基于 Qt/Embedded 的嵌入式 GUI 显示架构实现

王 凯
(南京航空航天大学 计算机科学与技术学院,江苏 南京 210016)

摘 要:嵌入式操作系统 VxWorks 原有的图形支持 WindML 对于开发高级 GUI 图形应用比较困难,WindML 不支持可视化的图形界面开发,也没有大量的图形控件供开发者使用。通过对 Qt/Embedded 的图形驱动架构和 Qt/Embedded 的服务器/客户端架构的研究,以及对 VxWorks 下的图形库 WindML 的显示体系结构和图形驱动支持能力的分析,提出了一个 VxWorks 下的基于 Qt/Embedded 的 GUI 显示架构。该显示架构通过引入 Qt/Embedded 嵌入式支持层,将 Qt/Embedded 库内帧缓冲驱动支持类和嵌入式系统 VxWorks 图形支持库 WindML 融合在一起,实现了 WindML 在源码层级对 Qt/Embedded 库的支持。给出了 Qt/Embedded 嵌入式图形显示支持的具体实现方法,通过修改 Qt/Embedded 库的 VxWorks 图形显示支持部分的源代码,使用 WindML API 获取到了上层 Qt 帧缓冲驱动需要的宽度、高度、像素格式、帧缓冲地址等重要信息,实现了图形显示设备的打开,注册以及访问功能,建立了基于 Qt/Embedded 的 GUI 显示系统。

关键词:Qt/Embedded;嵌入式系统;VxWorks;图形用户界面
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2017)05-0144-05
doi:10.3969/j.issn.1673-629X.2017.05.030

Implementation of Embedded GUI Display Architecture Based on Qt/Embedded

WANG Kai
(College of Computer Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing 210016, China)

Abstract: It is very difficult for the development of advanced GUI graphics applications that the original graphics support WindML of embedded operating system, VxWorks. WindML does not support the visual graphical interface development and can't provide large number of graphical controls for developers to use. Through the research on the Qt/Embedded graphics driver architecture and Qt/Embedded server/client architecture, as well as the analysis of the display architecture and graphics drivers support capabilities of the WindML graphics library under VxWorks, a VxWorks GUI with QT/Embedded display architecture has been implemented which can display architecture by introducing the embedded QT/embedded support layer. Combined both QT/embedded database frame buffer drive support and embedded system VxWorks graphics library WindML, the WindML support to QT/embedded library at the source code level has been achieved. The concrete realization method of Qt/Embedded embedded support has been presented. By modifying the source code part of Qt/Embedded Library that supported graphics display and by using WindML API to get the important information such as width, height, pixel format, frame buffer address that is needed by the upper Qt/Embedded layer, opening, registration and accessing to the graphics display device have been achieved and thus GUI display system with Qt/Embedded has been realized.

Key words: Qt/Embedded; embedded system; VxWorks; graphical user interface

0 引 言

VxWorks 操作系统是美国的 WindRiver 公司于 1983 年设计开发的一种嵌入式实时操作系统(RTOS),是嵌入式开发环境的关键组成部分。VxWorks 系统拥有良好的持续发展能力、高性能的内核以及友好的用户开发环境,以其良好的可靠性和卓越的实时性在军事、通信、航空、航天等实时性要求极高的领域中应用广泛[1]。近年来,嵌入式领域图形界面得到了越来越广泛的应用,为用户提供一个友好的人机交互界面已成为嵌入式开发的一个重要组成部分。

收稿日期:2016-06-01 **修回日期**:2016-09-08 **网络出版时间**:2017-03-13
基金项目:国防科工局"十三五"重点基础科研项目(JCKY2016206B001)
作者简介:王 凯(1984-),男,硕士研究生,研究方向为软件体系结构。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20170313.1546.070.html

随着嵌入式设备硬件的飞速发展,嵌入式领域图形 GUI 技术已经日趋成熟,用户对于嵌入式设备人机交互的要求也越来越高,人们需要的不是晦涩难明的字符界面,而是直观,操作逻辑简单,功能强大的图形 GUI 交互功能。

