基于 RTAI 的实时 Linux 系统构筑及其嵌入式程序移植

张 巍 李 俊 潘金贵

(南京大学计算机软件新技术国家重点实验室 南京大学计算机科学与技术系 南京 210093)

摘 要 实时 Linux 操作系统已逐渐被人们用作嵌入式应用软件的支撑平台,它在提供了一个优异的实时可控制性的同时,也不可避免地带来了一些相关问题。本文结合基于 RTAI 的实时 Linux 系统的构筑以及在其上开发一个 GPS 应用程序的经验,阐述了在 RTAI 系统上开发应用软件中遇到的诸如程序流程控制、串口实时传输和远程控制等几个关键问题,并探讨和给出了相应的解决方法。

关键词 实时 Linux,RTAI,内核,文件系统,串口

Real-time Linux Construction and Embedded Program Transplant Based on RTAI

ZHANG Wei LI Jun PAN Jin-Gui

(State Key Lab for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract In the field of embedded application, real-time Linux is used more and more as the support system of application software. Although real-time Linux provides good real-time property, it brings in some problems we must deal with. This paper states several key problems of developing software over RTAI, such as program process control, real-time data transmission through serial port and remote control, and gives the method for solving them according to the experience got from constructing RTAI and implementing a GPS application over it.

Keywords Real-time Linux, RTAI, Kernel, File system, Serial port

1 背景

如今,Linux 作为一个开源的操作系统被广泛地用于嵌入式系统,其主要优点是免费、支持平台多、开放性以及可定制性。另一方面,一些嵌入式设备为了能够工作,不仅需要一个功能强大的操作系统,同时要求系统能够满足实时性的要求,即系统除了逻辑正确外,还要能够在确定的时间内完成指定的任务。但是,Linux 是一个通用的分时操作系统,它被设计用来达到最佳平均性能,以让所有进程都可以在某段时间内获得 CPU[1],因而造成许多对时间要求严格的实时应用程序在 Linux 上得不到及时的响应,尤其在嵌入式领域,这种情况极其常见。因此,为了同时满足所有这些要求,实时 Linux 这个概念被提了出来,一些大学计算机系的实验室和开源组织各自对 Linux 系统进行研究,通过对 Linux 内核的改进,使其满足实时性要求。所以这些年出现了大量的实时 Linux 系统,如国外比较流行的 RT-Linux[1]、RTAI[2]、KURT[3]、REDICE-Linux[4]以及国内中科院软件所研究的基于红旗 Linux 的 RFRTOS[5]等等。

GPS 轮胎吊自动驾驶程序[6]是我们自行研制的用来完成集装箱码头堆场中的轮胎吊自动驾驶功能,它对整个堆场进行数字模型化,通过同时接收 GPS 信号和基站发出的差分数据,进行相应的计算和推导,得出控制命令,发送到控制器,控制器对轮胎吊进行自动的纠偏控制,从而实现自动驾驶。然而,此程序运行在 PC104 上的 DOS 系统之上,虽然目前可以满足基本的功能需求,但是由于 DOS 系统固有的缺点,诸如:①只能单任务顺序执行,不支持多任务;②对硬件的访问被直接嵌入到用户程序中,因此对硬件的操作不仅复杂,而且容易出错;③缺少系统接口,编程环境不完善,因此编写在 DOS 平台上运行的程序相对于 Linux 更加麻烦;④缺少对图形用户界面的支持,人机交互不友好,系统管理工具比较少,功能也比较单一,系统的可扩展性非常差;因此在程序维护和完善过程中,产生了很多麻烦,并且遇到了一些不可解决的问题,如不可对程序进行远程控制和调试。基于此点考虑,我们决定将此程序从 DOS 移植到实时 Linux 系统下。

在已存在的多个实时 Linux 操作系统中要选择一个作为 GPS 轮胎吊自动驾驶程序的支撑系统,需要做一番研究。首先,这个实时 Linux 系统必须支持现有的嵌入式设备,包括设备上的各种硬件接口,如串口、计数器、控制灯等等,并且要易于搭建;其次,这个实时 Linux 系统必须提供足够的实时性来满足应用程序的需要;再次,这个系统必须提供应用需要的功能 API 接口。按照以上要求,通过对各个实时 Linux 系统的调查研究,并根据查找到的数据资料和所做的实验,最终选择 RTAI 作为我们所用的实时 Linux 系统。RTAI 相比其他系统容易搭建、版本更新迅速、系统接口丰富,基本满足了所有的应用要求。

