基于 uC/OS-II 的嵌入式无线传感器网络平台的设计

王 峰[1] 王 健[2] 郭忠文[3]
(鲁东大学数学与信息学院 烟台 264025)[1] (曲阜师范大学 曲阜 273165)[2]
(中国海洋大学信息科学与工程学院 青岛 266100)[3]

摘 要 在对无线传感器网络体系结构、传感器节点、网关(Sink)节点的特点和功能分析的基础上,给出了传感器节点和网关(Sink)节点的软硬件设计与实现方案。在软件设计方面深入研究了嵌入式实时操作系统 uC/OS-II 的特点和内核结构,实现了 uC/OS-II 在 Nios 处理器上的移植。同时介绍了小型 TCP/IP 协议栈 LwIP 以及 LwIP 在 uC/OS-II 上的实现。

关键词 无线传感器网络,嵌入式系统,实时操作系统,Nios,移植,LwIP
中图法分类号 TP391 **文献标识码** B

Design of Embedded Wireless Sensor Netowrk Platform Based on uC/OS-II

WANG Feng[1] WANG Jian[2] GUO Zhong-wen[3]
(School of Math and Information, Ludong University, Yantai 264025, China)[1]
(Qufu Normal University, Qufu 273165, China)[2]
(College of Information Science and Engineering, Ocean University of China, Qingdao 266100, China)[3]

Abstract Design and realization scheme of sensor-node and sink-node in soft and hard were proposed based on the introduction of its system structure, characteirstics and function. This papers deeply researched characteirstics and kernel of the embedded RTOS uC/OS-II, and transported the uC/OS-II successfully to the controller Nios. The mini type TCP/IP protocol stack LwIP Realization of LwIP on uC/OS-II was introduced.

Keywords Wireless sensor network, Embedded OS, Realtime OS, Nios, Transplant, LwIP

1 引言

随着计算机运算速度、数据处理能力的迅猛发展和存储器容量的飞速增长,数据传输的效率随着网络的发展而日新月异,信息的获取和采集的手段却有待进一步的提高。因此,融合了传感器技术、嵌入式计算技术分布式信息处理技术和无线通信技术的无线传感器网络 WSN(wireless sensor networks)应运而生[1],它可实现对网络分布区域内各监测对象的数据采集量化、处理融合和传输,使得用户可以详尽及时地掌握被监测区域内感兴趣的信息和事件。无线传感器网络以其独有的特点无论是在军事还是民用领域都有着广阔的应用前景[2]。近年来,无线传感器网络技术的研究十分热门,国际上在该领域的研究工作中也取得了许多成果,WSN 技术也在逐渐产品化,慢慢走进我们的生活。但是,作为一项未来对人类生活产生重大影响的前沿科技,WSN 的研究应用可以说正处于起步阶段,并且在低能耗、系统的实时性、低成本、安全、抗干扰等方面依然需要很多进一步的研究。这是一个从理论到实践都很有意义的研究课题。

2 无线传感器网络

2.1 无线传感器网络体系结构

传感器网络系统通常包括传感器节点、汇聚节点和管理节点。大量传感器节点随机部署在监测区域内部或附近,能够通过自组织的方式构成网络。传感器节点监测的数据沿着其他传感器节点逐跳地进行传输,经过多跳后路由到汇聚节点,最后通过互联网或卫星到达管理节点。用户通过管理节点对传感器网络进行配置和管理,发布监测任务以及收集监测数据[3]。无线传感器网络体系结构如图 1 所示。

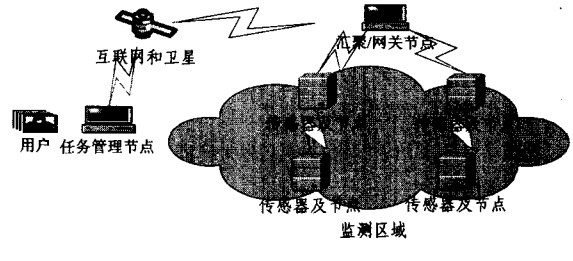

图 1 无线传感器网络体系结构

到稿日期:2009-07-16 返修日期:2009-10-09 本文受国家自然科学基金(项目批准号:60873248)资助。

王 峰(1965—),男,硕士,高级实验师,主要研究方向为嵌入式系统、无线传感器网络技术,E-mail:ytldwf@163.com;王 健(1983—),男,硕士,助教,主要研究方向为信息系统、软件程序设计等;郭忠文(1965—),男,博士,教授,博士生导师,主要研究方向为计算机网络、海洋信息分布式处理技术等。

从网络功能上看，每个传感器节点兼顾传统网络节点的终端和路由器双重功能，除了进行本地信息收集和数据处理外，还要对其他节点转发来的数据进行存储、管理和融合等处理，同时与其他节点协作完成一些特定任务。汇聚/网关节点的处理能力、存储能力和通信能力相对比较强，它连接传感器网络与 Internet 等外部网络，实现两种协议栈之间的通信协议转换，同时发布管理节点的监测任务，并把收集的数据转发到外部网络上。网关节点既可以是一个具有增强功能的传感器节点，有足够的能量供给和更多的内存与计算机资源，也可以是没有监测功能仅带有无线通信接口和网络接口的特殊网关设备。

