了，主要的工作就是对采样控制寄存器ADCCON进行设置[49]。设置主要是设置AD转换的控制寄存器中关于比例因子和转换通道的值，设置完采集的通道后就可以启动AD转换了，启动AD转换就是把控制寄存器的最低位置1，然后等待此位被复位成零后，说明转换过程结束，就可以读取转换结果数据寄存器ADCDAT0，因为寄存器是16位的，所以读取后的结果还应该屏蔽掉高三位，才是正确的AD转换结果。

由于 AD 转换要涉及到触摸屏所用的接口，所以在配置 linux 内核的时候应该首先禁用掉触摸屏选项，这样可以用到 8 路 AD 转换通道。如果还是不够用的话，则需要再通过其他 AD 转换芯片来扩展 AD 转换通道。在 AD 转换驱动程序中主要的子程序有 2 个，一个是 AD 转换的启动函数，另一个是 AD 转换的读函数。

AD 转换的启动，主要就是对 ARM9 的 AD 转换的寄存器进行设置，函数如下所示：

```
static int adc_start(int ch,int prescale)
{
   while(0)
  {
   S3C2410_ADCCON_PRSCEN | \
   S3C2410_ADCCON_PRSCVL(prescale) | \
   S3C2410_ADCCON_SELMUX(ch) | \
   S3C2410_ADCCON_ENABLE_START
   }
}
```

对于 AD 转的读函数需要先启动 AD 转的初始化程序 adc_start()函数，然后等待 AD 转换结束，再去读取 AD 转换数据寄存器并屏蔽掉其高 3 位，最后通过用户空间和内核空间数据交换的函数传递 AD 转换的结果。具体的程序如下所示：

```
static ssize_t s3c2440adc_read(struct file *filp, char *buffer, size_t
count, loff_t *ppos)
{
    unsigned long result = 0;
    unsigned int data;
    adc_start(adc.ch,adc.prescale);
    do  {
       result = S3C2410_ADCCON;
    }while(!(((unsigned int)result) & 0x8000));
    data = (S3C2410_ADCDAT0) & 0x3ff;
    copy_to_user(buffer, &data, sizeof(data));
    return sizeof(data);
}
```

在该驱动中用到了从内核空间拷贝数据到用户空间的函数，Linux中存在不同运行级别的用户进程和内核进程。为了避免用户进程改动内核空间的数据使得内核崩溃，所以内核空间不能和用户空间直接交换数据，如果要交换数据需要通过linux内核所提供的专用函数来实现。从内核空间拷贝数据到用户空间要用

copy_to_user()函数，而从用户空间拷贝数据到内核空间则用copy_from_user()函数[50]。

5.5 GPIO 驱动程序设计

通用输入输出接口（即 GPIO 接口），既可以扩展键盘等输入设备，也可以扩展 LCD、数码管等显示设备作为输出接口。以按键驱动为例，系统中设计了 6 个按键供应用程序来使用，该系统中按键 KEY1-KEY4 分别占用外部中断 EXINT0-EXINT3,对应的 GPIO 管脚是 GPF0-GPF3;KEY5 占用的是外部中断 EXINT11 即 GPG3; KEY6 占用的 EXINT19 即 GPG11。在 S3C2440A 处理器中外部中断 0-3 分别占用中断号 0-3，而外部中断 4-7 共享中断号 4，外部中断号 8-23 共享中断号 5。S3C2440 的 F 端口是一个 8 通道的输入输出端口,其端口控制寄存器 GPFCON 是以每 2 位来设定一个输入输出管脚。在设定的时候，为了不改变其余位原有的状态，通常要用程序读出寄存器的值做保存，然后再修改相应位的值。

该驱动编写过程中需要注意的问题如下：

(1) 利用 make menuconfig 选择驱动程序来配置内核的时候，既可以选择编译成模块又可以选择编译进内核。编译成模块时，以.ko 结尾，编译进内核时，以.o 结尾。如果选择以模块的方式编译，则需要在目标板的 linux 文件系统下输入 insmod s3c2440_buttons.ko 命令来动态加载内核模块。