Qt/Embedded 是 Trolltech 公司推出的一个跨平台的 C++图形用户界面应用程序框架。它给程序开发者提供了建立图形用户界面所需的所有功能[2]。Qt/Embedded 库是完全面向对象的,很容易扩展,并且允许真正的组件编程。Qt/Embedded 库是一套开源的 C++图形用户界面库[3],拥有种类丰富的图形对象可供开发者使用,采用名为 signal/slot 信号槽的事件信号传递机制。它拥有跨平台的集成开发环境 Qt Creator,对于开发者来说,开发难度低,开发环境友好。同时,它拥有面向嵌入式系统的版本,丰富的控件资源以及良好的可移植性,完全能够满足用户对 GUI 界面的需要[4]。通过分析 Qt/Embedded 库和 VxWorks 图形支持库 WindML 的体系结构、源码,提出了一种新的基于 Qt/Embedded 库的 VxWorks 下的图形支持架构。在该架构中,增加了 VxWorks 图形支持库 WindML 的支持,针对在嵌入式系统 VxWorks 下移植和使用 Qt/Embedded 库的问题,进行了深入研究。

1　Qt/Embedded 的体系结构

1.1　Qt 图形驱动架构

Qt/Embedded 库是 Qt 图形界面库的嵌入式版本,为了适应嵌入式操作系统环境,进行了许多改动和调整,它在构建 GUI 环境时放弃使用原来的 X Window 体系,因此也就不需要使用较大的 Xlib 库,所以内存占用十分小,能够适应嵌入式开发的需求。在放弃了 X Window 体系之后,Qt/Embedded 库底层使用帧缓冲体系作为底层图形接口[5],并使用输入事件作为具体的输入设备的抽象,比如 keyboard 和 mouse 输入事件等;在上层,Qt/Embedded 库继续使用原本的 Qt 架构,从而保证用户使用的便利性和一致性,并能够与 X Window 系统下的程序部分兼容,使得用户便于进行程序移植[6]。图 1 给出了 Qt/Embedded 库的软件架构。

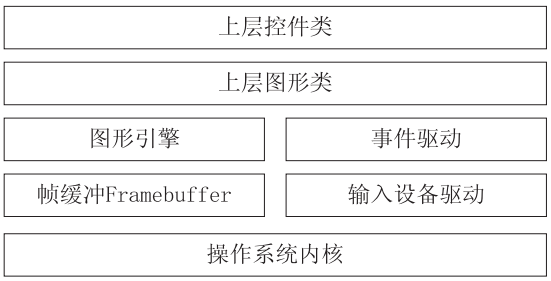


图 1　Qt/Embedded 的软件架构

Qt/Embedded 库的软件总体架构可分为以下三层:

(1)上层控件层:直接提供用户使用的 GUI 控件,给用户界面开发提供了基于 widget 的 GUI 开发控件。

(2)上层图形层:该层是低级别的绘图层,提供了对帧缓冲的绘图操作,在这一层中,需要获得帧缓冲驱动体系所需的具体硬件信息,从而为下一层驱动提供支持,与此同时,完成对底层帧缓冲驱动图形操作的抽象,供给上层使用。

(3)图形引擎和事件驱动层:在这一层中直接进行对底层帧缓冲驱动的访问,并将具体的输入设备抽象为事件。

图形帧缓冲(Framebuffer)体系架构是 Qt/Embedded 图形引擎的基础,Framebuffer 驱动最重要的功能就是给用户提供一个进程空间映射到实际物理显示内存的接口,该接口通过系统调用 mmap 来实现[7-8],具体原理如下:在帧缓冲体系中,显示硬件设备被抽象为帧缓冲区,从开发者的角度看,向帧缓冲写入指定格式的数据即是向屏幕输出内容,整个帧缓冲就对应屏幕上图像的一帧。通过不断向 Framebuffer 中写入数据,显示控制器就自动地从 Framebuffer 中取数据并显示出来。在 VxWorks 下的 GUI 显示体系结构中,帧缓冲位于上层 GUI 库和底层具体的显示硬件设备之间,目的是为了屏蔽各种显示硬件之间不同的操作[9]。这种图形帧缓冲体系结构也获得了 VxWorks 系统内核的支持,能够应用于 VxWorks 系统上。