2.2 传感器节点构成

传感器节点由传感器模块、处理器模块、无线通信模块和能量供应模块 4 部分组成，如图 2 所示。传感器模块负责监测区域信息的采集和信号的调理；处理器模块负责控制整个传感器节点的操作，存储和处理本身采集的数据以及其他节点发来的数据，处理器通常选用嵌入式 CPU；无线通信模块负责与其他传感器节点进行无线通信，交换控制信息和收发采集数据；能量模块为传感器节点提供运行所需要的能量，通常采用电池供电[4]。某些传感器节点可能还包括定位模块，外部存储器等。传感器节点首先采集与环境相关的数据，并对这些数据进行简单处理后传送到网关节点。

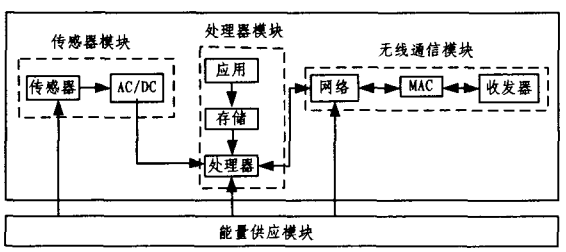

图 2 传感器节点构成

2.3 传感器网关节点构成

网关节点的功能是通过无线模块与传感器节点进行通信，发送命令给节点并接收来自节点的数据，并通过串口将数据发送给 PC 端监控软件。网关节点硬件主要包括中央处理单元、存储单元、射频收发模块、GPRS 通信模块和数据采集模块组成，如图 3 所示。网关节点用于组合从各个传感器节点得到的数据以及负责与外界的通信，网关节点对这些数据融合处理后，用 TCP/IP 模块封装成 IP 数据包，通过移动通信模块无线发送数据到检测中心服务器，对数据进行相关的管理，该节点基于嵌入式系统。

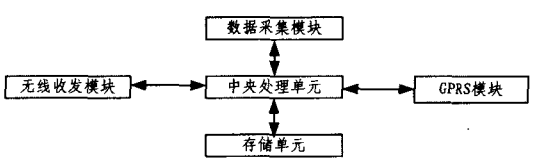

图 3 网关节点构成

3 系统硬件设计

在本文介绍的系统架构中，主要需要实现的是传感器节点和网关的硬件平台，下面介绍这两个平台的硬件设计。

3.1 传感器节点的硬件设计

传感器节点的功能是采集环境数据，并将数据发送给各感器节点组的网关。

传感器节点的传感器模块如图 4 所示。

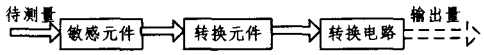

图 4 传感器模块

它由敏感元件、转换元件、转换电路等 3 部分组成。敏感元件是感受被测量，并输出与被测量成特定关系的某一物理量的元件；敏感元件的输出是转换元件的输入，它把某物理量输入转换成某电参数量；转换电路将转换元件输出的某些电参数量进行放大整形，并把该电参量模拟信号转换成数字信号输出。该数字信号与原始的被测量之间存在着对应关系，这种对应关系往往可以通过查找表或传递函数来进行描述。

传感器节点的处理器模块采用了 Atmel 公司 ATmega128L 为核心的处理器模块的 AVR 单片机，结构如图 5 所示。

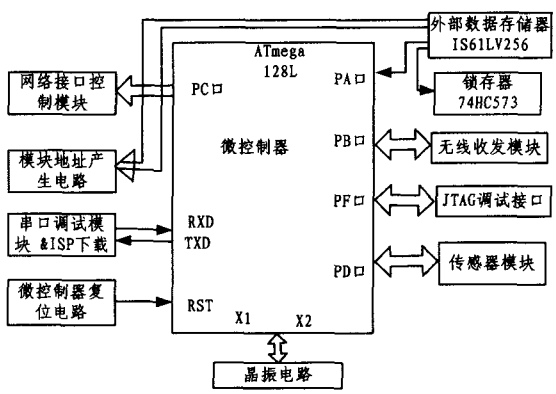

图 5 处理器模块结构图

该款处理器是 Atmel 公司推出的 51 兼容的高性能、低功耗处理器，其主要特点如下[5]：

(1)处理能力较强

该处理器采用 RISC 和 Harvard 结构，具有独立的数据和程序总线。程序存储器的指令通过一级流水线运行，CPU 在执行一条指令的同时读取下一条指令，这样就实现了指令的单时钟周期运行，大多数指令在一个周期内完成。在外接 8MHz 晶体时，其 CPU 的处理能力接近 8MIPS(百万条指令每秒)。

(2)丰富的片上资源和接口

该处理器存储空间很大，内置 128k 字节的程序存储器 Flash，4k 字节内部数据存储器 SRAM，4k 字节非易失性 EEROM。一共有 64 个管脚，56 个 I/O 口，内置 2 个 8 位计数器，2 个 16 位计数器，有 2 个 UART，有 SPI，I2C 接口和内置的 8 路 10 位 AD 转换器等资源。

(3)拥有多种电源管理模式

该处理器支持 6 种睡眠模式，分别为空闲模式、ADC 噪声抑制模式、掉电模式、省电模式和 Standby 模式。睡眠模式可以使应用程序关闭微处理器中没有使用的模块，从而降低功耗。AVR 微处理器具有不同的睡眠模式，允许用户根据自己的应用要求实施剪裁，从而可以尽量降低功耗。