(2) 如果编译模块时出现" assignment makes integer from pointer without a cast" 错误信息，说明一些指针的类型定义的不对，需要重新对指针进行定义。

(3) 如果出现 "ISO C90 forbids mixed declaration and code" 错误信息，说明是有些子函数的结尾部分没有加括号 "}"， 造成一个子函数里有 2 个函数声明部分。

(4) 如果在调试驱动模块的时候，按键只能用一次，说明按键中断只响应了一次，这是因为按键中断在读取引脚电平值的时候，需要把相应的管脚设置为输入输出模式，读取该管脚的电平状态后，需要重新用 linux 自己带的函数 set_irq_type()设置一下，把 GPIO 管脚仍然设置为中断模式。

6 应用程序设计

6.1 应用程序设计概述

应用程序是 linux 中不可缺少的一部分，应用程序运行在操作系统之上，利用 linux 内核提供的虚拟文件系统接口来控制和操作硬件设备，相对于驱动程序，它更加的灵活和易于移植。应用程序通常都是为了一个明确的目的或功能而设计，它调用标准的文件读写函数来实现对硬件的操作，每个模块或每个方案的主要功能或算法，都是在应用程序中来实现的。应用程序把系统的需求划分成每个小的功能模块，然后再调用驱动程序去处理每个模块所要请求的操作。应用程序的设计需要依赖操作系统提供的应用程序接口 API 和库函数，linux 下的应用程序需依赖的库函数是由交叉编译器提供的，在交叉编译器的 lib 目录下。在 Linux 下一般需要编写 Makefile 文件来管理应用程序的编译，特别是源文件比较多的时候。make 是一个编译源码的管理工具，它通过一条命令就可以完成对源代码的编译，并且可以自动判断哪些源文件做了改动，然后在编译源码的时候可以选择只编译那些改动过的源文件。make 管理源码是通过 Makefile 文件来实现的，Makefile 文件有其自己一定的规则即“目标…：依赖…”，其中目标表示要生成的目标文件，冒号后面的表示要生成此目标文件所需依赖的源码和头文件。另外，设计应用程序的时候还需要依赖一些必要的头文件，例如关于应用程序的头文件应该包含标准输入，标准输出函数的头文件 stdio.h，标准出错处理及其错误定义头文件 errno.h，标准 C 函数库的头文件 stdlib.h，标准文件操作 read、write、open 等函数原型的头文件 unistd.h，与文件打开属性相关的头文件 fcntl.h 等等。

6.2 温度传感器应用程序设计

温度传感器的应用程序设计主要包括采集粮仓内的温度、保存数据和通过网页的形式打印出保存的数据。PC 机通过浏览器，点击目标监测板提供的网页服务器上提供的查询温度按钮，触发 CGI 温度采集程序去调用温度查询应用程序，最后再把查询的结果提交给 CGI 程序打印成网页形式的数据表。传感器的排布和布局应该依照当地气象部门的数据来布置传感器的位置，其监测点平面