实时系统的确定只是第一步,接下来将进行系统的搭建和程序的移植。在将 GPS 轮胎吊自动驾驶程序移植到 RTAI 下时遇到了几个关键的问题。第一,RTAI 下应用程序的结构包括实时进程和非实时进程,所以必须对原程序的结构和流程进行重新设计;第二,一些程序中用到的接口(如串口)需要 RTAI 实时驱动的支持,同时需要对它的实时性能进行测试;第三,需要安装和配置一些工具软件来帮助远程控制和调试运行在 RTAI 系统中的应用程序。以下部分将详细叙述 RTAI 的搭建方法和程序移植过程中的这几个关键问题及其

张 巍 硕士研究生,主要研究方向为实时 Linux 和嵌入式程序开发;李 俊 讲师,主要研究方向为计算机图像处理和新型定位技术;潘金贵 教授、博士生导师,主要研究方向为多媒体信息处理等。

对策。

2 RTAI 系统的搭建

把程序从 DOS 移植到 RTAI 实时 Linux 下，必须完成两件事情：在嵌入式开发板上搭建一个 RTAI 实时 Linux 系统；重新设计和实现原来的基于 DOS 的程序。本节将介绍 RTAI 系统的建立过程。

2.1 系统搭建技术

建立一个 RTAI 系统，首先需要某一版本的 Linux 内核以及支持这一版本 Linux 内核的 RTAI 源程序压缩包。得到它们后，对其进行配置、编译，生成我们所需的二进制文件。启动一个操作系统还需要引导程序，Linux 下的两个经常使用的引导程序分别是 LILO 和 GRUB。GRUB 是目前大部分 Linux 发行套件的首选引导程序，对其配置相对于 LILO 更加容易，其功能也更强大，因此本系统选择 GRUB 作为引导程序。

一个完整的操作系统还必须包含有文件系统。文件系统管理整个系统的文件存储、创建、修改、删除等等。在一个只有内核的机器上建立新的文件系统有两种方法：一是拷贝一个已有的 Linux 文件系统，如从一个已安装 Redhat 的机器上拷贝一个文件系统；二是按照 Linux 文件系统的标准逐个建立目录和各个目录所必需的文件。第一种方法虽然比较方便，但是会造成文件系统非常庞大，占用大量的存储空间。第二种方法虽然更加复杂，但是由于是自己定制，所以可以去除大量不需要的文件，节省很多存储空间。还可以使用开源程序 busybox 来代替大量 Linux 下的工具，进一步减小文件系统占用的空间。由于嵌入式设备上的存储空间一般都很小，空间资源非常宝贵，所以选择第二种方法来构建嵌入式系统。图 1 展示了搭建 RTAI 系统需要完成的部分。

此外，必须对一些脚本文件进行配置。如开机将要执行的 shell 脚本文件 inittab 和 init.d 文件夹下的脚本文件，按照实际需要对它们进行修改。另外对一些网络配置文件进行修改来满足上网的需求。

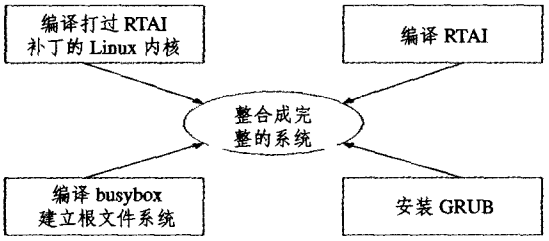

图 1 RTAI 系统搭建构成

2.2 系统搭建步骤

(1)编译 Linux 内核。首先选择 rtai-3.1 支持的内核版本 Linux-2.4.26，从 http://www.kernel.org 下载，文件名为 linux-2.4.26.tar.bz2。从 rtai-3.1 压缩包中找到文件 hal16-2.4.26.patch，然后按照说明对内核进行打补丁。完成后，内核版本被改为 linux-2.4.26-adeos。接着按照内核说明文件中的编译方法对内核进行配置和编译，根据硬件和软件的需求设置相应的配置选项。经过编译后生成一个名为 bzImage 的内核文件，大小为 1M 左右。

