浙江大学

硕士学位论文

嵌入式web服务器的实时性研究

姓名：郑一强

申请学位级别：硕士

专业：检测技术与自动化装置

指导教师：冯冬芹;金建祥

20070601

摘要

随着 web 技术的发展，嵌入式 web 技术在工业远程监控和过程控制系统得到广泛的应用。由于工业嵌入式系统是实时系统，要求 EWS（Embedded Web Server，嵌入式 web 服务器）也具备实时性。在单片机上设计的 EWS，通常不使用嵌入式实时操作系统，采用的是完全使用服务器硬件资源运行的 CGI（Common Gateway Interface，公共网关接口）应用程序接口模块实现动态数据交互，只能实现一些简单界面的远程网络监控。但随着智能仪表监控和组态功能的丰富，EWS 还需要在绘制动态曲线图形等复杂图形以及传输庞大数据的网页，这类设计方案难以达到工业仪表对实时性的要求。

针对上述问题，本文深刻分析影响实时性的各种因素，以 C3000 控制器为研究对象，提出了实时 EWS 设计方案：在 32 位高性价比微处理器 AT9140008 硬件开发平台上，采用了 μC/OS-II 实时操作系统和 LwIP 协议栈设计了嵌入式 web 服务器，使用多任务编程来提高系统的实时性。客户端程序利用 ActiveX 技术，使用 B/S（Brower/Server）和 C/S（Client/Server）混合监控模式，针对 C3000 的特性设计远程监控界面。控件可下载到客户机上，利用客户端资源绘制图形并动态刷新界面。在工业局域网内，可以通过 EPA 现场总线技术解决工业以太网网络不确定性传输问题。

工业仪表的实时响应速度要求为 4～10ms，C3000 最小的数据发送间隔为 0.125s；通过系统测试，ping 包的响应时间为 0.4ms、 HTTP 建立连接的时间为 4ms、数据包发送时间间隔为 0.025s；客户端的动态网页刷新速度也保持和仪表界面的速度几乎一致。通过上述实验结果，可以证实本设计完全满足工业仪表对实时性要求。

在本文中，首先研究影响 EWS 的实时性的三个关键因素：硬件、软件和网络通讯。接着本文描述通过软件设计提高系统的实时性。在服务器端，介绍了 μC/OS-II 实时操作系统和精简 TCP/IP 协议栈 LwIP 的移植，以及 HTTP 服务器和 Flash 文件系统的设计；为了提高数据交互速度，本文设计了实时性很高的快速数据交互任务。在客户端，本文采用面向构件思想设计一个监控界面，并尽量模拟真实仪表的功能和界面，设计了存储模块、网络通讯模块、实时监控模块、历史监控模块等。最后，本文对课题的工作进行了总结和展望。

【关键词】 EWS，μC/OS-II 操作系统，LwIP 协议栈，ActiveX 控件，C3000 控制器

Abstract

The technology of EWS (Embedded Web Server) applies widely in the industrial monitoring system and process control system. Industrial embedded System is real-time system, so the EWS must be the real-time system too. EWS, which designed in the single chip, doesn't have embedded real-time operating system and use the CGI (Common Gateway Interface) to realize the simple monitoring by network, so it can only realize simple internet monitoring. While the intellectual instrument's capacity of monitoring and configuration, the designing method of using the server's microprocessor source to draw and communicate the huge digit is not satisfied the real-time requirement of industrial instrument.

To this question, the paper researches the critical factors of real-time performance and brings forward the designing scheme in the platform of 32 bit microprocessor AT9140008 according to the C3000 controller. Server uses the μC/OS-II real-time operating system and LwIP protocol stack and adopts the multithreading method to improve the real-time. According to void-paper peculiarity, we utilize the ActiveX technique with the B/S (Browser/Server) and C/S (Client/Server) modes design the client interface. The software downloads to the client computer and draws and refreshes the interface by the client source instead of the server source. In the industrial Ethernet, the server realizes the certain Ethernet transfer by the EPA protocol which can improve the real-time property.

The real-time responding time of industrial system is less than 4~10ms, and the interval time between of two packs will sending of the C3000 is 0.125s. In the result of the software testing, the responding time of the ping is 0.4ms, and the responding time of HTTP is 4ms, the interval time between of two packs will sending is 0.025s, and the refurbishing speed of the web interface is the same as instrument's. From the results above, they can improve that this designing method can satisfy the real-time and stability of industrial instrumental internet monitoring system.

First, the paper studies three factors influencing the embedded web server real-time property. They are hardware, software and networks. According to the factors, the paper makes sure the scheme, and introduces the designing tools. Then, paper depicts the replant of μC/OS-II real-time operating system and lean TCP/IP protocol stack LwIP, HTTP server with file systems, digit communication and Ethernet driver, and test the performance of server. And then, according to peculiarity of void-paper, paper design the monitoring interface, which is composed of memory module, network communication module, real-time monitoring module, history monitoring module and so. At the end, paper summarizes the work and makes a prospect.

【Key Words】: Embedded Web Server, μC/OS-II real-time operating system, LwIP protocol stack, ActiveX, c3000 controller

第1章 绪论

1.1 课题研究背景

嵌入式系统是计算机技术、自动控制技术、现代网络与通信技术等高度融合的产物。根据 IEEE（国际电气和电子工程师协会）的定义：嵌入式系统是用于“控制、监视或者辅助操作的机器和设备的装置”，此定义是从应用上考虑的，嵌入式系统是硬件和软件的结合体，还可以涵盖机电等附属装置。而我们一般定义为：“以应用为中心，以计算机技术为基础，软件硬件可裁剪，功能、可靠性、成本、体积、功耗严格要求的专用计算机系统”。

嵌入式系统诞生于 20 世纪 70 年代[1]。那时出现以微处理器为核心的微型机。计算机控制专业人士将微型 机嵌入到一个对象体系中，实现对象体系的智能化控制。为了区别于原有的通用计算机系统，把嵌入到对象体系中，实现对象体系智能化控制的计算机，称作嵌入式计算机系统，这便是嵌入式系统。

嵌入式系统经历了 30 多年的发展，尤其是近几年来，计算机、通信、消费电子的一体化趋势日益明显，嵌入式技术已成为一个研究热点，嵌入式系统也给厂家带来良好商机。纵观嵌入式技术的发展过程，大致经历一下四个阶段。

1). 第一阶段

以单芯片为核心的可编程控制形成的嵌入式系统，具有与监测、伺服、指示设备相配合的功能。这类系统大部分应用于一些专用的工业控制系统中，一般没有操作系统的支持，通过汇编语言编程对控制系统进行直接控制。这一阶段的嵌入式小系统的主要特点是：系统结构和功能相对单一，处理效率较低，存储容量较小，几乎没有用户接口。由于嵌入式系统使用简单、价格低，早期在国内工业领域应用较为普遍，但是已经远不能适应高效的、需要大容量存储的现代工业控制和新兴信息家电等领域的需求。

2). 第二阶段

以嵌入式 CPU 和简单操作系统为核心的嵌入式系统。主要特点是：CPU 种类繁多，通用性比较弱；系统开销小，效率高；操作系统达到了一定的兼容性和扩展性；应用软件较为专业化，但用户界面不够友好。

3). 第三阶段

以嵌入式操作系统为标志的嵌入式系统。主要特点是：嵌入式操作系统能运行与各种不同类型的微处理器上，兼容性好；操作系统内核小，效率高，并且具

有高度的模块化和扩展性；具备文件和目录管理、多任务、网络支持、图形窗口以及用户界面等功能；具有大量的应用程序接口，应用程序开发变得更微简单；嵌入式应用软件界面更加丰富。

4). 第四阶段

以因特网为标志的嵌入式系统。这是一个正在迅速发展的阶段。目前大多数嵌入式系统还孤立于因特网外，但随着英特网的发展以及因特网技术与信息家电，工业控制技术结合日益密切，嵌入式设备与因特网的结合将代表嵌入式系统的未来。

嵌入式系统技术日益完善，32 位处理器在该系统中占主导地位，嵌入式操作系统已经从简单走向成熟，并随着 64 位处理器的出现以及应用，嵌入式系统的资源更加丰富，提供更强大的处理功能。嵌入式系统由原先的单一，非实时的控制系统发展成多元的实时控制系统，并亟待扩展网络功能，实现远程监控。

嵌入式 Internet 正成为当前国内外 IT 业发展的热点领域，也是工业控制网络发展的重要方向[2]。嵌入式系统已经在工业测控、环境监测、家居电器等各个领域得到了广泛应用，如果将这些分布在各处的起着各种作用的嵌入式系统接入 Internet 网，那么就可实现基于 Internet 的远程监控。

嵌入式 Internet 技术主要包括传感器技术、通信技术、计算机技术和集成电路技术等[3]。以前的设备控制系统一般是通过专用通信线中进行的，其通信介质、通信协议、相关软件和硬件都是专用的，而 Internet 技术的发展使嵌入式设备的远程控制和管理方式有了改变。不需要专用的通信线路，并且传输的信息不局局于数据信号，还有声音和图像。最重要的是其通信协议是标准而且公开的。随着 Web 技术的发展，几乎改变了现在的信息表达形式，很多应用都是基于 Web 技术的。由于 HTML 语言的标准统一性，只要在嵌入式设备中有一个微型服务器，就可以使用任意一种 Web 浏览器接收和发送信息。工业现场有极高的要求，设备必须要具备极高的实时性、稳定性和可靠性等。所以如何设计这种特别的 Web 服务器，如何在嵌入式设备中安装 Web 服务器，如何提高服务器的性能，就成了 EWS（Embedded Web Server，嵌入式 Web 服务器）的发展和研究方向。

目前，许多公司都在致力于嵌入式 Internet 技术的开发，已提出了多种嵌入式系统与 Internet 互联的解决方案。国外的情况，最具代表性的是 emWare 公司的 EMIT 技术（Embedded Micro Internetworking Technology）。 EMIT 由 emNet 和 emGateway 两部分组成，emNet 协议运行在 MCU 内部，是为嵌入式系统和其他网络（如 RS485、IR、RF 和电力线等）进行联接的网络协议。应用系统运行 MCU 内的 emNet，通过 emGateway 与 Internet 联接。emGateway 通过 RS232、RS485、CAN、红外、射频等总线将多个嵌入式设备联系起来，每个嵌入式设备

的应用程序中包含一个独立的通信任务，称为 emMicro，监测嵌入式设备中预先定义的各个变量，并将结果反馈到 emGateway 中；同时 emMicro 还可以解释 emGate－way 的命令，修改设备中的变量，或进行某种控制。用于 Web 的 emMicro 为适应嵌入式环境，仅保留 HTML 标记（emTag），大小简化到 1kB，这使它可在放在任何一个嵌入式系统中。除此之外，Access Systems America 公司为富士通微电子公司的 Web 浏览器 NetFront，用在富士通 32 位 RISC 结构的 MB91101 单片机上，占内存仅有 220kB。PharLap 公司的 Micro Web Server，占内存不足 300 kB，可以放进中等存储规模的单片机上。Agent System 公司的 EmWeb 服务器，它用一个硬件 ASIC 的状态机，支持对 HTML 的多用户、多任务的同时访问，以一个模块的形式向智能家电、信息家电以及工业控制提供应用。国内比较著名的有：沈阳东大新业信息技术股份有限公司研制开发的 Webit 嵌入式系统，它将 MCU 和以太网控制器集成到一块小板卡上，将它装入到嵌入系统中就可以完成嵌入系统与 Internet 网的联接。武汉力源公司开发的 Webchip，它是独立于微控制器的专用网络接口芯片，它通过标准的输入、输出口与各种 MCU 相连。MCU 通过 Webchip 与网关联接即可接收并执行经由 Internet 远程传来的命令或将数据交给 Webchip 发送出去。这类商用嵌入式 web 服务器各有优点，对于我们研究嵌入式 web 服务器的实时性和设计嵌入式实时 web 服务器有指导意义。

1.2 EPA 分布式网络控制系统简介及其关键技术

《EPA（Ethernet for Plant Automation）标准》（全称《用于工业测量与控制系统的 EPA 系统结构与通信标准》）是在国家"863"计划支持下，由浙江大学、浙江中控技术股份有限公司共同主持，联合中国科学院沈阳自动化所、清华大学、大连理工大学、重庆邮电学院、上海工业自动化仪表研究所、北京华控技术有限责任公司、机械工业仪器仪表综合技术经济研究所等国内部分高校、科研院所、高新技术企业，在解决了以太网用于工业现场设备间通信的确定性通信调度、总线供电、网络安全、可互操作等关键技术的基础上，起草的我国第一个拥有自主知识产权的现场总线国家标准[4]。

与此同时，该标准被国际电工委员会 IEC 作为 PAS 标准（Pulblic Available Specification）予以发布，并被 IEC 接收为正在制定的国际实时以太网标准 IEC61784-2 中的实时以太网类型 14（Common Profile Family 14, CPF 14），成为我国第一个被国际认可和接收的工业自动化领域的标准。

EPA 的关键技术和特点：

1). 确定性通信。EPA 解决了以太网由于采用 CSMA/CD（载波侦听多路访

问/冲突检测）介质访问控制机制而产生的网络不确定性问题。

2). “E”网到底。通过采用EPA网络，可以实现工业企业综合自动化智能工厂系统中从底层的现场设备层到上层的控制层、管理层的通信网络平台基于以太网技术的统一，即所谓的“E（Ethernet）网到底”。

3). 互可操作。几乎所有的控制系统都采用了以太网、TCP/IP协议作为其通信网络，实现了设备的互连。但是，如果仅采用以太网、TCP/IP协议，而没有统一的高层协议（如应用层协议），不同设备之间还不能相互理解、识别彼此所传送的信息含义，就不能实现信息互通，也就不可能实现开放系统之间的互可操作。

为此，EPA除了解决实时通信问题外，还为用户层应用程序定义了应用层服务与协议规范，包括系统管理服务、域上/下载服务、变量访问服务、事件管理服务等。至于ISO/OSI通信模型中的会话层、表示层等中间层次，为降低设备的通信处理负荷，可以省略，而在应用层直接定义与TCP/IP协议的接口。

为支持来自不同厂商的EPA设备之间的互可操作，EPA采用XML（Extensible Markup Language）扩展标记语言为EPA设备描述语言，规定了设备资源、功能块及其参数接口的描述方法。用户可采用Microsoft 提供的通用DOM技术对EPA设备描述文件进行解释，而无需专用的设备描述文件编译和解释工具。

4). 开放性。EPA 完全兼容 IEEE802.3、IEEE802.1P&Q、IEEE802.1D、IEEE802.11、IEEE802.15以及UDP（TCP）/IP等协议，采用UDP协议传输EPA协议报文，以减少协议处理时间，提高报文传输的实时性。为确保EPA系统运行的可靠性，EPA 中还针对工业现场应用环境，增加了媒体接口选择规范与线缆安装导则。商用通信线缆（如五类双绞线、同轴线缆、光纤等）均可应用于EPA 系统中，但必须满足工业现场应用环境的可靠性要求，如使用屏蔽双绞线代替非屏蔽双绞线。EPA网络支持其他以太网/无线局域网/蓝牙上的其他协议（如FTP、HTTP、SOAP，以及MODBUS、ProfiNet、Ethernet/IP协议）报文的并行传输。这样，IT领域的一切适用技术、资源和优势均可以在EPA系统中得以继承。

5). 分层的安全策略。对于采用以太网等技术所带来的网络安全问题，EPA规定了从企业信息管理层、过程监控层和现场设备层三个层次，采用分层化的网络安全管理措施。EPA 现场设备采用特定的网络安全管理功能块，对其接收到的任何报文进行访问权限、访问密码等的检测，使只有合法的报文才能得到处理，其他非法报文将直接予以丢弃，避免了非法报文的干扰。在过程监控层，采用EPA网络对不同网段进行逻辑隔离，以防止非法报文流量干扰EPA网络的正常通信，占用网络带宽资源。对于来自于互联网上的远程访问，则采用EPA代理

服务器以及各种可用的信息网络安全管理措施，以防止远程非法访问。

6). 冗余技术。EPA 支持网络冗余、链路冗余和设备冗余，并规定了相应的故障检测和故障恢复措施，如设备冗余信息的发布、冗余状态的管理、备份的自动切换等。

1.3 本论文研究的对象

C3000 是浙江中控集团仪表公司生产的一款采用 32 位微处理器和 5.6 英寸 TFT 彩色液晶显示屏的可编程多回路控制器，又叫做无纸记录仪。C3000 过程控制器主要有控制记录和分析等功能。可通过串口、以太网和 CF 卡实现与上位机的数据交换。内部有三个控制模块、4 个单回路 PID 控制模块、6 个 ON/OFF 控制模块，可实现串级、分程、三冲量、比值控制及用户指定等多种复杂的控制方案。

在 C3000 仪表中，包括旋钮、5 个自定义功能键和 4 个非自定义功能键，用来操作仪表的所有功能。其中选钮有左旋、右旋、单击、长按四种操作方式；菜单键可以在任意监控界面，单击此键，即进入主菜单画面；自定义功能键：5 个自定义功能键根据各个画面底部的提示实现相应的功能。

C3000 过程控制器的操作用户按权限分为四个等级：操作员 1、操作员 2、工程师 1、工程师 2。具体组态时可以根据上述几种方式登陆，其中工程师 2 拥有最高的权限，且进入组态界面需要通过密码验证。

C3000 过程控制器有 11 幅基本的实时监控画面，依次为总貌、数显、棒图、实时、历史、信息、累积、控制、调整、程序和 ON/OFF 画面。

1.4 本论文研究的意义和研究目标

嵌入式 web 服务器的实时性研究是 EPA 科研项目的子项目，其研究的环境是 EPA 系统应用的 10M 以太网网络。在 EPA 系统下，本文重点研究嵌入式 web 服务器的实时性和设计嵌入式实时 web 服务器以扩展工业现场设备的网络功能。

本文全面研究影响嵌入式 web 服务器实时性的各个方面，对涉及实时性的关键技术进行深入研究。在已有的硬件平台上，主要是通过软件设计提高系统的实时性。本文采取 μC/OS-II 实时操作系统和 LwIP 协议栈设计嵌入式通讯环境，合理划分任务，扩展 C3000 网络功能，实现服务器快速地响应客户端的请求和实时回传动态数据。针对 C3000 过程控制器的界面，本文采取 ActiveX 控件设计虚拟仪表界面，使其下载到客户端并运行，最终实现同步地显示仪表界面变化；

而且，为使用户的操作便利，界面设计尽力做到虚拟仪表界面与真实仪表界面基本上保持一致。

1.5 本论文任务和结构

本论文是在 ARM7 硬件平台上，研究嵌入式 web 服务器的实时性以及具体的软件设计和开发。本文采用 μ C/OS-II 操作系统和 LwIP 协议栈设计实时 web 服务器端；并采用 ActiveX 技术，根据实际需求设计实时客户端及采用面向构件的设计客户端界面。

本文共分五章，各部分内容安排如下：

第一章：介绍了论文的研究背景和现场总线技术 EPA、研究对象，并提出了本文的研究内容和意义。

第二章：本章首先介绍实时系统的概念和嵌入式实时 web 服务器的设计思路。根据控制系统的等级划分，确定本系统的实时性级别要求为级别 2，并将其作为整个文章研究实时性的前提条件。接着，本文对影响嵌入式 web 服务器的实时性因素展开具体的研究，分别硬件开发环境，软件以及网络通讯机制。硬件环境是基础，其主要的影响关键是微处理器和 Flash。软件设计是系统性能的关键，其中核心的部分是实时操作系统和协议栈以及 web 文档实现技术。工业以太网，由于其通信机制，也会影响实时性，因此提出了采用 EPA 通信调度实现确定性通信。最后，本文介绍 32 位 ARM 硬件开发平台、开发方式及开发工具。

第三章：本章首先介绍了嵌入式系统的软件结构。接着，对本系统所采用的 μC/OS-II 实时操作系统和精简的 TCP/IP 协议栈的移植工作进行了详细的描述。再则，本文在应用层开发中，根据任务优先级合理设计了四个任务：HTTP 服务器、文件系统的、数据库交互任务和网络数据发送任务。并使用数据库交互任务和网络数据发送任务构建一个实时的数据交互任务。然后，本章介绍以太网的驱动设计。最后，服务器端程序的测试结果证明服务器端的设计可以达到工业仪表实时性标准要求。

第四章：本章介绍嵌入式 web 服务器的实时客户端的实现。首先，介绍 ActiveX 技术。接着，本章提出了一种采用 ActiveX 设计 B/S 和 C/S 混合模式的实时 web 客户端。然后，本文在使用面向构件的设计理念上，根据仪表的特点，设计了一个虚拟仪表界面，主要包括实时监控界面模块、历史监控界面模块、网络数据通讯模块、存储模块等。最后，结合服务端程序对客户端进行测试，测试结果，证明设计方案的良好性。

第五章：本章是对课题的工作总结和对课题的未来展望。

第2章 嵌入式web服务器的实时性分析

嵌入式系统具备产品特征，它对系统的功耗、体积、成本、可靠性、速度、处理能力、电磁兼容等方面均有一定的要求。设计嵌入式 web 服务器，实时性是远程监控的最基本要求且关键要求。本章对影响系统的实时性因素和技术展开深入的分析。

2.1 嵌入式实时系统

2.1.1 实时系统概念

实时，表示“立即”、“及时”。实时性就是指能够在限定时间内执行完规定的功能并对外部的异步事件做出响应的能力[5]。实时系统是对外界事件在限定时间内能做出反映的系统。实时有确定性、可预测性、及时性、时限要求，不同于快速性。