布置方案如图 6-1 所示：

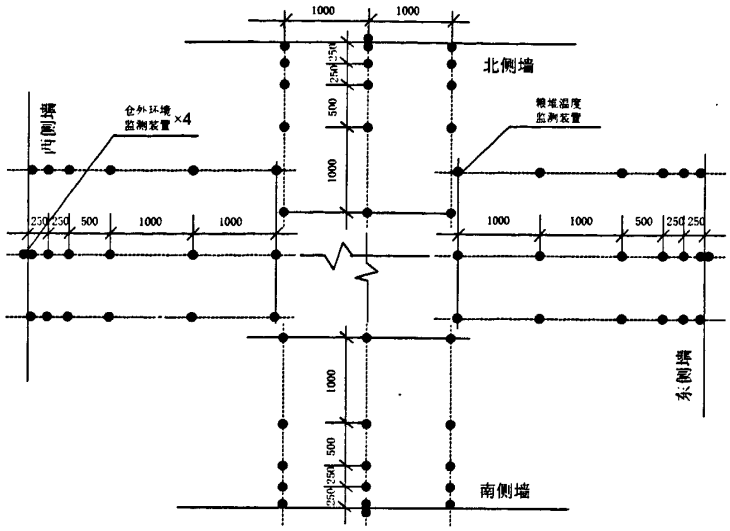


图 6-1 粮堆温度数据采集系统监测点布置平面示意图

其数据监测点立面布置方案如图 6-2 所示：

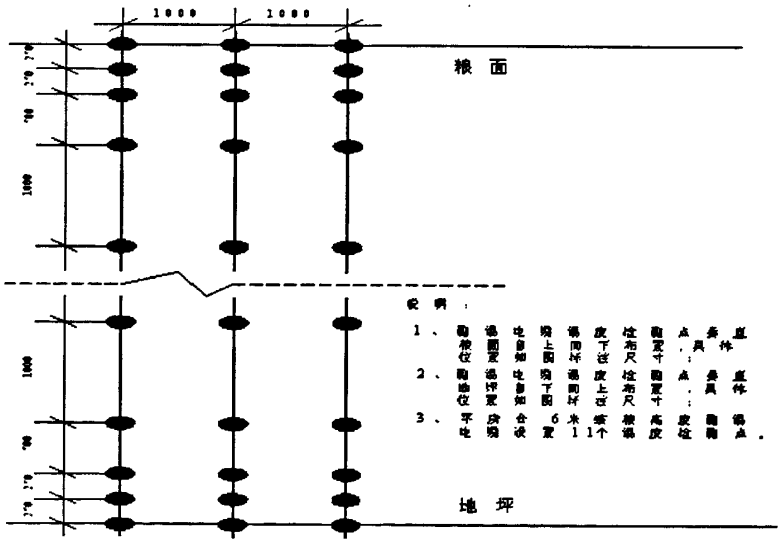


图 6-2 粮堆温度数据信息采集系统监测点布置立面示意图

平面布局的具体方案是：仓外的东、南、西、北部位各设置 1 套环境温度装置，共 4 套测温装置；仓内测温装置设定是东、南、西、北侧墙内，每侧各布置 3 组测温电缆，两条电缆的间距设定为 1 米;东、南、西、北各侧由墙向内

布置的间距应该依次拉大，例如设定分别是0.25m、0.25m、0.5m、1.0m、1.0m ……。

立体面布局的具体方案是：粮堆温度监测点应该同时按照由顶向下和由底向上来布置，按照粮仓内装粮的高度，来设定每根测温电缆需要安装的温度传感器个数。以6米的粮高为例，每根电缆应该安装11只传感器，间距为0.25m、0.25m、0.5m、1.0m、1.0m ……。

对于应用程序主要是打开温度传感器的设备文件然后读取数据，主要的代码如下：

```
#define DS18B20_DEV        "/dev/s3c2410-ds"
int main(int argc, char **argv)
{
     int fd，ret，i;
    char buf[64];
          fd = open(DS18B20_DEV, O_RDWR);
          if (fd < 0) {perror("can't open device /dev/s3c2410-ds");
                        exit(1);}
    ret=read(fd, buf, 64);//调用驱动函数
    fflush(stdout);//刷新缓冲区
    if (ret>0)
         {
         for(i=0;i<64;i++)
         {printf("the %dth temperature is:%d*********\n",i,buf[i]);}
         }
    else
         printf("-------read fail------\n");
          close(fd);
return 0;
}
```

6.3 压力传感器的应用程序设计

AD转换的驱动程序编写好后，同样需要修改Makefile和Kconfig文件，在

Makefile 文件中加入 obj-m :=s3c2440_adc.o，然后进入 linux 内核配置界面 Device driver 选项下 input device support 选项中的 touchscreen 选项移除掉触摸屏驱动，再执行 make uIamge 命令生成内核的映像文件。在 u-boot 下通过网络下载内核到目标板的内存中，启动目标板进入 linux 操作系统，用 insmod s3c2440_adc.ko 命令动态的加载 AD 转换驱动程序的模块到内核中。加载后可以通过查看 proc 文件夹下的 devices 文件，看到 AD 转换设备的设备号，然后根据主设备号可以手工创建设备文件，如：mknod /dev/s3c2440adc c 240 0

