基于嵌入式 ARM-Linux 无线 ZigBee 协调器驱动设计

李 婧,史智兴,崔哲伟,贾 方
(河北农业大学 信息科学与技术学院,河北 保定 071001)

摘 要:以 ARM9 处理器 S3C2410 作为嵌入式 Linux 的系统开发和运行平台,利用适合田间要求的无线 ZigBee 节点 CC2430,在 Linux 内核中实现协调器的字符驱动,使其通过 IO 进行数据传输,避免了串口数据传输速率低、数据冗余性差、程序设计繁琐等缺点。详细介绍了该系统的设计和开发过程,包括网络通信协议的选取,bootloader、内核、文件系统以及驱动程序的实现和移植。

关键词:ZigBee;ARM9;CC2430;内核;移植;驱动
中图分类号:TN915.09 文献标识码:A 文章编号:1674-7720(2011)15-0029-03

Design of the coordinator of wireless ZigBee driven based on embedded ARM-Linux

Li Jing, Shi Zhixing, Cui Zhewei, Jia Fang
(College of Information Science and Technology, Agricultural University of Hebei, Baoding 071001, China)

Abstract: Using ARM9 processor S3C2410 as the embedded Linux systems development and operating platform, using CC2430 as the wireless node which suitable for the requirements of filed. It implements the character-driven of the coordinator in the Linux kernel and transmits data through the IO. To avoid the low velocity of data transmission, data redundancy and the combersone of programe desigin by serial and so on. This paper introduces the system design and development process includeing the selection of network communication protocols, bootloader, kernel, file system and the impletement and migration of the process.

Key words: ZigBee; ARM9; CC2430; kernel; protable; drive

农田中大范围的环境信息监测已成为网络应用范围重点之一。针对农田布线不便的特点,ZigBee 无线节点网络成为农田信息采集系统的首选,可对其所分布区域内的各种环境和检测对象的信息进行实时的监控[1]。然而,控制下层整个网络状态的核心是上位机 ARM 处理器,而且上位机与下位机通信大多以串口模式来实现[2-3]。但串口通信模式存在串口传输速率低(波特率双方一致)、传送距离短[4]、数据冗余差(数据校验)以及设计串口协议繁琐(帧格式)等不足。因此本文研究了 ZigBee 在 ARM9 内核中的协调器字符驱动,利用 I/O 传输数据,控制具有协调器驱动的设备在农田任何位置即可组网,以减少协调器的布局,实现方便快捷的动态数据监测。

1 田间监测系统的要求

因监测节点需要零散分布在田间,以监测田间的空气和地表的温度,因此,田间监测系统所需要的技术指标应满足:(1)低功耗。田间采电受到布线限制,因此节点模块的耗电量应尽可能低。(2)低成本。田间需要大量布局节点,投资成本成为广泛实施的制约因素。(3)低复杂度和高可靠性。田间节点开发设备应采用结构简单、采集数据尽可能精确又廉价的设计。综合上述特点,ZigBee 可以作为田间无线协议首选。

ZigBee 协议是基于 IEEE802.15.4 标准的低功耗、低速率、低复杂度的双向通信技术。它可工作在国际上免授权的 2.4 GHz,具有 250 Kb/s 的最高数据传输速率和 10~75 m 的可靠传输距离。ZigBee 支持星型、树型、对等和混合型网络拓扑结构,网络中的从设备高达 254 个。根据如图 1 所示的节点在网络分布的特点,节点在网络中可实现多条数据链路通信,以选择最佳的路径进行传输,提高了网络通信的可靠性。

协调器是整个网络的核心部分,负责完成整个网络的无线接入和组建,是维持路由器和终端节点之间的数据通信的关键。在田间固定放置协调器节点会浪费大量的资源,若动态地测量田间任意位置的数据,把协调器作为移动设备动态地测量数据则是最好的选择,并且可

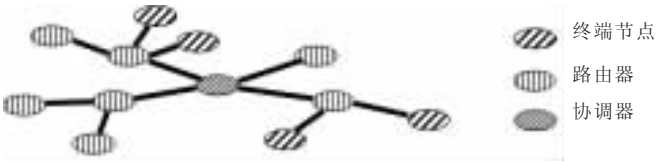

图 1 网络节点分布图

以减少田间协调器的放置,降低设计难度的成本。

2 嵌入式 Linux 驱动开发环境的搭建

Linux 操作系统环境的搭建如图 2 所示。

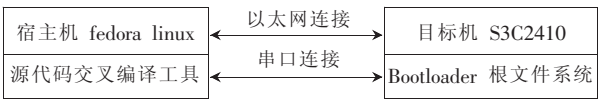

图 2 Linux 环境搭建

2.1 Bootloader 的移植

Bootloader 是操作系统内核运行之前运行的一小段程序,它为加载内核提供合适的硬件环境。Bootloader 分成 Stage1 和 Stage2 两个阶段,具体实现框图如图 3 所示。