在实时系统中，主要有三个指标来衡量系统的实时性：响应时间、生存时间、吞吐量。

响应时间：识别一个外部事件到做出响应的时间，在控制应用中它是最重要的指标，如果事件不能及时的处理，系统可能就会出现崩溃。对于不同的过程，有不同的响应时间要求。对于有些变化比较慢的过程，具有几分钟甚至更长时间的响应时间都认为是实时的。对于快速过程，其响应时间可能要求达到毫秒、微妙甚至更短。因此，不能单纯从绝对的响应时间长短上来衡量，应当根据不同的对象，在相对意义上进行评价。

生存时间：数据有效等待时间内，在这段时间数据是有效的。

吞吐量：是在给定时间内，系统可以处理的事件总数。例如通过控制器用每秒处理的字符来表示吞吐量，吞吐量可能是平均响应时间的倒数。

实时性分为 2 种类型：硬实时（Hard Real-time）和软实时（soft real-time）[6]。硬实时是指实时约束高度严格，即使出现超时都是不可接受的，因为可能导致灾难性的系统失效。硬实时的例子包括交通控制系统，医疗监控系统等。软实时的时间约束不是非常苛刻，对于时间约束的满足尽量做到就可以了。通常情况下，嵌入式实时系统是这两种的结合。

实时系统强调的是实时性和可靠性，这两方面除了和嵌入式硬件有关外，还与实时系统的软件密切相关。但在设计时，结合硬件价格来考虑硬件，不能一味

追求高性能硬件而增加成本[7]。

2.1.2 实时级别

对于工业控制系统来说，可按照不同过程对实时性的不同要求，将实时性能划分为 4 个级别，如图 2-1 所示。其中实时级别 4 是工控对实时性能要求最苛刻的，主要是机械传动和运动控制等，如对于高性能的同步运动控制应用，特别是在 100 个节点下的伺服运动控制应用场合，实时响应时间要求低于 1ms，同步传送和抖动小于 1us。过程自动化应用场合实时响应时间要求是 100ms 或更长。绝大多数的工厂自动化应用场合实时响应时间的要求最少为 5～10ms[8]，而本文的研究对象仪表，通信协议 R-BUS 协议（在 3.5.1 节具体介绍）设定的最小数据发送间隔为 0.125s。因此，我们知道扩展仪表网络时考虑的嵌入式 web 服务器的实时性，只要达到第二级级别即可满足。

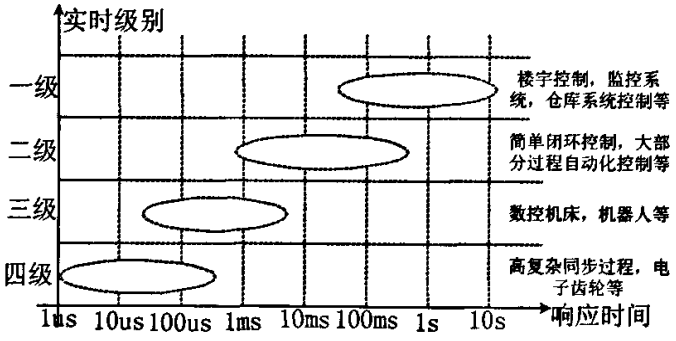

图 2-1 四种不同实时级别划分

2.2 嵌入式实时 web 服务器的设计思想

嵌入式实时 Web 服务器有其自身的特点，在进行设计时要注意以下几条思想原则：

1). 在嵌入式系统中，由于处理器能力和存储器容量等外部条件的制约，EWS 不能对系统资源占有量很大的程序提供支持，也不需要处理外部很大流量的访问，但要能够为远程访问、管理和监控设备提供服务支持，且嵌入式的 web 服务器要做到尽可能的小；

2). 为达到远程和监控设备的目的，必须使客户机能同本地系统进行交互，这种交互可使用很多种方法来实现；

3). 嵌入式设备大多没有硬盘，通常采用空间不大的 Flash，为实现基于文件的 HTTP，嵌入式的 Web 服务器必须用辅助的软件来实现文件的映射问题；

4). 嵌入式实时 Web 服务器服务的对象是实时性要求比较高的系统，必须解决 web 应用的实时性问题，这个问题也是本文的研究重点。

基于以上各种考虑，EWS 在嵌入式实时系统的制约条件下，必须满足管理和监控嵌入式系统的需要。

2.3 影响嵌入式 web 服务器的实时性的硬件因素

通常嵌入式系统硬件包括微控制器、存储器及外设器件和专门用于嵌入式产品的 I/O 端口等。其中核心是微控制器。

在硬件上，微处理器是影响设计实时系统的关键因素。影响微处理器的重要因素如主频率、处理器字节数、指令集、位数等[9]。

Flash 也是影响系统实时性的另一个关键因素，首先必须考虑到硬件的价格和所存放的代码大小，在 32 位微处理器上可以扩展 2M 存储空间的 Flash；其次是考虑到芯片读取速度和 Flash 的片内或偏外运行方式。

网卡芯片用于收发网络数据，因此它对网络信号的处理快慢，也影响着系统的实时性。

2.3.1 单片机设计嵌入式 web 服务器的局限性

在 8 位或则 16 位单片机上设计的 web 服务器，采用精简的 CGI 接口模块完成一些非常简单的远程监控。8 位或则 16 位单片机，其硬件资源非常有限。其内部的 RAM 一般都只有 1～2K，即便外部扩展也就几十 K；内部的 ROM 大的也只有 10K，扩展也只有几十 K。在这样的硬件平台上几乎无操作系统，无法使用硬件资源实现如内存管理、文件管理、进程管理等功能。系统采取的前后台模式，因此所花费的时间比较长。在这样的硬件上，只能设计单线程程序，无法设计实时性 web 服务器。

2.3.2 高性能微处理器设计嵌入式 web 服务器的优越性

设计一个高性能系统，必须要具备一个高性能微处理器。相比较单片机而言，微处理器具备充足的硬件资源，可以扩展大容量 Flash。本文采取采用 RISC 架构的 ARM7 微处理器，下面罗列出它的特点。

1). 体积小、低功耗、低成本、高性能；

2). 支持 Thumb（16 位）/ARM（32 位）双指令，能很好的兼容 8 位/16 位器件；

3). 大量使用寄存器，指令执行速度更快

4). 大多数数据操作在寄存器内完成；

5). 寻址方式灵活，执行效率高。

在 ARM7 等高性能微处理器上设计嵌入式 web 服务器，最大的优点是我们有充足的内存资源和良好的中断机制引入实时操作系统，来设计多任务系统和完成复杂的任务，如图形界面的绘制、各种运算以及网络数据的收发等。其次，ARM7 有丰富丰富 I/O 口，使得我们可以很容易扩展大容量的 Flash，为程序设计带来足够的存储空间。本系统所采用的硬件基础，在 2.6 节会详尽描述。值得注意的是，在设计时不可盲目一味追求高性能，还必须要考虑价格。

2.4 影响嵌入式web服务器的实时性的软件因素

合理的软件架构和编程技术影响系统的实时性。将这些因素细化为如下几点：嵌入式实时操作系统（实时系统的基础）、精简的 TCP/IP 协议栈、web 文档技术、代码源的编程技巧及编译器的效率等因素。本节将主要的因素在下面展开具体地描述。

2.4.1 实时操作系统

实时系统的软件是实时应用软件和实时操作系统的结合，其中实时操作系统起着核心作用，由它来管理和协调各项工作，为应用软件提供良好的软件运行环境。可以说，实时操作系统是实时系统的基础。

实时操作系统的采用，可以使应用软件开发人员避开琐碎的硬件管理和操作编程，而把主要精力放在目标应用的算法研究以及应用程序自身的构架上；同时应用实时操作系统提供的各种服务，可以更容易地构建复杂的嵌入式实时应用系统；并且在软件重用性和开放性方面实时操作系统也起到了非常重要的作用[10]。

一个实时操作系统的实时性评价根据以及在如何避免出现影响实时性的关键因素，对于我们在选择实时性操作系统设计嵌入式实时 web 服务器，都必须深入考虑。

2.4.1.1 衡量实时操作系统实时性能的重要指标

1). 任务切换

当多任务内核决定运行另外的任务时，它把正在运行任务的当前状态(即 MCU 寄存器中的全部内容)保存到任务自己的栈区之中。然后把下一个将要运行

的任务的当前状态从该任务的栈中重新装入MCU的寄存器，并开始下一个任务的运行。这个过程就称为任务切换。作任务切换所需要的时间取决于MCU有多少寄存器要入栈。MCU的寄存器越多，额外负荷就越重。

2). 中断响应时间

中断是一种硬件机制，用于通知微处理器有个异步事件发生。微处理器保存现场环境，然后跳转到中断服务子程序处理事件，触发相应的任务，最后退出时让进入就绪态的优先级最高的任务开始运行。对于一个实时系统，涉及到中断延迟、中断响应、中断恢复和中断处理时间。

实时内核最重要的指标就是关中断时间，因为这个指标影响用户系统对实时事件的响应能力。所有实时系统在进入临界区代码段之前都要关中断，保护临界区代码免受多任务或中断服务例程的破坏，执行完临界代码之后再开中断。在实时环境中，关中断的时间应尽量的短，太长可能会引起中断丢失。中断响应时间由下式：

中断响应时间 = 中断延迟 + 现场保存时间 + 进入中断服务函数的时间。

虽然中断服务的处理时间应该尽可能的短，但是对处理时间并没有绝对的限制。在大多数情况下，中断服务子程序应识别中断来源，从请求中断的设备取得数据或状态，并通知真正处理该事件的任务。当然应该考虑到在中断服务中通知一个任务作事件处理是需要一定时间的，如果事件处理需花的时间短于给一个任务发通知的时间，就应该考虑在中断服务子程序中直接处理事件并在中断服务子程序中开中断，以允许优先级更高的中断打入并优先得到服务。

虽然当今的实时操作系统已日臻完善，但仍有一些问题存在并干扰着实时的实现。我们应充分的重视，在设计时通过合理的安排程序减少它们的危害。

2.4.1.2 影响实时操作系统实时性的干扰问题

1). 优先级反转

这是实时系统中出现最多的问题。优先级反转是指一个任务等待比它优先级低的任务释放资源而被阻塞，如果这时有中等优先级的就绪任务，阻塞会进一步恶化。它严重影响了实时任务的完成。

为防止发生优先级反转，一些商业内核(如VxWorks)使用了优先级继承技术，当发生优先级反转时，优先级较低的任务被暂时地提高它的优先级，使得该任务能尽快执行，释放出优先级较高的任务所需要的资源。但它也不能完全避免优先级反转，只能称其减轻了优先级反转的程度，减轻了优先级反转对实时任务完成的影响。

优先权极限是另一种解决方案，系统把每一个临界资源与1个极限优先权相

联系，这个优先权极限等于系统此时最高优先权加1。当这个任务退出临界区后，系统立即把它的优先权恢复正常，从而保证系统不会出现优先权反转的情况。采用这种方案的另一个有利之处，是仅仅通过改变某个临界资源的优先级就可以使多个任务共享这个临界资源。

2). 任务执行时间的抖动

各种实时内核都有将任务延时若干个时钟节拍的功能。优先级的不同、延时请求发生的时间、发出延时请求的任务自身的运行延迟，都会造成被延时任务执行时间不同程度的提前或滞后，称之为任务执行时间的抖动。可能的解决方案有：

a). 增加微处理器的时钟频和时钟节拍的频率；

b). 重新安排任务的优先级；

c). 避免使用浮点运算等。

对于强实时系统，我们必须综合考虑，充分利用各种手段，尽量减少任务执行时间的抖动。

3). 任务划分

当使用可剥夺型实时内核时，长任务由于执行的时间较长，因而更容易被高优先级的任务打断；一旦高优先级的任务进入就绪态，当前任务的CPU使用权就被剥夺，或者说任务被挂起，那个高优先级的任务立刻得到CPU的控制权。这样会出现两个问题：一是长任务可能在一次执行的过程中被频繁打断，长时间得不到一次完整的执行；二是长任务被打断时，可能要保存大量的现场信息，其目的是为了保证在高优先级的任务执行完返回后，长任务能得以继续执行。然而，这样做要占用一定的系统资源，同时保存现场本身也是要占用CPU时间的，因此，实时性也会下降。

解决长任务问题最有效的途径是进行任务分割。所谓任务分割指将影响系统实时性的长任务分割成若干个小任务。这样单个任务的执行时间变短，系统的任务级响应时间变短，实时性提高。但是，过度的分解，将使系统有大量的任务，经常进行任务的切换。任务与任务之间，还需要进行很多同步和互斥控制，将增加大量的系统服务工作，降低系统的速度和有效性，影响系统的实时性。因此，把一个实时应用问题分解未若干任务时，还必须进行各种综合平衡和折衷。

长任务划分存在这样一对矛盾：如果任务太多，必然增加系统任务切换的开销；如果任务太少，系统的并行度会降低，实时性比较差。在任务划分时可以遵循H.Gomma原则[13]：

a). I/O原则：不同的外设执行不同任务；

b). 优先级原则：不同优先级处理不同的任务；

c). 大量运算：归为一个任务；

d). 功能耦合：归为一个任务；

e). 偶然耦合：归为一个任务；

f). 频率组合：对于周期时间，不同任务处理不同的频率。

如果我们在具体分析一个系统的时候发生原则冲突的话，则要为每一个原则针对具体的系统设定“权重”，必要的时候可以通过计算“权重”来最终确定如何去划分任务。

2.4.1.3 μC/OS-II 实时操作系统

μC/OS-II 是由 Jean J.Labrosse 在 1992 年编写的一个嵌入式多任务实时操作系统。最早这个系统叫做μCOS，后来经过近 10 年的应用和修改，在 1999 年 jean J.Labrosse 推出了μC/OS-II，并在 2000 年得到了美国联邦航空管理局对应用于商用飞机的、符合 RTCA DO-178B 标准的认证，证明了μC/OS-II 具有足够的稳定性和安全性[11]。μC/OS-II 是用 c 语言和汇编语言编写的。其中绝大部分代码是 c 语言编写的，只有极少部分与处理器相关的代码是用汇编语言编写的。用户只要做很少的工作就可以把它移植到像本题的 ARM32 位嵌入式处理器上。由于μC/OS-II 的构思巧妙，结构简洁精炼，可读性很强，同时又具备了实时操作系统的全部功能，所以虽然它只是一个内核，但非常适合初次接触嵌入式操作系统的学生，开发人员和爱好者学习，并且通过适当扩展之后，还可应用到世纪操作系统中[7]。μC/OS-II 提供了实时系统所需的基本功能，其包含全部功能的核心部分代码只占用 8.3K 字节，而且由于μC/OS-II 是可裁减的，所以用户系统中实际的代码最少可达 2.7K 字节，可谓短小精悍。这个相对于另一个公开源代码的 Linux 操作系统有着比较大的优势。μC/OS-II 操作系统的最大关中断时间和任务切换时间都是微秒级的。μC/OS-II 不仅可以使用户得到廉价的解决方案，而且由于μC/OS-II 的开放源代码特性，用户还可以针对自己的硬件优化代码，以获得更好的性能。目前μC/OS-II 在工业界得到了越来越广泛的应用。μC/OS-II 体系结构如图 2-2 所示。由于上述的诸多优越性，本课题采取的μC/OS-II 设计嵌入式 web 实时系统。

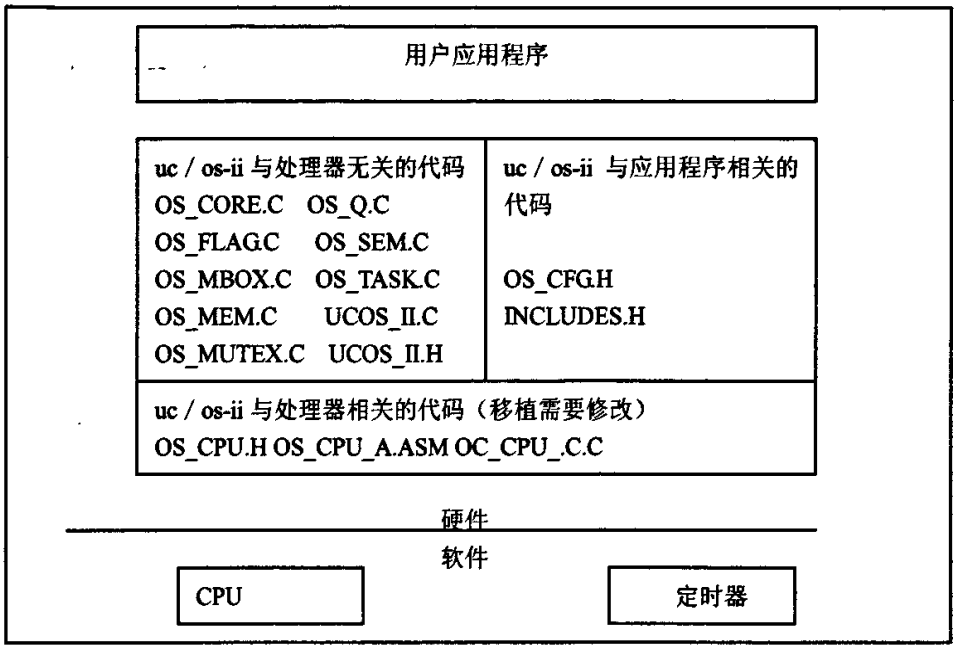

图 2-2 μC/OS-II 体系结构图

2.4.1.4 μC/OS-II 实时性实现使用的关键技术

1). 占先式内核

在多任务系统中，内核是负责管理各个任务的，或则说为每个任务分配 CPU 时间，并且负责任务间的通讯。内核是加在用户应用程序中的软件，会增加 ROM（代码空间）的用量，内核本身的数据结构增加了 RAM（数据空间）的用量，更主要的是，每个任务要有自己的栈空间，这部分占用内存相当多，因此 RAM 很有限的单片机一般不能运行实时内核。当系统时间响应很重要时，要使用占先式内核。其调度原理如图 2-3 基于优先级的抢占式调度过程所示。

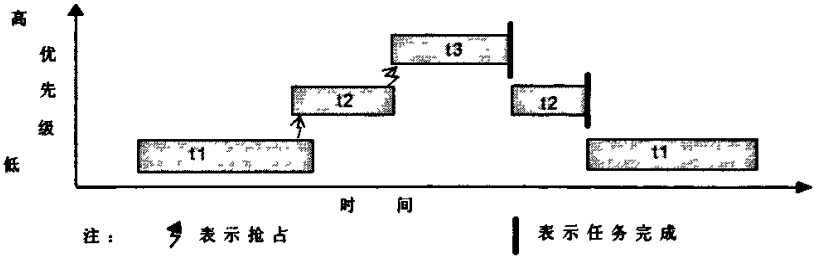

图 2-3 基于优先级的抢占式调度过程

多数的实时内核是基于优先级调度法的[12]。μC/OS-II 是一个可剥夺型内核，可以保证实时性要求高的任务可以最快的时间内分配到 CPU 时间。最高优先级的任务一旦就绪，总能得到 CPU 的使用权。使用可剥夺型内核，最高优先级的

任务何时可以执行，何时可以得到 CPU 的使用权，这些是可知的，可剥夺型内核使得任务级响应时间得以最优化，因此占先式内核是系统实时性的基础。

2). 调度策略分析

一个任务，也称作是一个线程，是一个简单的程序，该程序可以认为 CPU 完全属于自己，典型的是每个任务都是一个无限循环，都可能处于以下 5 种状态之一：休眠态，就绪态，运行态，挂起态，以及被中断态。μC/OS-II 操作性系统 5 种状态之间的转换如下图 2-4 所示。

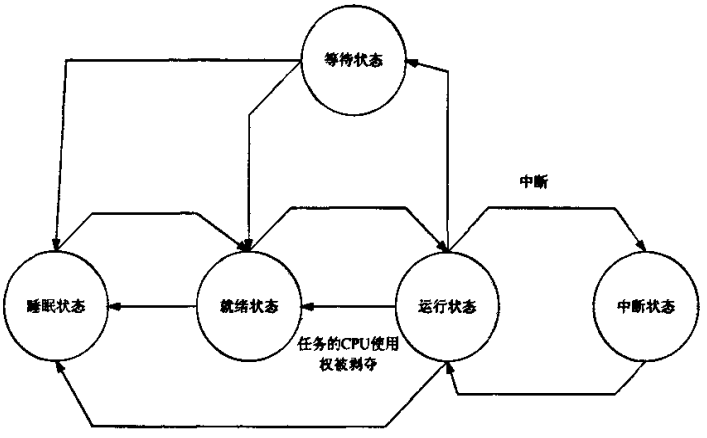

图 2-4 状态转移图

任务调度策略是直接影响实时性能的因素。实时系统的实时性主要反映在选择调度算法的优劣。选择基于优先级调度的算法足以满足准实时系统的要求，而且可以提供高速的响应和大的系统吞吐率。当两个或两个以上任务有同样优先级，通常用时间片轮转法进行调度。对硬实时系统而言，需要使用的算法就应该是调度方式简单，反应速度快的实时调度算法了。尽管调度算法多种多样，但大多由单一比率调度算法和最早期限优先算法变化而来。前者主要用于静态周期任务的调度，后者主要用于动态调度，在不同的系统状态下两种算法各有优劣。在商业产品中采用的实际策略常常是各种因素的折中。

3). 任务优先级分配

每个任务都有其优先级。任务越重要，赋予的优先级应越高。应用程序执行过程中其任务优先级不变，则称之为静态优先级。在静态优先级系统中，诸任务以及它们的时间约束在程序编译时是已知的。反之，应用程序执行过程中，任务的优先级是可变的，则称之为动态优先级。μC/OS-II 持基于固定优先级抢占式调度，用法比较简单。