(2)编译 rtai。选择 rtai-3.1，从 http://www.rtai.org 下载，文件名为 rtai-3.1.tar.bz2。根据 rtai 压缩包中的安装说明文件进行配置和编译。编译完成后，在指定的路径下生成一个新的名为 realtime 的文件夹，其中包含了所有的 rtai 文件。

(3)编译 busybox 和建立根文件系统。选择 busybox-1.1.0，从 http://www.busybox.net 下载，名为 busybox-1.1.0.tar.bz2。根据 busybox-1.1.0 压缩包中的安装说明文件进行配置和编译。编译完成后，在指定的路径下生成一个名为_install 的文件夹，其下包含 bin 和 sbin 两个子文件夹以及 linuxrc(为指向 bin/busybox 的链接)，其中 bin 文件夹下包含可执行文件 busybox，所有 bin 和 sbin 中的其它可执行文件均为指向 busybox 的链接。

根文件系统是一个树型结构，最上层的是根目录，在其下层创建以下目录：/bin /dev /etc /home /lib /mnt /proc /root /sbin /tmp /usr /var。在各个目录下分别创建系统需要的各种文件，其中 bin 和 sbin 目录下的文件正是上面编译 busybox 生成的 bin 和 sbin 目录下的文件，编译好的 rtai 文件也被拷贝到某个目录下。最后把以上建立的文件系统制作成 ramdisk 映像文件，名为 ramdisk.img.gz。

(4)安装 GRUB。把装有一个新的 CF 卡的读卡器通过 USB 口插入开发平台所用的 PC，对其进行格式化并建立一个新的分区，大小为整个 CF 卡的容量，设置为可 bootable。在这个新的分区里建立/boot/grub 目录，然后把 PC 中 Linux 系统已有的 GRUB 文件拷贝到此目录下，执行 grub-install 命令安装 GRUB。

(5)整合系统。把前面编译好的内核文件和文件系统映像文件拷贝到 CF 卡上。在/boot/grub/下建立 GRUB 配置文件 grub.conf，对其进行修改和设置。至此，完成了 RTAI 系统的所有搭建工作。

3 RTAI 上程序移植及其关键问题的解决方法

由于 DOS 只支持单任务，因此原 GPS 轮胎吊自动驾驶程序只有一个进程在运行，所有需要完成的操作都按照顺序一一执行。而 RTAI 实时 Linux 是一个支持多任务的操作系统，当程序移植到 RTAI 以后，将采用更符合程序执行逻辑的多任务方式。

为了让程序在 RTAI 下满足实时性的要求，还需要做一些额外的工作。RTAI 系统要求其实时进程中不能出现 Linux 系统调用函数，如不能包含 I/O 读写系统调用、显示部分的 curses 调用，所以必须把这些调用从实时进程中分离出来。这样就把原来的系统分为两个部分：实时部分和非实时部分。实时模块中包含串口读写、GPS 纠偏计算与流程控制等操作，这部分有严格的时间与周期要求，必须达到一定的实时性；非实时模块中包含用户界面显示、记录日志以及键盘响应等部分，这个部分没有严格的时间限制，所以把它们从实时进程中分离出来。实时进程和非实时进程通过 RTAI 提供的 FIFO 进行交互。

这种设计方式不仅可以满足原系统的实时性要求，且更符合实际系统的运行逻辑，更加易于扩展和维护。当然要完全实现这些功能，必然会遇到一些问题，下文将给出所遇到的关键问题和其解决方法。

3.1 程序流程控制

如图 2 所示，在 DOS 下此程序为一个单任务进程，包含一个主循环，程序运行后，按照固定的周期和指定的顺序一次一次重复执行循环中的任务。1 秒钟可以完成 5 个周期。由于 DOS 单任务的特性，没有其他进程会抢占此任务，因此当确定好每次循环的执行时间后，此任务会严格按照这个设定