传感器节点的通信模块的功能是由 CC2420 射频收发器来实现的。CC2420 是 ChipconAS 公司推出的首款符合 2.4GHz IEEE802.15.4 标准的射频收发器，CC2420 的选择

性和敏感性指数超过了 IEEE802.15.4 标准的要求，可确保短距离通信的有效性和可靠性。利用此芯片开发的无线通信设备支持数据传输率高达 250kbps，可以实现多点对多点的快速组网。CC2420 的主要性能参数如下：

(1)工作频带范围：2.400～2.4835GHz；采用 IEEE802.15.4 规范要求的直接序列扩频方式；

(2) 内部集成有 VCO，LNA，PA 以及电源整流器，采用低电压供电(2.1～3.6V)；输出功率编程可控；

(3)IEEE802.15.4MAC 层硬件可支持自动帧格式生成、同步插入与检测、16bitCRC 校验、电源检测、完全自动 AMC 层安全保护；其 MAC 层的帧格式为：头帧＋数据帧＋校验帧，PHY 层的帧格式为：同步帧＋PHY 头帧＋AM C 帧，帧头序列的长度可以通过寄存器的设置来改变。

每个传感器节点采用 AA 电池供电，电池采用 Panasonic 公司的 CR2054。CR2054 储存了 560mAh 的电能，并且体积较小。传感器系统在收发数据的工作模式下可以连续供电 35 个小时。

3.2 网关节点的硬件设计

网关的中央处理单元主要用来采集和处理从传感器节点送来的数据，并且完成对网络的控制功能。为了达到高性能、低功耗的目的，设计中采用 Altera 公司开发的 Nios 的可以嵌入在 FPGA 芯片中的软核处理器，Nios 处理器采用 16 位指令系统，包括 16 位和 32 位两种版本的体系结构，该处理器内核是由具有 5 级流水线的哈佛结构来实现的，主要特点包括[6]：

(1)处理器包含 512 个内部通用寄存器，并以寄存器窗口的形式组织，编译器使用这些寄存器来加速子程序调用和本地变量访问。

(2)32 位和 16 位的 Nios 处理器都使用 16 位宽的指令，减少了代码的大小和指令存储器的带宽。

(3)Nios 指令集包含加载和存储指令，可使用编译器来加速对结构和本地变量的访问。

Nios 是一种采用流水线技术、单指令流的 RISC 处理器，并针对 Altera 的可编程逻辑器件和片上可编程系统的设计思想做了相应优化，其结构如图 6 所示。作为一个采用硬件描述语言编写的软核，Nios 可以通过 Avalon 总线机制与其它采用 VHDL 语言描述的硬件接口模块组成 Nios 系统，应用 SOPC 技术把系统嵌入到 Altera 的 Stratix，Cyclone，APEX 系列 FPGA 中，从而构成一个可编程片上系统设计。

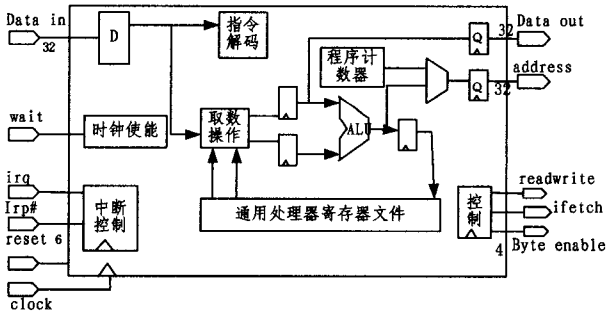

图 6 Nios 处理器结构图

4 系统软件设计

系统软件设计包括传感器节点软件设计和网关节点软件设计两方面工作。

4.1 传感器节点的软件设计

为保证良好的通信，微处理器和传感器之间的通信遵循一定的协议。在无线传感器节点设计中，传感器节点选用温度传感器 DS18B20、湿度传感器 SHT11 和光强传感器 TSL2561。其中，温度传感器采用 1-wire 总线与微处理器进行通信；湿度传感器采用自定义读写协议；光强传感器 TSL2561 采用标准 I2C 协议[7]与微处理器进行数据交互。

温度传感器软件设计的实现：微处理器和温度传感器 DS18B20 通过一线接口通信，数据和指令都是通过一线接口传递，而且没有时钟信号线，所以在模拟通信协议时对延时精确度比较高。同湿度传感器相同，仍然通过宏定义来实现 I/O 口状态的变化。延时函数仍然采用 WINAVR 自身提供的库函数_delay_loop_2()。

```
#define DQ_18B20(1<<3)//PD3
#define DQ_TO_0()(DDRD|=DQ_18B20)//PD3=0
#define DQ_TO_1()(DDRD&=~DQ_18B20)//释放总线，由于外接上拉电阻，总线被拉为高电平
#define DQ_status()(PIND&DQ_18B20)//读取总线值
```