1.2　Qt/Embedded 的客户端/服务器架构

Qt/Embedded 框架依然采用了客户端/服务器的架构,所以,所有基于 Qt/Embedded 的用户程序运行前必须连接到 Qt/Embedded 服务器。在服务器端,Qt/Embedded 对键盘、鼠标、屏幕等输入输出设备进行管理,同时也需要对各个客户端创建的窗口进行管理,处理窗口的创建与销毁、窗口的重叠与移动、窗口焦点的转换等操作。客户端重要的工作是处理服务器端发送的图形事件,在绘制完毕后通知服务器端并由服务器端显示。客户端的其他工作还包括处理窗口内部 GUI 控件的管理工作,绘制 Widget,对网络访问进行控制,处理文件系统的读写操作,等等。Qt/Embedded 的客户端/服务器架构如图 2 所示。

由于 Qt/Embedded 库为了适应嵌入式平台的硬件特点,做了许多兼容工作,并对 Server 端进行了简化,从而使得 Qt/Embedded 库移植的困难程度大大降低,只需要实现键盘、鼠标和屏幕显示驱动,就能实现 Qt/Embedded 的移植。

从灵活性和节省资源的角度考虑,Qt/Embedded 的 Server 端并不是一个独立的进程,而是包含在嵌入式系统下第一个启动的 Qt/Embedded 进程内;从开发

者的角度来说,它和普通的 Client 端进程表现相同,但这个进程控制着对所有输入输出设备的访问。

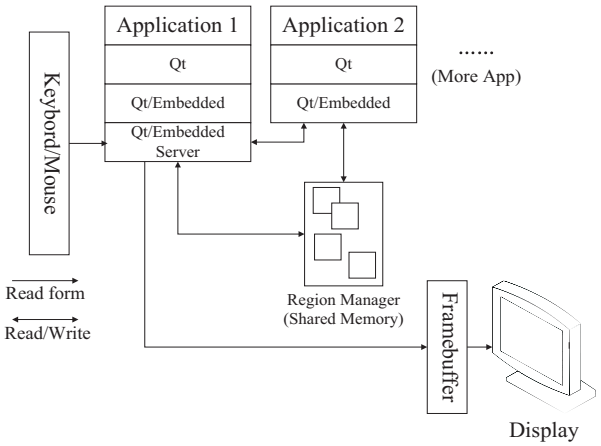


图 2 Qt/Embedded 的客户端/服务器架构

2 VxWorks 图形体系结构

WindML(Wind Media Library)是嵌入式操作系统 VxWorks 下的多媒体库[10],它提供基本的图形、视频、音频开发接口供 VxWorks 下的开发者使用。WindML 库提供了一套完整的多媒体设备驱动开发框架(见图 3),同时附带了一系列用来处理输入输出设备和过程事件的工具。

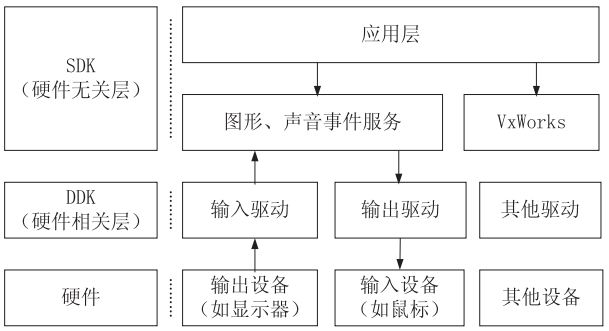


图 3 WindML 多媒体开发组件框架

WindML 中共有 2 个子组件:软件开发工具包(SDK)和驱动开发工具包(DDK)。

SDK 用于开发上层应用,它提供输入输出设备、字体、音频和图形的 API 集合,使得开发者可以编写面向硬件透明的代码,并在不同的硬件平台下测试和运行。

DDK 用于开发底层对应特定硬件的驱动程序。在 DDK 中,包含许多通用硬件的驱动程序模板,从而使得开发者可以快速开发出特定驱动的 API 函数,同时,DDK 为开发者提供了可定制性和可扩展性。