值一直运行，达到系统的实时性要求。

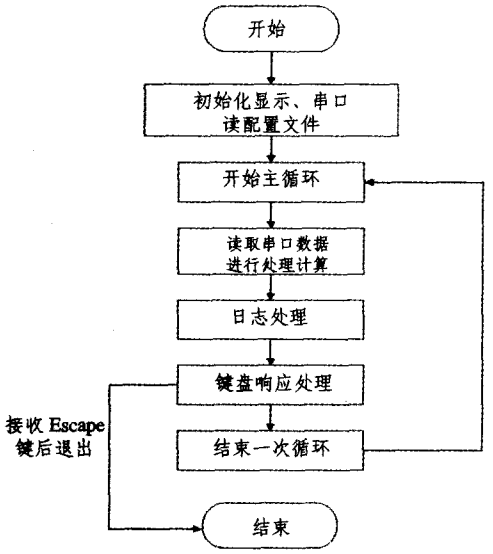

图 2　GPS 纠偏程序在 DOS 下的流程图

当把此系统移植到 RTAI 下时，将不得不对原程序的流程进行重新设计和实现。这里有个疑问，为什么不可以保持原有的程序执行流程不变，直接把它移植到 Linux 运行？的确，在最初移植的时候，曾经直接在普通 Linux 下对 GPS 轮胎吊自动驾驶程序进行了修改。但是，当实际运行的时候，出现了一些没有预想到的问题，GPS 串口数据经常丢失，导致结果错误。经过实验测试，找到了造成这个问题的根源：Linux 下串口响应和读取速度极其不稳定。下一小节将对此问题进行分析。

如图 3 所示，在 RTAI 下，原程序由一个进程转变为两个进程，即后端实时进程和前端非实时进程。这也是 RTAI 编程一个基本原则。在实时进程中包含三个主线程，分别是：GPS 纠偏与显示控制线程，日志记录控制线程和键盘响应控制线程。同样在非实时进程中也包含三个线程，分别是：界面显示线程，日志记录线程和键盘响应线程。非实时进程中的线程和实时进程中的线程一一对应，他们通过各自的管道进行数据传输和交互。

GPS 纠偏与显示控制线程是最重要的部分，它负责从串口读取 GPS 数据，然后进行相应的计算，最后把结果通过另外的串口传送给控制设备。它的响应时间决定了结果的正确与否，进而决定了整个应用程序的性能能否达到要求。GPS 纠偏与显示控制线程还控制着非实时进程中的显示界面。在每个执行周期，它都把需要在用户界面显示的实时数据写入 FIFO3，非实时进程中的界面显示线程在它的每个执行周期都从 FIFO3 中读取需要的数据，然后更新用户界面。实时进程中日志记录控制线程在每个执行周期把需要记录的信息写入 FIFO2，非实时进程的日志记录线程在每个执行周期从 FIFO2 中读取需要的数据，然后记录到日志文件。非实时进程中的键盘响应线程捕捉用户的键盘输入，然后把此信息写入 FIFO1，实时进程中的键盘响应控制线程在每个周期查看 FIFO1，如有数据，则读入，然后进行相应的处理。通过以上的设计方法，既可以达到程序的实时性要求，又能满足功能性需求，还具有良好的模块性和可扩充性。例如，当需要更改显示界面风格时，我们只需要对界面显示线程进行修改，其他部分保持不变。当需要增加功能时，只需要再加入一个线程。

