面向物联网应用的 UART-WIFI 网关设计*

王希朝[1],张 毅[1],程 鹏[2],马洪亮[2],吴 斌[2]
(1.重庆邮电大学,重庆 400065;
2.中国科学院微电子研究所,北京 100029)

摘 要:以飞思卡尔 MCIMX27 为控制器,设计了一款基于 WIFI 的串口数据无线收发模块,实现了通过串口配置模块参数以及用户串口数据的网络化。通过 UART-WIFI 模块,传统的串口设备也能轻松接入无线网络。着重介绍了系统软件设计整体架构、自定义协议以及实现中需要解决的关键问题。实验结果表明模块无线通信性能好、稳定性强。

关键词:物联网;网关;WIFI;UART
中图分类号:TP399 **文献标识码**:A **文章编号**:1674-7720(2013)08-0045-03

Research and design of UART-WIFI gateway of the Internet of Things oriented

Wang Xichao[1], Zhang Yi[1], Cheng Peng[2], Ma Hongliang[2], Wu Bin[2]
(1.Chongqing University of Posts and Telecommunications, Chongqing 400065, China;
2.Institute of Microelectronics of Chinese Academy of Science, Beijing 100029, China)

Abstract: With Freescale MCIMX27 as the control center, the serial data transceiver module design based on a WIFI to achieve the conversion between the data the by serial configuration module parameters, as well as user serial to wireless network via UART-WIFI module, the traditional serial devices can easily access the wireless network. The article focuses on the overall architecture of the system software design, custom protocols and the implementation of the key issues that need to be addressed. The experimental results show that the UART-WIFI module designed wireless communication performs well and stably.

Key words: Internet of Things; gateway; WIFI; UART

物联网作为互联网的延伸和扩展,使得通信的主体不再是人与人之间的通信,还有人与物、物与物之间的通信。随着物联网产业的发展,各种老式设备也有了接入网络的需求,在工业控制和通信设备中,很多是符合 RS232 标准的串口设备,通常很难在作业现场铺设有线网络。如何保证在原来设备不做太大改动的基础上实现这些设备的联网,便成了一个首先需要解决的问题。

面向物联网应用的 UART-WIFI 网关,基于目前成熟的 WIFI 无线传输解决方案实现串口数据的网络化传输。本文设计的方案,使得只要具有 UART 接口的设备即可接入网络,提高了设备的智能化水平,简化了设备组网流程[1],具有很重要的现实意义和使用价值。

1 系统整体架构

UART-WIFI 网关在用户侧留有标准 RS232 接口,使得设备只要具有串口就可以接入网络,网关的应用场景如图 1 所示。

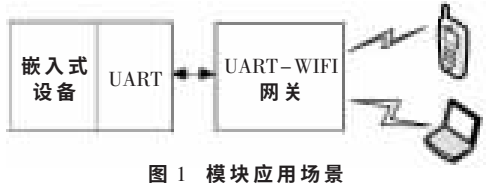

图 1 模块应用场景

网关从功能上分为串口数据收发模块、网络数据收发模块和用户数据处理模块等。依据数据流向,系统模块组成原理框图如图 2 所示。系统通过串口配置网卡参

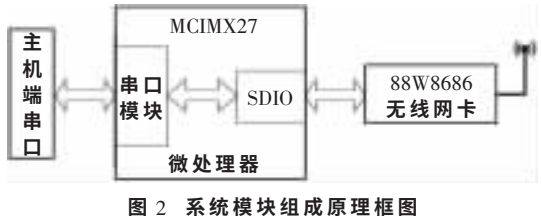

图 2 系统模块组成原理框图

* 基金项目:广东省院地合作项目(2010YS014)

数，实现串口数据的无线透明传输。

串口数据收发模块负责用户主机与网卡之间的数据交互；无线网卡模块负责收发网络数据，进行 802.11 与 802.3 协议之间的相互转换并交由操作系统处理；用户数据处理模块负责识别主机侧用户接口数据，用于配置网卡参数或进行数据透明传输。

2 系统硬件设计

2.1 系统硬件原理框图

硬件系统主要由处理器、无线网卡、标准 RS232 收发器模块、外部存储器以及系统电源组成。系统硬件原理框图如图 3 所示。

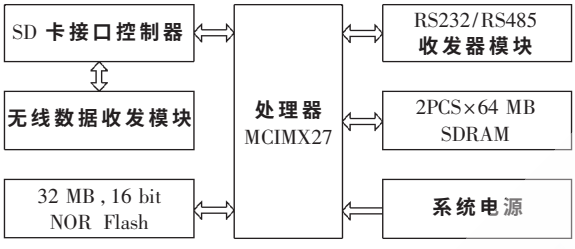

图 3 系统硬件原理框图

2.2 系统硬件设计及选型

系统原型开发选用飞思卡尔半导体面向多媒体应用的 MCIMX27 处理器，该处理器采用 ARM926-EJS 内核，工作频率可达 400 MHz，其内建的 MMU 功能，能很方便地实现嵌入式 Linux 的移植。