2.4.2 TCP/IP协议栈

由于本系统的实时操作系统没有TCP/IP协议栈，因此在系统中移植一个实时性比较高的协议栈。传统的TCP/IP协议在实现实时性方面做得不够好，它把大量的精力花在保证数据传送的可靠性以及数据流量的控制上[14]。而在实时性要求比较高的嵌入式领域中，因其实现过于复杂，需占用大量系统资源，而嵌入式应用的系统资源往往都很有限，故无法满足系统实时性要求。下面对几种公开源代码TCP/IP协议进行分析。

2.4.2.1 几款常用的TCP/IP协议栈分析及选择

1). BSD TCP/IP协议栈。在历史上，BSD栈是其他商业TCP/IP协议栈的起点，即大多数专业TCP/IP栈是BSD栈派生的。这是因为BSD栈在BSD许可协议下为其它商业协议栈提供了这些专业栈的雏形。

2). μC/IP是一套基于μC/OS操作系统且开放源码的TCP/IP协议栈，亦可移植到其它操作系统，是一套完全免费的、可供研究的TCP/IP协议栈。uC/IP大部分源码是从公开源码BSD发布站点和KA9Q(一个基于DOS单任务环境运行的TCP/IP协议栈）移植过来。μC/IP具有如下一些特点：带身份验证和报头压缩支持的PPP协议，优化的单一请求/回复交互过程，支持IP/TCP/UDP协议，可实现的网络功能较为强大，并可裁减。μC/IP协议栈被设计为一个带最小化用户接口及可应用串行链路网络的模块。根据采用CPU、编译器和系统所需实现协议的多少，协议栈需要的代码容量空间在30-60KB之间。

3). LwIP是一套专用于嵌入式系统的开放源代码TCP/IP协议栈[15]。LwIP的含义是Light Weight(轻型)IP协议。LwIP可以移植到操作系统上，也可以在无操作系统的情况下独立运行。LwIP协议栈实现的重点是在保持TCP协议主要功能的基础上减少对RAM的占用，一般它只需要几十K的RAM和40K左右的ROM就可以运行，这使LwIP协议栈适合在低端嵌入式系统中使用。LwIP的特性如下：支持多网络接口下的IP转发，支持ICMP协议，包括实验性扩展的的UDP(用户数据报协议)，包括阻塞控制、RTT估算、快速恢复和快速转发的TCP（传输控制协议），提供专门的内部回调接口（Raw API）用于提高应用程序性能，并提供可选择的Berkeley接口API。

4). μIP是专门为8位和16位控制器设计的一个非常小的TCP/IP栈。完全用C编写，因此可移植到各种不同的结构和操作系统上，一个编译过的栈可以在几KB ROM或几百字节RAM中运行。μIP中还包括一个HTTP服务器作为

服务内容。

5). TinyTcp 栈是 TCP/IP 的一个非常小和简单的实现，它包括一个 FTP 客户。TinyTcp 是为了烧入 ROM 设计的，并且现在对大端结构是可用的。TinyTcp 也包括一个简单的以太网驱动器用于 3COM 多总线卡。

在选择一个开源协议栈可以从如下四个方面来考虑：

1). 是否提供易用的底层硬件 API，即与硬件平台的无关性；

2). 与操作系统的内核 API；

3). 协议栈需要调用的系统函数接口是否容易构造，对于应用支持程度；

4). 最关键的是占用的系统资源是否在可接受范围内，是否有裁减优化的空间。

其中，BSD 栈可完整实现 TCP/IP 协议，但它的代码庞大，为 70KB-150KB，而且裁减优化有难度，μP 和 TinyTcp 代码容量小巧，实现功能精简，限制了在一些较高要求场合下的应用，如可靠性与大容量数据传输。LwIP 和 μC/IP 是同量级别的两个开源协议栈，两者代码容量和实现功能相似，LwIP 没有操作系统针对性，它将协议栈与平台相关的代码抽象出来，用户如果要移植到自己的系统，需要完成该部分代码的封装，并为网络应用支持提供了 API 接口的可选性。uC/IP 协议最初是针对 μC/OS 设计，为方便用户移植实现，同样也抽象了协议栈与平台相关代码，但是协议栈所需调用的系统函数大多参照 μC/OS 内核函数原型设计，并提供了协议栈的测试函数，方便用户参考。根据以上分析，从应用和开发出实时系统和研究的人及参考的资料比较多的角度考虑，本文采取的是精简 TCP/IP 协议栈 LwIP。

2.4.2.2 LwIP 协议栈分层结构

本课题在实现时使用的是 LwIP 的稳定版 V0.5.3。LwIP 协议栈的设计采用了分层的风格，每一个协议层分别解决了互联问题的一个部分，可分为物理层、数据链路层、网络层、传输层、应用层[16]。协议栈只实现了 ARP、IP、ICMP、UDP 和 TCP 协议，使微处理器从繁重的协议实现和数据运算中解放出来。图 2-5 为 LwIP 协议栈分层结构图。

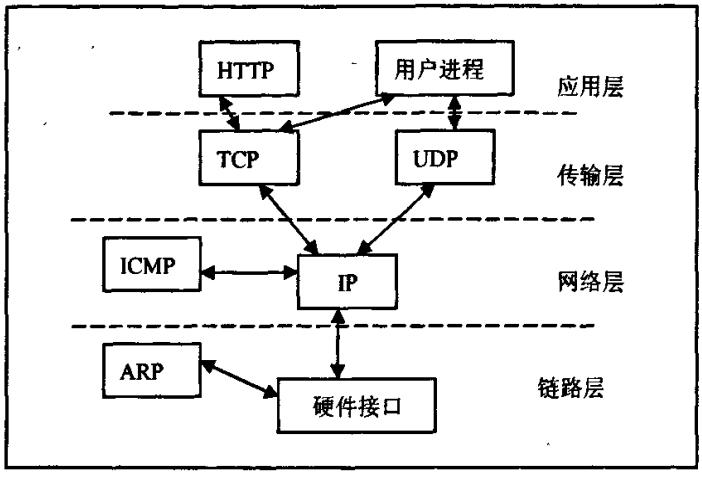

图 2-5 协议栈分层结构示意图

2.4.2.3 LwIP 协议栈进程模型

第一种通用的方法将一种协议执行的进程模型作为每一层协议作为一个标准的进程。在这种模型里，每一层协议都进行严格的区分，协议之间的通信必须要严格地定义。这种方法的好处是代码的理解和调试都相对来说比较简单。不足之处是数据报穿过不同的协议层时，会多次引起任务切换，增加系统负担。例如收到一包 TCP 帧，会引起三次任务切换，从网络接口层的设备驱动到 IP 层，再到 TCP 层，最后到应用层。在大部分的操作系统中，任务切换都是非常耗资源的，会使系统的性能下降。

第二种通用的方法是让通信协议进驻操作系统的内核。在这种模式下，应用程序和协议的通信是通过系统调用的。协议栈本身并不严格区分每个协议层，只是严格区分应用层和底层的协议，这样底层的协议可以较少的受到干扰。在大部分的操作系统中，底层的协议是作为操作系统核的一部分来执行的。应用程序只能看到抽象后的 TCP/IP 协议。应用程序与协议栈通信的过程与进程间的通信过程相同。

还有一种方法是让通信协议栈单独作为一个进程并和操作系统内核区分开。应用程序或则与协议栈在同一个进程中或则单独作为一个进程。如果应用程序和协议栈在同一个进程中，那么他们就通过函数调用的方式进行通信。如果应用程序和协议栈不在同一个进程中，那么就通过操作系统提供的消息来进行通信的。对于消息的操作一共有两种，一个是释放消息量，一个是接收消息量。

我们使用第三种方法将 LwIP 协议栈单独作为一个进程。主要的好处是协议栈独立作为一个进程可以方便实现在不同操作系统下的移植。

2.4.3 嵌入式web文档技术

2.4.3.1 嵌入式web文档分类

嵌入式 web 服务器实现，相当对于单机版的嵌入式系统，增添了一个网络监控功能。在实现基本的网络通讯后，还必须要一个监控的界面文档。所有 Web 文档可以划分为如下三类[17]。

1). 静态文档

静态 Web 文档是一个存储于 Web 服务器的文件。静态文档是在编写的时候已经确定文档内容，在网络中反映出来为静态网页。由于文档内容不会变化，所以对静态文档的每次访问都返回相同结果。

2). 动态文档

动态 Web 文档不是以一个预先定义的格式存在，而是在浏览器访问 Web 服务器时建立。当一个请求到达时，Web 服务器运行应用程序创建动态文档，服务器返回程序的输出作为应答。由于每次访问都要创建新的文档，动态文档的内容是变化的。

3). 活动文档

一个活动文档不完全由服务器一端说明，而是包括一个计算并显示值的程序。当浏览器访问活动文档时，服务器返回一个浏览器可以本地执行的程序。当该程序运行时，它可以和用户交互执行并不停地改变显示，如 ActiveX 控件。

在监控系统中各种信息是不断变化的，所以服务器必须具有提供动态文档的能力。在传统方式下，处理动态文档的 web 服务器需要三个特性。首先，服务器程序必须扩展，当每次请求到达时，能够执行一个单独的创建文档的应用程序。服务器必须编成能够捕获应用程序的输出，并且将该文档返回给浏览器。其次，必须为每个动态文档写一个单独的应用程序。第三，服务器必须配置成能够知道哪一个 URL 对应于动态文档和哪一个 URL 对应于静态文档。对每个动态文档，配置时必须说明产生文档的应用程序。动态文档可以下载到客户端，结合客户端资源实现界面变化。

2.4.3.2 嵌入式web文档实现技术分析

嵌入式 web 文档技术主要设计客户端实现方式，用于监控设备状态。Web 文档技术到目前为止，实现方法有很多种，从最早使用的 CGI，再到 JSP、ASP、Java Applet 和 ActiveX 控件等。本节主要是分析这几类技术的实现原理，根据具体需求选择那种实现方式。

1). CGI

CGI是Common Gateway Interface（通用网关接口）的简称，是早期Web开发最常用的Web服务器扩展方式。CGI可以使你能够运行不属于Web服务器的应用程序。许多CGI应用程序是用脚本语言编写的。由于这种语言的可移植性，因此这些语言是扩展Web服务器性能的流行方法。CGI程序具有灵活性和可移植性，但是由于CGI必须对每个CGI请求重新启动一个新的进程，需要耗费服务器大量的时间和内存，当并发的请求数目很大时，使用CGI效率较低。所以，CGI对大流量的Web站点不是最佳解决方案。在CGI完成对该请求的服务后，将取消该进程以及与它相关的任何信息。网页中的主要界面，是通过CGI绘制完整，以HTML网页的形式发送给客户端的，而此时的CGI如果是靠服务器资源运行的，会对系统的性能造成潜在的危险，不太适合现代工业远程监控系统的要求。

2). EGI

在用于远程监控的嵌入式Web服务器中，为了能够在客户端浏览器动态设置设备的状态以及获取设备的状态和传感器的实时值，要有和CGI功能相似的接口标准，用来调用其他可执行程序动态生成带有实时数据的Web页面。严格的讲，这不是一个标准的接口协议，将其称之为嵌入式网关接口(Embedded Gateway Interface,EGI)[18]。

EGI包含三部分：输入、输出、环境变量。当服务器接收到HTTP请求时，接收程序会判断用户请求的文档扩展名，若扩展名为“.html”，则直接从文件系统查找该文件并返回给用户，如果扩展名为“.egi”，则调用相应的EGI函数并将参数作为EGI函数的输入。EGI对参数进行解析以更新该次连接的环境。EGI的环境变量与CGI的环境变量有所不同，EGI的环境变量主要是用户对设备的启动、停止命令以及对设备的设置等参数，相应的EGI函数通过这些环境变量来对响应的设备发出控制命令或设置相应的寄存器。EGI的输出也并不完全是由EGI函数产生的虚拟文件。EGI函数首先会在文件系统中取出用户请求的文件，并在文件中寻找特殊标记，对特殊标记进行替换处理后再输出该文件。EGI函数首先会在文件系统中取出用户请求的文件，并在文件中寻找特殊标记，对特殊标记进行替换处理后再输出该文件。这种方法在实现动态页面刷新需要调用大量微处理器资源，且如果界面复杂使文件比较大，则在传输时，所需要花费大量的时间，也不太适合工业设备扩展实时性的要求。它的实现原理如图2-6所示。

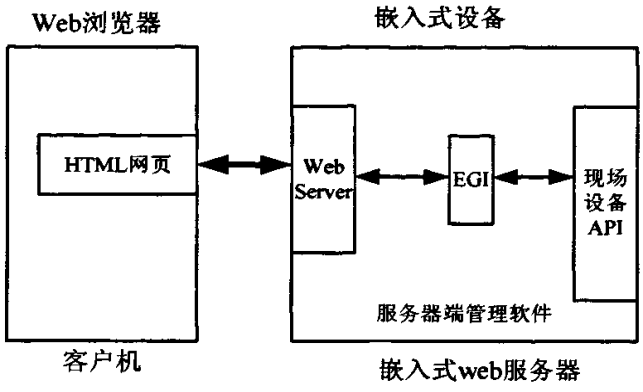

图 2-6 EGI 的嵌入式 web 体系框架

3). Java 技术

Java Applet 是用 Java 语言编写的一些小应用程序，这些程序是直接嵌入到页面中，由支持 Java 的浏览器(IE 或 Nescape)解释执行能够产生特殊效果的程序。Java Applet 可以解决网络传播图像速度缓慢问题。当浏览器访问嵌入了 Java Applet 的 HTML 网页时，Java Applet 代码就会被下载到本地计算机中运行，绘制图像的工作由 Java Applet 在客户机完成。嵌入式系统只负责将采集到的数据传输给 Java Applet 客户程序，相对于 EGI 接口，故可以提高系统的实时性。Java Applet 客户程序可以反复地连接到服务器以保持图片数据的数据更新。这样，在网络中仅需要传送嵌入式系统所采集的数据，而且解决嵌入式系统中如采用 JSP、ASP 等技术使得硬件资源不足的问题。同时由于 Java Applet 代码是下载到本地计算机中运行，所以嵌入式操作系统不需要 Java 虚拟机，实现如图 2-7 Java Applet 的嵌入式 web 服务器框架所示。但是 Java Applet 技术运用到嵌入式 web 服务器存在一定不足：在每次运行时，都要到服务器下载一遍，这显然要消耗服务器资源；Java 运行速度不够快，影响实时性。

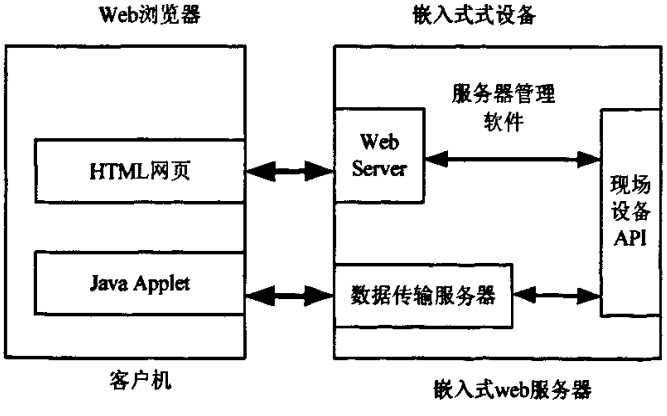

图 2-7 Java Applet 的嵌入式 web 服务器框架

4). ActiveX+java Script 技术

ActiveX 是微软公司开发的可以嵌入式网页等容器中直接运行的技术。采用 ActiveX 控件的嵌入式 web 服务器的实现方式和 Java Applet 类似，控件代码会下载到客户端运行，服务器不用绘制网页界面，而全部交给客户端绘制；服务器只在第一次请求网页时发送控件，之后只负责发送请求的动态数据，故可以节省大量的微处理器资源和时间；相对于采用 Java 技术实现的 web 文档，这种设计方案的优点是：在第一次下载后，无须再次传递界面，故可以节省网络资源和提高系统的实时性，而且，这种设计安全性相对比较高。除此之外，采用 Vsiual C++ 设计的界面设计比较容易。系统原理如图 2-8 所示，

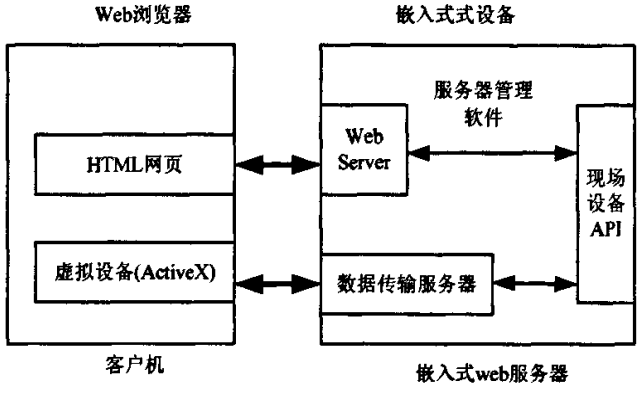

图 2-8 嵌入式 web 服务器的体系框图

2.5 影响嵌入式 web 服务器的实时性的网络通讯因素

2.5.1 网络通信通用技术分析

早期的设备与主机的通讯是通过 RS232/RS400/RS485 串口来实现的，存在数据交互慢和数据量小的缺点，以太网应用于工业中，解决了这个问题。但是，商用以太网由于其传输机制，存在着网络不确定性。

商业以太网 Ethernet 采用的介质访问控制方法是 CSMA/CD (Carrier Sense Multiple Access with Collision Detection，冲突检测载波监听多点访问)机制[19]。它的基本工作原理是：某节点要发送报文时，首先监听网络，如网络忙，则等到其空闲为止，否则将立即发送；如果两个或更多的节点监听到网络空闲并同时发送报文时，它们发送的报文将发生冲突，因此每个节点在发送时，还必须继续监听网络。当检测到两个或更多个报文之间出现碰撞时，节点立即停止发送，并等待一段随机长度的时间后重新发送。在网络负荷较高时，以太网上存在的这种碰撞

成了主要问题，因为它极大地影响了以太网的数据吞吐量和传输延时，并导致以太网实际性能的下降[20]。

对于以太网的通信延迟不确定性，国内外学者提出了各种改进方法。这些方法可分为两类：硬实时方法和软实时方法[39] [40]。硬实时是指通过设计适当的硬件电路，限制节点访问网络的时间和速率，来减少网络碰撞和排队延迟，以满足通信的实时性。软实时是指在不增加节点成本的同时，用软件调度策略对CSMA/CD和BEB机制进行改进，以提高通信的实时性。

2.5.2 EPA通信机制

EPA是一种软实时改进方案。EPA实现确定性通讯主要通过它的确定性通信调度机制实现的，原理如下：

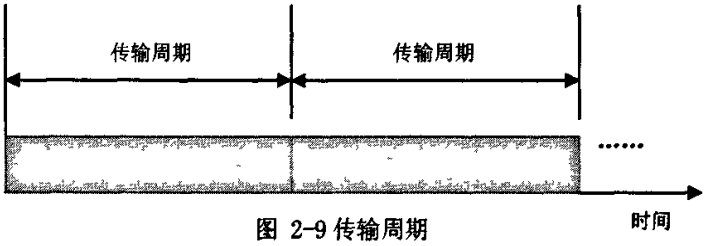

图 2-9 传输周期

对于网络中的周期信息，采用一种时隙访问的控制方式，以固定的时隙进行周期性的数据交换；而对于网络中的非周期信息，采用基于虚拟令牌的集中调度方式，在周期信息交换的间隙进行。为满足非周期信息的集中调度，要求网络上至少有一个主设备和一个或多个从设备，从设备只有在获得来自主设备的令牌后才能发送非周期信息；另外，网络上必须至少有一个时钟服务器，用于各个设备之间的时钟同步，保证每个设备在确定的时间发送周期信息[21]。

在保证网络上每个设备之间保持时钟同步的前提下，由主设备在系统组态时确定每个从设备发送周期信息的时间。主设备将整个网络传输时间划分为无限个等长的时隙，每个时隙称为一个传输周期。所有设备在各个时隙中发送和接收周期和非周期信息。每个传输周期由两个部分组成：一部分用于周期信息的通信，另一部分用于非周期信息的通信（如图 2-9 所示）。

在一个传输周期中，不同设备发送周期信息的时间相对于传输周期的起始时间的偏移量是不同的（如图 2-10 所示），这就避免了有多个设备同时访问网络资源，从而避免了在网络上发生冲突的可能。同时，每个设备的发送时间在发送周期时具备确定性。

在周期信息通信阶段，每个设备在预定的时刻发送完周期信息之后，还留有一段时隙，用于向主设备发送一个报文，通知主设备它在接下来的时间里是否还

有非周期信息要发送。如果有，还要标明该非周期信息所需要的发送持续时间，以及信息的优先级，主设备将有非周期信息要发送的设备的 IP 地址，信息的优先级以及信息的预期发送时间存放在一个队列中。如果设备没有非周期信息要发送，它就向主设备发送一个空闲报文。通过这种方式，主设备就可以知道每个从设备是否有非周期信息要发送，在此同时，主设备也可以监控每个从设备的状态。如果主设备在连续三个周期内没有收到来自某个从设备的空闲报文，那么就认为该设备已经失效。

在非周期通信阶段，主设备根据在周期信息通信阶段所获得的从设备的信息，按照要发送非周期信息的设备提供的非周期信息的优先级从高到低，依次发送非周期信息调度令牌。令牌的持续时间根据要发送非周期信息的设备所提供的非周期信息发送时间来确定，一般不小于非周期信息发送持续时间，以保证非周期信息完整发送。

