基于物联网技术的视频监控系统设计

王永强

(唐山学院 计算机科学与技术系,河北 唐山 063000)

摘要:设计了一种基于物联网技术的视频监控系统。该系统以基于 arm11 的 real6410 开发板作为嵌入式平台,并搭载 zc301 的 USB 摄像头以及 apm6658 SDIO WIFI 模块,在 PC 机上使用 Qt 进行图形界面开发,接收和显示采集到的每一帧图像,实现了可视化监控的功能。而且根据用户需求还可以将视频以 AVI 格式进行保存。

关键词:物联网技术;视频监控系统;硬件组成;软件环境

中图分类号:TP277.2 **文献标志码**:A **文章编号**:1672-349X(2015)06-0027-04

DOI:10.16160/j.cnki.tsxyxb.2015.06.011

Design of an IOT-based Video Surveillance System

WANG Yong-qiang

(Department of Computer Science and Technology, Tangshan College, Tangshan 063000, China)

Abstract: The author of this paper has designed an IOT-based video surveillance monitor system, which has real6410 of arm11 as an embedded platform, zc301 USB cameras, and apm6658 SDIO WIFI modules. The system can receive and display each frame image acquired to achieve a visual monitoring which can be saved as AVI files. The graphical interface of the system is capable of being developed with Pt.

Key Words: IOT; video surveillance system; hardware; software

随着计算机技术的快速发展,以及人们安全意识的提高,视频监控系统已被广泛应用到各个领域[1]。传统的视频监控系统主要分为两类:一类是基于闭路电视的,监控区域有限;另一类是基于 PC 机的数字化监控,体积大,组网较为死板,移动性差。基于物联网技术的视频监控系统,则可以很好地解决传统视频监控系统的弊端,它突破了有线网络的局限性[2-3],给信息交流提供了更大的便利,具有组网方便、便于安装、可靠性高、运行稳定等特点,而且随着物联网技术的不断提升,视频监控系统正向着数字化、网络化、智能化的方向发展。因此,基于物联网技术,本研究设计了一种视频监控系统。

1 系统组成

系统以嵌入式物联网平台为基础,对图像进行实时采集,并通过无线网络进行实时传输。系统的组成包括:硬件设备、USB 视频数据的采集、WIFI 无线数据的传输、Qt 平台数据显示及存储。

1.1 系统硬件

硬件设备主要包括 PC 机和开发板,PC 机上运行 ubuntu12.04;开发板型号为 real6410,使用 arm11 处理器[4],运行 linux2.6.28 版本内核,并搭载了 apm6658 SDIO WIFI 模块以及 zc301 的 USB 摄像头模块。对于 real6410 来说,开发板上集成了 USB 接口和 SDIO 接口,能够满足相应要求。

1.2 USB 视频数据采集

USB 视频数据采集部分主要是实现对 USB 摄像头的能力检查,初始化,并且进行一些相应的格式和流设置[5-6]。由于 USB 摄像头采集到的数据放在内核缓冲区,而程序需在用户空间运行,这就需要使用一种通讯方式来进行用户空间与内核空间之间的联络。一般通讯方式有三种:第一种是直接读写内核缓冲区,第二种是内存映射,第三种是使用用户空间指针。本系统采用的是第二种,即 mmap 内存映射方式。首先,向内核申请 5 块帧缓冲区,并将内核实际给予的缓冲区通过 mmap 映射到用户空间,这样程序便可以在用户

收稿日期:2015-07-10

作者简介:王永强(1980—),男,河北丰南人,讲师,硕士,主要从事物联网技术及机器视觉技术研究。

空间访问内核空间了。其次，开启视频采集，这时设备便会将采集到的帧数据放到内核缓冲区中，在内核空间开辟的缓冲区事实上是一个循环缓冲队列，用户空间的程序在取回一帧数据后需把该缓冲区重新放回内核缓冲队列中，以便内核继续使用它来存放接下来采集到的帧数据。