AD 转换的应用程序设计顺序是，先打开设备文件看设备是否已经注册，其次设置 AD 转换的转换通道和比例因子，然后再用标准的 read 操作来读取 AD 转换的结果。具体的程序实现如下：

```
#define ADC_DEVICE        "/dev/s3c2440adc"
.....
    for(i=0; i<=8; i++)
    {
     int PRESCALE=0xFF;
    int data=ADC_WRITE(channel, PRESCALE);//设置转换通道和比例因子
    write(fd, &data, sizeof(data));//写入转换通道和比例因子
    read(fd, &data, sizeof(data));//读取转换结果
    d=((float)GetADresult(i)*3.3)/1024.0;
    printf("a%d=%8.4f\t",i,d);
        }
    usleep(1);
    return 0;
}
```

6.4 串口应用程序设计

由于linux下串口的驱动程序一般作为一个类别来单独存放在内核源码目录下driver目录下的serial目录中，它并不是像其他字符设备的驱动程序一样直接把file_operations结构体作为驱动程序的入口点来执行的。在linux下串口设备被看做是一个终端设备，从这个角度来看串口的驱动程序实际上又被封

装了一层，它们与具体的硬件操作的关联是通过uart_ops这个结构体来联系在一起的，uart_ops结构体相当于file_operations结构体，它是串口驱动程序的入口点，在这个结构体中定义的串口的各种操作。uart_ops结构体的类型是结构体tty_operations，这样就又链接到了终端tty的驱动程序。

串口应用程序的设计比较复杂，主要是因为对串口的设置，包括波特率设置、停止位设置和校验位设置是由一系列的结构体来实现的。设置串口主要是设定struct termios结构体中各个成员的值，对这个结构体的设置比较复杂，需要参看详细的在线文档。

设置完了就可以把串口当做普通文件来操作。在通过串口操作以发送命令控制字的方式来控制单片机控制板，也可以用 C 语言编写相应的应用程序。用命令的时候其实就是利用 shell 的输出重定向功能，然后向串口的设备文件发送命令控制字就好。例如可以在 shell 下输入：echo “S”>/dev/ttyS1 命令就可以把参数发送给串口。由于串口的最高传输速率的限制,在串口的读写过程中可能需要加入一些延时，使得设备准备好数据和完成上位机发出的命令操作。

6.5 WEB 服务器及其 CGI 程序设计

在计算机中，软件系统的体系结构一般可以分为 C/S 结构和 B/S 结构，它们各自都有其自身的优点和缺点。该系统软件部分采用程序控制来控制各个模块的运行，当需要查看的时候主要是查看当前检测的报表，因此选用 B/S 结构来实现。当然 B/S 结构的缺点也是显而易见的，就是当有一个客户访问的时候，系统会通过网页来触发一个 CGI 程序，而 CGI 程序就会创建一个进程去处理客户的请求，当有多个用户触发 CGI 程序的时候，就会创建多个进程，这样就会加大处理器的负担，甚至造成服务器的死机，所以该系统必须建立相应的用户并设置权限，防止多个用户同时登陆。采用 linux 操作系统的 PC 机和服务器计算机上一般用 Apache 来构建 web 服务器，而在嵌入式系统中却不能用这种耗费资源很大的服务器程序，而是采用 boa、httpd 和 mini_httpd 等轻量型的 web 服务器。该系统采用 boa 来作为嵌入式 web 服务器。

在 B/S 结构中数据信息的传递通常采用公共网关接口(CGI)技术来实现。CGI 只是提供了一种数据交换的接口的标准，并没有规定具体的编程语言，所以 CGI 程序可以用任何一种语言来实现，选择的时候应该选用自己比较熟悉的一种语