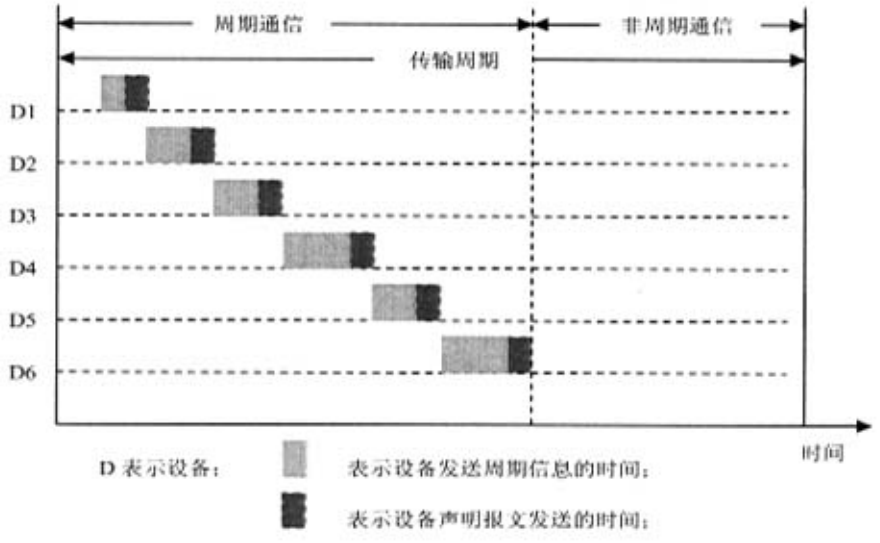

图 2-10 周期信息通信过程

从设备收到来自主设备的非周期信息调度令牌后，它得到在一定时间内发送非周期信息的权利，时间长度在令牌中指定。为了保证周期信息的正常发送，令牌持有时间不能超过周期信息通信的间隙。设备发送完成或到达最大允许发送的时间时，向主设备发送一个确认信息，同时释放令牌。在保证周期信息实时传输的前提下，为了保证重要而紧急的非周期信息（如报警信息）能及时发送，为每种非周期信息划分不同的优先级，分别为 0 级，1 级，2 级。其中，0 级优先级是最高优先级，2 级优先级是最低优先级，1 级优先级处在两者之间。同时，非周期信息调度令牌也划分为同非周期信息相同的优先级类别。设备收到令牌后，比较发送缓冲区中的非周期信息和令牌的优先级，只有当非周期信息的优先级不

低于令牌的优先级，并且，令牌持有时间不小于信息发送时间时，才发送非周期信息，否则，直接向主设备释放令牌。

2.6 系统硬件开发环境

2.6.1 硬件开发平台介绍

嵌入式 web 服务器的硬件环境主要由嵌入式微控制器、存储器模块、网络接口模块等构成。设计嵌入式 Web 服务器硬件，要充分考虑各部件工作的协调性和实现的简洁性，以获得稳定、可靠、优化和高效的硬件支持平台。本论文采用的嵌入式 web 服务器系统硬件构架图，结构如图 2-11 所示。

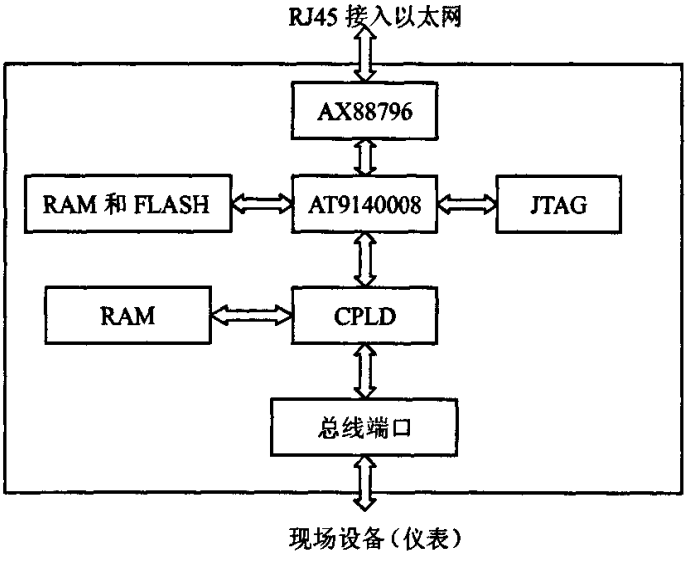

图 2-11 嵌入式 web 服务器系统硬件构架

微控制器是嵌入式系统的硬件核心，本论文采用硬件资源相对丰富的 ARM7 系列 32 位芯片 AT9140008，它是 RISC 微处理器结构，具备 32 位寻址方式，可以很容易构件较大的 Flash 存储空间，带有 256k 片内 32 位 SRAM，可以给予程序方便在线调试[22]。ARM40008 是一款性价比极高的芯片，是复杂嵌入式系统开发应用硬件平台的良好选择。

在嵌入式系统中，扩展以太网接口常用的方法如下表 2-1 所示。

从降低开发成本和设计简单考虑，本论文采用第一种方法实现以太网接口模块。采用台湾 Asix 公司推出的 NE2000 兼容快速以太网控制器 AX88796。这种方法不需要嵌入式处理器有网络接口控制器，只要把以太网芯片连接到嵌入式处理器的总线即可。此方法通用性强，不受处理器的限制，且价格适当。

表 2-1 嵌入式系统接入以太网方式

以太网扩展方式	实现方式	优点	缺点
嵌入式处理器+以太网接口芯片	用纯软件来实现TCP/IP的协议栈，让嵌入式Web服务器软件在此之上运行，并可直接连接到网络上。这种直接连接的方式，通常需要CPU具有很强的运算处理能力，要求32位处理器，以太网芯片组连接到以太网上，从而让TCP/IP协议栈软件与外界进行通信	设备可直接挂接到网络上；整体性好，不需要其他辅助硬件，全部硬件就是CPU和接口芯片，硬件设计简单，成本低。	需要处理能力较高的CPU，通常为32位运算能力的处理器；代码量和数据内存耗费多，需要较大的存储空间；纯软件协议栈，软件调试复杂。
选择带有以太网接口芯片的嵌入式处理器	完全由硬件来实现TCP/IP协议栈，只需在硬件规划时设计好接口即可。也可采用能实现TCP/IP的硬件电路板，通过串口接入到现场智能设备上，从而实现硬件TCP/IP协议。	设备可以直接挂接到网络上；全部的TCP/IP协议栈由外围芯片硬件实现，减轻了调试软件的负担；CPU不用运行TCP/IP协议栈软件，减轻了处理负担，对CPU的性能要求降低，减少了存储器使用空间。	增加了外围芯片，成本增加；增加了硬件设计复杂度和产品成本。
网关形式	让一台PC机来充当外部网关，在此之上运行完整的TCP/IP协议和部分嵌入式Web服务软件，通过串口等方式使每台PC机可控制一个或者多个现场设备。外界对现场设备的访问，先要通过网关进行解析，现场设备交换信息，提取网页等，并将最终的信息送给外部访问者。	现场设备不需其他辅助的硬件，只需有简单的RS-232/ RS-485等通信接口；极大减轻了CPU的负载，对CPU的性能和存储器空间的要求大大降低；基于8位或16位微处理器使得控制设备比较适合采用外部网关形式Web服务器；由于软硬件的修改很少，极大缩短了产品的研发时间。	由于需要外部的网关(通常是PC机)，增加了产品的成本和系统复杂性；在网关与现场设备之间的协议没有标准可循，通常不同的厂商之间的协议各不相同，增加了互联的难度。

2.6.2 开发方式

虽然嵌入式系统已经应用于各个领域，但由于嵌入式设备都具有功能专一、

针对性强的特点，其硬件资源一般都很有限，几乎不可能在嵌入式设备上建立一套开发系统。支持 JTAG[23]技术的嵌入式芯片，通常采用交叉开发（cross Developing)模式[24]。交叉开发模式：开发系统是建立在硬件资源丰富的 PC 机上，称其为宿主机 (host)，应用程序的编辑、编译、链接等过程都在宿主机上完成，而应用程序的最终运行平台却是和宿主机有很大差别的嵌入式设备，称为目标机。调试在二者之间交互进行。本系统也是使用交叉开发系统完成的，其包含两部分：作为开发平台的宿主系统和面向嵌入式 CPU 的目标系统。

宿主系统包含交叉编译器、交叉链接器等工具，具体的开发软件在下节会具体介绍。系统的开发还需要有交叉调试工具，本系统使用 HITOOL 调试软件把程序下载到目标系统的 CPU 的 RAM 里进行交叉调试，具体的宿主机和目标机之间的通讯在这个软件内部已经实现。通过 HITOOL 完成应用程序的下载、目标机内存和寄存器的读写、设置断点、单步调试等功能。宿主机和目标机的通信连接采用串口和并口连接。调试系统的结构图如图 2-12 所示。

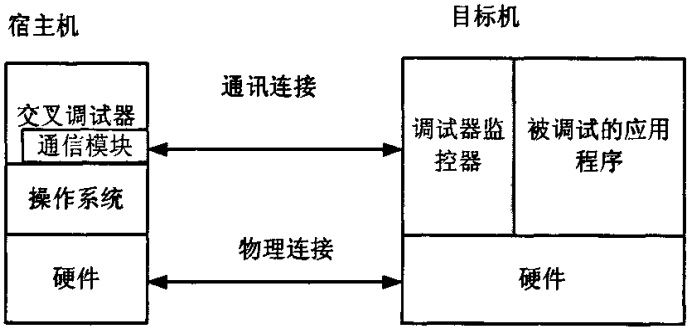

图 2-12 调试系统结构

在该交叉开发系统的支持下，系统的开发流程如下：

1). 在宿主机上编写程序的源代码，生成可执行程序；

2). 将可执行的目标代码下载到目标机的 RAM 上；

3). 使用调试器进行调试；

4). 若程序正确，则转到 8)；

5). 在调试器的帮助下定位错误；

6). 在宿主机上修改程序的代码，纠正错误；

7). 转到 2)；

8). 将可执行程序固化到目标机 Flash 中。

交叉调试方法降低了调试的难度，缩短了产品开发周期，有效的降低了开发成本，只需要一个 JTAG 进行硬件调试。

2.6.3 开发工具

软件开发工具主要分两个部分：服务端和客户端。

1). 编译软件 ARM_SDT

ARM SDT 的权英文名字微 ARM Software Development Kit，是 ARM 公司微方便用户在 ARM 芯片上进行应用软件开发而推出的一套集成开发工具。

2). 程序调试软件 HITOOL_ARM

Hitool for ARM 是一种 ARM 嵌入式应用软件开发系统，包括 Hitool ARM Debug、GNU Compiler、JTAG cable、评估板以及嵌入式实时操作系统 ThreadX 等，用于实现交叉调试的软件[24]。

3). source insight/ultraedit 程序宿主机编写软件

这是两款在宿主机上用于编写 C 代码的软件。

4). 程序下载软件 FLASH_PROGRAM

主要完成程序的下载，执行.hex 文件。

5). Ethereal 网络抓包软件/Sniffer 抓包软件，分析网络中数据包。

6). Visual C++6.0 开发环境，用于开发客户端程序。

2.7 本章小结

本章主要研究嵌入式 web 服务器的实时性，在此基础上确定本课题实时 web 服务器的设计方案，并有后面的具体设计来验证本方案的实时性。

对于一个 web 服务器，影响实时性的关键因素主要有三个：硬件、软件和网络通讯。

硬件是基础，提供服务器的硬件资源，其中微处理器和 Flash 是关键。

软件是实时性的核心，必须充分考虑优良的软件设计架构，采用实时操作系统是实时系统的基础，精简的 TCP/IP 协议栈是必要条件，在编程时要考虑到多任务的实现，可以提高程序的运行效率；采取的 web 文档技术决定客户端监控界面的优良性，这在系统构架时根据资源情况必须慎重考虑。

网络通讯是关键因素之一，传统的网络通讯如串口通讯的速度局限性以及普通商业也太网的不确定性通信机制严重影响系统的实时性，采用 EPA 现场总线技术则是一个很好的解决方法。

除此之外，本章还对硬件平台进行了简单的介绍，课题设计时所采取的开发方式和开发工具。

第3章 嵌入式实时 web 服务器的服务器端实现

通过第二章的实时性分析，我们知道，在设计一个嵌入式实时 web 服务器，软件方面是一个非常关键的因素。如何采取操作系统和协议栈、如何划分好任务和它们之间的任务同步以及如何设计实时任务，是一个实时 web 服务器能否实现其功能所必须考虑的环节。

3.1 嵌入式实时 web 服务器设计方案

嵌入式系统的软件体系结构可以分为四个层次：设备驱动、操作系统、应用中间件和应用系统，如图 3-1 所示。

设备驱动接口负责嵌入式系统与外部设备的信息交互。操作系统分为基本和扩展两部分。前者是操作系统的核心，负责整个系统的任务调度、存储分配、时钟管理和中断管理，并提供文件管理等基本服务；后者是为用户提供操作系统的扩展功能，包括网络、数据库等。应用编程接口 API 是应用编程中间件，为编制应用程序提供各种编程接口库，减轻应用程序开发负担。应用系统即为实际的嵌入式系统应用软件。

实时操作系统和 TCP/IP 协议栈，都会提供很多 API 函数。我们要设计实时服务器，主要从应用层软件出发，区别对待实时性任务和一般任务。

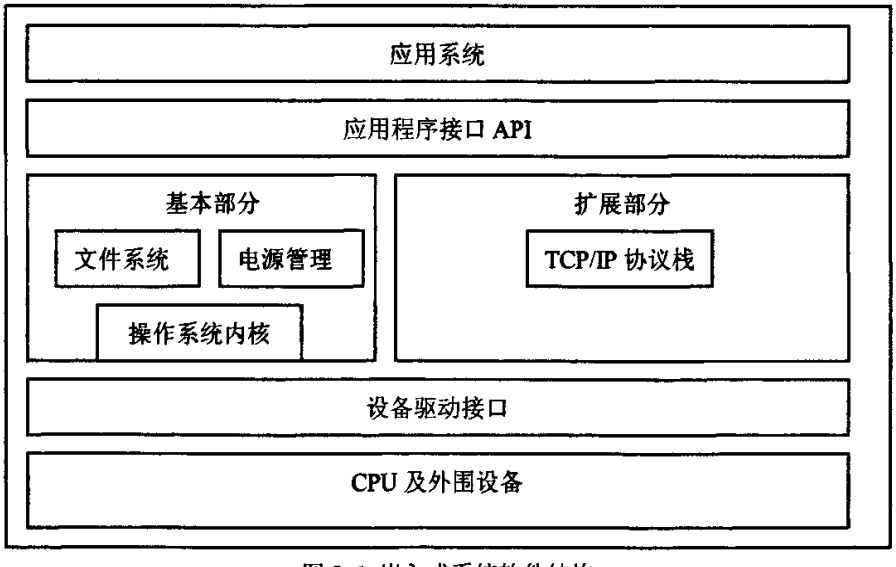

图 3-1 嵌入式系统软件结构

设计一个满足工业仪表实时性的嵌入式 web 服务器，至少需要提供如下几个条件：

1). 采用一个高性能的硬件环境，本文采用的是32位的微处理器ARM平台，并具备以太网接口硬件；

2). 选择精简的TCP/IP协议栈LwIP，使仪表成为一个微型的网络节点，与外界网络通讯；

3). 选择实时操作系统μC/OS-II作为实时系统的基础，结合协议栈完成任务调度，设计多任务系统；

4). 合理划分任务，创建HTTP服务器和文件系统、快速数据交互任务，通过信号量使它们同步；

5). 采用EPA分布式控制系统，实现在工业以太网中的确定性发送和接收，提高系统实时性；

6). 设计一个采用ActiveX技术的客户端应用程序，用于与服务器交互信息，完成用户对设备的操控。

3.2 μC/OS-II结合LwIP协议栈设计实时开发环境

嵌入式实时操作系统是系统实时性的基础。在嵌入式开发中，μC/OS-II实时操作系统，可以高效地完成系统任务的调度，提高系统的实时性能，但是它本身不具备TCP/IP协议栈。TCP/IP协议栈是工业以太网络接口的重要组成部分。简单的说μC/OS-II是一个任务调度器，在操作系统内核中没有实现TCP/IP协议栈，因此在此方案中是通过软件来实现TCP/IP协议栈的。因此，如果要设计一个网络设备，必须在这个系统的基础上，增加TCP/IP模块，而目前的公开源代码LwIP是一款精简的TCP/IP协议栈，两则的结合不但不会给微处理器增加很大的负担，因为两则都是比较简单的模块。在μC/OS-II下编写LwIP协议栈，比传统的前后台系统要方便很多，用它分配内存缓冲区，使用前申请，使用后释放，可有效地利用系统资源。在程序设计时将LwIP协议栈做成任务，而用户程序在另外的任务中运行，而且它们各自的代码量都非常精小，运行占用服务器的资源少。因此，这两则的结合可以开发实时的网络仪表。

3.2.1 μC/OS-II操作系统在AT9140008上的移植

从μC/OS-II操作系统的体系结构图2-2μC/OS-II体系结构图可以看出，μC/OS-II的源代码分为三部分：与处理器无关的内核代码，与应用相关的配置文件，与处理器相关的代码。因此，整个移植过程分为两个步骤：第一步，针对不同的处理器，修改与处理器相关的代码；第二步，根据具体的应用程序开发需要，

修改与应用相关的配置文件。因此移植时，必须编写 OS_CPU.H、OS_CPU_C.C、OS_CPU_A.ASM 三个文件。本课题中使用的编译器允许在 C 语言中插入汇编语言，所以可以将所需的汇编语言代码直接放到 OS_CPU_C.C 中，故可以不需要 OS_CPU_A.ASM 这个文件。这三个文件是与 MCU 的特性有关的，主要提供任务切换与系统时钟的功能。其它文件用 C 写成，它们为系统提供任务管理、任务之间通信、时间管理以及内存管理等功能。

下面针对 32 位高性能微控制器 AT91R40008 介绍具体的移植过程。由于操作系统的移植有相当一部分工作和所用处理器的体系结构密切相关，因此首先简要介绍 ARM7TDMI 的体系结构[25]。

ARM 微处理器可以在 7 种运行模式下工作：

1). 用户模式：ARM 处理器正常的程序执行状态。

2). 快速中断模式：用于高速数据传输或通道处理。

3). 外部中断模式：用于通用的中断处理。

4). 管理模式：操作系统使用的保护模式。

5). 数据访问中止模式：当数据或指令预取中止时进入该模式。

6). 系统模式：运行具有特权的操作系统任务。

7). 未定义指令中止模式：当未定义的指令执行时进入该模式。

除了用户模式之外，其余的 6 种模式称为特权模式；除去用户模式和系统模式以外的 5 种模式又称为异常模式。在此项目中只使用了用户模式和中断模式，μC/OS-II 内核运行期间采用的是用户模式，当有硬件中断时，MCU 自动完成从用户状态进入中断状态，在中断程序结束处，需要通过编程的方法从中断状态恢复成用户状态。

ARM 共有 37 个 32 位寄存器，但是这些寄存器不能被同时访问，这与 ARM 处理器的工作模式有关。在不同的工作模式下能够访问到不同的寄存器。

在 5 种异常模式下还分别有一个专用的物理状态寄存器，称为 SPSR（备份的程序状态寄存器），当异常发生时，SPSR 用于保存 CPSR 的当前值，从异常退出时可由 SPSR 来恢复 CPSR。

μC/OS-II 可以简单地看作是一个多任务调度器，在这个任务调度器之上添加了和多任务操作系统相关的一些系统服务，如信号量、邮箱等，因此移植的关键就是构造任务堆栈及任务切换时的上述寄存器的出入栈。移植工作主要集中在多任务切换的实现上，因为这部分代码用来保存和恢复 MCU 现场，不能用 C 语言来实现，只能用汇编语言实现。

首先在 OS_CPU.H 文件中，需要修改和设定数据类型、堆栈类型和堆栈的生长方向。因为这些都是和处理器相关的且不可以移植的，不同的微处理器有不

同的设置，故需要定义类型；根据 AT9140008 芯片的性能，设置栈的生长方式从上往下递减。而与代码相关的一部分，为了保护临界段代码免受多任务或中断服务子程序的破坏，需要两个宏用于在进入临界区程序时关闭中断 OS_ENTER_CRITICAL，离开临界区程序时打开中断 OS_EXIT_CRITICAL。为了避免出现在进入临界段代码前已经禁止了中断，而运行完临界段代码，中断却被被打开了这种情况；实现这两个宏时，进入临界区首先保存处理器的状态，然后再关闭中断，离开临界区时恢复前面保存的处理器状态字。

其次，在 OS_CPU_C.C 文件中，移植时需要编写 10 个简单的 C 函数：

OSTaskStkInit()、OSTaskCreateHook()、OSTaskDelHook()、OSTaskSwHook() OSTaskIdleHook()、OSTaskStatHook()、OSTimeTickHook()、OSInitHookBegin()、OSInitHookEnd()、OSTCBInitHook()。

其中唯一必要的函数是 OSTaskStkInit()，其他 9 个函数必须申明，但并一定要包含任何代码。应用程序在使用一个任务前首先要任务，并在其中调用堆栈初始化函数 OSTaskStkInit()，主要是模拟任务被中断后的堆栈结构，定义了 14 个寄存器和让系统开启中断请求。而至于其它函数，本课题只申明了，但没有具体编写。

3.2.2 LwIP 在μC/OS-II 操作系统中的移植

LwIP 协议栈在设计时就考虑到移植问题，因此把所有与硬件、操作系统、编译器相关的部分独立出来，放在一个目录下。因此 LwIP 在μC/OS-II 上的实现只需修改这个目录下的文件[26]。下面分几部份来说明相应文件的实现过程。

1). 与 MCU 或编译器相关的 include 文件

在文件 cc.h、cpu.h、perf.h 中有一些与 CPU 或编译器相关的定义，如数据长度，字的高低位顺序等。这些应该与用户实现μC/OS-II 时定义的数据长度等参数是一致的，还要定义处理器的大小端存储系统，我们采用的 ARM 处理器是小端存储系统（系统存储的栈地址是从高地址往低地址递减）。