系统存储系统由两片 DDR SDRAM 和一片 NAND Flash 构成。处理器内嵌的 SDIO 主控制器提供 4 bit 模式下最高 100 Mb/s 的数据速率。无线网络收发模块选用 Marvell 的 WLAN 片上系统芯片 88W8686 设计，芯片集成 IEEE802.11a/g/b MAC/Baseband/RF 等功能模块，支持典型的 WLAN 数据速率；支持 SDIO 接口的主机接口单元（HUI），允许主机控制器使用 SDIO 总线协议与 WLAN 设备进行通信。

3 系统软件设计

3.1 系统软件整体架构

在 Linux 操作系统下通过对各个功能模块的划分实现了对 WLAN 的完美支持，图 4 所示为 Linux 下典型的 WLAN 层次模型[2]。

硬件层是软件运行的承载体和通信的物理实现层；固件（Firmware）层向驱动程序屏蔽具体的物理细节，提供驱动访问硬件的接口；驱动程序向 WT（Wireless-Tools）或者其他配置工具提供访问底层的接口；配置程序层向上层提供一个统一的 Linux 用户接口；用户使用相关配置工具来访问和配置不同的无线网卡。本设计软件整体架构参照 Linux 操作系统下典型的 WLAN 层次模型来设计，系统软件整体架构如图 5 所示。

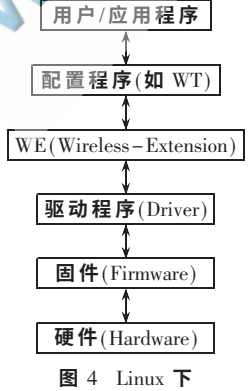

图 4 Linux 下典型 WLAN 层次模型

图 5 系统软件整体架构

3.2 Linux 操作系统

嵌入式 Linux 以其内核高效稳定、网络功能强大等特性，成为嵌入式系统领域中的佼佼者[3]。本设计选用成熟的 Linux-2.6.19.2 操作系统内核，编译器使用专用 arm-926ejs-linux-gcc 交叉编译器。

3.3 WT 工具

Wireless Tools for Linux 是一个 Linux 命令行工具包，用来设置支持 Linux Wireless Extension 的无线设备。Wireless-Extension（WE）是一组通用的 API，能在用户空间对通用 Wireless LANs 进行配置和统计。

WT（Wireless-Tools）就是用来操作 Wireless-Extensions 的工具集合，支持所有 Wireless-Extension，它主要包括 iwconfig、iwlist 和 iwpriv 等工具。

3.4 网卡驱动和固件

按照主机驱动与设备驱动分离设计的思想，一个标准的无线收发系统软件包含主机驱动和 WLAN 固件两部分。

WLAN 固件主要实现 802.11 和 802.3 协议帧格式之间的转换。在系统接收数据时，网卡接收符合标准 802.11 协议的数据，由 WLAN 固件转换成符合 802.3 协议的帧格式，通过 SDIO 接口送达主机驱动；在发送串口用户数据时，应用层程序将符合 802.3 协议的数据通过 SDIO 接口发送到 WLAN 固件，固件将其转换为 802.11 协议数据帧，通过无线方式发往服务器端。

主机驱动模块主要包含三部分：Ethernet Driver、802.11Extensions 和 Hardware Interface Driver。Ethernet Driver 实现标准的以太网驱动；802.11Extensions 扩展标准以太网驱动以控制 WLAN Adapter 的状态；Hardware Interface Driver 即硬件接口驱动控制在主机侧的硬件接口。

3.5 通信协议设计

通过 UART 接口在用户终端设备与 UART-WIFI 之间传输的数据称之为用户接口数据，接口数据分为控制

数据和用户数据两种类型：

(1)控制数据

用于用户终端设备与 UART-WIFI 之间的控制信息传输,配置模块的网络参数和系统参数。

(2)用户数据

模块用于数据透明传输时遵循用户数据帧格式,协议由用户自定义。

3.6 系统软件工作流程设计

在系统中软件将分为两个线程工作，一个线程接收串口数据,用于系统参数配置或者网络数据传输;另一个线程用于接收服务器端信息，并通过串口发送至主机端。

系统完成初始化相关参数设置后,创建两个线程来使系统进入稳定工作状态。主要代码如下：

```
i=pthread_create(&thread_a,NULL,(void *)
                                    UartDataProcessFun,NULL);
if(i==0)                                          //创建线程 1
printf("Create thread1 success! \n");
else{
printf("Create thread1failure! \n");
     exit(0);
     }
j=pthread_create(&thread_b,NULL,(void *)
                                    NetDataProcessFun,NULL);
if(j==0)                                          //创建线程 2
printf("Create thread2 success! \n");
else{
printf("Create thread2 failure! \n");
exit(0);
}
pthread_join(thread_a,NULL);                      //等待串口数据
                                                  处理线程结束
pthread_join(thread_b,NULL);                      //等待网络数据
                                                  处理线程结束
```