在 VxWorks 下,由 WindML 为 Qt/Embedded 库提供帧缓冲驱动的嵌入式支持。WindML 图形驱动支持主要通过 2D 应用层、通用图形库和图形设备驱动层与图形硬件进行通信[11],WindML 图形驱动支持体系结构如图

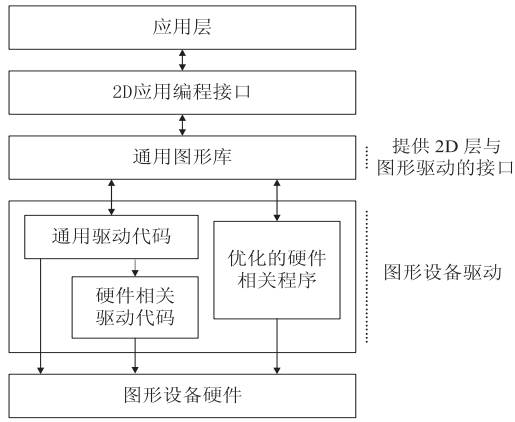


图 4 WindML 图形驱动支持体系结构

其中,2D 应用层通过通用图形库中的 UGL 图形接口对图形设备驱动(ugl_ugi_driver 结构)进行访问。驱动程序主要提供符合上层 UGL 图形调用接口并能驱动显卡硬件的一些接口函数[12]。ugl_ugi_driver 结构如下所示:

```
typedef struct ugl_ugi_driver * UGL DEVICE ID;
typedef struct ugl_ugi_driver{
/*数据成员*/
UGI_MODE *pMode; /*显示模式*/
UGI_PAGE *pPageZero; /*第一个显示页面地址*/
void * extension; /*可选的扩展驱动部分*/
/*通用函数*/
UGI_STATUS( *info)(struct ugl_ugi_driver * pDriver,
UGI_INFO_REQinfoRequest,void * info);
UGL STATUS( *destroy)(struct ugl_ugi_driver *
pDriver);
/*支持的模式*/
UGL STATUS( *modeSet)(struct ugl_ugl_driver *
pDriver,UGL_MODE * pMode);
.....
}
```

VxWorks 下任何显卡驱动都是将主要功能函数,如模式设置、公用画点、画线等图形函数的函数指针注册到 UGL_DEVICE_ID 结构体中,因此,虽然底层硬件不同导致函数功能实现方式不同,但面向上层时的函数接口保持一致性,函数的名称和参数都是相同的。

3 Qt/Embedded 图形界面架构实现

3.1 VxWorks 下的 Qt/Embedded 架构

通过 Qt/Embedded 的图形引擎类提供给 Framebuffer 设备的支持和 WindML DDK 提供的 API 集合支持,可以提出一种自上而下的嵌入式实时系统 VxWorks 下的 GUI 显示体系结构,如图 5 所示。

从图 5 可以看出,Qt 帧缓冲驱动依赖于 Qt/Embedded 嵌入式支持,Qt/Embedded 嵌入式支持又是依靠 WindML DDK 函数与底层 VxWorks 操作系统进行