言。数据交互的过程是：远程客户机通过浏览器访问 web 服务器，根据可以查询的功能提交数据查询的请求，各个查询功能的按钮分别链接到不同的 CGI 程序，当点击不同查询功能的按钮的时候就会触发不同功能的 CGI 程序，然后 CGI 程序去调用具体的应用程序来请求硬件设备以获得数据信息，再通过 html 文本的格式打印出所采集到的数据信息，完成一次查询请求。该系统中需要编写三个 CGI 程序，分别是查询温度、查询重量、控制通风。该系统的 CGI 程序采用 shell 脚本语言来编写，然后用 shell 以超文本传输协议的格式输出得到的数据。下面以 AD 转换的 CGI 程序 ad7.cgi 文件为例解释一下怎么以脚本语言来实现 CGI 程序的编写，如下所示：

```
#!/bin/sh
echo "Content-type: text/html"
echo ""
/bin/cat << EOM1
<HTML>
<HEAD>
<TITLE>粮食重量检测 </TITLE>
</HEAD>
<BODY bgcolor="#ccffdd" text="#8080ff">
<HR SIZE=5>
<H1> 当前压力传感器值</H1>
<HR SIZE=5>
<P>
<!--SMALL>
<PRE>
EOM1
./ad>adcstr
cx=`cut -b1-11 adcstr`
echo " 当前测得的压力值如下:"
echo "$cx"
cat << EOM2
</PRE>
```

<!----/SMALL>

<P>

</BODY>

</HTML>

EOM2

在这个 CGI 程序中采集数据的关键是第 17、18 行，它是此脚本程序的核心，第 17 行是直接调用 AD 转换的应用程序，应用程序再去调用 AD 转换的驱动程序，启动转换并等待 AD 转换完毕，应用程序再把得到的结果存起来。第 18 行的意思是读取 AD 转换的结果然后送给变量 cx，得到变量的值后就可以按照 html 文本的格式显示。

6.6 USB 键盘捕获程序设计

在嵌入式系统中一般用 GPIO 端口来扩展按键，扩展的按键一般采用行列扫描的接法来扩展成矩阵键盘。如果系统需要的按键比较多，就可以考虑采用外加键盘芯片的方法。常用的方法是用 8279 通过 I/O 端口来扩展，但是此种芯片连接比较麻烦，同时还需要编写相应的驱动程序，该系统已经集成了 USB 接口，因此可以采用 USB 键盘芯片来扩展键盘。Linux 作为一种操作系统，其内核源码中已经包含了 USB 键盘的驱动，因此可以采用专用键盘芯片来扩展按键。该系统中采用 HT82K629A 芯片来扩展，直接通过 USB 接口与嵌入式目标板相连接，这款键盘芯片支持 PS/2 模式和 USB 模式自动识别，按键的扫描方式也是行列式扫描。在 linux 系统中 USB 键盘是一种输入设备，通过查看 /proc/bus/input/devices 文件可以获得 USB 键盘的设备号和设备文件名称，设备文件一般是 dev 目录下 input 目录中的 event0、event1、event2 等文件。在编写应用程序的时候，可以把其看做是字符设备来直接进行读写，然后再把读到的键值做简单的处理，就可以根据不同的键值去处理不同的子程序。

7 总结与展望

7.1 本文总结

针对现阶段粮情监测系统的现状，该课题从网络化角度提出了一种粮情监测方案，该方案以 S3C2440 为硬件平台结合嵌入式 linux 操作系统搭建了开发环境，在此平台上移植了 u-boot 和 linux 内核，完成了温度监测、压力监测和通风控制的驱动程序及其应用程序的编写，设计了基于网页服务器模式的 CGI 程序，实现了监测系统的基本功能。