1.3 WIFI 无线数据传输

WIFI 无线数据传输部分主要是实现接收客户的连接请求、处理用户的传输请求，以及判断用户是否退出。由于 tcp 传输会进行一系列的差错检验，一旦报文出错，例如乱序、乱码等，便会请求进行重发，这样当网络状态不理想时，会造成视频不能流畅显示，因为大部分的时间都被 tcp 的重发机制占用了。鉴于 udp 传输没有 tcp 传输那样的差错控制机制，所以本 WIFI 无线数据传输使用的是 udp 传输，这样当传输的报文出错时，可以在程序中直接简单地丢弃这帧数据。另外，由于网络负载问题，每帧数据都是先进行分包传输，然后在 PC 机上再进行组包重新形成一帧数据。

1.4 Qt 平台数据显示及存储

Qt 平台部分主要是实现与用户的交互，如输入服务器的 IP 及端口、显示当前连接的状态、实时视频显示、是否暂停视频的实时显示、是否进行视频的录制等。Qt 端需要对接收到的帧数据进行显示，显示是通过 Qtimer 来控制的，将 Qtimer 的信号与相应的槽函数绑定，定时器时间一到便会执行槽函数，该槽函数的功能便是向开发板索取一帧数据并且显示。Qtimer 不断索取视频帧数据，并且连续显示每一帧，这样便形成了流畅的视频，而且如果此时用户选择了录制，则会在显示实时数据的同时，调用 mencoder，将接收到的数据转换为视频存储起来，以实现后期查看。

2 系统工作流程

系统中，主要涉及到两个平台：一是以 real6410 为基础的嵌入式平台，二是以 PC 机为基础的 x86 平台。开发板作为服务器负责帧数据的采集和传输，PC 机作为客户端获取视频数据并显示和存储。

2.1 服务器工作流程

服务器（开发板）上电后，先启动服务器程序，程序启动后，服务器会启用 udp 服务，帮助提取服务器的网络信息。根据客户端发来的命令作相应处理。如果传输报文错误，则采用丢帧的方式进行处理。系统中的无线传输主要传输帧数据以及一些 PC 机与开发板交互的命令，概括来说，主要有以下几种交互方式：

（1）客户端发送“# # # #”表示请求连接，服务器收到“# # # #”后会等待接收用户命令。

（2）客户发送“GOON”表示请求服务器发送一帧数据，服务器收到该指令后会采集一帧数据，并且将该帧数据拆成多个包循环发送给客户端，直到发完为止。

（3）客户端发送“STOP”表示客户端即将退出，收到该命令，服务器不再为该客户服务。

服务器工作流程如图 1 所示。

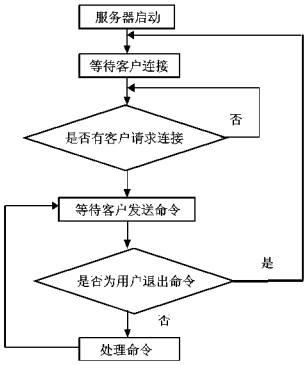

图 1 服务器工作流程

2.2 客户端工作流程

在 PC 机上使用 Qt 搭建客户端。启动客户端程序后，输入服务器端口以及 IP，点击连接，此时状态标签会显示当前连接的状态。如果连上，用户可以点击“start（开始）”按钮进行视频实时显示。如果用户点击“AVI（录制）”，视频则以 AVI 格式存储在家目录的相应目录下。客户端工作流程如图 2 所示。