通信。通过关键部分 Qt 帧缓冲驱动和 Qt/Embedded 嵌入式支持,构建起 VxWorks 下的 GUI 显示体系结构。

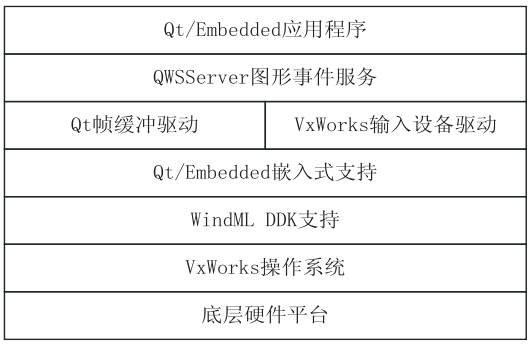


图 5　VxWorks 下的 Qt/Embedded 架构

Qt 帧缓冲驱动以类的形式存在于 QtGui 源代码中,该类继承自 QScreen,通过实现这个类的具体功能代码,使得 Qt 中的 QWSServer 图形事件服务高效而稳定的运行。在实现功能的过程中,使用 WindML DDK 函数优化帧缓冲驱动获得硬件相关信息的过程(比如屏幕分辨率、缓冲地址等),快速而有效地完成帧缓冲驱动作为上层应用和底层显示设备之间抽象层的功能。同时,通过对图 5 中 Qt/Embedded 嵌入式支持中间层的修改,实现 WindML DDK 对 Qt 帧缓冲驱动的功能支持。

3.2　Qt/Embedded 嵌入式支持实现

VxWorks 下构建一个完整的 GUI 显示体系结构需要从上层界面到底层系统的完整支持。上层准备使用 Qt/Embedded 图形引擎,它是基于 C++类库实现的,Qt/Embedded 类库封装了适应不同操作系统的访问细节,能够跨平台实现 GUI 的显示功能。Qt/Embedded 库对目标系统的要求是:只要这个系统有一个线性地址的缓冲帧并支持 C++的编译器[13]。它通过自带的帧缓冲驱动与底层操作系统进行通信联系。VxWorks 系统本身支持 GUN C++的编译[14],同时该系统的显示驱动模式支持帧缓冲驱动抽象的工作方式,通过帧缓冲这个桥梁,Qt/Embedded 库可以很好地与 VxWorks 系统结合在一起。

Qt/Embedded 中,QScreem 类是抽象出底层显示设备的基类,其中声明了对底层显示设备的基本描述和操作方式,Qt/Embedded 库对具体的显示设备,如 VxWorks Framebuffer 做的抽象类接口都是继承并重载基类中的函数来实现。Qt/Embedded 库帧缓冲驱动的实现类为文件 qscreenvxworksfb_qws. cpp 中的 QvxWorksFbScreen 类,下面给出 qscreenvxworksfb_qws. cpp 文件中的 Qt/Embedded 帧缓冲的主要代码:

```
bool QVxWorksFbScreen::connect(const QString &)
{
    if(! vxDisplayOpen()) return false;
    //调用设备打开主函数,失败则返回
    int format;
    vxDisplayGet(&w,&h,&d,&format,&data,&lstep);
    //获取显示设备信息
    dw=w;
    dh=h;
    mapsize=size=h * lstep;
    ……(略)
    QWSServer::setDefaultMouse("vxworks");
    QWSServer::setDefaultKeyboard("vxworks");
    //设置鼠标键盘
    return true;
}
```

从代码中可以看出,Qt/Embedded 帧缓冲驱动的功能主要通过 vxDisplayOpen 和 vxDisplayGet 两个函数来实现。vxDisplayOpen 实现打开显卡设备并向 VxWorks 系统注册该设备,获取设备资源;vxDisplayGet 实现获取显示的具体信息,比如 LCD 屏幕的宽度、高度等相关信息,同时,给基类 QScreen 的成员变量 data 赋值,该变量保存了在 Qt/Embedded 嵌入式支持层获取的帧缓冲映射地址。在 Qt/Embedded 帧缓冲驱动代码的最后,向 QWSServer 图形事件服务注册了使用的输入设备驱动,在这里,键盘鼠标都使用了 VxWorks 系统驱动。