该课题所做的主要工作有：

（1） 采用 S3C2440A 为监测系统的硬件平台，设计了简单的外围应用电路。

（2） 移植了 bootloader、linux 内核和 yaffs2 文件系统的驱动。解决了 u-boot 下的网卡驱动程序存在的问题，实现了利用网卡来下载调试的功能。建立了嵌入式 linux 下的文件系统。

（3） 设计了 linux 下的温度传感器、键盘、AD 转换等驱动程序，实现了对这些设备的控制。编写了温度监测、键盘输入、压力采集等应用程序。

（4） 构建了嵌入式 web 服务器，实现了查询粮仓温度、控制通风、监测压力等 CGI 程序的设计。

7.2 展望

由于时间的限制，该粮情监测系统还存在许多的问题，在监测功能和实用性方面还需要做大量的工作：

（1） 应该增加对粮仓中粮食的湿度的监测。

（2） 需要扩展更多通道的 AD 转换接口，实现压力传感器网络数据的采集。

（3） 应完善系统的无线通信功能，以便和无线压力传感器网络进行互联，实现对数据的采集。

参考文献

[1] [韩]俞永昌著. 李红姬，李明吉译. Linux 设备驱动开发技术及应用[M]. 北京：人民邮电出版社，2008.

[2] 王强, 彭继慎等. 基于 ARM 控制器的渗炭炉温度控制系统的设计[J]. 电子技术应用, 2006 年第 6 期.

[3] 赖于树. ARM 微处理器与应用开发[M]. 北京：电子工业出版社, 2007.

[4] 博创 ARM2410S 试验箱指导书. 北京博创科技.

[5] 刘振永, 高恒志. 嵌入式系统在远程监控中的应用[J]. 安防科技，2008 年 6 期.

[6] 周宏霖. CramFS 在 Linux 嵌入式环境的应用——应用 CramFS[J]. Internet：共创软件，2002.

[7] 董红政, 史晓鹏, 王忠勇. IPv6 环境下信息家电系统网络终端设计[J]. 微计算机信息 2007 年第 35 期.

[8] Jonathan Corbet,Alessandro Rubini,Greg Kroah-Hartman 著. 魏永明, 耿岳, 钟书毅译. Linux 设备驱动程序（第三版）[M]. 北京：中国电力出版社，2006.

[9] 张崙. 32 位嵌入式系统硬件设计与调试[M]. 机械工业出版社, 2005.

[10]成 洁，吕遵明，敖 雪. 基于嵌入式 Linux 的嵌入式 GIS 的设计与实现[J]. 电子工程师, 2006. 11.

[11]RedBoot User's Guide. March 2001. RedHat Corp.

[12]任泰明. TCP/IP协议与网络编程[M]. 西安:西安电子科技大学出版社. 2004.

[13][美]Douglas E. Comer 著. 林瑶，蒋慧，杜蔚轩译. 用 TCP/IP 进行网际互联[M]. 北京电子工业出版社. 2006. 1

[14]孙琼. 嵌入式linux应用程序开发详解[M]. 北京：人民邮电出版社，2006. 07.

[15]王宇行. ARM 程序分析与设计[M]. 北京：北京航空航天大学出版社，2008.

[16]华清远见嵌入式培训中心:嵌入式 Linux C 语言应用程序设计[M]. 北京：人民邮电出版社， 2008. 02.

[17]李亚锋著. ARM 嵌入式 Linux 设备驱动实例开发[M]. 北京：中国电力出版社，2008.

[18]李亚锋，欧文盛等编著. ARM 嵌入式 Linux 系统开发从入门到精通[M]. 北京：清华大学出版社，2007.

[19]欧文盛编著. ARM 嵌入式 Linux 应用实例开发[M]. 北京：中国电力出版社，2008.

[20]杨水清，张剑，施云飞等编著. ARM 嵌入式 Linux 系统开发技术详解[M]. 北京：电子工业出版社，2008.

[21]刘凯编著. ARM嵌入式接口技术应用[M]. 北京：清华大学出版社，2009.

[22]白玉霞. 基于嵌入式 Linux 的多媒体信息终端技术的研究与应用[D]. [硕士学位论文]. 西安电子科技大学硕士学位论文，2006. 01.