其中(void *) UartDataProcessFun 和(void *)NetDataProcess-Fun 是指向在函数外部定义的两个线程处理函数的指针,线程创建后将同时开始执行。

3.7 系统设计中的多线程同步方法

多线程的引入降低了系统实现的复杂度,提高了系统执行效率。多线程程序在执行时,除了局部变量外,其他所有变量都将在一个进程中的所有线程之间共享,为了改变程序执行时序,保护共享资源,需要引入线程同步方法。

Linux 提供了多种线程同步的机制,其中有互斥锁、信号量、条件变量等[5]。单独使用互斥锁容易发生死锁;信号量分为简单的二进制信号量和计数信号量;条件变量通过允许线程阻塞和等待一个线程发送信号的方式弥补了互斥锁的不足,常和互斥锁一起使用。使用时条件变量用来阻塞一个线程,条件不满足时,线程解开互斥锁等待条件发生变化。当某个线程改变了条件变量后,它将通知相应的条件变量唤醒一个或者多个正被此条件变量阻塞的线程。本设计中的线程同步采用互斥锁加条件变量的方式。使用条件变量需要的头文件是 pthread.h。使用条件变量标识符 pthread_cond_t 创建一个静态条件变量时使用 PTHREAD_COND_INITIALIZER 常量,例如:pthread_cond_t cond=PTHREAD_COND_INITIA-LIZER。等待条件变量使用到的函数有：

```
int pthrea_cond_wait(pthread_cond_t *cond,pthread_mutex_
                                                    t *mutex)
int pthread_cond_timedwait(pthread_cond_t *cond,
    pthread_mutex_t *mutex,const struct timespec *abtime)
```

激发条件变量使用到的函数有：

```
pthread_cond_signal()        //激活一个等待该条件的线程
pthread_cond_broadcast()     //激活所有等待线程
```

互斥锁加条件变量实现多线程同步的一般方法如下：

线程 1 代码：

```
pthread_mutex_lock(&mutex);                       //上锁
if(条件满足)
pthread_cond_signal(&cond);//激活等待该条件变量的线程
pthread_mutex_unlock(&mutex);                     //解锁
```

线程 2 代码：

```
pthread_mutex_lock(&mutex);                       //上锁
while(条件不满足)
pthread_cond_wait(&cond,&mutex);  //阻塞等待条件满足
pthread_mutex_unlock(&mutex);                     //上锁
```

在上述代码执行过程中,条件变量一旦满足,线程 1 立即通知等待该条件变量的线程 2，同时释放互斥锁,线程 2 在捕获到条件变量满足的消息便被激活,执行相关操作。

本文设计了一款面向物联网应用的 UART-WIFI 网关，重点分析了基于 Linux 操作系统的 WLAN 软件架构,介绍了 Linux 下多线程同步的处理方式,重点分析了使用互斥锁和条件变量进行同步的方法。本文设计的 UART-WIFI 网关具有成本低、易部署的特点。经过测试，证实该网关具有良好的稳定性和通信性能,可满足当前物联网发展的需求,具有很好的使用和推广价值。

参考文献

[1] 范炜,徐洪泽.基于 TCP/IP 协议的嵌入式多串口网关的设计[J].计算机工程与设计,2008,29(1):80-82.

[2] 高扬,石秀民.基于嵌入式平台的 WLAN 实现[J].吉林大学学报,2006,24(1):103-107.

[3] 董志国,李式巨.嵌入式 Linux 设备驱动程序开发[J].计算机工程与设计,2006,27(20):3737-3740.

[4] 贾晶鑫,蒋健,宋彬.ARM9 工控平台上的多串口网关及视频采集传输的实现[J].电子产品世界,2012,19(7):37-39.

[5] MATTHEW N,STONES R.Linux 程序设计(第 4 版)[M].北京:人民邮电出版社,2010.

[6] MCIMX27 multimedia applications processor reference manual MCIMX27RM Rev.0.2[OL].[2007-09-27].http://www.freescale.com.

(收稿日期:2012-12-22)

作者简介:

王希朝,男,1986 年生,硕士研究生,主要研究方向:短距离无线通信。

张毅,男,1963 年生,教授,硕士生导师,主要研究方向:通信与信息系统。

程鹏,男,1982 年生,助理研究员,主要研究方向:宽带无线通信系统。