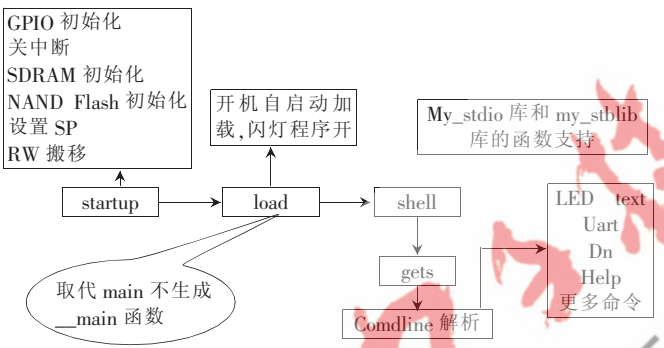

图 3 bootloader 实现原理图

Stage1:主要由汇编实现,包括 GPIO 驱动、使开发板上电 LED 闪烁、关闭所有中断、设置系统时钟、关闭看门狗、SDRAM 初始化、实现相应驱动(提供更大的执行空间)、NAND Flash 初始化(驱动开发板上唯一的固态存储掉电不消失设备)以及设置 SP 栈指针为 Stage2 中的 C 语言代码执行做好准备。

Stage2:实现加电自搬移过程、串口调试信息、函数库、shell 命令等扩展功能。

2.2 内核的编译和移植

本硬件移植 2.6.27 版本的 Linux 内核:(1)解压缩 tar xf linux-2.6.27.tar.bz2,进入该目录。(2)移植平台为 ARM 体系结构,修改 Makefile 中的 ARCH? =arm CORSS_COMPILE? = arm-linux-(交叉编译器的前缀)。(3)配置内核:make deconfig(清除原来编译的 config,如果是第一次配置可省略);make menuconfig 进入配置菜单,选择硬件所需的驱动。大部分可选择默认选项,但注意网卡驱动一定必选,硬件类型也要匹配。(4)编译内核 make bzImage 在~/linux-2.6.27/arch/arm/boot/bzImage 生成内核映像,通过 tftp 把 bzImage 烧到地址为 0x30008000 内存上,然后用 nand erase kernel 擦除 kernel 分区上的数据,最后用 nand write 0x30008000 把内存上的数据烧到 Flash 对应的 kernel 分区上。

2.3 根文件系统的移植

运行 Linux 操作系统,除了内核外还需要根文件系统。用 mkdir 创建 rootfs 文件夹,在其中创建根文件系统目录并安装 busybox。busybox 是专门为嵌入式系统设计的,它把大多数常用的命令(如 ls,cp,cd,tar 等)拼接在一起,在根文件系统中只有一个可执行文件/bin/busybox,其余都是 busybox 的链接。安装 busybox 与安装内核类似,在~$tar xf busybox-1.9.1.tar.bz2、cd busybox-1.9.1/下修改 ARCH? =arm CROSS_COMPILE? = arm-linux-;make defconfig、make menuconfig 设置 busybox 安装路径 rootfs 文件夹。将 make、make install、busybox 文件与一系列链接文件安装在 rootfs 下。其他链接文件在/bin、/sbin、/usr/bin、/usr/sbin 中,配置 Linuxrc 启动文件、安装 glibc 共享库,在/dev 目录下创建设备文件,将主机系统时钟拷贝到根文件系统中去,并配置网路和 http 相关配置文件。最后将文件系统配置成 YAFFS 文件系统,可直接对文件系统进行读写。设置开发板为 NFS 方式,启用可以直接在主机上操作开发板的根文件系统并进行调试。

3 硬件设计及驱动实现

3.1 系统硬件设计

本系统平台是采用 ARM 体系结构的 S3C2410 作为处理器,通过移植的字符设备驱动与 ZigBee CC2430 无线收发节点进行数据的传输。系统硬件框架图如图 4 所示。

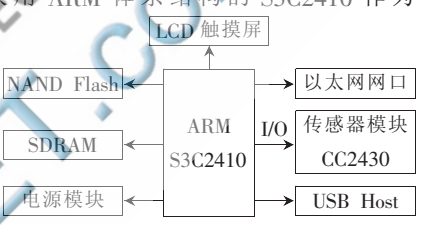

图 4 系统硬件框架图

CC2430 是一个真正的片上系统(SoC),以高性能和低功耗的 8051 为内核,专门针对 IEEE802.15.4 和 ZigBee 应用,它可以用很低的费用构成 ZigBee 节点。

现有的硬件是通过串口实现数据传输,数据传输的格式要按照串口通信协议的格式封装,大量数据的传输还需要在串口通信格式的基础上再进行设计封装,不仅数据传输速度慢,而且容错能力低。如果在内核中加入 ZigBee 的字符驱动则可省去数据发送时的封装以及接收时需要解析的麻烦。