此外还有一点：一般情况下 C 语言的结构体 struct 是 4 字节对齐的，但是在处理数据包的时候，LwIP 使用的是通过结构体中不同数据的长度来读取相应的数据的，所以，一定要在在需要紧凑结构的地方添加了_packed 关键字来定义结构体，让编译器放弃 struct 的字节对齐。使用了紧凑结构的结构有：以太网地址结构 eth_addr、以太网帧头部结构 eth_hdr、IP 地址结构 ip_addr、ARP 帧头部结构 arp_hdr、IP 帧头部结构 ip_hdr、icmp 帧头部结构 icmp_echo_hdr、TCP 帧头部结构 tcp_hdr、UDP 帧头部结构 udp_hdr。使用了_packed 关键字，使得处理器

在处理这些结构体时是按照字节来处理的。

2). sys_arch 操作系统相关部份

sys_arch.c 和 sys_arch.h 中的内容是和操作系统相关的一些结构和函数，主要可以分为四个部份：

a). sys_sem_t 信号量

LwIP 中需要使用信号量通信和任务进程同步，所以需要在 sys_arch 中应实现信号量结构体和处理模块，实现的功能包括创建一个信号量结构、释放一个信号量结构、发送信号量、请求信号量。

由于μC/OS-II 已经实现了信号量 OS_EVENT 的各种操作，并且功能和 LwIP 上面几个函数的目的功能是完全一样的，所以只要把 uC/0S-II 的函数重新包装成上面的函数，就可以直接使用了[27]。

b). sys_mbox_t 消息

LwIP 使用消息队列来缓冲、传递数据报文，因此要在 sys_arch 中实现消息队列结构 sys_mbox_t，以及相应的处理程序。需要实现的功能有创建一个消息队列、释放一个消息队列、向消息队列发送消息和从消息队列中获取消息[28]。

μC/OS-II 操作系统同样实现了消息队列结构及其操作，但是μC/OS-II 没有对消息队列中的消息进行管理，因此不能直接使用，必须在μC/OS-II 的基础上重新实现。为了实现对消息的管理，我们定义了以下结构：

```
typedef struct {
OS_EVENT* pQ;
void* pvQEntries[MAX_QUEUE_ENTRIES];
} sys_mbox_t;
```

在以上结构中，包括 OS_EVENT 类型的队列指针（pQ）和队列内的消息（pvQEntries）两部分，对队列本身的管理利用μC/OS-II 自己的 OSQ 操作完成，然后使用μC/OS-II 中的内存管理模块实现对消息的创建、使用、删除回收，两部分综合起来形成了 LwIP 的消息队列功能。

c). sys_arch_timeout 函数

LwIP 中每个与外界网络连接的线程都有自己的 timeout 属性，即等待超时时间。这个属性表现为每个线程都对应一个 sys_timeout 结构体队列，包括这个线程的 timeout 时间长度，以及超时后应调用的 timeout 函数，该函数会做释放连接和回收资源的工作。如果一个线程对应的 sys_timeout 为空，说明该线程永远等待连接。移植时需要实现超时函数 sys_arch_timeout，此函数的功能是返回目前正处于运行态的线程所对应的 timeout 队列指针。函数返回对应于当前任务的指向定时事件链表的起始指针。

d). sys_thread_new 创建新线程

LwIP 有一个主线程（tcpip_thread），负责处理所有的 tcp/ucp 连接。各种网络应用程序都作为其他线程和 tcpip 任务合作来实现网络功能。因此需要用户实现创建新线程的函数。由于在μC/OS-II 中，没有线程（thread）的概念，只有任务，因此将创建新任务的系统函数 OSTaskCreate 封装一下，就可以实现此功能。需要注意的是 LwIP 中的线程并没有μC/OS-II 中优先级的概念，实现时要由用户事先为 LwIP 中创建的线程分配好任务优先级。

3). lib_arch 中库函数的实现

此外还需要实现几个与用户使用的系统或编译器有关的函数，包括 16 位、32 位数据高低字节交换函数，以及内存拷贝、内存清零等函数。

4). 网络设备驱动程序

在 LwIP 中可以有多个网络接口，每个网络接口都对应了一个 struct netif，这个 netif 包含了相应网络接口的属性、收发函数。LwIP 调用 netif 的方法 netif->input()及 netif->output()进行以太网 packet 的收、发等操作。在驱动中主要做的，就是实现网络接口的收、发、初始化以及中断处理函数，利用前面实现的网络驱动程序即可。

另外，opt.h 文件设定了一些协议栈的选项，有一些是与移植有关的，在移植时需要做一些改动。另有一些选项放在了自己添加的头文件 lwipopts.h 中。

宏定义 PBUF_LINK_HLEN 表示数据链路层头部的字节数。以太网的头部占用 14 字节，但是在 ARM 处理器中，内存是 4 字节对齐的，因此在将以太网头部定义成结构体时实际占用了 16 字节的空间。所以将此宏定义设为 16。

宏定义 MEM_SIZE 表示了分给协议栈使用的内存字节数。如果应用程序需要发送接收大量的数据，则需要将此值设置的大一些。MEMP_NUM_UDP_PCB 表示最多 UDP 连接数，MEMP_NUM_TCP_PCB 表示最多 TCP 连接数，都需要根据应用来修改。TCP_SND_BUF TCP 发送缓冲区的大小，TCP_MSS 表示 TCP 报文段最大的长度，都可以根据实际应用进行修改。

应用程序有两种方法来使用 TCP/IP 协议栈，一种是通过回调函数的方法，直接调用 TCP、UDP 模块中的函数。由于函数调用不能够跨线程，因此应用程序必须和协议栈在同一个任务中。另一种是使用 lwip 提供的 API 接口。API 程序使用了操作系统提供的进程间通信方法来与 TCP/IP 协议栈进行通信。应用层接口为应用层提供了使用 TCP/IP 协议的方法，具体包括：创建连接、释放连接、套接字绑定、建立连接、网络监听、接收数据、发送数据、关闭连接等函数。

3.3 实时任务创建和同步

在本课题中，各个任务可以通过 sys_thread_new（由 OSTaskCreate 封装成的函数）或则 OSTaskCreate 来建立。μC/OS-II 操作系统在任务切换和中断请求时间都小于 10us。

为完成整个系统基本操作，根据任务优先级以及对实时性的要求，设计了 5 个任务。其中各个任务优先级和功能如下表 3-1 所示。

表 3-1 系统任务表

任务名称	功能	优先级
tcpip_thread	通过这个任务把整个协议栈封装成一个任务，所有的网络通讯协议都是由这个任务完成，是一个基本任务	15（最高）
http_thread	HTTP 服务器，实现与客户端的交互	16
tcpecho_thread	文件系统，实现控件和网页的下载，	17
real_data_send	实时任务：数据发送任务，采用 UDP 协议实现，发送速度快	31
LEDTask	实时任务：与仪表数据库的数据交互	32

为了实现一个服务器的通讯，嵌入式 web 必须建立多个任务合作。为了提高运行效率和实时性，必须安排好几个任务的同步。在本课题中，采取的是消息 net_mutex 来实现各个任务之间的同步。首先对于 tcpip_thread，这是一个 TCP/IP 主线程，所有任务都必须在它完成初始化后才能使用，故安排为优先级最高；其次，就是要考虑到 HTTP 服务器了，在运行了系统后，如果客户端调用服务器端资源，则必须通过这个服务器响应，故它是作为系统的一个网络接口来操作的，如果没有响应，则这个任务就会处于永远等待网络连接；再此，就是数据发送任务 real_data_send，这个任务与 LEDTask 是联合一起使用的，因 LEDTask 比 real_data_send 任务优先级低，为了实现一个数据先从数据库提取出并发送到网络的循环过程，我们可以采取一个消息邮箱，具体实现会在本章第 5 节描述。同步实现原理如图 3-2 所示。

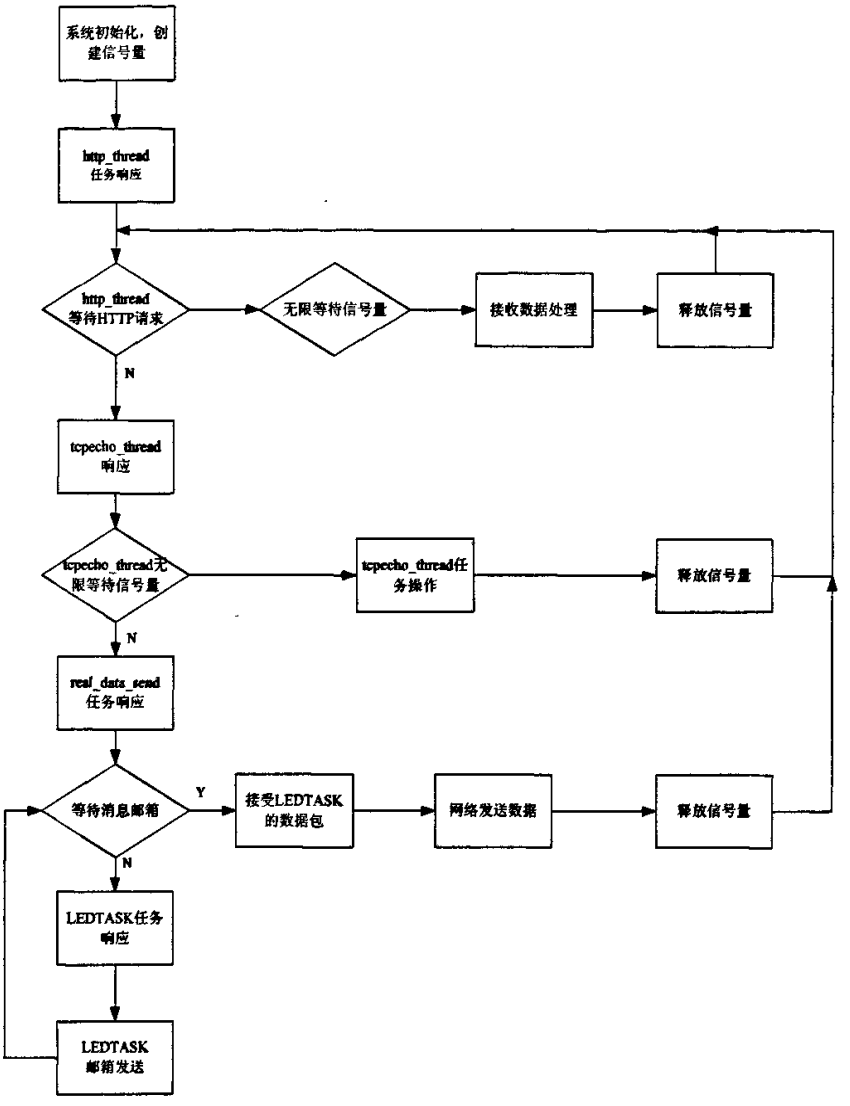

图 3-2 系统任务同步原理图

3.4 HTTP 服务器和文件系统

3.4.1 HTTP 服务器的设计

HTTP 服务器是为嵌入式系统提供数据展示的完善的模块，是服务器中相当重要的部分。HTTP 服务设计为一个独立的任务，优先级次高，通过中断可以随时响应客户机。

HTTP 服务器设计为基本的请求/响应模型：1、客户端发出 HTTP 请求（Request）；2、服务端返回响应信息（Respond）。本文对 HTTP 服务器进行简

单的构造，并将其驻留在非常小的代码段中。为了简化设计，将服务器固定在一个固定的 IP 地址，由于工业现场应用中有穿过防火墙的要求，所以套接字绑定在 80 端口。服务器提供发布网页和控件外，还提供了对动态内容的支持。虽然提供静态内容的服务适用于某些应用场合，但在嵌入式系统中，HTTP 服务器经常要作为远程监控来使用，为了完成数据交互和界面的动态刷新，不可能仅采用静态网页形式。

本文中，HTTP 服务器完成数据交互、网页和控件的下载，为其设计了一个任务 http_thread，任务优先级仅次于协议栈的 tcpip 任务，结合客户端程序，设计了许多标志位用于判断是什么任务。主要流程图如图 3-3 所示。

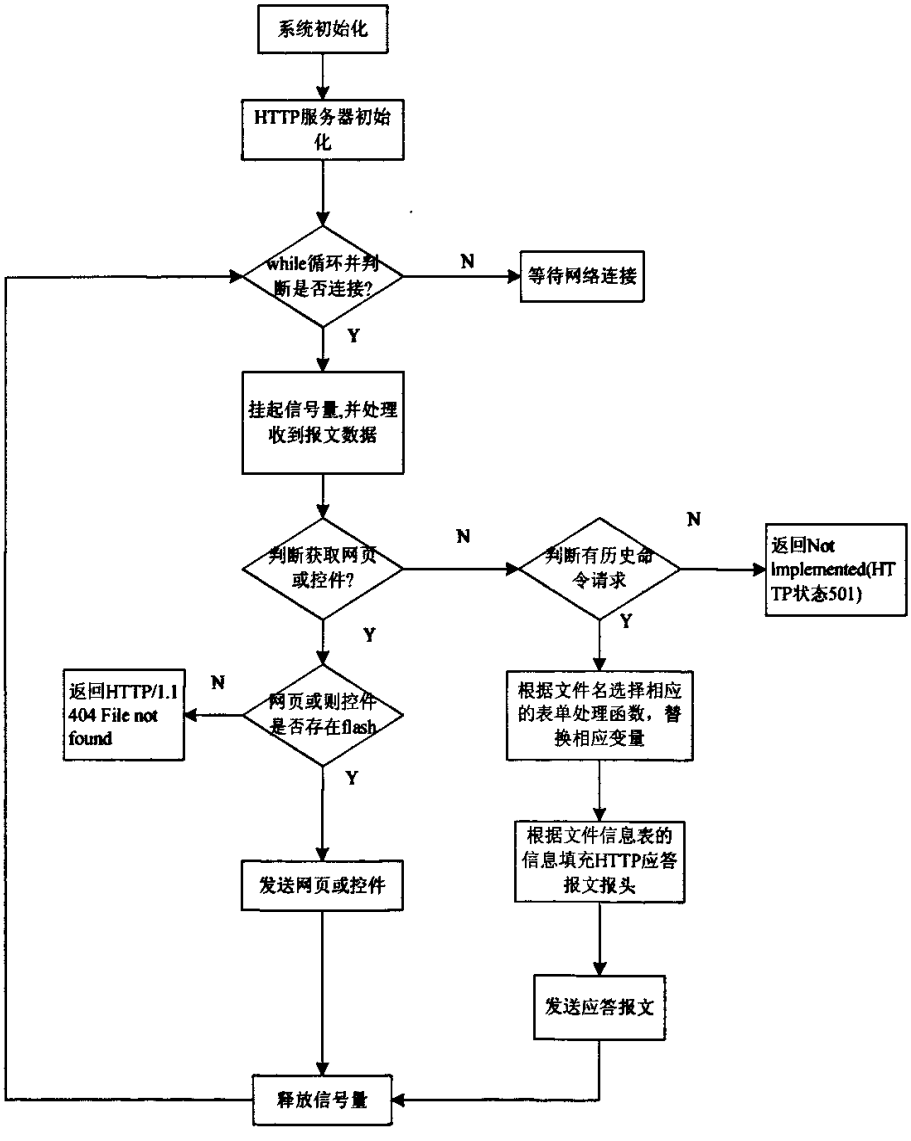

图 3-3 HTTP 服务器实现原理图

在设计中，必须注意到某些文件非常大，不能一次性全部发送完，这样会占用过多的内存。比如，本课题中的控件就是比较大的一个文件，大概有500～600K左右，故不能一次全部发送，否则会使内存溢出。在处理这类问题，可以考虑将这些打文件分成多次拷贝，每一次拷贝6k内容，最后一次拷贝剩余的内容。

3.4.2 文件系统实现

系统中使用的FLASH存储器型号为SST38VF160，NOR型的可重写存储芯片，能够在断电的情况下继续存储内容。它容量为1M×16位，分为32个块，512个扇区[29]。读数据的时间为70ns，块、扇区擦除的时间为18ms，整片擦除的时间为70ms，双字节的写入时间为14us，可以满足系统实时性要求。

由于Web服务器需要有文件读取和写入功能，因此嵌入式Web服务器需要文件系统的支持[30]。文件系统用于向用户提供一种格式化的读写存储器的方法。它需要实现的功能有管理文件的存储空间、管理文件目录、完成文件的读写操作、实现文件的共享和保护等。嵌入式系统一般都采用Flash作为ROM只读存储器，本节针对Flash存储器的特点，设计了一种嵌入式环境下的Flash文件系统[31]。

针对嵌入式系统的需求，本文设计一种小型的、简单的文件系统完成文件的读写功能。基于Flash的文件系统不能完全套用已有的文件系统，Flash能够擦除的范围越小，对文件的改动就越小，所执行的I/O操作就越小，减少了I/O时间，提高文件系统的实时性能。

本课题的文件系统实现分为两大部分：文件系统的下载模块和服务器端模块。

1). 文件系统的下载模块（客户端程序）

在本课题中，文件系统的下载模块完成控件和网页的下载到Flash，作为一个独立的可执行软件模块来设计。在开发中，如服务器端程序在编译后生成.hex文件，则可以直接使用FlashPgm下载到Flash，但这个软件不支持.ocx和.html文件，因此设计了一个下载软件，通过网络通讯下载到Flash。

下载模块主要实现在网络发送新文件给服务器端，采用创建TCP套接字建立链接，防止网络发送时出现的文件数据丢失。实现原理如下：

a). 查找要发送的文件目录和文件名

b). 初始化TCP协议并建立套接字并连接

c). 读取文件并存储在缓存中等待发送

d). 判断是否为.htm或则.ocx文件，若为.htm则跳入5，若为.ocx则跳入6，若不是，则跳出程序

e). 建立.htm 文件的标志符号，作为前几个字符结合文件所有数据存储到发送缓存中并发送

f). 建立.ocx 文件的标志符号，作为前几个字符结合文件所有缓存到发送缓存中并封装 4096 个字节作为一个数据包发送，最后不 4096 字节则全部发送完毕

2). 服务器端模块

由于系统中 FLASH 存储器还要存放系统的代码，因此可以考虑将 Flash 分作两半，前 256 个扇区用于存放程序代码，后 256 个扇区用于存放文件。文件系统参照 FAT 文件系统实现，通过 FAT 表来管理文件系统。第 256 个扇区用于存放 FAT 表，其余扇区用于存放文件。FAT 表记录[32] [33]了每一个扇区的使用情况文件名，文件长度等。由于 Flash 存储器是有擦写次数限制的，因此如果 FAT 表一直放在 Flash 中就会造成反复擦写这一块 Flash。为此在系统运行时将 FAT 表拷贝至内存中，当 FAT 出现改动时，首先在内存中进行，当改动完成时再更新 Flash 中的 FAT 表。在这里根据 Flash 的型号，4096 字节作为一个扇区。

本文的文件系统主要管理两个文件：网页和控件。网页的字节比较小，仅有 477 个字节，故一个扇区已经足够其存放，本文将其保存在后半存储区的第 23 个扇区。相反，对于控件，到目前为止，控件的大小已经达到了 655596 字节了，而且会随着界面功能的完善，会变化。因此，在设计时，预估了大小，给它开辟了一个 800K 左右的存储空间，地址从后半存储区的第 24 个扇区开始存储。

Flash 的读写是按一定时序操作的，在擦除操作时，任何其他操作都无效。Flash 擦除的时序如下[34]：

a). 在相对地址 0x5555H 处写入数据 AAH

b). 在相对地址 0x2AAAH 处写入数据 55H

c). 在相对地址 0x5555H 处写入数据 80H

d). 在相对地址 0x5555H 处写入数据 AAH

e). 在相对地址 0x2AAAH 处写入数据 55H

f). 扇区擦除则在扇区地址处写入 30H，块擦除则在块地址处写入 50H，片擦除则在地址 5555H 处写入 10H。

Flash 写数据也是按一定的时序操作，如下：

a). 在相对地址 0x5555H 处写入数据 AAH

b). 在相对地址 0x2AAAH 处写入数据 55H

c). 在相对地址 0x5555H 处写入数据 A0H

d). 写入数据

文件系统从上到下有三个层次：第一层为 API 层、第二层为中间转换层、

下层为介质驱动层。第一层 API 层：API 层是文件系统和用户应用程序之间的接口，它有一个标准 C 函数库，其中包含有诸如读文件（Read_Flash）、写文件（Write_Flash）等函数。本层的功能是将用户调用传送给中间转换层。这是整个系统设计的核心，也是嵌入式文件系统中用户唯一可见的部分。第二层中间转换层：中间转换层要为文件系统的实现提供与硬件无关的统一接口，是文件系统结构规整性的基础。中间转换层包含有文件系统子层及逻辑块子层，其中文件系统子层将文件操作解释到逻辑块子层，然后文件系统调用逻辑块子层并根据不同的设备定义出相应的设备驱动程序；逻辑块子层主要是同步对设备驱动程序的访问，向上提供友好界面。第三介质驱动层：介质驱动层是访问硬件的最低端的程序，该程序的结构要能够便于实现对硬件的访问。本层的功能主要是完成对介质的访问。本层的重要任务就是提供统一的设备驱动程序接口。

在应用层上，将文件系统作为一个独立的任务 tcpecho_thread 操作。服务器端根据客户端的发送数据包标志符，来区别接收到的文件类型，并根据相应的函数来处理。文件系统的读写其流程图如图 3-4 所示。