如上所述,Qt/Embedded 帧缓冲驱动通过调用 vxDisplayOpen 和 vxDisplayGet 两个函数来获得 Qt/Embedded 嵌入式支持层的支持,这两个函数具体实现在 qvxworksdisplay. c 文件中,vxDisplayOpen 函数主要代码如下:

```
int vxDisplayOpen(void)
{
    char * pDisplayName=UGL_NULL;
    UGL_UINT32displayNumber;
    UGL_STATUS status;
    UGL_REG_DATA * pRegistryData;
    UGL_FB_INFOfbInfo;
    status = uglDisplayOpen ( UGL _ NULL, &pDisplayName,
&displayNumber);
    //使用 UGL 库函数根据默认名打开显示设备
    ……(略)
    pRegistryData = uglRegistryFind ( UGL _ DISPLAY _ TYPE,
&displayNumber,0,0);
    //使用 UGL 库函数注册显示设备
    if(pRegistryData==UGL_NULL)
    {
    printf("Display not found. \n");
    uglDisplayClose( UGL_NULL);
    return FALSE;
```

```
}
vxDisplayId=(UGL_DEVICE_ID) pRegistryData->id;
//显示设备注册成功后,从注册结构体中提取显示 ID
……(略)
return TRUE;
}
```

在上述代码中,通过 WindML DDK 函数 uglDisplayOpen 打开显示设备,然后通过调用 uglRegistryFind 函数向 VxWorks 系统注册显示设备,并通过 pRegistryData->id 获得了 UGL_DEVICE_ID 结构体的相关信息,该结构体包含了所有的图形接口信息,并把 UGL_DEVICE_ID 结构体的信息保存在 vxDisplayId 变量中。

在 vxDisplayGet 函数中,使用 vxDisplayId 变量,调用 WindML DDK 的 uglInfo 函数,获取到了上层 Qt 帧缓冲驱动需要的宽度、高度、像素、像素格式、帧缓冲地址等重要信息,主要代码如下:

```
void vxDisplayGet(int *width,int *height,int *bitsPerPixel,
int *pixelFormat,unsigned char **data,int *strideBytes)
{
……(略)
uglInfo(vxDisplayId,UGL_MODE_INFO_REQ,&modeInfo);
//根据显示设备 ID,获取显示相关信息
uglARGBSpecGet(modeInfo.colorFormat, &colorSpec);
//获取颜色相关信息
*width=modeInfo.width;
*height=modeInfo.height;
*bitsPerPixel=colorSpec.numBytesPerARGB*8;
*pixelFormat=modeInfo.colorFormat;
……(略)
*data=(unsigned char *)modeInfo.fbAddress;
//通过 modeInfo 结构体提取帧缓冲地址传入到 Qt 类中
……(略)
}
```

至此,通过对 Qt/Embedded 内部的 VxWorks 系统支持源代码的分析和修改,获取到了 Qt 帧缓冲驱动的所有重要硬件信息,完成了帧缓冲驱动的基本功能,从而实现了 Qt/Embedded 嵌入式支持层对 Qt 帧缓冲驱动层的支持,也实现了对 Qt/Embedded 在 VxWorks 系统下的移植。

3.3 Qt/Embedded 显示支持代码框架结构

在 Qt 库的 QVxWorksFbScreen 类中,通过在 connect 函数中调用的两个重要函数 vxDisplayGet 和 vxDisplayOpen,实现了 Qt 库与 WindML 库 UGL 库函数的结合,通过 UGL 库函数获取到了显示宽度、高度、像素信息等图形显示关键参数,同时获取到了显示帧缓冲内存地址等重要信息,从而实现了 Qt/Embedded 库在 VxWorks 环境下对显示设备的支持。显示支持代码框架结构如图 6 所示。