湿度传感器软件设计的实现：由于微处理器通过二线串行接口访问湿度传感器 SHT11，访问协议是芯片生产商自定义的协议，因此需要用通用 I/O 口来模仿该通信协议。ATmega128L 微处理器的 I/O 口可以根据需要通过对 I/O 口寄存器的编程来设置成输入、输出、高阻等状态，这为模拟该通信协议提供了条件。在软件实现过程中，通过宏定义来实现 I/O 口状态的改变。

```
#define set_data_0()DDRB|=(1<<PB5);PORTB&=~(1<<PB5)//DATA 输出 0
#define set_data_1()DDRB|=(1<<PB5);PORTB|=(1<<PB5)//DATA 输出 1
#define release_data_1()DDRB&=~(1<<PB5)//释放总线，将 DATA 设为输入状态，因为外接上拉电阻，DATA 总线被上拉为高电平
#define set_sck_output()DDRB|=(1<<PB4)//设置 SCK 为输出
#define set_sck_1()PORTB|=(1<<PB4)//SCK 输出高电平
#define set_sck_0()PORTB&=~(1<<PB4)//SCK 输出低电平
```

通过以上宏定义，可以实现 SCK 和 DATA 总线的各种输出和输入状态，为了模拟该二线串行协议，还需要一延时函数，WINAVR 库函数提供一延时函数_delay_loop_2(unsigned char s)；该延时函数运行 4 个时钟周期，所以自定义延时 1us。

函数可以如下定义：

```
#define CPU_CRYSTAL 7.3728//系统晶振(MHz)
void delay_us(unsigned char us)
{
_delay_loop_2((unsigned int)((us) * CPU_CRYSTAL/4));
//延时 1us 程序
}
```

基于以上宏定义和延时函数我们可以方便地模拟该二线串行协议。

光强传感器软件设计的实现：微处理器通过 I2C 协议和光强传感器 TSL2561 通信，由于 ATmega128L 本身集成了 TWI 两线接口控制器，TWI 和 I2C 接口是兼容的，因此在程序设计方面会很方便。TSL2561 内部含有 17 个寄存器，通过对内部寄存器的读写，再经过计算就可以很方便地得到光强度值[8]。

TSL2561 程序设计流程如图 7 所示。

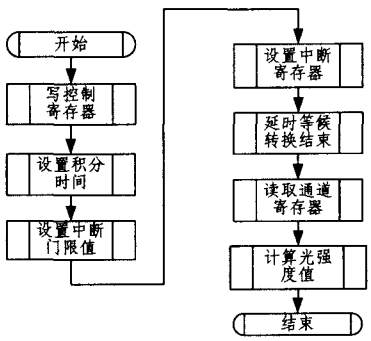

图 7　TSL2561 程序设计流程图

经过实际测试表明，以上 3 种传感器可以完成对温度、湿度和光强度的测量，运行稳定。有关参数如表 1 所列。

表 1　传感器参数表

传感器	精度	响应时间	误差	功耗(mw)：工作/睡眠
TSL 2561	—	400ms	—	15/0.75
SHT11	0.03%	2s	3%	2.75/0.005
DS18B20	0.0625℃	750ms	0.5℃	5/0.005

4.2　网关节点的软件设计

网关节点软件主要完成的功能是管理传感器节点送来的数据，对数据进行处理和数据融合后经过 TCP/IP 功能模块封装打包后由 GPRS 通信模块发送到监控中心，它主要由 GPRS 通信软件、RF 通信软件以及任务管理软件组成。我们采用开源的嵌入式实时操作系统 uC/OS-II，TCP/IP 协议栈采用 LwIP。尽管 uC/OS-II 是一个开放源码的 RTOS，但是目前它的第三方 TCP/IP 支持都是商业化的，很少给出源代码，用户需要付费才能获得。通过在 Nios 上移植 uC/OS-II 和开放源码的 TCP/IP 协议栈 LwIP 就可以实现 uC/OS-II 的网络功能，并建立一套嵌入式网络开发平台。该系统模型，如图 8 所示。

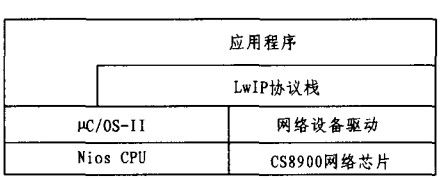

图 8　系统模型

4.2.1　嵌入式实时系统 uC/OS-II[9]

uC/OS-II 是现在流行的一种免费公开源代码的实时操作系统。它广泛应用于从 8 位到 64 位单片机的各种不同类型、不同规模的嵌入式系统。uC/OS-II 的特点为以下几个方面：公开源代码，可移植性好；可裁剪，可固化；内核属于抢占式，最多可以管理 60 个任务。uC/OS-II 的实时性、稳定性和可靠性也得到认可。在系统中嵌入式 uC/OS-II 可以把整个程序分成许多任务，每个任务相对独立，然后在每个任务中设置超时函数，时间用完后，必须交出 MCU 使用权。uC/OS-II 的每个任务都有自己单独的堆栈，。uC/OS-II 允许每个任务有不同的栈空间，以便减小应用程序对 RAM 的需求。

但 uC/OS-II 是一个实时操作系统内核，缺少应用部分，如 TCP/IP 通信软件库。如果只用单片机实现 TCP/IP 协议中的某些功能，可选用免费公开源代码的小型 TCP/IP 协议栈。uC/OS-II 的结构以及它与硬件的关系如图 9 所示。