[23]杜春雷编著. ARM 体系结构与编程[M]. 北京：清华大学出版社，2003.

[24]S3C2440A -32-Bit COMS RISC Microprocessor.User's Manual, Revision 1 (July 2004)

Samsung Electronics.

[25]DM9000A-DS-F03 Datasheet. Davicom Semiconductor Inc. 2004.

[26]K9F2G08U0M NAND FLASH User's Manual Revision 0.6 (2004.3) Samsung Electronics.

[27]韩山，郭云，付海艳编著. ARM 微处理器应用开发技术详解与实例分析[M]. 北京：清华大学出版社，2007.

[28]孙红波等编著. ARM 与嵌入式技术[M]. 北京：电子工业出版社，2006.

[29]网络文档：移植 U-Boot 到博创试验箱上 2410s，网络详细链接 http://blog.chinaunix.net/u1/34474/showart.php?id=410294.

[30]陈文智等著. 嵌入式系统开发原理与实践[M]. 北京：清华大学出版社，2005.

[31]杨永志，唐玉华著. 高度可移植嵌入式系统设备驱动体系结构[M]. 计算机工程，2006.7.

[32]程路，郑毅，向先波著. protel 99SE 多层电路板设计与制作[M]. 北京：人民邮局出版社，2007.4.

[33]袁丽慧，彭磊. 可重用 Linux 设备驱动程序框架[J]. 计算机工程，2008.05.

[34]徐爱钧，彭秀华著. 单片机高级语言编程与 uVision2 应用实践[M]. 北京：电子工业出版社，2008.5.

[35]吴炳胜等编著. 8051 单片机原理与应用技术[M]. 北京：冶金工业出版社，2003.9.

[36]童诗白，华成英著. 模拟电子技术基础第三版[M]. 北京：高等教育出版社，2003.4.

[37]阎石著. 数字电子技术基础第四版[M]. 北京：高等教育出版社，2004.4.

[38]Karim Yaghmour, Jon Masters, Gilad Ben-Yossef, Philippe Gerum. building embedded linux systems. O'Reilly Media 2003.

[39]蒋句平著. 嵌入式可配置实时操作系统 eCos 开发与应用第 2 版[M]. 北京：机械工业出版社，2008.9.

[40]薛闵闵著. USB 应用开发技术大全[M]. 北京：人民邮局出版社，2007.8.

[41]谭浩强. C 程序设计第二版[M]. 北京：清华大学出版社，2000.2.

[42][美]Ellie Quigley 著. 吴雨浓译. Linux Shell 实例精解[M]. 北京：中国电力出版社，2003.02.

[43]马忠梅编著. ARM & Linux 嵌入式系统教程[M]. 北京：北京航空航天大学出版社，2008.

[44][美]W. Richard Steven 著. 尤晋元等译. UNIX 环境高级编程[M]. 北京：机械工业出版社，2005.8.

[45]韦东山著. 嵌入式 Linux 应用开发完全手册[M]. 北京：人们邮电出版社，2008.07.

[46]范书瑞，赵燕飞，高铁成编著. ARM 处理器与 C 语言开发应用[M]. 北京：北京航空航天大学出版社，2008.9.

[47]田军营，韩建海，马志荣著. uclinux 源代码中 Make 文件完全解析—基于 ARM 开发平台[M]. 北京：机械工业出版社，2005.7.

[48]李佳编著. ARM 系列处理器应用技术完全手册[M]. 北京：人民邮电出版社，2006.12.

[49]潘巨龙，黄宁，姚伏天等著. 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，2006.08.

[50] 刘焱著. 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，2006.5.