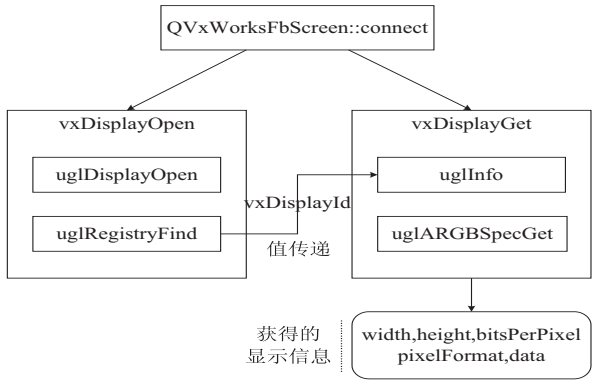


图 6 Qt/Embedded 显示支持代码框架结构图

最后,在交叉编译环境下编译修改后的 Qt/Embedded 库源代码,生成. so 库文件,即可在 VxWorks 操作系统环境下加载使用。

4 结束语

为了实现 VxWorks 下便捷、易用的图形 GUI 系统,实现 Qt/Embedded 库与 VxWorks 的完美兼容,在研究分析 Qt/Embedded 库的图形驱动架构和 VxWorks 系统的图形显示体系结构以及在 VxWorks 下实现图形 GUI 原理的基础上,通过对 Qt/Embedded 库帧缓冲驱动的二次开发修改,实现了 Qt GUI 界面在 VxWorks 下的完美运行。Qt/Embedded 延续了 Qt 在桌面系统的所有功能,丰富的 API 接口和基于组件的编程模型使得嵌入式 VxWorks 系统中的 GUI 应用程序开发更加便捷。

参考文献:

[1] 向孝龙,陈 欣,李春涛. 基于 RTW 和 VxWorks 的无人机仿真系统设计与实现[J]. 计算机测量与控制,2014,22(4):1291-1293.

[2] 吴燕燕,贺锋涛. 基于 ARM9 平台上 Qt/Embedded 的移植与开发[J]. 液晶与显示,2013,28(2):261-265.

[3] 杨 柳,岳 坤,庞和明,等. Qt/Embedded 及嵌入式 Linux 在智能监控系统控制中的应用[J]. 计算机应用,2010,30(S1):289-291.

[4] Chen F,Fan X. Embedded system's performance analysis with RTC and QT[C]//International symposium on advanced parallel processing technologies. Guangzhou, China:[s. n.], 2007:569-579.

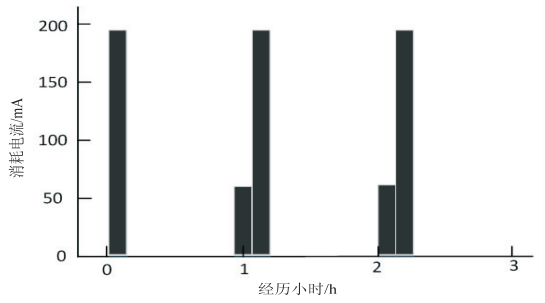
[5] 连照亮,徐世国. 基于 Qt/Embedded 在嵌入式 linux 下的应用研究[J]. 微计算机信息,2010,26(17):81-82.

[6] 曾剑元. ARM9 平台上基于 Qt/Embedded 的嵌入式 GUI 的研究与实现[D]. 长春:吉林大学,2010.

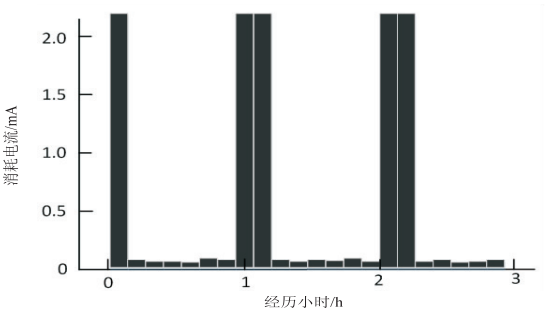
[7] Yang L,Sander P V,Lawrence J. Geometry-aware framebuffer level of detail[J]. Computer Graphics Forum,2008,27(4):1183-1188.

(下转第 153 页)

调度时间,提高了空箱的利用率。



(a)运行 3 小时功耗图



(b) 放大后的功耗图

图 6　监控终端功耗测试效果图

5　结束语

集装箱的实时调度不仅可以提升工作效率并且大大降低了管理运营成本,实时性关键在于对集装箱的实时监测并及时处理。实时调度系统中的关键点是集装箱空箱的高效利用,其作为促进集装箱货物运输的根源,国内外都在研究相关的空箱调度的有效解决方案,但大多是以调度的管理模式和优化方案为主,很少涉及监测技术。为了弥补相关的技术空缺,将具体问题量化成为各种待监控参数,研发监控终端和可视化平台,运用 DCM 技术架构将具体问题用物联网的手段解决并将结果通过平台呈现出来,从而从技术层面克服了空箱闲置鲜为人知的缺陷和箱体运输过程状态未知的问题。

参考文献:

[1] Kang T W, Ju S M, Liu N. Research on the empty container transportation management innovation for import and export enterprises[C]//International symposium on management of technology. [s. l.]:[s. n.], 2012:262-265.