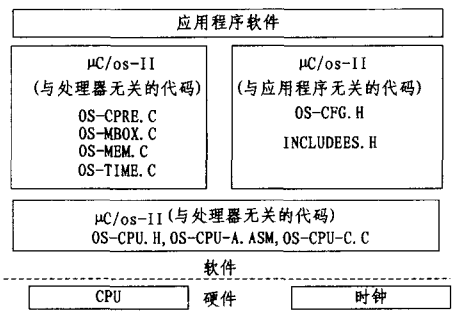

图 9　uC/OS-Ⅱ的硬件和软件体系结构

4.2.2　嵌入式实时系统 uC/OS-II 的移植

移植工作主要集中在多任务切换的实现上。这部分代码主要是用来保存和恢复处理器现场(即相关寄存器)，因此不能用 C 语言，只能使用特定处理器的汇编语言完成。在 Nios 上移植 uC/OS-II 非常简单，只需修改 3 个和 Nios 体系结构相关的文件即可。下面分别介绍这 3 个文件的移植工作。

1. OS_CPU. H 文件

数据类型定义：这部分的移植是和所用的编译器相关的，我们使用的编译器是 nios-elf-gcc。需要定义的数据类型包括无符号和有符号的 8 位、16 位和 32 位整型变量等。堆栈单位：因为处理器现场的寄存器在任务切换时都将被保存在当前运行任务的堆栈中，所以 OS_STK 数据类型应该与处理器的寄存器长度一致。

typedef unsigned int OS_STK；堆栈增长方向 堆栈由高地址向低地址增长，这和选择的编译器有关。

#define OS_STK_GROWTH 1 宏定义(包括开、关中断的宏定义，以及进行任务切换的宏定义)

#define OS_ENTER_CRITICAL() disable_interrupt();

#define OS_EXIT_CRITICAL() enable_interrupt()

#define OS_TASK_SW() OSCtxSw

2. OS_CPU_C. C 文件

该文件必须实现任务初始化时的堆栈设计，也就是在堆栈增长方向上如何定义每个需要保存的寄存器的位置。我们将堆栈空间设计为按任务堆栈空间由高至低依次保存寄存器 ra，ISTATUS，r1－r31。该文件还需要实现几个操作系统规定的 hook 函数。通常都实现为空函数。

3. OS_CPU A. S 文件(由汇编语言实现)

(1)OSStartHighRdy()函数：此函数是在 OSStart()多任务启动后，负责从最高优先级任务的 TCB 控制块中获得该任务的堆栈指针 sp，通过 sp 依次将 CPU 现场恢复。这时系统就将控制权交给用户创建的该任务进程，直到该任务被阻塞或者被其他更高优先级的任务抢占 CPU。该函数仅仅在多任务启动时被执行一次，用来启动优先级最高的任务执行，以后多任务的调度和切换就由下面的函数来实现。

(2)OSCtxSw()函数：任务级的上下文切换。它是当任务因被阻塞而主动请求 CPU 调度时被执行的。它的工作是先将当前任务的 CPU 现场保存到该任务堆栈中，然后获得最高

优先级任务的堆栈指针，从该堆栈中恢复此任务的 CPU 现场，使之继续执行。

(3)OSIntCtxSw()函数：中断级的任务切换，它是在 ISR(中断服务例程)中执行任务切换。若发现有高优先级任务就绪，则在中断退出后并不返回被中断任务，而是直接调度就绪的最高优先级任务执行。这样做的目的是能够尽快地让高优先级的任务得到响应，保证系统的实时性。它的原理基本上与任务级的切换相同，但是由于进入中断时已经保存过被中断任务的 CPU 现场，因此这里就不用再保存。

(4)OSTickISR()函数：时钟中断处理函数。它的主要任务是负责处理时钟中断，调用系统实现的 OSTimeTick 函数，如果有等待时钟信号的高优先级任务，则需要在中断级别上调度其执行。

(5)OS_ENTER_CRITICAL()函数和 OS_EXIT_CRITICAL()函数：分别是进入临界区和退出临界区的宏指令。主要用于在进入临界区之前关中断，在退出临界区的时候恢复原来的中断状态。

4.2.3 TCP/IP 协议栈 LwIP

TCP/IP 是 Interenet 的基本协议。嵌入式设备要与 Internet 网络交换信息，就必须支持 TCP/IP 协议。LwIP[10]是 Light-weight Internet Protocol 的缩写，即轻量级网络协议。LwIP 是瑞典计算机科学院的 Adam Dunkels 等开发的用于嵌入式系统的 TCP/IP 协议栈。LwIP 实现的重点是在保持 TCP/IP 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 kB 的 RAM 和 40kB 左右的 ROM 就可以运行，适于在嵌入式系统中使用。

在 LwIP 中，所有 TCP/IP 协议栈都在一个进程当中。应用层程序既可以是单独的进程，也可以驻留在 TCP/IP 进程中。如果是单独的进程，可以通过操作系统的邮箱、消息队列等和 TCP/IP 进程进行通讯；如果驻留 TCP/IP 进程中，那么利用内部回调函数接口(Raw API)和 TCP/IP 协议栈通讯。对 uC/OS-II 来说，进程就是一个任务。LwIP 的进程模型(Process Model)，如图 10 所示。在图 10 中，整个 TCP/IP 协议栈都在同一个任务(tcpip_thread)中。应用层程序既可以是独立的任务(图中的 tftp_thread 和 cpecho_thread)，也可以在 tcpip_thread 中利用内部回调函数接口和 TCP/IP 协议栈通讯。