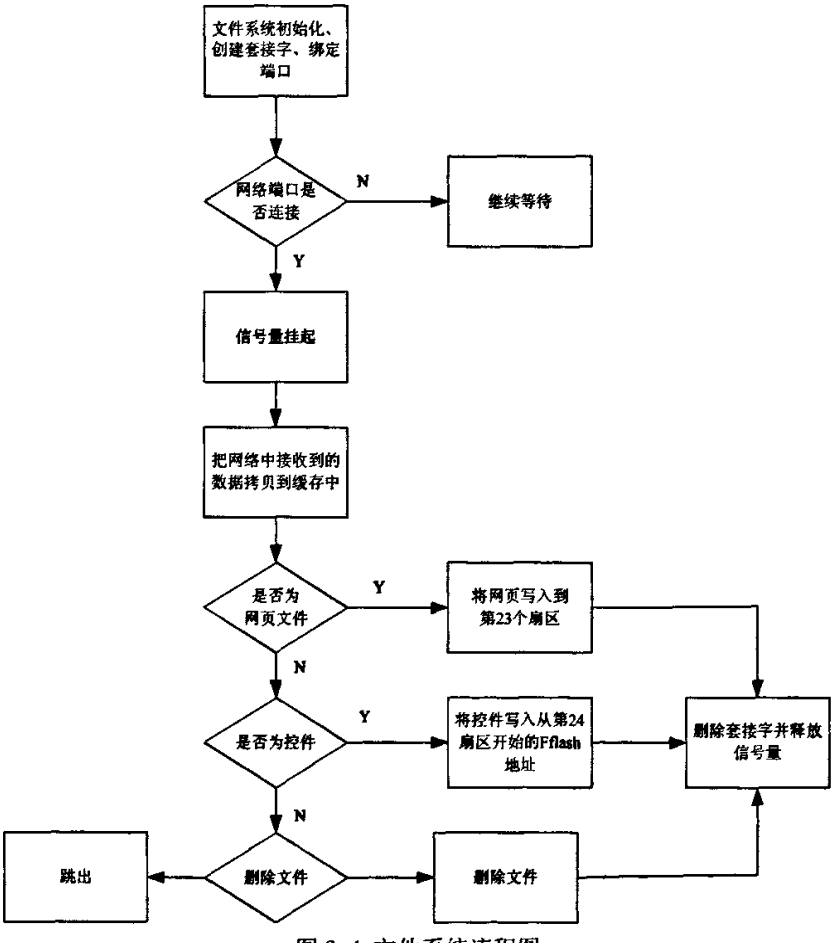

图 3-4 文件系统流程图

3.5 快速数据交互设计

远程客户端与服务器建立连接后，则必须快速的发送和接受数据，在本文中对于实时数据和历史数据交互采取的是 UDP 协议完成。

任何一个服务器，最起码要完成客户端和服务器端数据信息的交互，反映出设备的基本状况和远程监控功能。无纸记录仪本身具备一个数据库，用于存放仪表检测的数据和组态信息等。发送到客户端的数据主要包括实时数据、历史数据等，本文将其分为两种获取数据方式，主动发送和请求发送模式。实时数据需要在开机后就要发送的，故将其定为主动发送模式，即在仪表启动后，就在网络上发送实时数据；而对于历史数据和组态信息等，则需要在客户请求的情况下，才给予发送数据，故可将其定分为被动模式。现场工业仪表对于网络数据交互必须满足实时性，故我们在设计时将它独立成一个实时任务，并为它创建套接字，使这个任务和客户端直接进行交互。而与数据库的连接，是通过 R-BUS 协议来实现的，由 R-BUS 协议中的两条 API 函数 UpLoadData（请求数据库数据）和 DownLoadData（发送数据库数据）完成具体操作。下面，首先来介绍 R-BUS 协议。

3.5.1 R-BUS 协议

R-BUS 协议是无纸记录仪与 PC 机之间的通信协议，在系统设计无纸记录仪服务器和客户端之间进行通信，无纸记录仪通信模块和主模块之间通信都是按照 R-BUS 协议实现的，R-BUS 层次结构如图 3-5 所示。

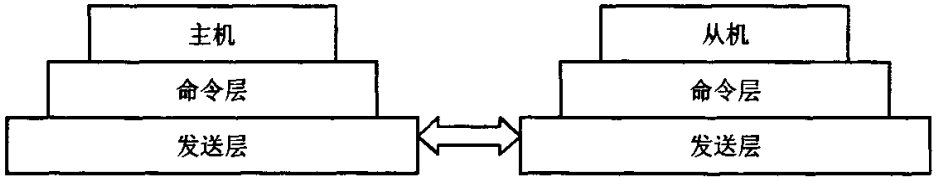

图 3-5 R-BUS 的层次结构

仪表和用户之间采用主从方式进行通讯，R-BUS 协议共分为 2 层。

发送层：负责数据传输，目前分为两种方式（RS232/485 串口通讯和 Ethernet 以太网通讯）

命令层：发送和接收数据报

主机/从机：负责数据内容的填充。在方案设计中把无纸记录仪端作为服务器，用户作为客户端。

在该层协议中有两种发送方式：

第一种发送方式是通过 RS232/485 总线进行，在这种发送方式是不可以路由的。

第二种发送方式是通过以太网实现的，在这种方式下发送是可以路由的。在该设计中采用是这种方式实现。

因协议的再继续开发中，故不是很完善，本论文在获取数据时，主要使用了如下几条命令号：

1). 10H，读取实时数据；

要想读取数据库里的实时数据，必须根据 R-BUS 协议来操作。获取实时数据，无须发送命令好，而回应的数据最大长度＝32*3+8*2+32+32+4=180，回应格式如表 3-2 所示。

表 3-2 读实时数据回应格式

名称	长度(byte)	典型值
AI 数量	1	0～20H（0 表示没有）
AI 值	N1*3	百分量(根据总体信息 03H) +质量码（包括报警信息） D2-D0:上下限报警 000 没有报警 101 上上限报警 110 下下限报警 001 上限报警 010 下限报警 D3:速率报警，上行 D4:速率报警，下行 D5:信号可疑 D6:变送器故障或断线 D7:保留
AO 数量	1	0～8（0 表示没有）
AO 值	N2*2	百分量(根据总体信息 03H)
DI 数量	1	0～20H（0 表示没有）
DI 值	N3*1	ON-1 OFF-0
DO 数量	1	0～20H（0 表示没有）
DO 值	N4*1	ON-1 OFF-0

2). 62H，读取 C3000 组态信息

本命令根据子命令号确定内容，下面详细列出子命令的内容

22 命令（读取实时数据信息）

发送数据：无

回应数据如表 3-3 所示。

表 3-3 组态信息回应格式

名称	长度(byte)	典型值
AI 数量	1	0～20H（0 表示没有）
AI 信息	N1*14	量程下限(定点两个字节＋量程上限（定点两个字节）+显示格式（）＋指定单位（9 字节）
AO 数量	1	0～8（0 表示没有）
AO 信息	N2*5	量程下限(定点两个字节＋量程上限（定点两个字节）+显示格式（）

3). 64H 读 C3000 历史数据

本命令根据子命令号确定内容，下面详细列出每个子命令的内容，读历史数据总共有两条子命令号，分别是读历史数据和续读历史数据命令号。

读命令：

发送数据：子命令号＋子命令发送数据

回应数据：子命令号＋子命令回应数据

02 命令(读历史数据)

子命令发送数据和回传数据格式分别如表 3-4 和表 3-5 所示。

表 3-4 读历史数据请求格式

名称	长度(byte)	典型值
通道号	1	0～1FH
起始时间	6	02H-0BH-1CH-03H-1CH-20H=2002-11-28 3:28:32
结束时间	6	02H-0BH-1CH-03H-1CH-20H =2002-11-28 3:28:32

表 3-5 读历史数据回应数据

名称	长度(byte)	典型值
通道号	1	0～1FH
通道类型	1	0：模拟量通道；1：开关量通道 模拟量：显示工程量、单位 开关量：显示开关状态（小于 50%为关，反之开）
显示类型	1	0~5：N 位小数，6：科学计数法（3 位有效数）
单位	8	ASCII（特殊符号单位见 C3000 特殊单位符号表）
量程上限	4	浮点型
量程下限	4	浮点型
开状态描述	8	ASCII 以 0x00 表示字符结尾，位号长度在 0－8 范围内
关状态描述	8	ASCII

		以 0x00 表示字符结尾，位号长度在 0－8 范围内
起始时间	8	02H-0BH-1CH-03H-1CH-20H'02H =02-11-28 3:28:32'2(注意 '2 代表 2/8 秒 最后 1 个字节保留)
记录间隔单位	1	0：0.125s 1：s 2：min 3：hour
记录间隔	1	间隔数量
总共需要读的数据个数	4	总数量
数据的个数	2	1 字节长时 512　2 字节长时 256
历史数据	字节数*个数	百分量(根据总体信息 19H 1AH)

子命令回应数据（完成）：错误码为 08 的错误帧

04 命令为续读命令号，它的发送数据和请求数据格式分别如表 3-6 和表 3-7 所示。

表 3-6 子命令发送数据

名称	长度(byte)	典型值
通道号	1	0～1FH
操作	1	0－继续 1－停止 2—重发

表 3-7 子命令回应数据（正常）

名称	长度(byte)	典型值
通道号	1	0～1FH
起始时间	8	02H-0BH-1CH-03H-1CH-20H'02H =02-11-28 3:28:32'2(注意 '2 代表 2/8 秒 最后 1 个字节保留)
记录间隔单位	1	0：0.125s 1：s 2：min 3：hour
记录间隔	1	间隔数量
数据的个数	2	1 字节长时最多 512　2 字节长时最多 256
历史数据	字节数*个数	百分量(根据总体信息 03H 04H)

子命令回应数据（完成）：错误码为 08 的错误帧

3.5.2 实时数据交互设计

对于现场仪表，通过实时监控仪表状态来反应现场工作情况，是极其关键重要的，如果不能实时的传输数据，造成对于仪表的不及时操作，可能影响到整个

系统，甚至带来设备损害甚至人员伤亡，造成严重经济损失。

在具体设计时，为数据库的信息交互创建任务LEDTASK。考虑到任务的重要性，把它的优先级设置为最低（32），即最后执行的任务，因为操作系统是强占式的，所以在其他高优先级任务如HTTP任务，响应时，服务器会响应中断，剥夺这个任务CPU资源，先去完成高优先级任务，结束后在来完成这个任务。R-BUS协议，规定实时数据的请求命令为：cmd_ent[10] = {0x06,0x10,0x00,0,0,0}，具体设计将其作为LEDTASK的一个局部变量。

为网络发送任务创建任务优先级31（高于数据交互任务）real_data_send它具体负责从数据库获取到数据，存放到发送缓存中，通过socket发送到网络中。但是，设计时需要注意的是，为防止出现浮点数运算，读实时数据命令10H只发送百分量数据，大小为0~30000，而不包含数据的上下限、数据单位等，这部分信息我们可以从读取组态信息命令62H补充进去，故在设计时，把这两条命令都同时操作，获取到的数据打包成同一个数据包发送。

实时数据交互设计，通过上述两个任务协同实现的。TEDTASK数据库交互任务和real_data_send网络发送任务，它们本身都是无重循环任务，为了使他们通讯，创建一个邮箱对这两个任务同步。主要设计思路如下：网络发送任务优先级高于数据库交互任务，在网络发送任务里设置一个无限时间等待的请求邮箱OSMboxPend，而数据交互任务里创建一个OSMboxPost发送邮箱。两个任务都建立有超级循环。首先，程序进入网络发送任务，并等待邮箱消息，这时这个任务挂起，这时最低优先级的数据交互任务LEDTASK运行，任务初始化，并进入任务的超级循环。它获取R-BUS命令号，并通过CPLD把命令号发送给数据库，从数据库获取数据并拷贝到邮箱内存中，最后邮箱发送OSMboxPost把邮箱信息发送给网络发送任务，这时网络发送任务real_data_send的OSMboxPend函数获取到了消息，网络发送任务运行。把邮箱内存中的数据拷贝到发送缓存，并通过套接字发送到网络中去和将消息量释放，在完成这此发送任务后，超级循环又进入邮箱挂起等待。实时操作系统的内核管理重新把CPU返回给数据交互任务，并定位到发送邮箱函数后一条代码开始运行。接下来需要设置发送时间间隔。因为无纸记录仪的存储数据是由时间间隔的，分别为0.125s，0.25s，0.5s，1s，因此在获取到数据时，则需要根据时间间隔操作，比如间隔时间一秒，而发送间隔时间是0.5，那就意味着1秒内发送两个相同的数据，这样是毫无意义的。时间间隔计算也很简单，就是用时间间隔减去所需要花费的代码时间，不过必须知道每条指令所耗费的时间。在完成时间间隔计算后，程序由于超级循环回到邮箱发送，于是又回到网络发送任务，这两个任务就是在这种循环中反复操作，实现实时数据网络交互功能。在设计这个任务时，因为这两个任务的优先级是最低的，

故不存在所谓的优先级反转问题而带来的实时性问题。具体的流程图实现图如图 3-6 所示。

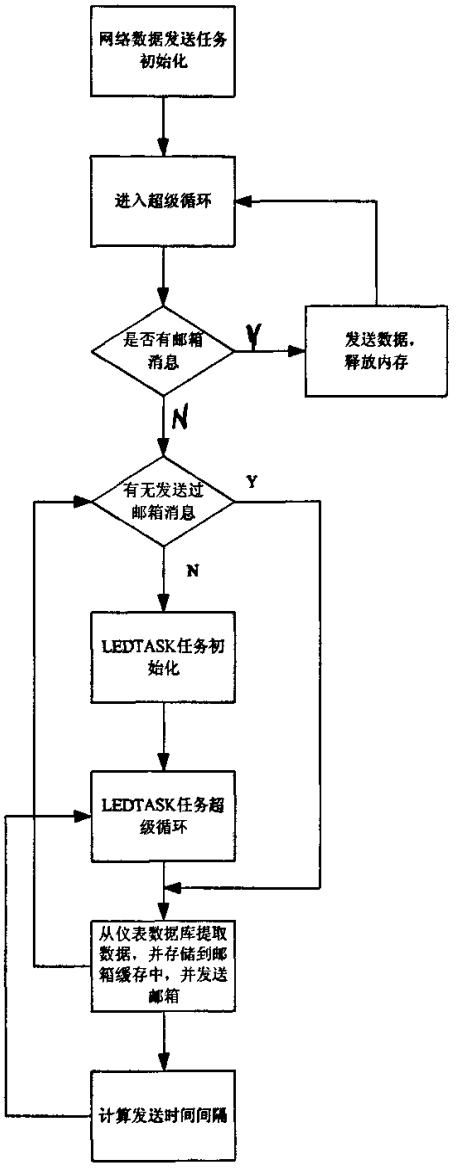

图 3-6 实时数据流程

3.5.3 历史数据交互设计

无纸记录仪可以把每一时刻的实时数据存放起来，方便以后观测以前的工作状态，观察设备状态的正常与否。而存放的数据就是历史数据，历史数据可以帮助操作人员观察现场的工作情况，与现在的实时数据比较，得出这刻之前的工业现场的工作状态。

历史数据发送定义为被动模式，它的实现和实时数据交互比较类似，数据库的数据获取完全一样，唯一的区别是它是在客户请求后才发送的。历史数据发送只有在建立 HTTP 连接后才有效。具体的原理图如图 3-7 所示。历史数据的操作涉及到三个任务：HTTP 服务器任务、TEDTASK 数据库交互任务和 real_data_send 网络发送任务。

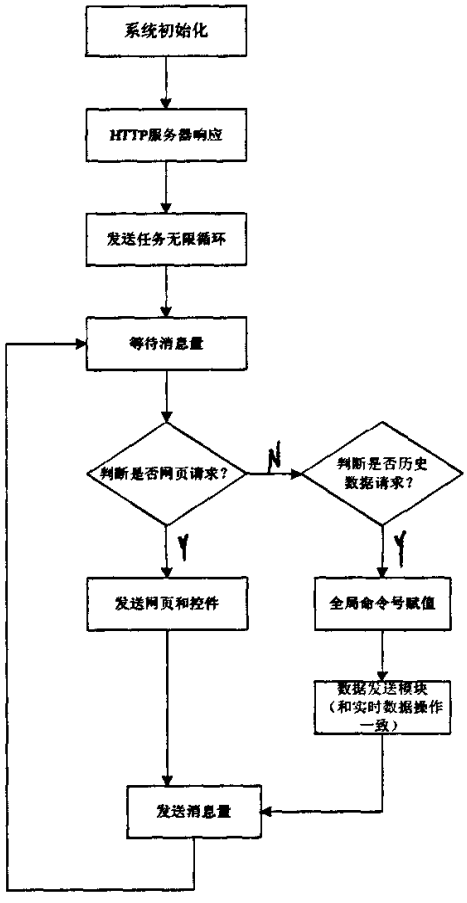

图 3-7 历史数据发送原理图

根据 R-BUS 协议，我们通过命令号只能获取某个时间段内的历史数据，但这个时间长度是任意的。历史数据在客户端的请求，可以参看 4.3.6 节。历史数据命令号在 HTTP 服务器内接收到，并拷贝到缓存中。在 R-BUS 协议中，我们可以看到两条命令号，读取历史数据和续读历史数据命令号。这是因为在获取历史数据时，数据量可能很大，从数据库获取到的数据量只有 256 个字节，因此一次不够发送，则需要通过续传命令号来继续发送，可以分别通过历史数据命令 64H 的子命令 02H 和 04H 完成。实时数据和历史数据发送不能冲突，否则会导致某个时刻的实时数据断掉。设计时，我们在实时数据发送的 2 个间隔单位时间内发送历史数据，如果一个时间发不完，则下个时刻先发送实时数据，再发送历

史数据，通过实际测试，每个历史数据包的获取时间是毫秒级的，因此可操作。接下来如是，直到历史数据发送完毕。而对于两个任务之间的操作，与实时数据的发送过程是一样的。

3.6 以太网驱动程序

本文是通过微控制器和以太网网卡实现系统接人 Internet，为此必需做好两方面的准备：(1)在硬件上，要给系统主控器一微处理器加一个网络接口；(2)在软件上，要提供相应的通信协议。当给一个系统配上一个以太网卡芯片，并提供 TCP/IP 协议和 IEEE802.3 协议时，这个系统就可以通过以太网上 Internet。本文是在 AX88796 以太网芯片上实现网络通讯。

AX88796 的驱动程序包括发送、接收和初始化三部分功能。在接收和发送数据以前要进行必需的检测和初始化。网卡的初始化主要是设置所需的寄存器状态，建立网卡发送、接收的条件。

网络接口通过两个 DMA 操作来完成数据的接收和发送，本地 DMA 完成网卡存储器和 MAC 层之间的数据传送，远程 DMA 则完成网卡缓冲区与微处理器之间的数据传送。

网络接口接收数据的过程是：首先，数据进入 MAC 层的 FIFO 队列，然后，通过本地 DMA 将数据送到接收缓冲区，再通过远程 DMA 送到微处理器中。

网络接口的发送过程刚好相反：首先微处理器通过远程 DMA 将数据送到网络接口的发送缓冲区中，然后再通过本地 DMA 将数据送到 MAC 层的 FIFO 队列进行发送。网卡初始化的一项重要任务是完成划分接收和发送缓冲区以及建立接收缓冲环。

3.7 服务器端测试

为测试服务器性能，服务器端的本地 IP 地址设置为 128.128.2.2，局域网网关为 128.128.2.1，子网掩码为 255.255.255.0，程序编译通过后，下载到通讯板的 Flash，通讯板通过以太网经 Hub 与局域网连接(或对角线把仪表和电脑直接相连)，作为局域网内的一台小型 HTTP 服务器运行。

1)．Ping 包响应速度测试数据如

2)．表 3-8 所示。

表 3-8 ping 包的响应时间表

序号	源地址发送时间 Tsource(s)	目标接收时间 Tdest(s)	响应时间 T(s)
1	18.791614	18.791993	0.000379
2	19.848744	19.849151	0.000407
3	20.871685	20.872091	0.000406
4	21.893094	21.893501	0.000407
5	22.914514	22.914921	0.000407
6	23.964756	23.965160	0.000404
7	24.987556	24.987961	0.000405
8	29.103424	29.103829	0.000405
9	30.124888	30.125294	0.000406
10	31.146515	31.146915	0.0004
11	32.196490	32.196892	0.000402
12	33.219367	33.219775	0.000408
13	34.240798	34.241205	0.000407
14	35.263079	35.263484	0.000405
15	36.313219	36.313647	0.000428
16	37.336177	37.336584	0.000407

最后得出平均 ARP 包响应时间为：

Tave=0.0004058s，即约为 400us。而且系统几个测试点时间都是比较平稳的，速度是相当快的。

3). HTTP 包功能实现和速度测试数据如表 3-9 所示。

表 3-9 TCP 响应时间数据

序号	源地址发送时间 Tsource(s)	三次握手接收时间 Tdest(s)	响应时间 T(s)
1	6.467158	6.471574	0.004416
2	37.039251	37.043640	0.004389
3	41.228716	41.233141	0.004425
4	44.809813	44.814199	0.004386
5	48.176672	48.181153	0.004481
6	56.282575	56.286991	0.004416
7	60.385125	60.389551	0.004426

平均 TCP 响应时间 T=0.00442488S，约为 4 毫秒，响应时间为毫秒级时间。

从上面数据可以看出，几个测试点时间都是比较平稳，都趋于平均时间，所以系统响应 TCP 时间也比较平稳的。

4). 两个数据包发送时间间隔的一组数据如表 3-10 所示。

表 3-10 数据包间隔发送时间

序号	发送时间 Ts(s)	时间间隔 T(s)
1	41.012700	
2	41.038245	0.025545
3	41.068862	0.030617
4	41.090580	0.021718
5	41.112715	0.022135
6	41.134404	0.021689
7	41.156260	0.021856
8	41.192892	0.036662
9	41.219821	0.021600