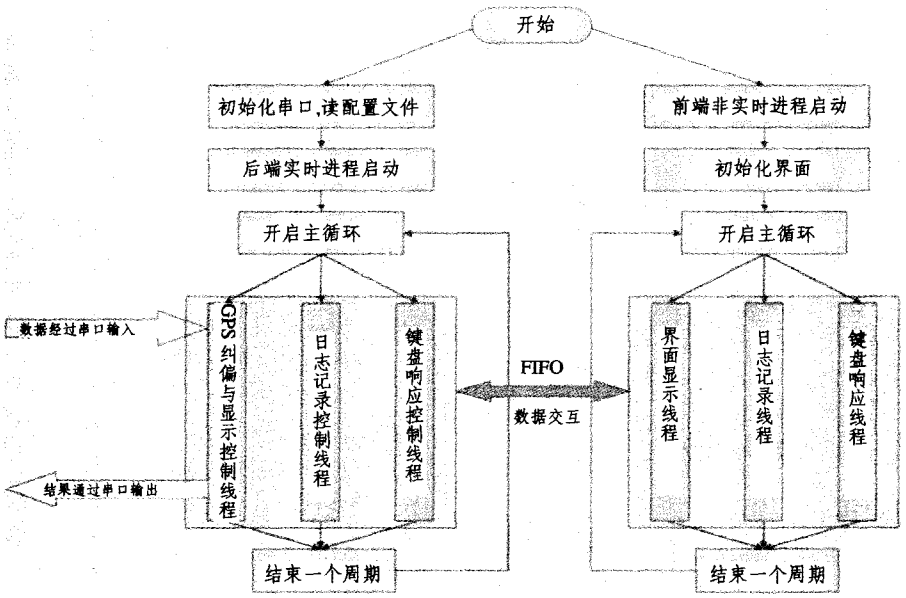

图 3　GPS 纠偏程序在 RTAI 下的流程图

3.2　串口通信和处理实时性

串口在 GPS 轮胎吊自动驾驶程序中起着至关重要的作用，对串口的处理速度决定着程序是否可以按照预期的目标正常运行。图 3 显示 GPS 纠偏与显示控制线程负责与串口进行通信，它是主循环中的三个线程之一，严格按照程序设置的周期运行，因此它读写串口的频率和程序的运行频率完全一致。当程序主循环的频率很高时，它对串口的读写频率相应地也很高，这时就必须确保它能够快速且正确的读写串口。

在将此程序移植到 RTAI 系统之前，曾经把此程序进行简单的修改后，放在普通的 Linux 系统下运行。但是经过实际的测试后发现，在 Linux 下 GPS 轮胎吊自动驾驶程序完全不能正常运行。经过详细分析后得出原因：在 Linux 下对串口响应和数据的读取速率相当不稳定，造成在有些周期数据丢失，进而导致程序失效。为此，设计了一种测试方法，分别

在DOS、Linux和RTAI对串口读写速度进行测试，给出一个统计结果，以此来作为操作系统平台选择的依据。

目标测试平台为PCM3348，主频为133MHz，内存为64M；辅助测试平台为一台装有P4双核处理器的PC。测试方法如下：分别在PCM3348上运行DOS、Linux和RTAI下的串口数据接收程序，在PC上用串口调试助手以115200b/s的速度分别以1000ms、200ms、100ms的周期发送128字节数据，运行在PCM3348上的串口数据接收程序以同样的周期接收数据，每个周期记录从接收第一个字节开始到接收最后一个字节结束的时间差。以上每种情况都分别进行1000次串口数据发送，记录1000个样本值，然后对其求平均值和标准差。

表1给出了以100ms为周期发送数据的统计结果，另外两种周期的统计结果类似，此处省略。图4给出了平均读取时间统计柱状图。从表中可以看出RTAI下读取串口数据的速度最快，读取128个字节平均只需要5us左右；Linux其次，约需要10ms；DOS最慢，约需要16ms。DOS和RTAI相差竟然达到3000倍。另一方面，RTAI和DOS下的标准差都很小，说明其读取速度非常稳定，而Linux下读取时间相差较大，慢的时候甚至比DOS下读取还要慢，这也解释了当GPS轮胎吊自动驾驶程序中运行在Linux系统时，当系统负载过重时，可能导致串口数据丢失，程序失效。

表1 在不同系统下读取串口数据的速度统计值(1000次)

系统类型	平均时间	标准差	最大值	最小值
RTAI	0.00518ms	0.00028ms	0.00596ms	0.00440ms
Linux	10.1989ms	2.0592ms	20.7790ms	3.6830ms
DOS	16.2349ms	0.1060ms	16.4175ms	15.5777ms

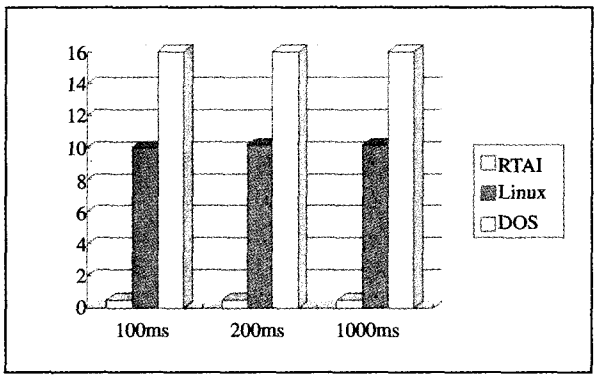

图4 串口平均读取时间柱状图

这一现象也反映了各个操作系统的不同实现机制。RTAI下通过串口驱动对串口数据进行读取时，串口数据已经放在了系统的临时缓冲区中，实时任务不经过系统调用，直接一次把所有数据返回，速度稳定而又快速。DOS下对硬件的操作直接包含在应用程序中，由应用程序开发者决定，一般采用一个字节一个字节的读取，故速度较慢，但是时间稳定。Linux下读取串口的程序为普通优先级任务，读取数据时可以被其他进程抢占，因此会产生不稳定的表现。