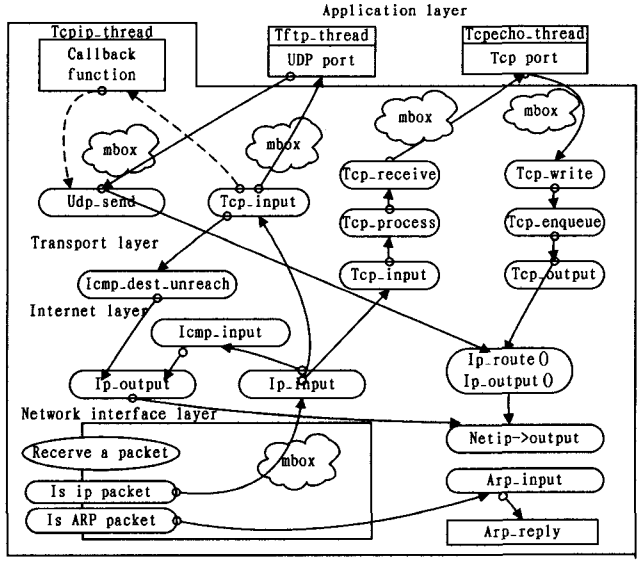

图 10 LwIP 的进程模型

4.2.4 LwIP 在 uC/OS-II 上的实现

在设计 LwIP 时，就考虑到移植问题，所有与操作系统、编译器相关的部分被独立出来，放在/src/arch 目录下。因此，LwIP 在 uC/OS-II 上的实现就是修改这个目录下的文件。下面分别说明相应文件的实现。

1. 与 CPU 或编译器相关的 include 文件

在 LwIP/src/arch/include/arch 目录下，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。这应该与用户实现 uC/OS-II 时定义的参数一致。通常，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 是通过结构体中不同数据的长度来读取相应的数据，所以，一定要在定义 struct 的时候使用_packed 关键字，让编译器放弃 struct 的字节对齐。LwIP 也考虑到了这个问题，所以，在它的结构体定义中有几个 PACK_STRUCT_xxx 宏，在移植的时候添加编译器所对应的_packed 关键字。比如在 nios-eft-gcc 上对应的定义为：

```
# define PACK_STRUCT_FIELD(x) x_attribute_((packed))
# define PACK_STRUCT_STRUCT _attribute_((packed))
```

2. sys_arch 操作系统相关部分

sys_arch. h[c]中的内容是与 OS 相关的一些结构和函数，主要可以分为 4 个部分：

(1)sys_sem_t 信号量

LwIP 中需要使用信号量进行通信，所以在 sys_arch 中应实现信号量结构体和处理函数：

```
struct sys_sem_t
sys_sem_new() //创建一个信号量结构
sys_sem_free() //释放一个信号量结构
sys_sem_signal() //发送信号量
sys_arch_sem_wait() //请求信号量
```

由于 uC/OS-II 已经实现了信号量 OS_EVENT 的各种操作，并且和 LwIP 上面几个函数的功能是完全一样的，因此要把 uC/OS-II 的函数重新封装成上面的函数。

(2)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 以及相应的操作函数：
sys_mbox_new() //创建一个消息队列
sys_mbox_free() //释放一个消息队列
sys_mbox_post() //向消息队列发送消息
sys_arch_mbox_fetch() //从消息队列中获取消息
```

uC/OS-II 虽然实现了消息队列结构 OS_Q 及其操作，但是 uC/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 uC/OS-II 的基础上重新实现。为了实现对消息的管理，定义了以下结构：

```
typedef struct{
OS_EVENT * pQ;
void * pvQEntries[MAX_QUEUE_ENTRIES];
sys_mbox_t;
typedef PQ_DESCR sys_mbox_t;//LwIP 中的 mbox 是 UCOS 的消息队列
```

该结构包括 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 uC/OS-II 自己的消息队列相关函数来完成,然后使用 uC/OS-II 中的内存管理模块来实现对消息的创建、使用和删除,两部分综合起来便实现了 LwIP 的消息队列功能。

(3)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,它包括这个线程的 timeout 时间长度,以及超时后应调用的 timeout 函数,该函数会做一些释放连接、回收资源的工作。timeout 结构体已经在 sys.h 中定义好了,而且对结构体队列的数据操作也由 LwIP 负责,所要实现的是如下函数:struct sys_timeouts * sys_arch_timeouts (void) 。这个函数的功能是返回目前正处于运行状态的线程所对应的 timeout 队列指针。timeout 队列属于线程的属性,因此是与操作系统相关的函数,只能由用户实现。

(4)sys_thread_new 创建新线程函数

LwIP 可以是单线程运行,即只有一个 tcpip 线程(tcpip_thread),负责处理所有的 TCP(Transmission Control Protocol,传输控制协议)或 UDP(User Datagram Protocol,用户数据报协议)连接,各种网络程序都通过 tcpip 线程与网络交互。它也可以多线程运行,以提高效率。这时就需要用户实现创建新线程的函数:

void sys_thread_new(void(* thread)(void * arg),void * arg);

在 uC/OS-II 中,没有线程(thread)的概念,只有任务(Task)。它提供了创建新任务的系统调用 OSTaskCreate,因此只要把 OSTaskCreate 封装一下,就可以实现 sys_thread_new。需要注意的是 LwIP 中的 thread 并没有 uC/OS-II 中优先级的概念,实现时,用户要事先为 LwIP 中创建的线程分配好优先级。

3. lib_arch 中库函数的实现

LwIP 用到 8 个外部函数,这些函数通常与用户使用的系统或编译器有关,因此要求用户实现。

u16_t htons(u16_t n);//16 位数据高低字节交换

u16_t ntons(u16_t n);

u32_t htonl(u32_t n);//32 位数据大小端对调

u32_ t ntonl(u32_t n);

int strlen(const char * str);

int strncmp(const char * str1,const char * str2,int len);

void bcopy(const void * src,void dest,int len);

void bzero(void * data,int n);

4.3 网络设备驱动程序

我们采用的网络芯片为 Cirrus Logic 公司的 CS8900 芯片。LwIP 的网络驱动有一定的模型,/src/netif/ethernetif.c 文件即为驱动的模板,用户为自己的网络设备实现驱动时应参照此模板。在 LwIP 中可以有多个网络接口,每个网络接口都对应了一个 netif 结构,该结构体包含了相应网络接口的属性、收发函数。LwIP 调用 netif 的函数 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。

void ethernetif_init(struct netif * netif); //网卡初始化函数

void ethernetif_input(struct netif * netif); //网卡接收函数

err_t ethernetif_output(struct netif * netif,struct pbuf * p,struct ip_addr * ipaddr); //网卡发送函数

void ethernetif_isr(void);//网卡中断处理函数

5 实验测试

完成上面的移植修改工作后,就可以在 uC/OS-II 中初始化 LwIP,并创建 TCP 或 UDP 任务进行测试了。这部分是用 c 语言实现的。关键部分的代码和说明如下:

```
main()
{
  OSInit();
  OSTaskCreate(lwip_init_task,&task1_data,&lwip_init_stk
  [TASK_STK_SIZE-1],0);
  OSTaskCreate(user_task,&task2_data,&user_stk[TASK_
  STK_SIZE-1],1);
  OSStart();
}
```

主程序中,创建了 Lwip_init_task 初始化 LwIP 任务(优先级 0)和 user_task 用户任务(优先级 1)。lwip_init_task 任务中除了初始化硬件时钟和 LwIP 之外,还创建了 tcpip_thread(优先级 3)和 tcpecho_thread(优先级 4)。实际上 tcpip_thread 才是LwIP 的主线程,多线程的 Berkley API 也是基于这个线程实现的,即上面的 tcpecho_thread 线程也要依靠 tcpip_thread 线程来与外界通信。tcpecho_thread 是一个 TCP echo 服务,监听 7 号端口。程序框架如下:

```
void tcpecho_thread(void arg)
{
    conn = netconn_new(NETCONN_TCP);//建立新的连接
    netconn_bind(conn,NULL,7);//绑定到 7 号端口
    netconn_listen(conn);//监听 7 号端口
    while(1)
    {
    newconn = netconn_accept(conn);//接收外部连接
    bur=netconn_recv(newconn) //获取数据
    …… //处理数据
    netconn_write(newconn,data,len,NETCONN_COPY);//发送数据
    netconn_delete(newconn);//释放本次连接
    }
}
```

编译下载运行,用 ping ip 地址命令可以得到 ICMP reply 响应,用 telnet ip 地址 7(登录 7 号端口)命令可以看到 echo server 的回显效果。说明 ARP,IP,ICMP,TCP 协议都已正确运行。

结束语 本文着重论述了无线传感器网络节点和网关节点的硬件、软件设计,整个设计建立在嵌入式实时操作系统 uC/OS-II 和 NIOS 软核心处理器的基础上,在此基础上,提出了 uC/OS-II 的移植方案,并使 LwIP 在 uC/OS-II 上实现这一

(下转第 219 页)

了上面出现的无法撤销的问题。

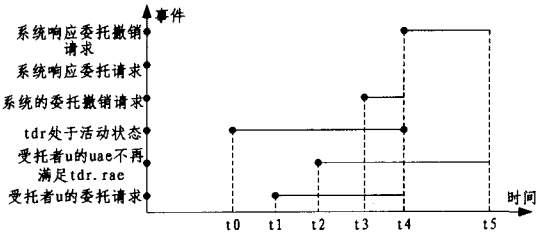

图 4 “先撤销后委托”中的委托和撤销时间关系

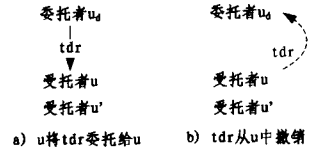

图 5 “先撤销后委托”中委托和撤销的结果

因此，当同时存在针对同一个临时委托角色的委托和委托撤销请求时，要求系统优先响应委托撤销请求。此处用临时委托角色 tdr 的当前状态来表示在其上的两种操作和所处的状态。

定义 1 记用户 u 的临时委托角色 tdr 的当前状态为 $\lambda(u,tdr)$，其取值定义为：

$$\lambda(u,tdr)=\begin{cases}1 & \text{当在 } u \text{ 中的 } tdr \text{ 上只存在委托请求}\\ 2 & \text{当在 } u \text{ 中的 } tdr \text{ 上存在委托撤销请求}\\ 0 & u \text{ 中的 } tdr \text{ 两种请求均不存在}\end{cases}$$

式中，只有当 $\lambda(u,tdr)=1$ 时，才能够委托成功。

前面例子在引入了临时委托角色状态后，系统的操作时间顺序以及临时委托角色的 λ 值的变化如图 6 所示。

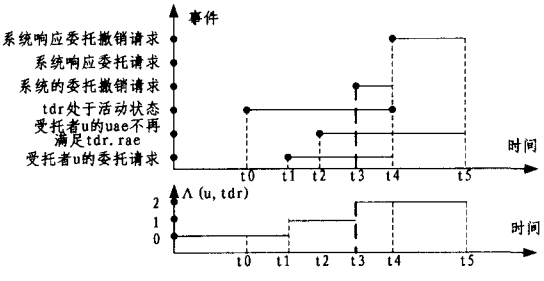

图 6 引入 $\lambda(u,tdr)$ 后的委托和撤销时间关系及 $\lambda(u,tdr)$ 变化过程

引入状态函数后，委托系统在处理受托者将临时委托角色再次委托出去时必须保证该状态函数取值为 1。因此，撤销对多步委托就产生了影响。本文并不在此深入探讨，可作为进一步的研究之用。

结束语 撤销是委托模型必不可少的功能，作为 ABDM 模型的重要组成部分，本文和文献[8]分别从撤销和委托两个方面说明了基于属性的委托模型中撤销与委托操作。本文认为为保证委托和撤销操作的正确性，在对某个临时委托角色同时存在委托和撤销操作时，需优先执行撤销操作。本文只给出了最基本的撤销操作，进一步的工作将包括如何进行基于属性的多步和多重撤销。

参 考 文 献

[1] Sandhu R, Coyne E, Feinstein H, et al. Role-Based Access Control Models[J]. IEEE Computer, 29(2): 38-47

[2] Barka E, Sandhu R. Framework for Role-Based Delegation Models[C]// Proc. of 16th Annual Computer Security Application Conference (ACSAC2000). New Orleans, USA: IEEE Computer Society Press, 2000