两个数据包发送平均间隔时间约为 0.025s，而实际 C3000 控制器的发送最小间隔为 0.125s，完全符合要求。

5). 服务器的最大并行访问量

为了可以测试系统的最大负荷量，测试可以同时几台客户机访问仪表。在客户端下载控件，当访问服务器时，目前测试过 6 台，都可以非常顺利的访问数据，可以满足多台电脑同时访问设备。

绝大多数的工厂自动化应用场合实时响应时间的要求最少为 5～10ms，而仪表的发送最小发送时间间隔要求是 0.125s，而实际测试的时间间隔为 0025s，我们可以知道这个系统设计是符合工业仪表无纸记录仪扩展它的网络功能。

3.8 本章小结

这章内容是在实时性研究的基础上开展具体软件设计。根据第二章的实时性分析，提出了整体的系统设计方案。

本文采取μC/OS-II 操作系统和 LwIP 协议栈作为构成系统实时环境的基础，本章简单介绍了它们的移植要点。除此之外，应用层软件的合理设计也很大程度上影响到实时性。本文把系统所完成的工作分为五个任务来操作，设计了各个任务的同步和之间的通讯。一个实时嵌入式 web 服务器，必须具备一个 HTTP 服务器，在 Flash 上设计一个应用于嵌入式系统的简单文件系统。而快速数据交互则很大程度上影响服务器实时性，为避免出现长任务对实时性的影响，采取两个任务分别操作。

最后，对嵌入式服务器的性能进行了测试，一个 ping 包的响应速度是 0.4ms，HTTP 建立连接的响应速度是 4ms，而工业仪表的实时响应速度要求是 6～10ms，因此，可以说这个服务器是满足仪表实时性要求的。

但一个完整网络仪表系统还需要客户端程序来对设备状态的监控，而客户端

的运行方式、图形的绘制也会影响系统的实时性，下面一章是详细介绍客户端程序的设计。

第4章 嵌入式实时web服务器的客户端实现

通过第二章对实时性的研究，我们知道要一个实时 web 服务器的性能，除了需要实时的web服务器是一个关键因素外，web文档实现方式也是决定系统能否实现实时性的另一个关键因素。

4.1 ActiveX技术

ActiveX是Microsoft提出的一组使用COM（Component Object Model，部件对象模型）使得软件部件在网络环境中进行交互的技术集。它与具体的编程语言无关。作为针对Internet应用开发的技术，ActiveX被广泛应用于web服务器以及客户端的各个方面。同时，ActiveX技术也被用于方便地创建普通的桌面应用程序。

ActiveX控件是OLE控制的更新版本[37]。控件是建立可编程部件的主要元素。ActiveX控件可以用于所有支持COM规范的容器中，或者作为Internet控件嵌入到 web 页面中。用户访问该页面时将下载该控件并自动在本地注册。利用脚本描述语言（Script）可以在控制之间以及客户与服务器之间通过设置属性（Property）、调用方法（Method）和激活事件（Event）进行通信[38]。

采用ActiveX控件实现远程监控的优点[42]：

1). ActiveX 控件发布形式简单，无须用户安装，只需要一次下载，自动注册。

2). ActiveX 控件作为独立的软构件，便于维护和升级，同时又便于软件复用。

3). ActiveX 控件以动态链接库的形式发布，相对于其他形式具有更高的安全性和代码保密性。

4). 作为控件的实现语言，Visual C++开发环境和 C++语言功能强大，可以生成高质量的可执行代码。

4.2 实时客户端设计方案

在实现仪表的远程监控中，web界面是不可或缺的部分，它用来监控仪表状态和实现客户端和服务器之间的交互。在2.3.3节，本文分析了常用的web文档实现技术，充分考虑到嵌入式 web 服务器资源的局限性，本文提出一种采取客户端资源运行的客户端设计方案，具体设计思想如下几点：

1). 客户端所采取的 web 文档技术

在 2.3.3.2 节嵌入式 web 实现技术中，本文介绍了几种 web 文档实现技术，并分析了它们的优缺点，最终采取的是 ActiveX 和嵌入 Java Script 的 HTML 网页。

客户端和仪表通过 HTTP 服务器建立连接，如果客户端是第一次请求服务器端 web 文档，服务器会将控件和网页一起上载到客户端。如果客户端不是第一次请求，则会在直接运行上次获取到的控件。

在使控件和网页通讯方面，考虑到 web 的安全性，为了在服务器与客户端建立良好的信任关系，必须为每个在 web 上使用 ActiveX 控制设置一个“代码签名”（Code Signing）。在 web 页面中使用 ActiveX 控件，还要对之进行包装，将有关的动态连接库及信息文件压缩到一个扩展名为 CAB（Cabinet）文件中，可以使用 Visual C++开发环境提供的工具包装 ActiveX 控件。在 HTML 文件中，使用 object 标签插入 ActiveX 控件，并利用 Java Script 访问该 ActiveX 控件。客户端用 Internet Explorer 浏览此页面时，可以自动解包该文件。每一个控件在其编译后，会有一个 ID 号来标志它，并使用 object 标签插入这个 ID 号。

```
<object classid = "clsid: 0D088E43-34ED-4463-8F8A-03EF82C25AAD"
id="NEW41"codebase="NEW4.ocx" width="100%" height="450" border="0">
</object>
```

出于安全性的考虑，Internet Explorer 为 ActiveX 部件的下载、初始化、是否拥有合法的代码签名以及是否允许 Java Script 等设置了不同的安全级别，用户可根据需要进行设置。控件通过 Java Script 和网页进行通信，先由 Java Script 获得 URL，然后发送给控件的属性。具体方案是将 URL 设置为通过 Get/Set 函数实现，通过 Java Script 将 URL 传进去，然后，通过 Get 函数进行处理[43] [44]。具体实现代码如下。

```
<script language="javascript">
Voidprinter.url = document.location;
</script>
```

2). 客户端实现的远程监控模式

在实现远程监控程序我们有两种设计模式：B/S（Browser/Server）和 C/S（Client/Server）。我们采取的是 B/S[35]和 C/S[36]模式的结合。最初控件是存放在服务器端，第一次请求，必须先将控件上载到客户端，这时实现的是 B/S 模式；之后，每次运行，自动调用已经存放的控件，这时实现的 C/S 模式。

上述的设计模式，既发挥了 B/S 模式的维护简易、开发容易特点，又发挥 C/S 监控界面易实现的特点。因为我们显然可知，相比较采用服务器端 25M 晶

振的微处理器来处理客户端界面程序而言，采取单位为 G 的客户端处理器处理 web 动态界面，速度上是不可比拟的。这种设计方案，可以使我们有能力设计一个界面非常复杂且功能丰富的智能仪表的监控程序，并且它完全可满足实时性的要求。

3). web 文档存储方式

在 3.5 节，我们介绍过文件系统，文件系统是用来存储和上传文件等。在本课题中，文件只有网页和控件。对于它们的存放，本文在 Flash 中的下半区开辟一块存储区，用于存放文件。

4). 控件采取的语言

本文采取 Visual C++开发环境开发客户端程序。控件的开发的可以使用多种语言和集成开发软件，如 C 语言，VB，VC 等[41]。考虑到 VC 编写界面的方便性，本论文采取的是使用 VC 来编写控件。Visual C++开发环境提供两种开发 ActiveX 控件的类库：MFC 和 ATL。使用 MFC 来开发 ActiveX 组件最大的好处是 MFC ActiveX Control Class Wizard 能帮我们产生程序代码[41]，几乎把所有能和 COM 沟通的都做完了，包含了启动的程序码、处理数据存储的程序码和显示属性的程序码等。你只需要加入相对应的事件处理函数即可，无需对 COM 或 ActiveX 组件的底层技术有太多了解。但使用 MFC 开发 ActiveX 组件需要用到另外两个体积庞大的 MFC 动态连接库文件。采用 ATL 开发 ActiveX 组件则无需 MFC 动态连接库，但不及 MFC 简单便利。综合考虑，在本文中采取的是 MFC 开发 ActiveX 控件。

5). 实时的虚拟仪表界面

针对仪表的界面，本文设计一个类似的控件界面，这里就叫它为虚拟仪表。虚拟仪表的使用，是本文的一个关键点，提供了一种解决复杂服务器端界面动态显示的方法。

同时为了提高系统的代码运行效率，控件采用双线程编程。一个是网络事件接收线程，用于完成网络数据的接收和发送；另一个是主线程即用户界面线程，主要完成除网络事件外的其它一切工作，包括数据的存储、界面的绘制等。两个线程间通过消息响应函数 SendMessage 和 Postmessage 等方式来通讯[50]。实现如图 4-1 所示。

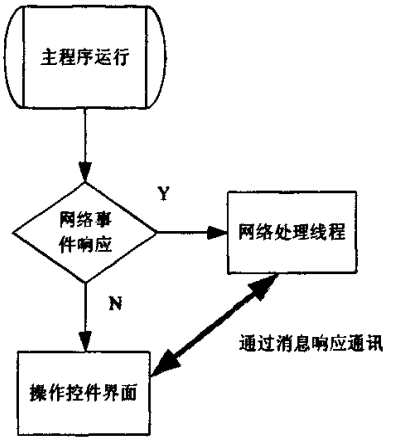

图 4-1 控件线程通讯和同步

为了提高软件的可重用性，方便软件的维护和升级工作，本方案采用面向构件的方法对网络无纸记录仪客户端进行设计和开发。面向构件的基本思想是以定义良好的、独立开发的部分即构件为基础开发系统[49]。每个构件代表一个完整的功能单元，通过适当的方式组装各个构件，以此完成系统的开发。本文根据无纸记录仪的特性，客户端设计的模块有： 1)实时监控模块 2）存储模块 3）数据通讯模块 4）历史监控模块 5）报警模块 6）网络登录模块等。

在 Visual C++开发环境中，采取 ActiveX 控件设计客户端程序。这种方案的优点主要有如下几点：

1). 服务器端只需传递一次控件，之后都只需传递动态数据，响应速度快；

2). 图形的绘制可以交由客户端，界面刷新速度快；

3). 采用 B/S 模式监控模式，具备分布式特点，随时可以进行业务处理，且开发简单，维护方便；

4). 服务器大大减轻负担；

5). 采用 Visual C++开发环境，可以缩短开发时间。

4.3 虚拟仪表界面

4.3.1 通讯模块

ActiveX 控件是通过套接字来实现网络通讯，但在此之前，必须要知道服务器和客户端的 IP 地址，在绑定之后才可以通讯。ActiveX 控件的属性实现是通过 Get/Set 函数实现的，当控件的用户需要获取属性的值时调用 Get 函数，当需要设置控件的值时，调用 Set 函数。当运行时需要计算属性的值，或是改变实际属性之前要对传入的值进行预处理以及需要实现一个只读/只写属性时，可以使用

这种方法实现。

在本课题中，采取的是异步阻塞式套接字，所谓的阻塞套接字意思是指，当试图对服务器端文件进行读写时，如果当时没有东西可读，或者暂时不可写，程序就进入等待状态，直到有东西可读或者可写为止。异步套接字，指客户端在发送请求后，不必等待服务端的回应就可以发送下一个请求。阻塞方式使客户端永远请求远程的数据，而异步方式，则使客户端程序可以继续操作下面的代码。因此，我们可知道，通过异步阻塞套接字，可以提高系统的网络收发速度，因此提高系统的实时性。

客户端设计了两种套接字，分别是基于TCP协议的流套接字的和基于UDP协议的数据报套接。相对于实时性要求高的数据的收发，我们采取的是创建UDP协议，交互速度比较快。而从客户端主动发送到服务器历史信息或组态信息等用于改变仪表状态的信息，更多考虑出的是现场的安全性，因此采取的是TCP协议。

4.3.2 存储模块

ActiveX控件可以通过消息响应使两个线程通讯，而网络事件发生后，必须要有个内存用于存储接收到的数据。为此，主线程创建了一个WM_REFURBISH_EVENT响应网络事件，具体的数据存储则在这个函数里操作。考虑到不同数据的作用不一样，设计了两种队列，一般队列和循环队列。

队列是从日常排队现象抽象出来的一种数学模型。当然数据结构中的队列远没有生活中的排队灵活。数据结构中的队列规定：数据只能从队尾进，从队首出来。已经进入队列的数据次序不能再做改变。这就叫做先进先出(FIFO)或者说后进后出(LILO)。允许插入的一端称为队尾，通常用一个称为尾指针（rear）的指针指向队尾元素，即尾指针总是指向最后被插入的元素；允许删除的一端称为队首，通常也用一个队首指针（front）指向队首元素的前一个位置（当然也可以直接指向队首元素，只是许多数据结构的书上都习惯这么定义）。

而循环队列是在队列的基础上，将存储空间的首尾相接构成一个圆环，存储在其中的队列称为循环队列[44]。

在R-BUS协议中，采取协议中不同的标记号区分数据包的类型，如，实时数据是10H，而历史数据则是64H。但是对于实时数据，因为它的数据是源源不断的读取。如果仅使用普通队列，随着长时间的监控，会导致内存消耗量积累越来越多，最终可能导致内存耗尽。对于实时模块中的实时曲线界面，每一页只有240个数据点，故可以为其设计一个循环队列，队列长度只要稍大于240个就足

够了。

一个实时数据包里包含了数据大小值，上下限，单位，通道号，时间等一些描述实时信息的量，故在保存时设计一个结构体用于存储，但对于网络直接接收到的数据大小值是四个字节表示为 0～30000 的值，需要经过处理转换为实际百分量，8 个字节的单位存储也要转换为真实的单位符号。

而对于历史数据，每次调用的是某个时间段的数据，因此则需要为其动态申请内存，一次多大，就开辟多大的内存空间，在其重新调用前就删除原先内存数据，则会节省大量内存空间，提高系统的运行速度。

而对于历史数据，因为 R-BUS 协议限定一次最多只能发送 256 个字节。如果两点的时间段大于 256，则要续传接下来的数据，故 R-BUS 协议中有两个读取历史数据子命令号，分别为 02 和 04，历史数据包也包含数据大小值，上下限，单位，通道号，时间等信息，存储历史数据的结构设计如下。

4.3.3 总态图

总貌画面显示当前所有通道的运行状况，显示实时数据或者状态，包括模拟量输入 AI、开关量输入 DI、频率量输入 FI、模拟量输出 AO、开关量输出 DO、时间比例输出 PWM、模拟量虚拟通道 VA 和开关量虚拟通道 VD。本课题已经实现的图如图 4-2 所示。

图 4-2 总态图

4.3.4 实时监控界面

对于实时数据的监控，可以由几个界面从不同方式显示，图形有数显图、棒

图、实时曲线图，它们都是用来显示实时数据的变化趋势，故可以把他们统一作为实时监控模块设计，但界面绘制是由不同类来操作的。一个实时监控模块实现分两部分：数据的提取，界面绘制。

1). 数据的提取

网络接收的数据报在数据存储这个模块已经经过处理的设计数据，因为网络中的实时数据是按一定的时间间隔 t 发送，我们在 t 时间获取一个数据。根据队列的操作，我们获取队尾的数据，即此时最新的数据。

2). 界面绘制

每一个界面都是有静态和动态刷新模块构成。在 Visual C++6.0 集成开发环境，绘图由 OnPaint 和 OnDraw 完成。OnDraw 在进行屏幕显示时是由 OnPaint 进行调用的，但是在重绘时界面会出现闪烁问题。因为当窗口由于某些原因需要重绘时，总是先用背景色将显示区清除，然后才调用 OnPaint，而背景色往往与绘图内容反差很大，这样在短时间内背景色与显示图形的交替出现，使得显示窗口不一致，看起来在闪。如果将背景刷设置成 NULL，这样无论怎样重绘图形都不会闪了。当然，这样做会使得窗口的显示乱成一团，因为重绘时没有背景色对原来绘制的图形进行清除，而又叠加上了新的图形。绘图的显示速度对闪烁的影响不是根本性的。闪烁就是反差，反差越大，闪烁越厉害。

解决的界面闪烁的常用方法有双缓冲法和逆向操作法。

双缓冲[45] [46]就是除了在屏幕上有图形进行显示以外，在内存中也有图形在绘制。我们可以把要显示的图形先在内存中绘制好，然后再一次性的将内存中的图形按照一个点地覆盖到屏幕上去（这个过程非常快，因为是非常规整的内存拷贝）。这样在内存中绘图时，随便用什么反差大的背景色进行清除都不会闪，因为看不见。当贴到屏幕上时，因为内存中最终的图形与屏幕显示图形差别很小（如果没有运动，当然就没有差别），这样看起来就不会闪。

逆向操作[47]的基本原理：要对文档进行回退重做功能，要做两方面的工作，一方面要保留删除的文档（在操作过程中，删除的文档资料一定能够保留），另一方面，系统必须能够记录进行文档操作的全过程及每个操作过程的参数。为了保留历史操作，所有数据非常占用内存空间，这就是一些系统只能进行有限次退步逆向操作的原因。需要建立如下存储机制：建一个临时文件储存数据模拟堆栈，进行一次操作时将相关操作数据入栈。回退一次将相关数据弹出栈，重做一次又依据相关数据重新恢复原有数据。它的好处是在回退和重做时只入一次栈即申请一次内存，这样操作就不会使界面出现闪烁。

棒图和实时曲线图因为都有动态刷新的曲线和方块图形，在本课题中采取的是双缓冲绘图方法。而其它几幅实时监控图如总态图、数线图只显示数据变化，

无界面闪烁，因此无须采用双缓冲方法。

数显图画面显示通道位号内容、实时数据和单位；正常情况下为绿色显示，若处于报警状态，则以实时数据红色显示，并于该通道显示区域左下方显示报警类型，数显图如图 4-3 所示。

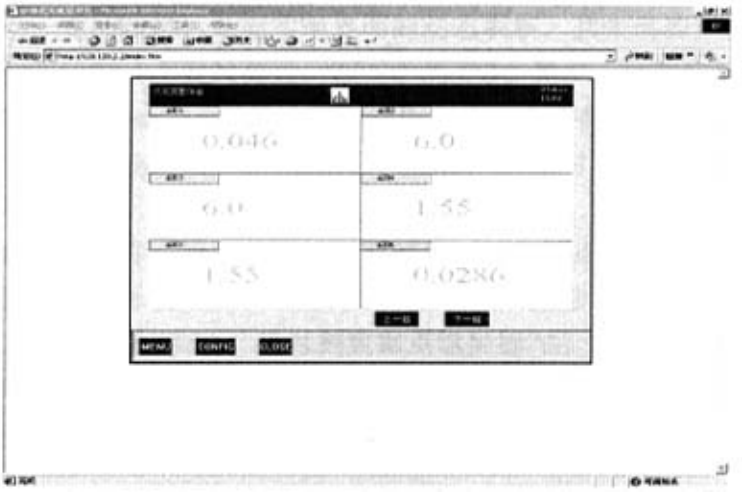

图 4-3 数显图

棒图画面显示该通道当前时刻的工程数量在整个量程中所处的位置。正常情况下为绿色显示，当该通道出现报警时，则以红色显示。棒图自上而下显示该通道信号信息、工程量量程、设置的报警点位置，当前报警类型、实时数据和单位。棒图的界面变化为上下的图形的填充来显示数据变化情况，即矩形面积的增大缩小来反映数据变化情况。实现这种界面的刷新变化原理很简单，本文通过从队列获取的数据上下限，并将当且的数据与下限的值的差值与上下限差值比较得到每次变换的矩形的面积。实现图如图 4-4 所示。

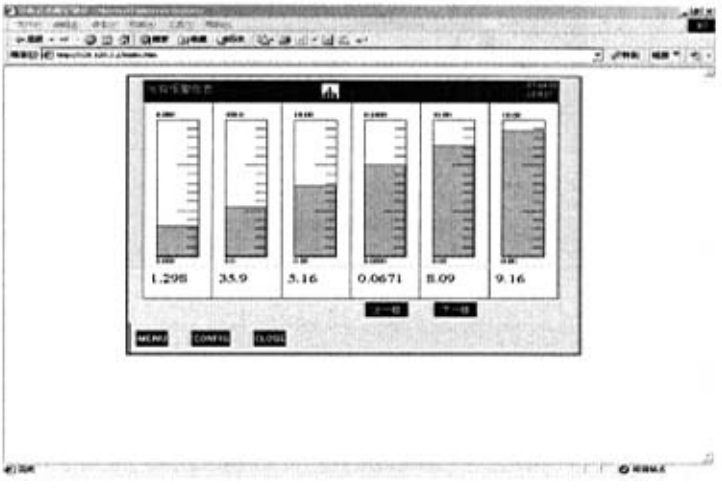

图 4-4 棒图

实时曲线监控界面是实时监控界面最形象的一副界面，图形中间左部分区域显示曲线，右边区域同时显示当前数据值。实时曲线图在实时监控模块里算是比较复杂，界面要动态显示出当前时刻数据的变化和其他几个通道之间的比较，以及数据值。中间的实时曲线和坐标是按发送数据间隔时间而变换，曲线和纵坐标从左到右移动一定的距离，从而实现界面的动态刷新。其实现的算法如下：

首先，如果绘制动态实时曲线，在 VC 的 CDC 类中，封装好了绘制直线的函数 Moveto 和 Lineto[51]，因此只需要获取到每个实时数据点的信息 RealDigit（X_i，Y_i，T_i），其中 X_i 为某时刻数据点横坐标，Y_i 某时刻数据点纵坐标，T_i 某时刻数据点时间值。