从表1中还可以看出，当串口传输速率达到几十毫秒数量级时，Linux和DOS由于自身的缺陷将很难再满足应用程序的需求，此时RTAI就是最好的选择了。

3.3 远程控制和文件传输

许多嵌入式开发板使用CF卡作为外存储器使用，为了修改系统配置、修改程序或者调试程序时，过去的做法通常是把CF卡拔下，通过读卡器在PC上进行修改。这种做法的缺点显而易见。首先，在系统配置和开发阶段，每做一次系统或者程序的修改，必须完成一遍如下过程：关闭嵌入式开发板上的RTAI操作系统，拔下CF卡，插入读卡器后把读卡器再插入PC上，修改其上的配置文件或者程序，拔下CF卡读卡器，把CF卡重新插入嵌入式开发板，启动RTAI操作系统。由此可见这是一个极其浪费时间的过程。另外，在程序投入运行阶段，如果程序出现问题，进行调试和修改时仍然必须重复如上的步骤。而且由于开发板安装位置的原因，会造成更大的时间和人力代价。造成这一结果的主要原因是由于过去DOS系统不支持网络设备，因此没有办法进行远程调试和控制。

Linux操作系统为我们改变这种情况提供了一个良好的基础。许多开源组织和个人专为Linux开发了各种各样的使用工具，我们可以使用其中的一些工具来帮助我们对RTAI系统进行远程控制以及文件传输。Busybox中为我们提供了telnetd服务器端，只要在PC上装有telnet客户端，就可以通过telnet方式登录开发板上的RTAI系统就行远程控制，进行在线修改(即在开机的同时进行同步的修改)等等。如图5(A)所示。

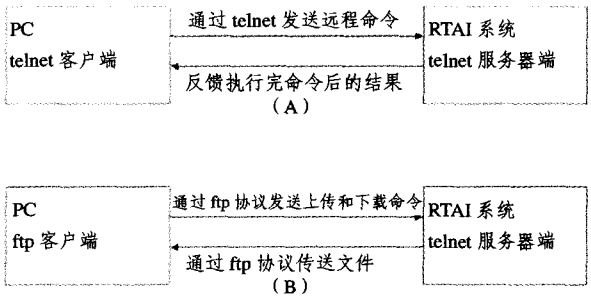

图5 远程交互图

为了实现文件的传输，在嵌入式设备的RTAI系统中安装一个FTP服务器，在PC上装一个FTP客户端，这样通过FTP就可以实现从PC到嵌入式设备的文件传输。在PC上修改一个文件，然后通过FTP传输到嵌入式设备上，完成对原有的文件的更新。如图5(B)所示。

结论 通过对以上问题的探讨和实验，成功地解决了GPS轮胎吊自动驾驶程序中所遇到的程序流程控制、串口实时传输和计算、远程控制等问题。通过全面的测试，表明运行在RTAI上的GPS轮胎吊自动驾驶程序能够以100ms为周期完成实时数据接收、实时计算、实时发送控制命令，辅助码头中轮胎吊的自动驾驶。在程序运行期间，可以通过网络进行telnet远程控制和调试，通过ftp对文件进行更新。对这些方法进行扩展，可以用在各种不同的嵌入式应用程序开发中。

参考文献

1 Barabanov M, Yodaiken V. Introducing real-time Linux. Linux Journal, 1997,(34):19～23

2 Mantegazza P. Dissecting DIAPM RTHAL-RTAI (with some comparisons to NMT-RTL, here and there) (draft). Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano. http://www.aero.polimi.it/～rtai/documentation/articles/paolo-dissecting.html

3 Srinivasan B, Hill R, Pather S, et al. A firm real-time system implementation using commercial off-the-shelf hardware and free software. In: Proceedings of the Real-Time Technology and Applications Symposium. Denver, 1998. 112～119

4 Wang Y C, Lin K J. Implementing a general real-time framework in the RED-Linux real-time kernel. In: IEEE Real-Time System Symposium, 1999. 246～255

5 李小群，等. RFRTOS：基于Linux的实时操作系统. 软件学报，2003,14(07):1203～1212

6 闫德鑫，等. GPS在轮胎吊自动驾驶中的应用. 计算机工程与应用，2005,41(33):206～210