[3] Barka E, Sandhu R. A role-based delegation model and some extensions[C]// Proc. of 23rd National Information Systems Security Conference (NISSC 2000). Baltimore, USA, 2000

[4] Zhang Longhua, Ahn Gail-Joon, Chu Bei-Tseng. A rule - based framework for role-baseddelegation[C]// Proc. of SACMAT '01. Chantilly, VA, USA: ACM press, 2001

[5] Tamassia R, Yao Danfeng, Winsborough W H. Role-based cascaded delegation[C]// Proc. of the SACMAT '04. Yorktown Heights, New York, USA: ACM press, 2004

[6] Zhang Xinwen, Oh S, Sandhu R. PBDM: A Flexible Delegation Model in RBAC[C]// Proc. of the SACMAT '03. Como, Italy: ACM press, 2003

[7] 赵青松，孙玉芳，孙波. RPRDM：基于重复和部分角色的转授权模型[J]. 计算机研究与发展，2003，40(2)：221-227

[8] 叶春晓，吴中福，符云清，等. 基于属性的扩展委托模型[J]. 计算机研究与发展，2006，43(6)：1050-1057

[9] Stoupa K, Vakali A, Li Fang, et al. XML-based revocation and delegation in a distributed environment[C]// Proceedings of the EDBT International Workshop on Database Technologies for Handling XML information on the Web. Heraklion, Greece, 2004: 299-308

[10] 孙波，赵庆松，孙玉芳. TRDM--具有时限的基于角色的转授权模型[J]. 计算机研究与发展，2004，41(7)：1104-1109

(上接第 141 页)

关键问题得以解决。实践证明，该嵌入式系统功能完整、集成度高、稳定可靠、简洁实用，具有可编程性、可裁减性、易操作性、灵活性以及低成本等特点，具有广泛的应用价值。

参 考 文 献

[1] Pottie G J, Kaiser W J. Embedding the Internet: Wireless Integrated Network Sensors[J]. Communications of the ACM, 2003, 43(5): 51-58

[2] Cerpa A, Elson J, Estrin D, et al. Habitat Monitoring: Application Driver for Wireless Communications Technology[C]// Proceedings of the ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean. San Jose, Costa Rica, 2001

[3] 孙利民，李建中，陈渝，等. 无线传感器网络[M]. 北京：清华大学出版社，2005(4)

[4] Raghunathan V, Schurgers C, Park S, et al. Energy-AwareWireless Microsensor Networks[J]. IEEE Signal Processing Magazine, March 2002: 40-41

[5] Barnett R, Larry O'cull, Cox S. 嵌入式 C 编程与 Atmel AVR [M]. 周俊杰，译. 北京：清华大学出版社，2003：75-76，52-53

[6] AlteraCorporation, Nios3. 0CPUDatasheet[Z]. Version2. 2, 2004. 10

[7] Semiconductors P. The I2C Bus Specification, 2003: 11-12

[8] 何立民. I2C 总线应用系统设计[M]. 北京：北京航空航天大学出版社，1995：178-180

[9] LabrosseJeanJ. uC/OS-II —源码公开的实时嵌入式操作系统[M]. 北京：中国电力出版社，2006

[10] Dunkels A. LwIP. A free small TCP/IP implementation for 8 and 16-bit microcontroller[Z]. 2006

[11] 高雷，郑相全，张鸿. 无线传感器网络中一种基于三边测量法和质心算法的节点定位算法[J]. 2009，23(7)：138-141