根据界面已知条件，历史曲线的框图的坐标轴左端点值 X_L、右端点值 X_R，曲线最右边界面横坐标值 X_E，一个界面共有 240 个数据点；曲线的纵坐标最上面点位置 Y_T、最低点坐标 Y_B、某时刻数据点纵坐标 Y_i；第一个获取到的数据点时间为起始时间点 T_0，某时刻数据点时间值 T_i 在前面的数据存储里，我们从服务器可以获取到上下限值 D_{MAX}，D_{MIN}，和某时刻数据值 D_i。实现见公式(4-1)～(4-2)。

$$X_E=(X_R-X_L)*4/5 \tag{4-3}$$

$$X_i=X_E+(T_i-T_0)*X_E/240 \tag{4-4}$$

$$Y_i=\frac{19}{20}(Y_B-Y_T)-\frac{4}{5}(Y_B-Y_T)(D_i-D_{MIN})/(D_{MAX}-D_{MIN}) \tag{4-5}$$

实现图如图 4-5 所示。

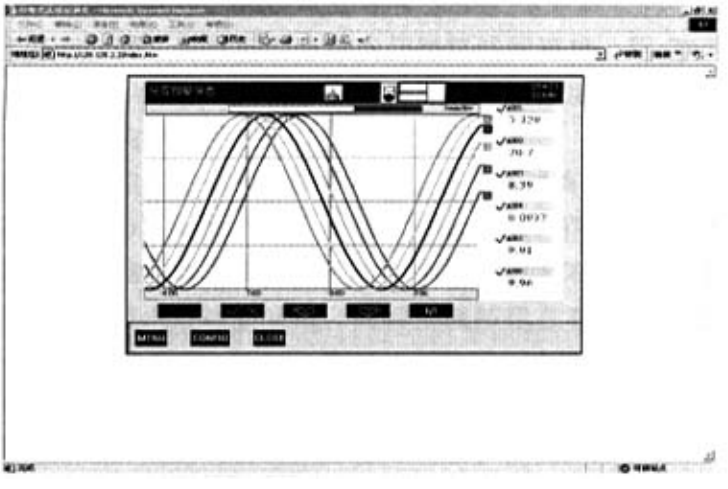

图 4-5 实时曲线图

4.3.5 历史监控模块

相对于实时监控界面[48]，历史监控模块主要反映一段时间仪表运行状态，可以为客户了解此刻之前的现场状况。历史监控模块设计与实时曲线曲线比较类似，也包括两方面：数据的提取和界面的绘制。

1). 数据提取

从存储模块可以知道，一个历史数据信息是非常全面的，包含数据上下限、时间值等。队列的数据存储是先入先出（FIFO）方式，因此我们获取到最新的数据也是采取从队尾获取。相对于实时数据的循环队列，VC 中是没有类直接实现的，而只有一个一类 CArray 来实现一般的队列，这个类集成了很多的函数供调用。

2). 历史界面绘制

历史界面包括一个监控曲线和请求发送界面两块组成。对于历史数据是被动请求的，不像是由实时曲线那样由服务器主动请求的。已经在 3.4.1 节介绍了历史数据读取的命令格式，因此在这个请求界面必须封装成请求命令格式一致的数据格式，如下：

Char Historycommand[] ={7，8，6，100，20，2，channel，year，month，day，hour，minute，second，lastyear，lastmonth，lastday，lasthour，lastminute，lastsecond}；

其中前面两位是历史曲线读取的标志位，100 是与数据库读取的一个命令号，20 是这个数组的长度，2 是子命令号，而不包括续传命令号，考虑的是减少网络发送时间和提高实时性，续传命令号 4 直接在服务器处实现。接下来的 12 是起始时间和终止时间。并通过消息响应与网络线程连接，用 TCP 协议将这个命令数组发送给服务器，HTTP 服务器接收这个命令数组，并交给给定函数进行处理，最后回传响应的数据。

历史界面是所有监控界面中最复杂的，它不仅需要有发送命令数据的独立对话框，而且在主界面的界面操作包含曲线的左右移动、界面的左右翻页、最前面界面和最后界面的跳翻、红色定位线的左右移动、以及右侧数据的显示等操作，而这些界面操作主要是通过按钮实现。目前实现按钮左右移动、左右翻页四个，基本上实现了页面的翻页、红色定位线的左右移动、以及在红色靠边时整个界面移动等一系列操作。因为历史曲线是手动操作的，界面的刷新不是很快，闪烁不明显，故无需用双缓冲绘图。实现如图 4-6 所示。

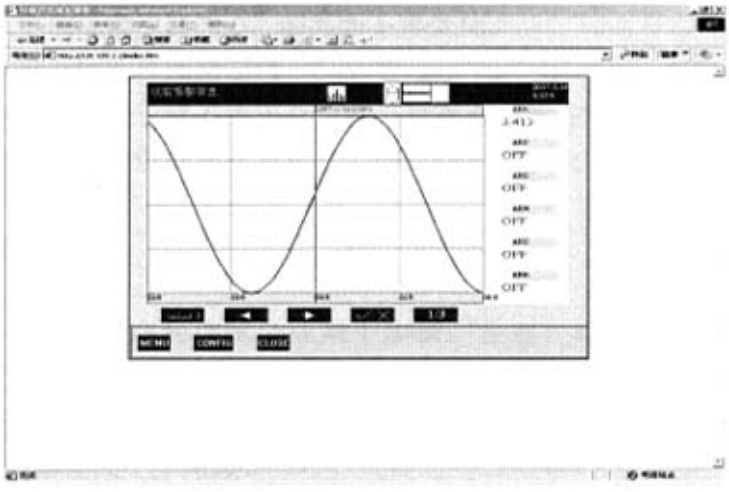

图 4-6 历史曲线图

界面可以显示某一时刻的历史曲线的信息。界面信息包含一条红色定位线、坐标和历史曲线，下面具体介绍翻页和移动按钮对它们的影响。

首先翻页按钮，对于界面的操作，可以分成几种情况。

1). 起始界面

a). 起始页

如果数据总个数少于 120 个，界面不能翻页，因此坐标和曲线都不会变化。

b). 翻回的第一页

当从后面几页翻回到第一页时，绘制曲线起始点位置变为左边点。即界面绘制的起始点索引值为 0，红色定位线的索引值变化为未翻页前红色定位线的索引值于当前页的最左边点的索引值的差值。

2). 中间页

中间页的翻页，红色定位线的索引值为增大或则减小 240。

3). 最后一页

翻到最后一页时，则将索引值最大的值作为界面最右边的点的索引值，而红色定位线的索引值化为未翻页前的界面最右边点的索引值与红色索引值的差。

左右移动对曲线显示影响，可以分为如下几种。

1). 当红色定位先在界面中间

这时，点击左右按钮，只会移动红色定位线，改变红色索引值的大小，而不会改变界面。

2). 红色定位线在界面最左边

a). 红色定位线的索引值大于 0，如果继续点击左移按钮，会使界面继续变化，但红色定位线位置不变

b). 红色定位线索引值为 0，则界面不再变化。

3). 红色定位线在界面最右边

a). 红色定位线索引值不大于总个数，则界面继续左移动；

b). 红色定位线索引值等于总格数，则界面不再变化。

4.4 客户端测试

本文对客户端程序的调试分成两个步骤：在客户端的在线调试和下载到 Flash 的测试。在线调试时，服务器端程序必须运行，当前的客户端的 IP 地址设置为 128.128.2.130，局域网网关为 128.128.2.1，子网掩码为 255.255.255.0，保证 IP 地址和服务器在同一个网段。

VC 集成开发环境拥有极强的在线调试能力。VC 调试特点：设置断点，跟踪程序段的执行；进入函数调用或跳过函数调用；设置监视窗口观察变量和表达式。

本文是在控件的基础上对程序进行的调试，故需要一个容器，因此要改变调试环境，有两个选择，一个是专门的控件调试容器和 IE 浏览器，需要注意的这两容器的栈大小是不一样的，最终必须要在 IE 里通过跳调试。在调试过程中，因为是按照面向构件的思想设计的，不同的界面有不同的类，方便我们按模块来调试。而调试时，必须是建立在服务器端运行的前提下，这时客户端才会响应服务器的网络事件。在客户端的调试可以方便用户直接修改代码，对于错误进行纠正。

在完成上步后，则必须将控件下载到服务器 Flash。在客户端请求后，控件下载到服务器客户端程序的最后实现界面如图 4-2～图 4-6 所示。

4.5 本章小结

设计实时 web 服务器，客户端也是必须考虑的关键环节。为了利用客户端资源操作动态界面和使得系统方便维护，本文采取 ActiveX 控件和带有 Java Applet 的 HTML 网页的文档技术。在监控模式方式上采用 B/S 和 C/S 混合监控模式，极大提高了系统的实时性。

本文采用 Visual C++来编程动态界面，可以容易实现界面绘制。在网络编程上，采用了异步阻塞式套接字设计通讯方式，可以大大提高网络收发速度。为提高代码运行效率，采取双线程实现。同时，为了提高代码的易读性和理解，界面的绘制采取模块化设计，不同界面采用独立的类实现；并为防止出现界面闪烁问

题，采取双缓冲绘图方法。在这种思想上，本文对虚拟仪表界面划分为通讯模块、存储模块、动态界面模块、实时监控模块、历史监控模块等。在测试中，我们可以得到这些界面的图形如图 4-2～图 4-6 所示，而且它们的动态显示几乎可以保持和真实仪表的一致性。因此，本文可以证实这种设计方案可以很好实现实时性的监控界面。

第5章 结论与展望

5.1 结论

本课题的研究的嵌入式web的实时性以及实时web服务器的实际应用设计。在本课题中，基于无纸记录仪的工业实时性的要求，考虑如何具体设计实时的嵌入式web服务器。然而要实现一个实时的可靠的嵌入式web服务器，必须要从多方面考虑。本文从具体的硬件、软件、以及所应用的网络来分析实时性。

在本系统中，采用AT9140008微处理器为核心和AX88796以太网芯片实现网络通讯的硬件开发平台。主要完成了如下工作：

1). 本文从硬件、软件、网络传输机制，深刻研究了它们对系统的实时性影响。

2). 在实时性研究的基础上，本文采取μC/OS-II实时操作系统和LwIP协议栈设计了一个实时软件开发环境。

3). 在应用层上，本系统采取多任务编程，合理划分任务，设计了HTTP服务器和文件系统、以及与实时性相关的快速数据交互任务。

4). 动态web文档实现没有采取传统的CGI等一些应用程序接口，而是使用ActiveX控件嵌入到Java Script脚本的HTML网页中，下载到客户端使用客户端资源运行，故而减轻服务器绘制工作以及无须传递代码量大的网页，提高实时性。

5). 客户端监控程序，针对无纸记录仪的特点，采取ActiveX控件利用面向构件的思想编写界面模块。在通讯模块中，采取异步阻塞方式，大大提高应用程序的性能。

在完成系统的软件开发工作后，对服务器进行测试。请求一个ping包的响应时间平均约为400us，建立HTTP连接的响应时间约为4ms；仪表的最小间隔发送时间为0.125s，实际测得的发送数据间隔为0.025s。通过实际测试的数据和仪表需要的时间对比，我们可验证这个服务器是满足工业仪表实时性要求的。在客户端采取异步阻塞通讯方式和多线程设计，web界面在使用客户端资源操作时性能非常高效。最后通过观察界面，可以看到客户端界面的变化几乎和仪表的界面保持同步。

5.2 课题展望

由于时间的限制，虽然完成嵌入式web服务器的开发以及在无纸记录仪中

的应用，基本上达到了仪表实时性的要求，但仍然有很多需要研究和完善的地方。

1). 嵌入式系统具备产品特征，它对系统的功耗、体积、成本、可靠性、速度、处理能力、电磁兼容等方面均有一定的要求。由于本系统具有一定的实验性质，所以在本设计中未能全面考虑这些问题。所以还需要对系统进一步的测试，观察系统的资源耗费情况，尽可能提高系统的稳定性和可靠性。

2). 客户端界面设计上还需要进一步完善，目前只是完成了一些基本界面的绘制，而且由于无纸记录仪的R-BUS协议目前不完善，导致部分界面的功能无法实现，若要完成商业用途，则需要在这些方面进一步研究和设计开发。

3). 无线网络是刚刚兴起的通讯技术，有着其特有的优点，本课题设计的仪表只能在以太网中运用，没有具备无线通讯功能。在这方面可以考虑增加无线报警功能，将仪表和手机等方便携带的通讯工具建立连接，方便时刻监控工业现场状况。

伴随这现代网络技术芯片技术的迅猛发展，嵌入式 web 服务器的稳定性和可靠性都将会有快速的提高，使其不仅应用于高要求的工业现场，而且也能惠及普通家庭，为智能家庭设备提供服务。

参考文献

[1].朱珍民，隋雪青，段斌编. 嵌入式实时操作系统及其应用开发[M]. 北京：北京邮电大学出版社，2006

[2].蒋智勇. 基于嵌入式实时操作系统 Web 服务器的研究与实现 [学位论文]. 交通信息工程及控制研究所，2004

[3].张曦煌，柴志雷. 嵌入式 Web 服务器中 CGI 的特点及实现[J]. 小型微型计算机系统，2003，10：48-53

[4].张鑫龙，王平，孙攀. EPA 通信协议栈的设计与实现[J]. 重庆邮电学院学报，2006，18(4)：512-515

[5].IEEE 1588, Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. IEEE 1588-2002 Standard, November 2002

[6]. Jean J.Labrosse 编. 嵌入式实时操作系统μC/OS-II（第 2 版）[M]. 北京：北京航空航天大学出版社，2003

[7]. 电子科技大学软件研究中心. 嵌入式操作系统[M]. www.taikebj.com.cn，1999

[8].实时的工业以太网 Ethernet Powerlink. http://designnews.com.cn，2005

[9].马道钧，张敬怀. 微处理器指令集的发展与研究[J]. 中国科技成果，2005，15：32-34

[10].刘云生，方丹. 嵌入式实时操作系统的内核抢占机制研究[J]. 计算机工程，2005，31(24)：80-82

[11].任哲编. 嵌入式实时操作系统μC/OS-II 原理及应用[M]. 北京：北京航天航空大学出版社，2005

[12].王海燕. μC/OS-II 任务调度的改进与实现[J]. 现代电子技术，2006，13(2)：25-31

[13].程璞，李奇，倪超. 基于嵌入式系统的家庭智能网络终端设计[J]. 微计算机信息，2006，13(3)：200-205

[14].JEAN J.LABROSSE uC/OS-II 代码公开的实时嵌入式操作系统[M]. 中国电力出版社，2002，15(2)：115-118

[15].AdamDunkels. Design and Implementation of the lwIP TCP/IP Stack. 2001

[16].李鸿强、苗长云. lwIP 移植到 μC/OS-II 中的实现[J]. 天津工业大学学报，2006，25(4)：38-40

[17]黄英. 基于嵌入式 linux 的 Web Server [学位论文]. 信息与通信工程，2002

[18].余亚东. 基于嵌入式 WEB 服务器的远程监控系统的研究与实现 [学位论文].信号与信息处理，2003

[19].李新，朱永宣，张德琨. 一种支持实时业务的新型以太网协议的研究[J]. 中国数据通信，2003，15(3)：80-82

[20].陈昕光，许勇. 以太网应用于工业控制系统的实时性研究[J]. 自动化仪表，2005，26(8)：10-14

[21].黄文君，金建祥. 基于 EPA 工业以太网的现场控制器研制[J]. 仪器仪表学报，2006，27(8)：949-953

[22].宣荣喜. 基于 AT91M40800 微控制器的嵌入式 Web 技术[J]. 西安电子科技大学学报，2006，23(3)：482-486

[23].IEEE1149.IEEE standard test access port andboundary-scan architecture[S]. 2001

[24].阳富民，柯滔，涂刚. 基于 JTAG 技术的嵌入式交叉调试软件[J]. 计算机工程与设计，2005，26(10)：392-397

[25].马忠梅，徐英慧，叶勇建，林明编. AT91 系列 ARM 微控制器结构与开发[M].北京：北京航空航天大学出版社，2003

[26].罗军宏，谢余强，舒辉，张有为. LwIP 在 u C/OS II 操作系统中的实现[J]. 微计算机信息，2005，20：33-38

[27].韩光洁，王金东. 基于 Web 管理的 Embedded Web Server 研究与实现[J]. 东北大学学报：自然科学版，2002，23(11)：115-159

[28].刘敏. 嵌入式 web server 中 TCP/IP 的设计与实现[J]. 计算机工程，2004，31(21)：321-326

[29].钟忻慕，慕春棣. 基于闪存的文件系统的实现[J]. 计算机工程与应用，2003，21(14)：25-29

[30].阎航. Flash 文件系统研究综述[J]. 现代计算机，2006，31(24)：64-68

[31].乔峰，林平分，Yu John. 基于 Vxworks 的 Flash 文件系统[J]. 北京工业大学学报，2005，31(5)：352-355

[32].金晶，浦汉来，朱莉. 基于 FLASH 存储器的嵌入式文件系统的设计与实现[J].电子器件，2005，26(2)：162-166

[33].谭小安，侯成刚，徐光华. 一种嵌入式 Flash 文件系统的设计与实现[J]. 仪器仪表用户，2006，16(2)：113-119

[34].曾祥辉. 嵌入式工业以太网络接口的开发与应用 [学位论文]. 先进控制研究所，2006

[35].王磊，史烈，陈小平. 基于 B/S 结构的嵌入式超文本编辑器的设计与实现[J].计算机工程，2002，19(1)：31-35

[36].施莹. 嵌入式 WebServer 及在 SNMP 网管系统中的设计实现[J]. 程度信息工程学院报，2005，23(12)：361-364

[37].李彦华，刘仁芬. 基于 ActiveX 实现的 web 页面电子图章的插入[J]. 科技资讯，2006，31：16-19

[38].李建霞，陈鲁汉. ActiveX 与其它应用程序的交互使用方法[J]. 机电产品开发与创新，2006，16：94-98

[39].费情. 实时通讯以太网研究[J]. 南京大学学报，2003，23(2)：335-339

[40].陈磊，冯冬芹，金建祥，褚健. 以太网在工业应用中的实时特性研究[J]. 浙江大学学报，2004，38(6)：361-365

[41].叶金杰，孙宝强. 基于 ActiveX 控件建构监控和数据采集系统[J]. 自动化与仪表，2001，13(4)：112-116

[42].姚勇 陈金勇. 用 MFC 制作 ActiveX 控件[J]. 无线电通信技术，2002，19(8)：64-69

[43].黄学良，温传新，陈锦桂. VC 环境下数控系统与上位机的通讯实现[J]. 制造业自动化，2004，24(12)：52-56

[44].李勇，曹文亮，王兵树，崔凝，张冀. 电厂厂级监控信息系统中的数据通信实现[J]. 电力技术，2004，18(4)：16-19

[45].黄峰，单家方，匡光力. 基于 RTLinux 的低杂波数据采集系统[J]. 工业仪表与自动化装置，2005，6：59-62

[46].侯金波. 基于 VC++6.0 开发的多功能数字显示程序[J]. 仪器仪表标准化与计量，2005，6(3)：69-73

[47].任易. VC6.0 实现逆向操作并防止界面闪烁[J]. 自动化仪表，2004，25(16)：23-27

[48].姬文亮，姜平，朱海荣. 基于 VC++6.0 的工业监控软件中历史曲线控件的开发[J]. 南通大学学报，2005，19(5)：33-36

[49].童庆勇，王盼卿. 构件化软件体系结构研究[J]. 科学技术与工程，2006，9：123-128

[50].陈作柄，熊涛，李欣. 工控网中基于 Linux 的嵌入式 HTTP 服务器设计[J]. 单片机与嵌入式系统应用，2004，6(3)：22-27

[51].孙霞. 基于 Java 的高效多线程 HTTP 服务器的研究及实现[J]. 微型电脑应用，2004，14(4)：31-36

致谢

在攻读硕士的两年时间里，我的学习和科研开发得到许多老师和同学关心和帮助，在此对所有的老师和同学表示由衷的感谢！

首先感谢我的导师冯冬芹教授，在我攻读士学位期间他在学习、工作上给予我的悉心指导和亲切关怀。冯老师广博的学识、敏锐的思维、实事求是的治学态度、耐心和蔼的教导，给我留下了深刻的印象，使我终生受益。在毕业设计中，冯老师更是在白忙之中抽出时间提供我帮助，指导我的研究方向和论文的写作方面，这为我圆满完成学业奠定了坚实的基础。在此，谨向冯老师表示诚挚的敬意和感谢，祝福冯老师工作顺利、事业顺心，创造更多的辉煌。

同时还要感谢我的导师金建祥研究员，尽管金老师忙于集团公司的管理，但金老师还是抽出他宝贵的时间，给予我不少帮助与指导。他的严谨求实的治学态度和开拓创新的精神也使我受益非浅。在此，向金老师表示由衷的谢意。在完成毕业设计时，还有黄文君老师、胡协和老师、毛维杰老师也抽出他们宝贵的时间，给予我很多的帮助和指导。

在我刚开始课题研究时，马浩师兄给予我很大的帮助和指导，使我的毕业设计能够顺利完成，在此对马浩师兄表示深深的敬意。

同样要感谢课题组的所有同学蔡庆荣、陈布亮、方瑜、金振都、全剑敏、羊海龙、遇彬、章涵。很高兴有机会跟他们在一起共事，成功的路上有他们相伴，我倍感荣幸。在即将离开我们这个充满生机、蓬勃向上发展的实验室之时，我向他们表达我的祝福，祝福他们都能够有个美好的未来！我还得到了浙江中控技术有限公司的施妙华等众多员工的支持和帮助。对他们表示衷心的感谢。

两年的学习时光匆匆而逝，回想起老师和实验室同学的关心和帮助，仍然历历在目，然而无以为报，谨在此表示最真挚的谢意！

郑一强

2007年6月于浙大求是园

嵌入式web服务器的实时性研究

作者：郑一强
学位授予单位：浙江大学