[2] Drewry. Annual container market review &:forecast 2006/07 [R]. London:Drewry Shipping Consultants Ltd, 2006.

[3] 吉清凯,胡祥培,孙丽君. 集装箱空箱调度问题的研究现状与发展[J]. 系统工程理论与实践, 2014, 34(6):1578-1586.

[4] Jiang Y N, Hao S C. Research on the development of intelligent logistics based on Internet of Things[C]//International conference on remote sensing. [s. l.]:[s. n.], 2011:5254-5257.

[5] 吕洁印,周受钦,曹广忠. 面向冷藏箱箱温的智能化监控系统研究[J]. 自动化与信息工程, 2013, 34(2):28-32.

[6] 吕洁印,周受钦,曹广忠. 基于无线射频传输的大型车辆智能辅助倒车系统设计[J]. 电子设计工程, 2013, 21(17):67-70.

[7] 邓延洁,倪　鹏,蒋　鑫,等. 内河水上危险货物集装箱运输监测系统研究[J]. 交通信息与安全, 2011, 29(5):99-105.

[8] Xu D H, Zhang X F, Deng Q C. Study on intelligent logistics system:based on Internet of Vehicles[C]//International conference on ubiquitous intelligence and computing. [s. l.]: IEEE, 2015:681-685.

[9] Ma J, Tang S Y. Container real-time positioning on the terminal[C]//International conference on logistics systems & intelligent management. [s. l.]:IEEE, 2011:1069-1071.

[10] 余　雷,许宏科,胡　欣. 基于物联网的远程视频监控系统设计[J]. 计算机技术与发展, 2016, 26(4):139-143.

[11] 关　勇. 物联网行业发展分析[D]. 北京:北京邮电大学, 2010.

[12] 孙科学,洪　樾,章康宁,等. 一种联合检测门禁系统的设计与实现[J]. 计算机技术与发展, 2016, 26(1):155-159.

[13] Xu Z Q, He J L, Chen Z Y. Design and actualization of IoT-based intelligent logistics system[C]//IEEE international conference on industrial engineering & engineering management. [s. l.]:IEEE, 2012:2245-2248.

[14] 程青青,姚振强,胡永祥. 危险品集装箱运输远程监控平台设计与实现[J]. 计算机工程, 2009, 35(14):192-194.

(上接第 148 页)

[8] Mao C, Johnson K M. Fast-switching liquid-crystal-on-silicon microdisplay with framebuffer pixels and surface-mode optically compensated birefringence[J]. Optical Engineering, 2006, 45(12):1269-1278.

[9] 吴子平,徐爱钧. 基于 Qt/Embedded 的嵌入式 GUI 的研究与构建[J]. 电脑开发与应用, 2012, 25(1):13-16.

[10] Battye T G G, Kontogiannis L, Johnson O, et al. iMOSFLM:a new graphical interface for diffraction-image processing with MOSFLM[J]. Acta Crystallographica, 2011, 67(4):271-281.

[11] 王　璐,官　琴. 基于 PowerPC 嵌入式平台的 WindML 图形驱动设计[J]. 兵工自动化, 2013, 32(5):95-96.

[12] 张继伟. 基于 WindML 环境下的显卡驱动设计[J]. 现代电子技术, 2010, 33(14):78-80.

[13] 李　鸽. 基于 VxWorks WindML 组件的 SM502 显卡驱动开发[D]. 西安:西安电子科技大学, 2014.

[14] 谢周标. 嵌入式二维图形硬件加速引擎研究与设计[D]. 长沙:湖南大学, 2013.