3.2 Linux 设备驱动实现

Linux 的输入输出设备分为字符设备、块设备和网络设备三类。字符设备是发送和接收都按照字符方式进行。块设备则是传输固定大小的数据给设备。网络设备则是通过 BSD 套接口访问设备。驱动程序一般以模块方式编写,加载和卸载主要由 module_init()和 module_exit()完成[2]。

(1)模块加载和卸载

模块需要入口函数 module_init(zigbee_init)的实现代码如下:

```
int __init zigbee_init(void)
{
    if(zigbee_major){
        dev=MKDEV(zigbee_major,zigbee_minor);
        result=register_chrdev_region(ev,1,"zigbee");
        }else{
```

```
        result=alloc_chrdev_region(&dev, zigbee_minor, 1,
                                                "zigbee");
        ……
        zigbee_major=MAJOR(dev);
        ……
        }
        cdev=cdev_alloc();
        cdev->ops=&zigbee_fops;
        rc=cdev_add(cdev, dev, 1);
        ……;
        return 0
}
module_exit(zigbee_exit)
{
    cdev_dev(cdev);
    return 0
}
```

在不同的系统中，同一设备的设备号不尽相同，如果静态设置设备号，则在换另外的平台时，设备号有可能冲突，所以动态分配是最佳选择。

(2)模块驱动实现

注册设备编号后要将设备驱动与之连接，因此必须用 file_operation 结构建立链接，并建立中断通知相关数据。其实现代码如下：

```
Struct file_operation zigbee_fops={
    .owner=THIS_MODULE,
    .open=zigbee_open,
    .read=zigbee_read,
    .write=zigbee_write,
    .ioctl=.zigbee_ioctl,
    .relese=zigbee_release,
}
```

当上层应用调用驱动程序时，驱动程序需要完成以下功能：

①初始化设备。S3C2410 与下层 ZigBee CC2430 连接管脚处于工作状态，注册并使能中断。

②按照 ZigBee 协议规则构建数据包并发送给 CC2430，实现不同控制命令，使芯片完成数据发送和状态间的转换。

③当下位机接收到的数据与协议包格式不符时，产生中断，用户须重新发送数据。

其实现代码如下：

```
Int zigbee_open(struct inode *inode, struct file *filp)
{
    Rc =request_irq (IRQ_EINT0, zigbee_interrupt,
SA_INTERRUPT, "zigbee", NULL);
    Enable_irq();
    …
    Set_io();                                   //初始化 I/O
    …
}
```

用户发送数据通过 ssize_t zigbee_write (struct file *filp,const char __usr *buf, ssize_t count,loff_t *f_ops)传到内核空间，然后调用构建数据包函数把数据打包发送出去。

用户控制下层命令，实现代码如下：

```
int zigbee_ioctl (struct inode *inode, struct file *filp,
unsigned int cmd, unsigned long arg)
{
    Switch(cmd)
    Case A:
        Set_state();                            //设置设备类型
    Case B:
        Set_restart();
    Case C:
        Set_start();
    Case D:
        Set_printf();                           //输出网络地址信息
}
Static zigbee_interrupt(int irq, void *dev_id)
{
    Flag=1;
    Set_restart();
    Outb(&buf, &add);
    Return IRQ_HANDLED;
}
```

除实现以上函数外，还需要实现 zigbee_relese(struct inode*inode, struct file*filp)，释放程序运行中所有资源。

本文通过上位机处理器 ARM9CS3C2410，设计了 ZigBee 内核字符驱动，轻松地实现了对下位机的控制，也方便了用户的上层开发，提供了用户与下位机数据传输的接口，避免了用串口进行数据传输时程序设计的繁琐性。由于篇幅限制本文没给出控制下层模块命令的具体实现代码。希望通过本文能促进 ZigBee 协调器驱动的进一步实现和研究。

参考文献

[1] 杨帆，廖桂平，李锦卫，等.无线传感器网络在农田环境信息监测中的应用[J].农业网络信息，2008(3)：20-23.

[2] 甘勇，王华，常亚军，等.基于 ARM 平台的 ZigBee 网关设计[J].通信技术，2009，42(1)：199-201.

[3] 魏守包，唐慧强.基于嵌入式 ARM-uClinux 的 ZigBee 网络设计[J].仪表技术与传感器，2009(1)：62-64.

[4] 包长春，石瑞珍，马玉泉.基于 ZigBee 技术的农业设施测控系统的设计[J].农业工程学报，2007，23(8)：160~163.

(收稿日期：2011-03-03)

作者简介：

李婧，女，1984 年生，硕士研究生，主要研究方向：计算机测量与控制技术研究。

史智兴，男，1954 年生，博士，教授，博导，主要研究方向：计算机检测控制教学与研究。

崔哲伟，男，1986 年生，硕士研究生，主要研究方向：计算机测量与控制技术研究。