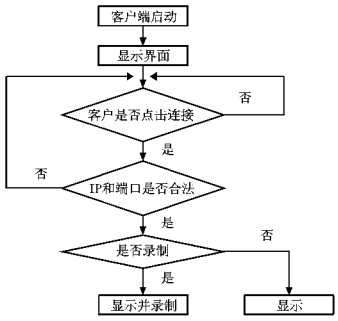

图 2 客户端工作流程

客户端部分关键代码如下：

```
//记录是否点击 AVI 录制
avi_flag=ui->record_button->isChecked();
//发送 GOON
sendto(client_fd,"GOON",4,0,(struct sockaddr *)
&server_addr,len);
//接收数据
memset(image_buf,0,118784);
while(1)
{
    //select
    FD_ZERO(&fds);
    FD_SET(client_fd,&fds);
    timeout.tv_sec=1;
```

```
        timeout.tv_usec=0;
        ret = select(client_fd+1, &fds, NULL, NULL,
&timeout);
        if(ret==0)
        {
            fprintf(stderr,"timeout\n");
            return;
        }
        else
        {
            ret = recvfrom(client_fd, p, PAGENUM, 0,
(struct sockaddr*)&server_addr,(socklen_t*)&len);
            p+=ret;
            sum+=ret;
            if(sum==118784)
                break;
        }
    }
    //显示
    *image = QImage::fromData((uchar*)image_buf,
118784);
    *pixmap=pixmap->fromImage(*image);
    ui->my_display->setPixmap(*pixmap);
    ui->my_display->setFixedSize(800,480);
    //是否记录 AVI
    if(avi_flag==true)
    {
        time_t t;
        struct tm *now;
        int fd;
        char buffer1[1024]={0},buffer2[1024]={0};
        t=time(NULL);
        now=localtime(&t);
        if(now==NULL){
            perror("localtime");
            return;
        }
```

3　软件环境的搭建

嵌入式软件的开发工作一般都在 PC 机上完成，交叉编译后通过串口或者网口烧写到板子上，这就涉及到交叉编译工具链、串口调试工具以及 nfs 或 tftp 服务器的搭建。交叉工具链用于编译 uboot、内核和应用程序等，以便生成适合于目标板系统的代码；串口调试工具是为了在 PC 端登录开发板，以便运行并调试程序；nfs 或 tftp 服务器一般用于 PC 机与目标板之间的文件拷贝。在开发板上，需要烧写 uboot、linux2.6.28 版本的内核、cramfs 文件系统、ubitfs 文件系统以及相关的库和可执行文件，以下是软件环境的搭建步骤。

3.1　交叉编译工具链

要编译出 arm 平台的可执行代码，需要在 PC 机上安装交叉编译工具链。本设计中的系统处理器是 armv6 架构的，经过查阅相关的资料，发现早期的交叉编译器，例如，arm-linux-gcc-3.4.1 不支持 armv6 构架，因此需要使用更新的通用的交叉编译器，这里选用 arm 公司新推出的 EABI 编译器，即 arm-none-linux-gnueabi-4.3.2。该编译器使用了新的 glibc 库 2.8，并在编译器中预先安装好了各种需要用到的库文件，如编译 qtopia 时需要用到的 jpeg，zlib，libts，libuuid 等。首先，通过 windows 与 linux 下的共享文件夹把 arm-none-linux-gnueabi-4.3.2.tar.gz 压缩包拷贝到 linux 下，解压；其次，在家目录下的.bashrc 文件中编辑 PATH 路径，PATH =/home/zhang/arm-none-linux-gnueabi-4.3.2//=bin:：$PATH；最后，在家目录下执行 source .bashrc 命令更新 PATH，可以使用 which arm-linux-gcc 命令来查看是否配置正确。

3.2　配置内核并重新编译

linux2.6.28 发布的内核中没有 USB 摄像头驱动，并且 WIFI 模块的驱动部分不支持 AP，需要重新编译内核源码将驱动加载进去，并重新烧写到开发板上。编译内核时，一些库和程序需要安装。进行安装时执行以下几条命令：

```
sudo apt-get install libncurses*
sudo apt-get install kernel-package
sudo apt-get install build-essential
sudo apt-get install libqt3-compat-headers
```

进行 WIFI 模块的内核相关配置。先添加固件，从网上下载 helper_sd.bin 和 sd8686.bin 两个固件复制到内核源代码目录下的 firmware 目录下，将 helper_sd.bin 改名为 sd8686_helper.bin。然后进行网络支持的配置和网卡驱动支持的配置。

配置 USB 摄像头驱动。虽然使用的摄像头是非 UVC 的，但为了实现摄像头能即插即用，因此在配置 USB 驱动时，应使内核尽量支持多种摄像头。本设计中使用的是 zc301 的摄像头驱动支持和 UVC 摄像头驱动支持。