致 谢

三年紧张而又充实的研究生生活即将结束，在即将毕业之际，回首往事历历在目，仿佛就像是昨天发生的一样，在此我要向所有指导、帮助过我的老师和同学们表示最诚挚的谢意。

由衷感谢我的导师张元教授，张老师敏锐的思维、严谨的治学态度、渊博的学识、果断干练的作风、谦虚的品格，永远值得我学习。在我的学业上尤其是选择课题的过程中，张老师给予了细心的指导和教诲，使我终生受益。对张老师的感激之情是无法用语言来表达的，我只有在人生的路上努力进取，以不辜负恩师的期望。同时，在此一并感谢河南工业大学廉飞宇老师和付麦霞老师在我完成论文的过程中所给予的细心指导！

在专业课程学习和实验期间，还得到了创新实验室程明教授的热心帮助和指导，感谢程老师给我提供了实验的空间和锻炼的机会，在此对程老师表示深深的感谢！ 在程老师及其学生的帮助和努力下，创新实验室养成了良好的学术交流氛围，对我完成课题产生了很大的推动作用。在此向他们表示衷心的感谢。

感谢我的同学刘启军、肖祖胜、孟海成、张锦朋、李伟、余世朋、李建华，在他们的帮助下，我克服了软硬件调试过程中的种种困难。同时感谢给予我关心和帮助的明奇、张永奇、王俊山、陈京育等同学。

特别感谢我的父母和姐姐，父母含辛茹苦，对我却从来没有任何要求，只有鼓励和无私的奉献。在我求学的这些年里，父母始终是我坚强的后盾和精神上的支柱。养育之恩，无以为报，惟祝愿父母健康长寿。此外我的两位姐姐在物质上给予了我很多的资助，在此一并表示感谢，希望她们工作顺利！

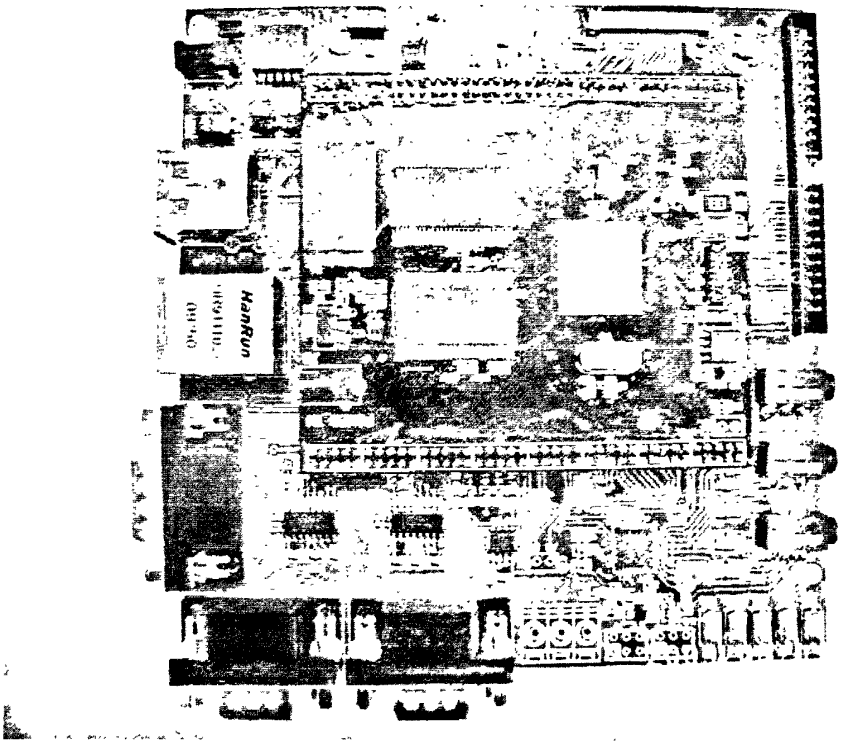
个人简历 在学期间发表的学术论文与研究成果

攻读学位期间发表论文：

[1]王朋涛，张元，付麦霞. 粮库储粮空洞检测过程中证据冲突研究.河南工业大学学报，2010年 第1期.

[2]王朋涛，张元. 基于ARM9的粮情监测子系统设计.计算机数字与工程，2010年 第2期.

附录A 系统实物图



附录B 图表清单