3.3　烧写 uboot，kernel 和文件系统

烧写在 windows 下执行，使用 DNW 进行烧写。需要安装 DNW 的 USB 驱动，可以从网上下载，也可以使用开发板自带的。

3.4　移植 wpa_supplicant

首先移植 wpa_supplicant 工具包。下载此工具包依赖的 openssl 库和 wpa_supplicant，并解压。进入 wpa_supplicant 目录，将 patch 目录下的补丁文件 openssl-0.9.8e-tls-extensions.patch 拷贝到 openssl 目录：

cp openssl-0.9.8e-tls-extensions.patch openssl-0.9.8e/

其次进入 openssl 目录下执行一下命令来打补丁：

patch-p1＜openssl-0.9.8e-tls-extensions.patch

最后执行 make，make install，这样就生成了 openssl 库，并且存放在/usr/local/ssl 目录下。

接下来交叉编译 wpa_supplicant。进入 wpa_supplicant 目录下，将该目录下的 defconfig 文件拷贝为.config 文件，因为 defconfig 文件只是一个模板文件，里面所有的行都是注释，只供参考，只有将其拷贝为.config 文件才能使用。对该文件进行修改，添加如下行：

CC＝arm-linux-gcc-L/usr/local/ssl/lib/

CFLAGS＋＝-I/usr/local/ssl/include/

LIBS＋＝-L/usr/local/ssl/lib/

3.5 nfs 服务器搭建

由于使用的是在 PC 机上开发的编译程序，此程序需烧写到开发板上才能执行，所以需建立 nfs 服务器，在目标板上挂载进而来传输文件。在 PC 机上下载 nfs 服务器，修改配置文件，将 PC 机上的目录设为 nfs 共享目录，并且开放给所有能连接 PC 机的用户，开发板每次启动时执行：

mount-t nfs 192.168.1.103:/home/wang/nfs/nfs-o nolock

这样便将 PC 机的 nfs 目录挂载到了开发板的 nfs 目录上，其优点是放到 PC 机的 nfs 目录下的文件，可以从开发板的 nfs 目录下取出，十分方便。因为每次开机都要输入这个命令，所以把它写到脚本里命名为 nfs.sh，每次开机后./nfs.sh 挂载即可。

4 图形用户界面的实现

通过 qtcreator 进行可视化的 Qt 界面开发，可以减少工作量，提高开发的效率。界面的构成主要有：中央的视频显示窗口、四个按键（分别是“connect（连接）”“close（关闭）”“start（开始）”“stop（暂停）”）、一个状态标签（用于显示当前的连接状态以及一些出错信息等）、两个输入框（分别是 IP 和端口，在程序中已经用正则表达式规范了 IP 的书写方式，这样可以避免 IP 输入不合法）、一个 AVI 选择按钮（当选中时会进行视频的录制与保存）。基于 Qt 的界面设计显示结果如图 3 所示。

5 运行效果

启动服务器，点击客户端连接按钮，然后点击“start”按钮，进行视频捕获，实现视频数据的实时传输。客户端运行效果如图 4 所示。

6 结论

设计的基于物联网技术的视频监控系统，通过搭载 zc301 的 USB 摄像头和 apm6658 SDIO WIFI 模块，以及在 PC 机上使用 Qt 进行图形界面开发，可以接收和显示采集到的每一帧图像，实现可视化监控的功能。而且根据用户需要

图 3 基于 Qt 的界面设计

图 4 客户端运行效果

还可以将视频以 AVI 格式进行保存。运行效果显示，本视频监控系统可对现场进行实时有效地监控，而且设计成本较低，易于组网，可扩展性强，运行稳定，具有较高的实用价值。

参考文献：

[1] 娄德成，韦银. 基于 B/S 架构的嵌入式远程视频监控系统的设计[J]. 微型电脑应用，2014，30(9)：51－53.

[2] 陈奕舟，吕勇，许远向，等. 基于 ARM9 构架的热释电车载视频监控系统设计[J]. 北京信息科技大学学报，2014，29(1)：82－84.

[3] 刘涛，吕勇，毛海波. 基于 Wi-Fi 技术的无线视频监控系统设计[J]. 现代计算机，2015(1)：49－52.

[4] 江俊杰，王志明. 基于 X264 的嵌入式视频监控系统设计[J]. 计算机工程与设计，2013，34(12)：4109－4203.

[5] 王溢琴，秦振吉，芦彩林. 基于嵌入式的智能家居之视频监控系统设计[J]. 计算机测量与控制，2014，22(11)：3623－3626.

[6] 罗金玲，刘罗仁. 基于 ARM＋Linux 的物联网远程监控终端设计[J]. 计算机系统应用，2013，22(1)：189－191.

（责任编校：李秀荣）